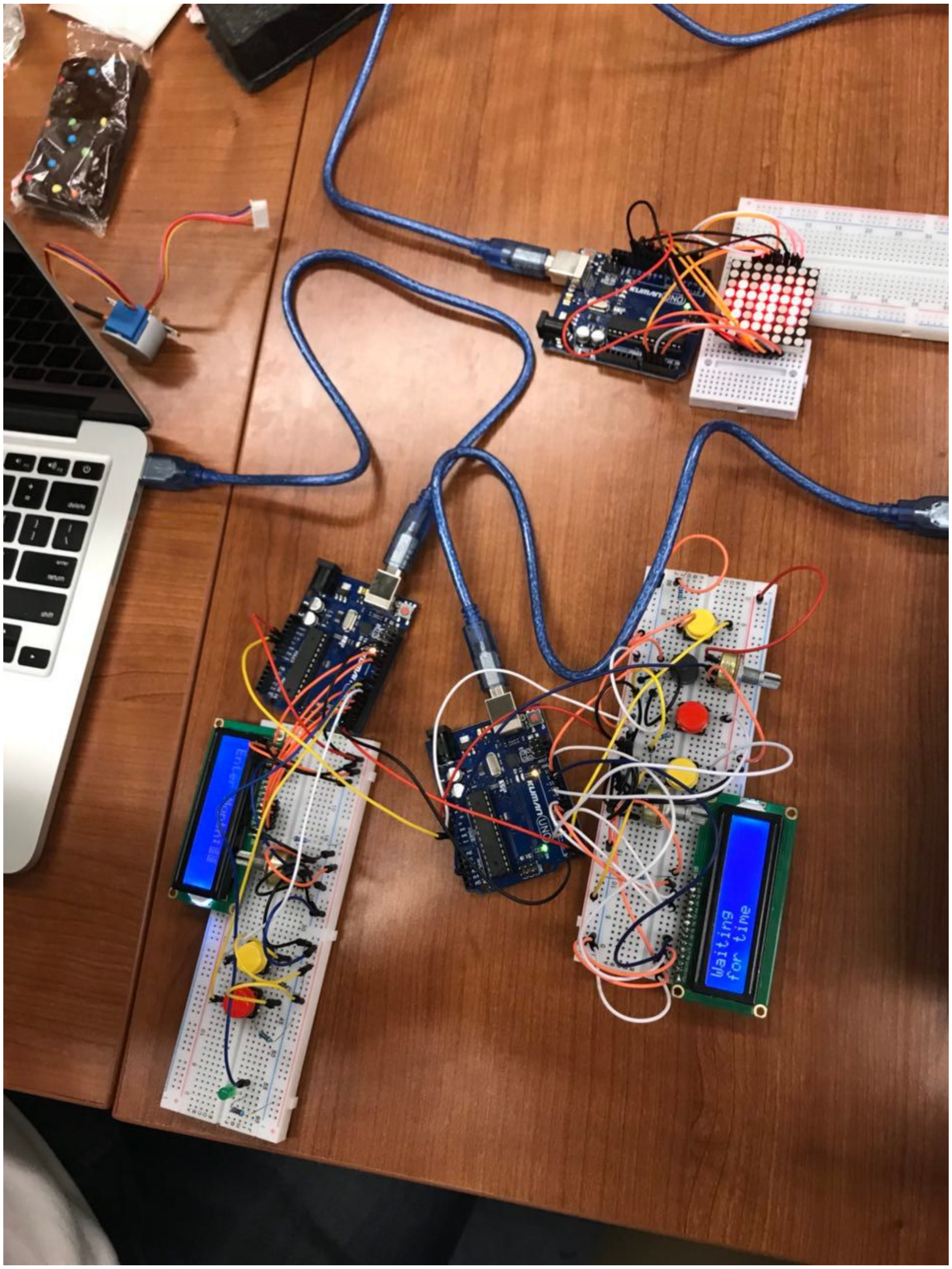
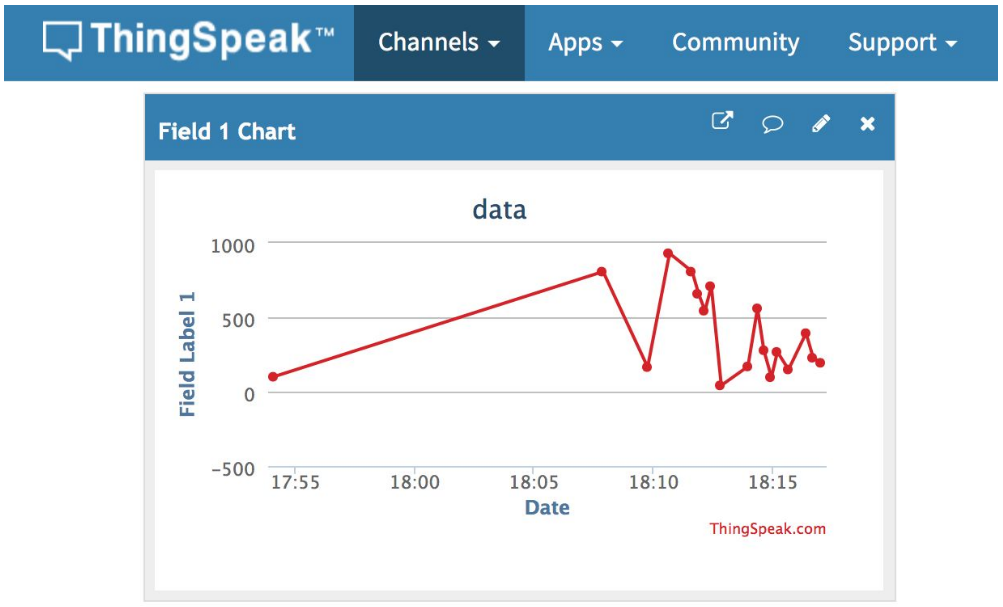
Ahmed Khan
Edgar Martinez
Ryan Trokey

# Smart Clock



## Overview

We began our project with the goal to create a smart clock that would have multiple features. These features included: displaying the date and time, displaying the weather temperature and conditions through the use of wifi, having music play through bluetooth connection to a phone, setting an alarm and having an alarm go off, having led lights turn on depending on how much light was present in the room, showing different US time zones, and displaying inspirational quotes. We began on our work with features that we already had some background knowledge in from our labs. One of the first parts of the project that we completed was setting up an lcd display and outputting the date and time to the screen. Another feature we soon after implemented was the ability to communicate between two arduinos. One arduino would send the date and time to the other arduino, making it available to that arduino to display on it's own lcd display. The receiving arduino was also the arduino responsible for being able to set an alarm and sound the alarm. Once this arduino successfully set an alarm, that information would be passed back to the first arduino, which turns on an led light to signal that alarm has been set. Whenever the alarm is turned on or off the signal is sent to the first arduino in order to turn on or turn off the led light. One of the next steps we decided to take was to attempt to connect our arduinos to wifi to receive date, time and weather information from the internet. In order to do this we ordered a few esp8266 wifi chips off the internet that would allow us to connect. After many attempts with the wifi chip, we were finally able to connect them to the internet and push data from our arduino to the internet which was pictured on a graph shown below.



With this success also came some failure. After countless attempts we were unsuccessful in obtaining or “pulling” information from the internet into our arduinos. It was at this point that we decided that we would remove the wifi component from our project, and continue on with input from the user through the serial command prompt in our arduino IDE. With this new implementation we were able to input weather information from the serial command prompt, and display the temperature in degrees and then a picture of a sun if it is sunny, a cloud if it is cloudy, a rain cloud if it is raining, and a snowflake if it is snowing on the 8x8 led matrix. This is also where we decided to add our inspirational quote of the day, depending on if it is sunny, cloudy, raining or snowing. In conclusion, our first arduino allows for setting the date and time, and lighting up an led light to signal that an alarm has been set, or turning off the led light if the alarm has been turn off or set to snooze. This arduino also sends the date and time information to our second arduino. The second arduino receives the date and time information, displays it on the screen, and also allows for the user to set an alarm like an alarm clock would. This is when the signal is sent back to the first arduino to tell the led light to turn on or off depending on if the alarm has been set or turned off. When this alarm time is hit on the lcd display the buzzer goes off, simulating an actual alarm clock, with the feature to turn off the alarm with the use of a snooze button. The third arduino receives information from the user through the serial command prompt for weather. The 8x8 led matrix connected to this arduino will then display the temperature in degrees and then a picture depending on the conditions outside followed by an inspirational quote depending on these conditions.

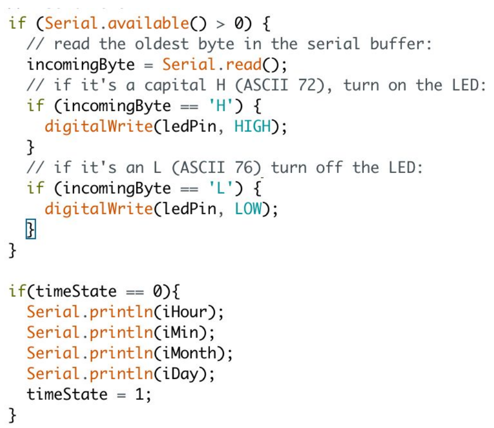
## Prototype/Design

The overall design of the smart clock was broken up into three different parts:

### *Date and Time Setter:*

- Display time thru LCD display, originally the idea was to use a 4 7-segment display for the time, but it ended up requiring to much time and thus we switched back to using an LCD display
- Date was originally not really a part of the prototype since we were planning on using a 4 7-segment and that wouldn't have allowed us to

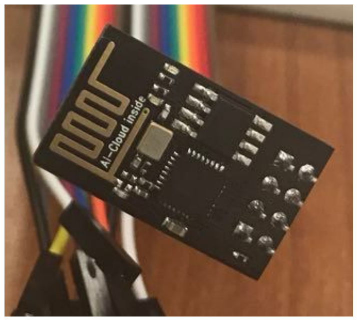
display time nicely, but since we ended up switching to using the LCD display that allowed us to display more information, in this case the date



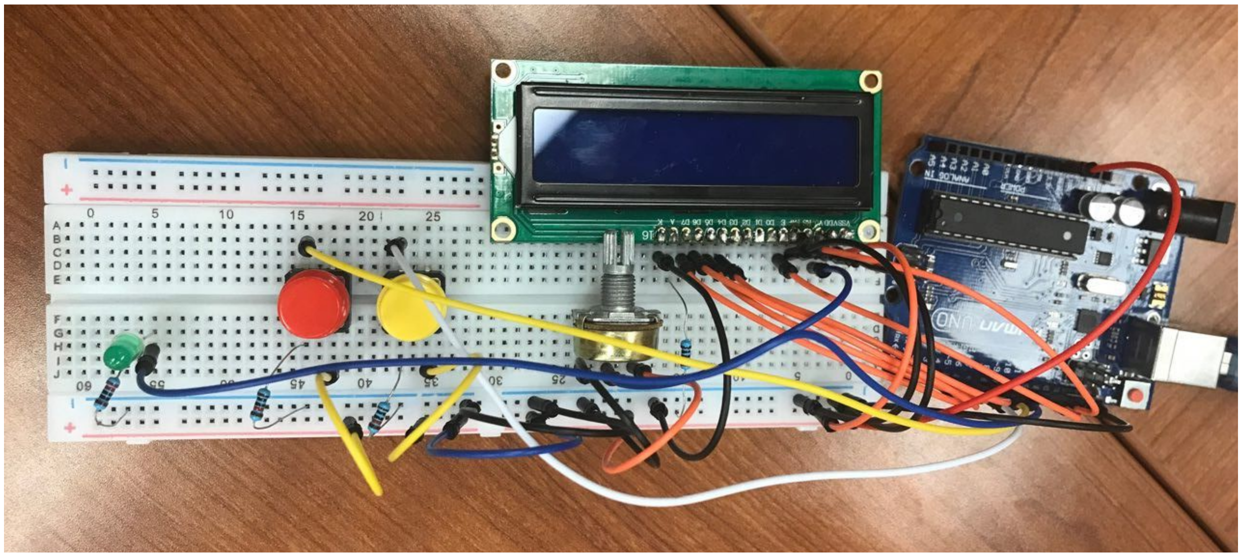
```
if (Serial.available() > 0) {
  // read the oldest byte in the serial buffer:
  incomingByte = Serial.read();
  // if it's a capital H (ASCII 72), turn on the LED:
  if (incomingByte == 'H') {
    digitalWrite(ledPin, HIGH);
  }
  // if it's an L (ASCII 76) turn off the LED:
  if (incomingByte == 'L') {
    digitalWrite(ledPin, LOW);
  }
}

if(timeState == 0){
  Serial.println(iHour);
  Serial.println(iMin);
  Serial.println(iMonth);
  Serial.println(iDay);
  timeState = 1;
}
```

- Was also to get time thru Wifi, but ended up using user input through buttons to get time and then use that time to talk to the alarm clock manager through serial communication and send the current time and date, so that alarms can be set. The method that we were gonna use to get the time was through an NTP server and get the time that way. When we were unable we tried using a prebuilt API, but that also wouldn't work correctly with the WIFI module.



- Has an LED that is lit up whenever an alarm is set up by the alarm manager to notify the user in a more visual sense that an alarm is active. Once an alarm is armed it turns on and can be seen from anywhere. After an alarm is snoozed or disarmed the LED is turned off as to indicate that there is currently no alarm active.

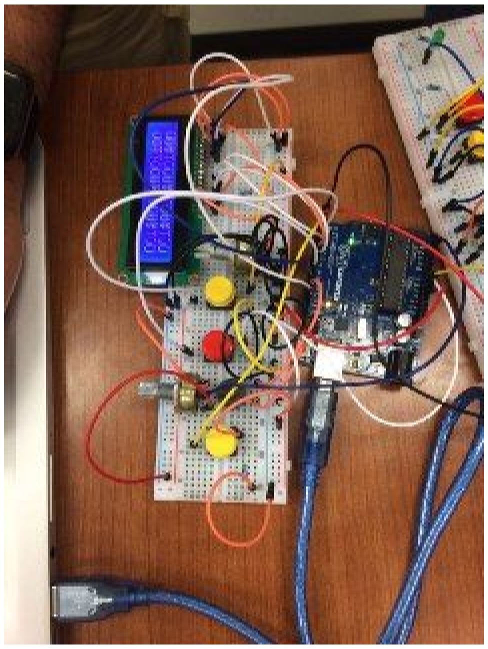


*(This is a picture of the set up used for setting the date and time. The yellow and red buttons are used to choose the month, day, hour and minute.)*

## *Alarm Clock Manager*

- Uses the LCD display, a button for activating the alarm, a potentiometer for changing with hour and minute the alarm will be set to, another button for setting the alarm, and a third button for a snooze button, and the buzzer for the alarm itself
- The LCD display connects to 4 digital pins, each button uses a digital pin, the buzzer uses a digital pin, and the potentiometer connects to an analog pin
- With all of the wiring set up correctly, the rest of the work is done through the IDE
- When an alarm is not set the LCD is programmed to display “No alarm set”

- The LCD display will also display the time that is received from the first arduino via serial connection. The serial connection between arduinos only sends bytes at a time and not ints, so we needed to send these values typecasted to chars and then typecasted back to ints for display purposes
- When activating an alarm, the button is programmed to do a digital read and when that signal is received the LCD display shows numbers for setting the alarm, which can be changed using the potentiometer.
- The potentiometer is programmed to scroll through numbers(1-12) for the hour and (1-59) for the minute when setting the alarm. When twisted all the way to the right the number reads 12 or 59, all the way to the left reads 1 for both, and then the numbers in between when not on the far right or left.
- In order to set the hour the second button is coded to do a digital read for the button press signal, and when that is done whichever number is on the screen from the potentiometer will be the hour used to set the alarm. The same thing is done to set the minute for the alarm.
- There is a simple if statement to check if the time that was received from the first arduino matches the time set for the alarm. If they match, then the buzzer will emit its sound(the same way an alarm clock will go off when the time hits 7:00 A.M. as long as the alarm is set to 7:00 A.M.)



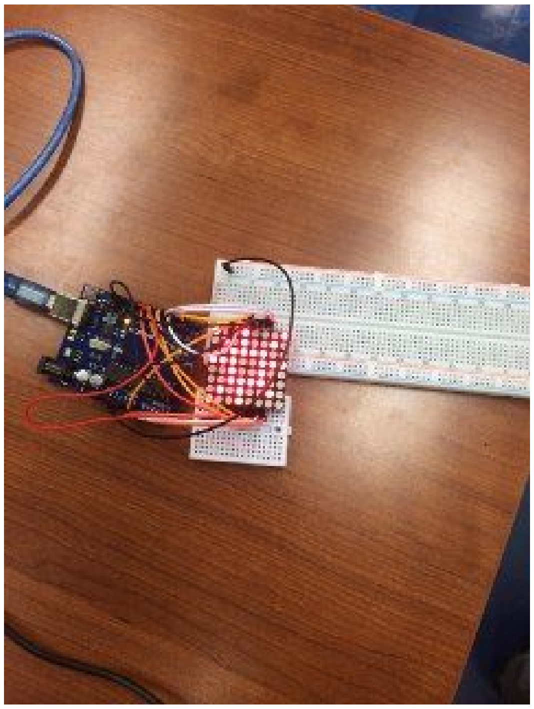
## *Weather/Quote Displayer*

- Uses the 8x8 LED matrix to display temperature, outside conditions and the inspirational quote of the day
- The matrix has 8 connection pins on 2 parallel sides, resulting in a total of 16 pins to be connected
- These 16 pins connect through digital pins 2-13 on the arduino and also the first 4 analog pins, A0-A3
- The matrix also requires the use of both breadboards given in the kit because of its size
- Once successfully wired, the rest of the work was done inside the IDE in order to transmit the correct lights to be lit up on the matrix
- This was done by creating a byte[] array which had 8 rows of 8 bits inside each row. Each of the 8 bits in each row symbolized on of the led lights on

the 8x8 matrix board. In order to set a light to be turned on, the placeholder for that light would be a 1, and if it was to be turned off the placeholder would be a 0

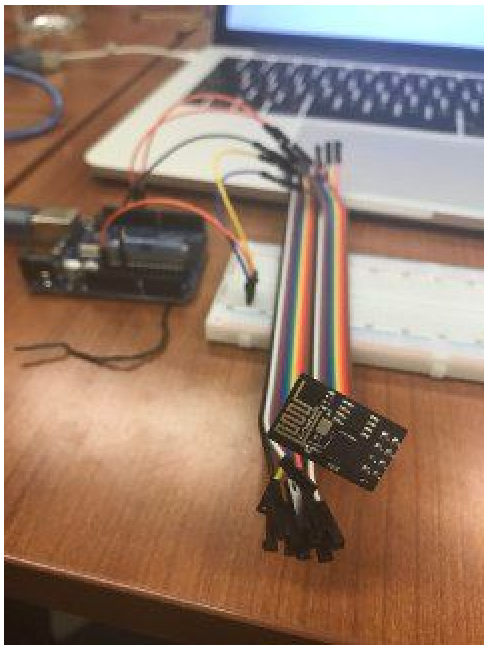
- For example, this is what the byte[] array would look like for the letter ‘A’
- byte A[] = {B00000000,B00011000,B00100100,B00100100,B00111100,B00100100, B00100100,B00000000};
- Each letter and number would require a different array setup in order to display the correct letter/number to the matrix. Since it would be redundant to create 100 different byte[] arrays for numbers 1-100, a simple series of if statements were used to display the correct temperature to the screen
- For example, if the user entered 75 for the temperature in the serial command prompt, 10 if/else statements were in place seeing if the number entered was between 0-9, 10-19, 20-29, etc. to display the first number of the temperature to the matrix. Then 10 more if/else statements were in place to determine the second number of the temperature to be printed to the screen by using the modulus operator and printing the remainder
- The user is also asked to enter a number 1-4 depending on if it is ‘1’: Sunny, ‘2’: Cloudy, ‘3’: Raining, or ‘4’: Snowing. Each of these 4 options have their own byte[] array which displays a picture of each weather condition

- The last implementation in this part of the project is the inspirational quote. We decided to display a different inspirational quote based on the weather condition outside. So if it was sunny it displayed a certain quote, cloudy had its own quote and so on.


(This example of the 8x8 matrix is currently displaying the picture of the sun, depicting that it is sunny outside.)

## Problems/Issues

The biggest issue that we encountered when building this project was getting the wifi to work to get the weather and time based on your location. The goal of the project was to make a smart clock and that involved getting date, time, and weather from the exact location that the user is in from the internet, thus taking away the hassle of setting up a clock. The first problem that we encountered with the wifi was a tragedy. When attempting to connect the wifi component to the correct wires of the arduino, one of us accidentally dropped the small piece onto the floor. We said guys don’t move I just dropped the wifi module on the floor. At this very moment Faraaz moved his rolling chair to look for the component and we all heard a cracking sound. The wheel of the chair Faraaz was sitting on rolled over the module, crushing and breaking it. We were then sadly forced to order another component off of the internet, delaying our production for a couple of days. Fortunately, this was early on in our development process, so we had time to spare, and this time we ordered a few extra components in case of a disaster like this occurring again. After weeks of working through the esp8266 wifi component, we were successfully able to connect to wifi and push data from our arduino to our online connection database which is shown above in this document. After a couple more weeks of attempting to receive/’pull’ the weather and time information off the internet, we were unsuccessful and finally decided to cut out the wifi component as a whole. Another problem we ran into was with the serial connection between two arduinos. We used the serial connection to send the date and time information from one arduino to another. We initially were trying to send these values as ints to the other arduino but kept running into the problem of it not working correctly. The values we were getting kept reading -1. After doing some research on the internet about Serial.read(), we discovered that Serial.read() returns the first byte read, and a -1 if no data is received, for example an int. In order to fix this issue we decided to typecast the int as a string(which is the same as a char array) and send the numbers one at a time as chars. This allowed us to receive the correct numbers and then type cast them back to ints in order to use them correctly in the receiving arduino.



## Improvements

There are two main improvements that would really help turn our version of a smart clock into a highly functional smartclock that could be available for retail sale. The first and main improvement would be the wifi feature. If we were able to successfully include wifi we could be able to upload date, time and weather information from any location that the smart clock was used. This would include temperature, humidity, wind levels, conditions(including sunny, rainy, cloudy, snowy), chance of weather for preceding days in the week, just like how the weather app works on our smartphones. On top of the wifi, if we had more available digital pins and components to use from our arduino kits we would be able to add more feature boards and display features to show all of this information that we are trying to display. We discovered that the 8x8 led matrix is a pretty cool part to use. We would have liked to used multiple of these side by side but they take up all of the digital pins and four of the analog pins so we were only able to use one of them. Also, when setting the alarm we use a potentiometer to scroll through the numbers to set the hour and minute for the alarm. When setting the date and time on the other arduino we have to use buttons for that feature because we ran out of potentiometers for use. So if we had more potentiometers we could have made our set up a little cleaner but our implementation is still nice.

## Conclusion/What we learned

From this experience we learned a lot about the arduinos and writing code in general. First off, we became a lot more familiar and comfortable with a lot of the features these arduinos provide such as using external devices for digital reads(potentiometer, buttons, LCD display). We figured out that using the potentiometer for choosing the time is a lot nicer than using the buttons, but we ended up using the buttons in one portion because we ran out of potentiometers to use. Another feature of the arduinos that we learned a lot more about is the serial communication between two arduinos. In our project, we were sending integers over the serial connection, and we quickly learned that was not going to work the way we planned. After reading the explanation pages for Serial.read(), we learned that the serial communication reads one byte at a time so we had to send our ints through as chars, and then typecast them back to ints when received. This was one of the highlights of our production. We also learned how difficult implementing a wifi connection would be. We were pretty proud for being able to connect to wifi and push data to the internet, but we learned that we are still too inexperienced with the arduinos to collect weather information off the internet and pull it into our arduino systems. One of the last things we learned was how to connect and use the 8x8 LED matrix. It was interesting creating the different arrays and rows of each matrix in order to have the matrix display the letter, number or picture that we wanted. Overall, we learned a lot more than we expected about the arduino and its features. We all thoroughly enjoyed working on this project and look forward to learning more in the future.

# Resources

WIFI Connection Tutorials

- https://medium.com/@cgrant/using-the-esp8266-wifi-module-with-arduino-uno-publishing-to-thingspeak-99fc77122e82
- https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/examples/NTPClient/NTPClient.ino
- https://www.esp8266.com/viewtopic.php?f=29&t=6007&start=4

Clock Management

- * Melody code and pitches.h code found at: https://www.arduino.cc/en/Tutorial/PlayMelody

- * Time Library Source Code and Documentation found at:
  https://github.com/PaulStoffregen/Time
- * Serial.available() example found at:
  https://www.arduino.cc/en/Serial/Available#.UwYy2PldV8E
- * convert function code found at:
  https://stackoverflow.com/questions/929103/convert-a-number-range-to-another-range-maintaining-ratio

Weather Station

- https://create.arduino.cc/projecthub/SAnwandter1/programming-8x8-led-matrix-23475a