

Solder Time Desk Clock

Andres Morales, John Lee

Professor Zia, PhD (Mentor)

CET 4811-D488

Computer Engineering Technology

New York College of Technology - CUNY

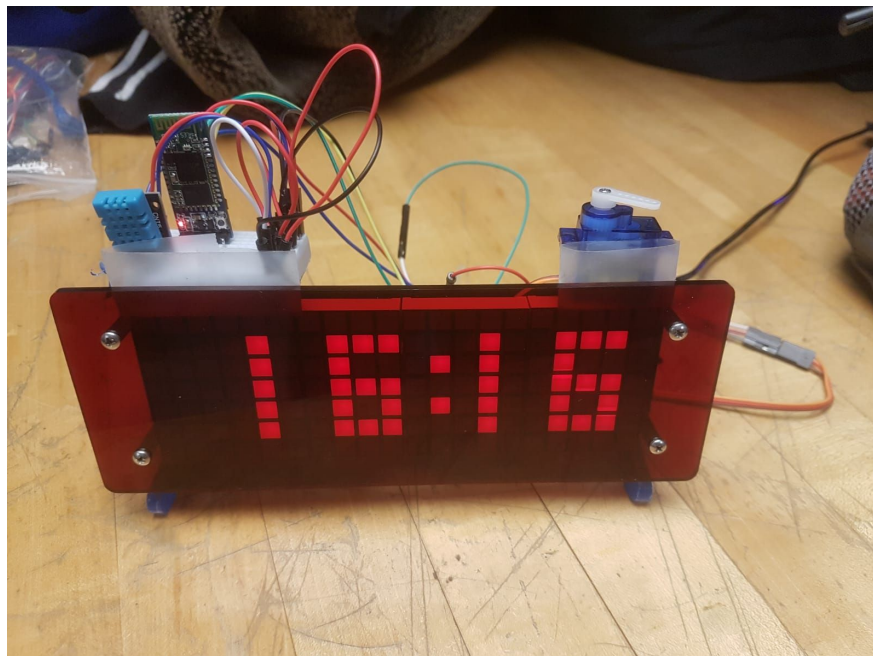


Table of Contents

1. Cover Page.....	1
2. Table of Contents.....	2
3. Project Summary.....	3-5
4. Brief Code Sample.....	5-7
5. Project Design.....	8-10
6. Components Datasheets.....	10-11
7. Project Management.....	12
8. Conclusion.....	12
9. References.....	

14

1. Introduction

This desk clock has a vast range of functionalities such as the more basics being time, date, and alarm. We continued to modify the device by adding functions such as weather display and bluetooth. To further enhance the functions of a simple clock and make it more useful in the everyday life of a person. We have a digital clock that displays the time digitally (in numerals or other symbols), as opposed to an analog clock, where the time is indicated by the positions of rotating hands. This device will not only display time but also functions as a multi-purpose device by displaying weather, written messages, and figures. The real life applications would be to use this item as an everyday device since the moment you wake up able to customize it to the specifics users desires from everything even a message display for the users favorite weather temperature range a sort of thing to brighten their day.

2. Challenges

Various challenges were encountered while making the device some go as far as hardware such as the soldering process, which created a few problems that had to take time to resolve with delicacy if not one of the components could have been damaged in the process. The issue came about when we were trying to implement another component on the PCB since the space was really narrow the components were almost shorten by soldering other

components together by accident but it was resolved by removing the solder slowly and delicately. One of the biggest issues was the Arduino UNO microcontroller(custom designed), which was faulty and wasn't compiling the code correctly. Many things were done such as burning the chip and trying to replace the chip but nothing seemed to work until the microcontroller was resetted was the problem solved. Another challenge was to get the whole components to work such as the Sensor. The code was more complex and adding it to the already included code seemed to cause a whole lot of issues for us. The kit we used had a lot of limitations with a lot of I/O being used up by other components.

3. DHT-11

The DHT-11 was meant to be our Temperature & Humidity sensor to display the weather on the clock. It was gonna be working together with our servo motor which would display a smiley face if the weather was in range to the users choice of Favorable Weather Settings. It would have been an easy way to let the user know the temperature and start the day right.

3.1 HC-05 Bluetooth Module

The HC-05 was our way of connecting the clock to a phone via bluetooth to display the setting on the phone. It was going to display the temperature and humidity collected by the Device and keep a record to keep track on the changing weather and humidity throughout the week/days.

3.2 Overview

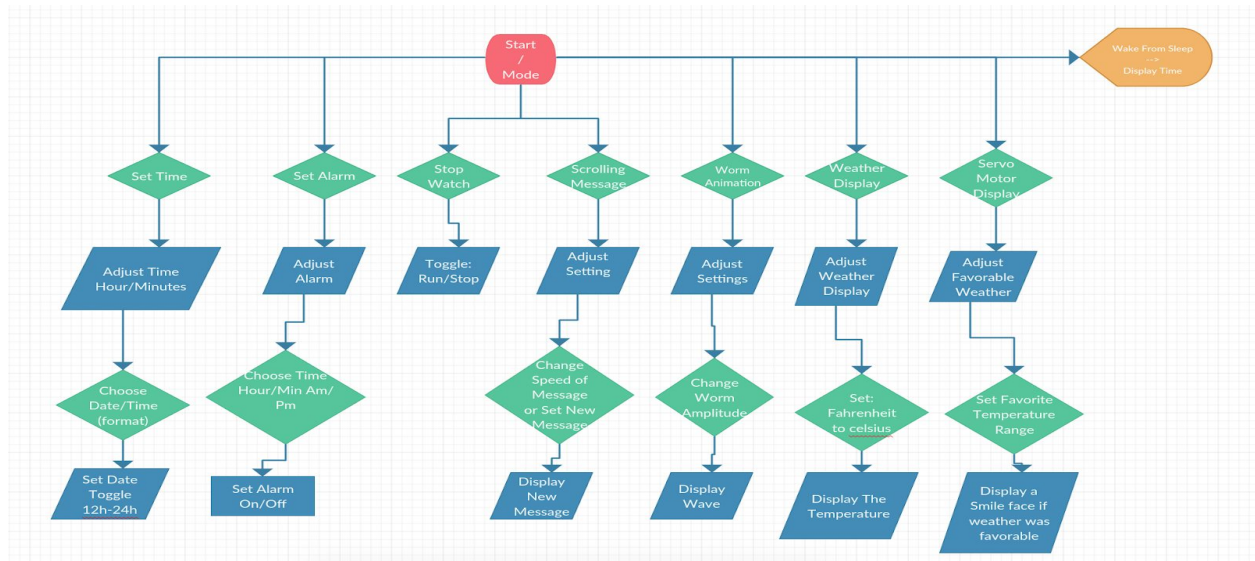


Figure 1: Overview of Commands

This is flowchart of the diagram when user press Mode(button 1) and Set(button 2) to see reacts in the subsystem.

4. Code

The following images display the code created for this project.

```

// ***** Main Loop *****
void loop()
{
  // Test for Sleep -----*
  currentMillis = millis();
  OptionModeFlag = false;

  if(((currentMillis - SleepTimer) > SleepLimit) && SleepEnable)
  {
    if(STATE= 1) // New for ST Desk Clock - goto Time vs Sleep
    {
      SUBSTATE = 1;
      blinkON = true;
      blinkFlag = false;
      blinkMin = false;
      blinkHour = false;
    }else
    {
      STATE= 1; // was STATE= 99;
      SUBSTATE = 0;
    }
  }
}

```

Figure 2: Main Code

This is the code compiler where we combined all of the codes into 1 to be compiled and uploaded to the device (clock) for it to run with the modified components.

```

// ***** Called by Timer 1 Interrupt to draw next column in LED matrix *****
// ***** Only light one ROW (and one column) ie one pixel at a time. = lower current draw, but lower refresh rate. *****

void LEDUpdateTWOC() // ONE ROW of selected column at a time
{
  if(ROWBITINDEX >6)
  {
    Mcolumn = Mcolumn+1; // Prep for next column
    if(Mcolumn >19)
    {
      Mcolumn =0;
    }

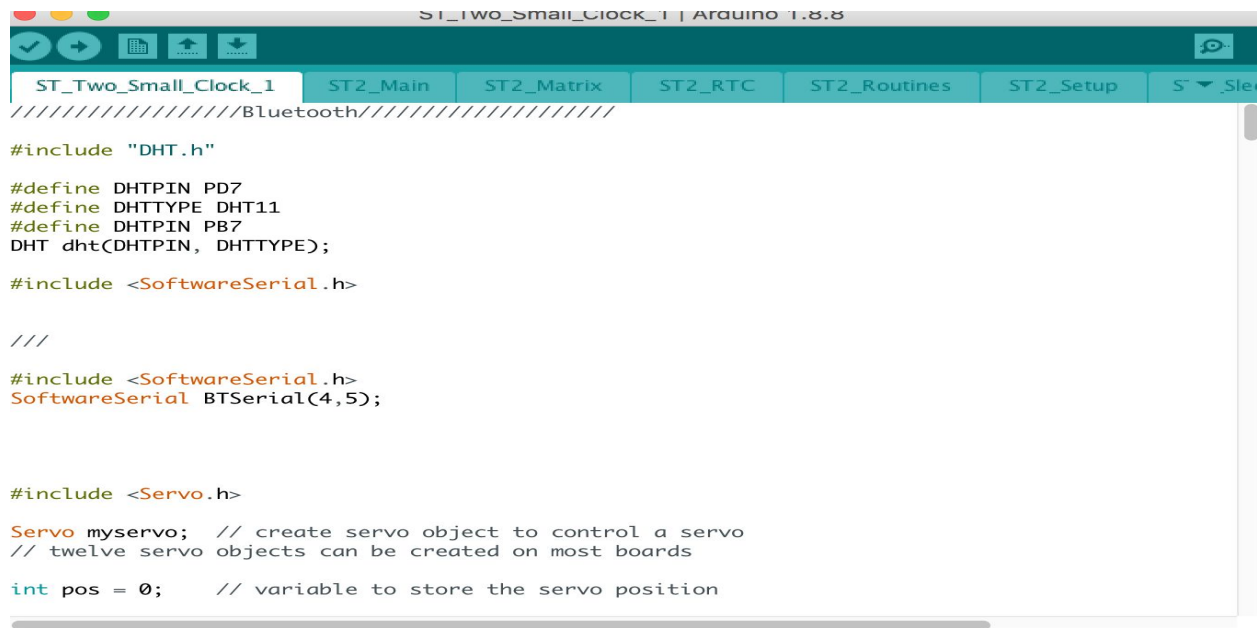
    PORTB = (PORTB & B10000000); // Clear last column
    PORTC = (PORTC & B11110000) | B00001111;

    if(Mcolumn <16) // Matrix column (from 0 to 19)
    {
      PORTB = (PORTB & B01111111); //! (0<<PORTB7); // Decode digit Col. 1 to 16 - Select De-Mux ch
      PORTD = (PORTD & B00001111) | (Mcolumn << 4); // Decode address to 74HC154
    }
    else
    {
      PORTB = (1<<PORTB7); // Decode digit Col. 17 to 20 - UN-Select De-M
    }
  }
}

```

Figure 3: Code for the LED matrix module

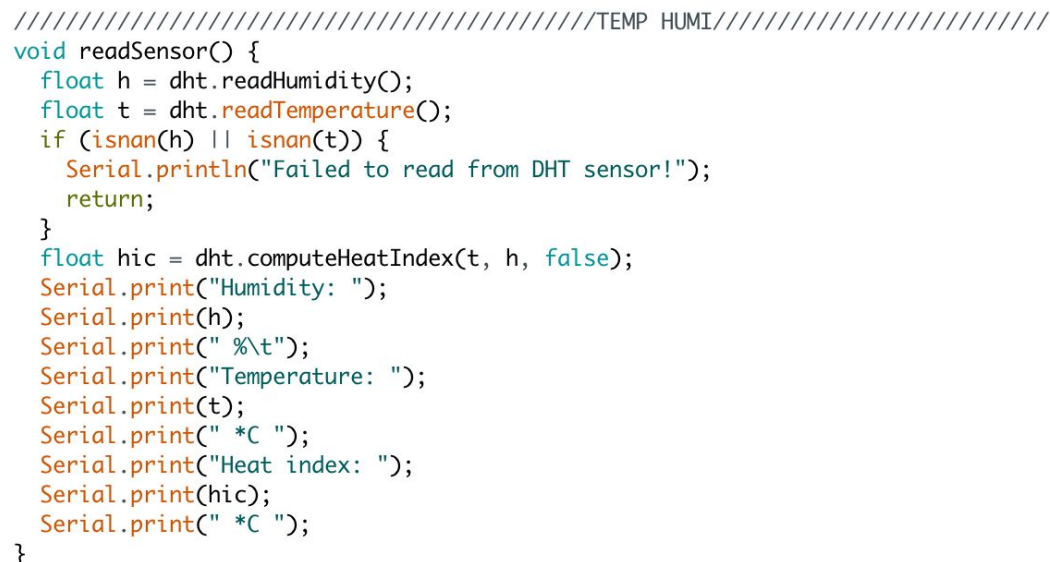
Code for the LED Matrix to display the required figures using matrices.

A screenshot of the Arduino IDE interface. The title bar shows 'ST_Two_Small_Clock_1 | Arduino 1.8.8'. The top menu bar includes 'ST_Two_Small_Clock_1', 'ST2_Main', 'ST2_Matrix', 'ST2_RTC', 'ST2_Routines', 'ST2_Setup', and 'ST2_Sleep'. The main code editor displays the following code:

```
//////////////////////////////////Bluetooth//////////////////////////////////  
  
#include "DHT.h"  
  
#define DHTPIN PD7  
#define DHTTYPE DHT11  
#define DHTPIN PB7  
DHT dht(DHTPIN, DHTTYPE);  
  
#include <SoftwareSerial.h>  
  
///  
  
#include <SoftwareSerial.h>  
SoftwareSerial BTSerial(4,5);  
  
  
#include <Servo.h>  
  
Servo myservo; // create servo object to control a servo  
// twelve servo objects can be created on most boards  
  
int pos = 0; // variable to store the servo position
```

Figure 4: Code for the Bluetooth Module

Here we have the basis code for the Bluetooth where we declare the bluetooth as well as the servo motor in one of our libraries to later on add to our main code. We define DHT-11 our temperature and humidity sensor.

A screenshot of the Arduino IDE showing a portion of the code. The title bar is partially visible. The code editor displays the following code:

```
//////////////////////////////////TEMP HUMI//////////////////////////////////  
void readSensor() {  
  float h = dht.readHumidity();  
  float t = dht.readTemperature();  
  if (isnan(h) || isnan(t)) {  
    Serial.println("Failed to read from DHT sensor!");  
    return;  
  }  
  float hic = dht.computeHeatIndex(t, h, false);  
  Serial.print("Humidity: ");  
  Serial.print(h);  
  Serial.print(" %\t");  
  Serial.print("Temperature: ");  
  Serial.print(t);  
  Serial.print(" *C ");  
  Serial.print("Heat index: ");  
  Serial.print(hic);  
  Serial.print(" *C ");  
}
```


Figure 5: Code for the DHT-11

5. Design

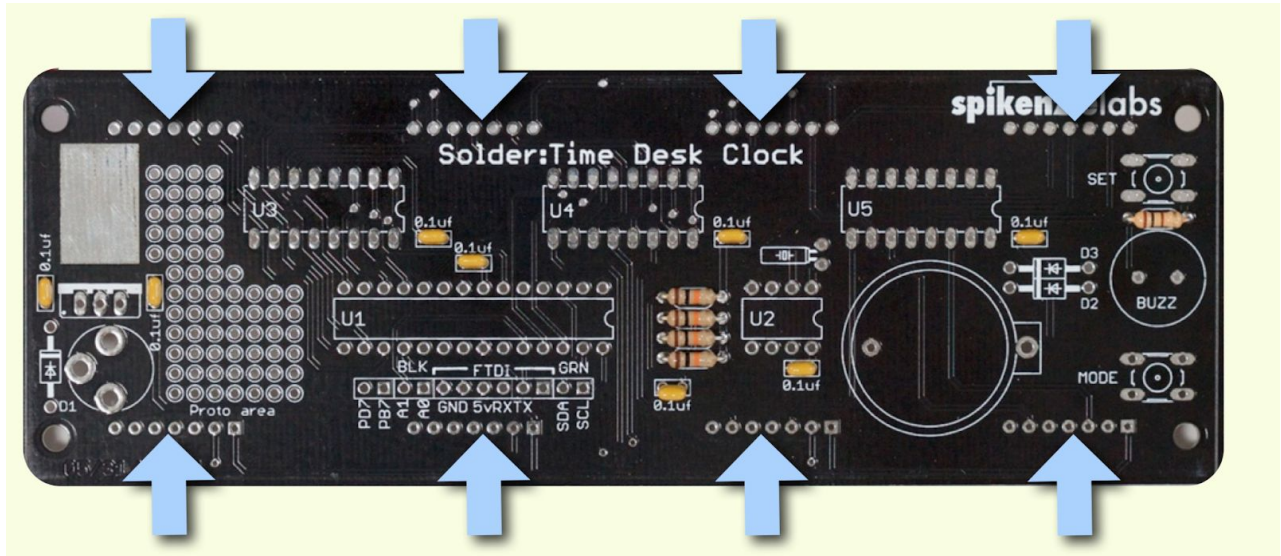


Figure 6: Design of PCB board with resistors and diodes



Figure 6.1: All components Mounted

5.1 Components

- Time Desk Clock Kit
- SG90 (Servo Motor)

-
- A** ATmega + socket
B Tail buttons
C Battery
D Crystal
E 0.1uF caps
F Buzzer holder
G Diodes
H Clock Chip
I Decoder
J DC jack
K Resistors
L LED matrix modules
M Voltage regulator
N Standard male pins
O Breadboard
P Buzzer
Q Tail buttons
- * Not to scale

5.2 Electrical Design

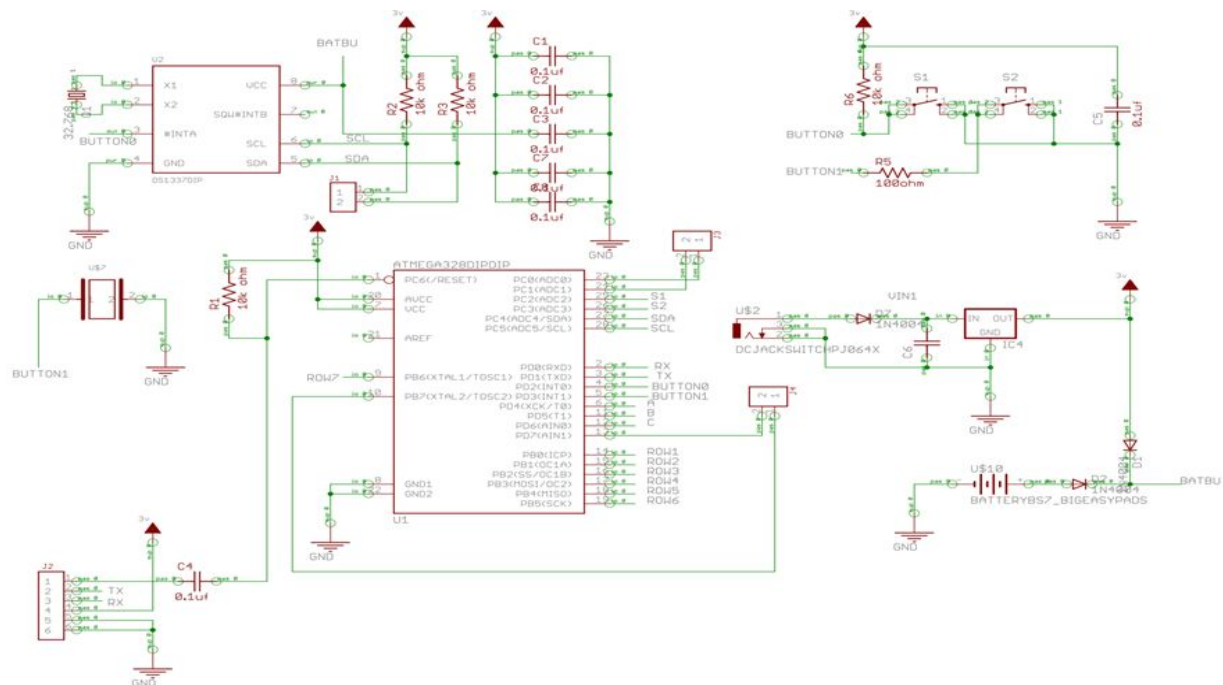


Figure 7: Schematic Diagram of the Circuit Board with Power Supply

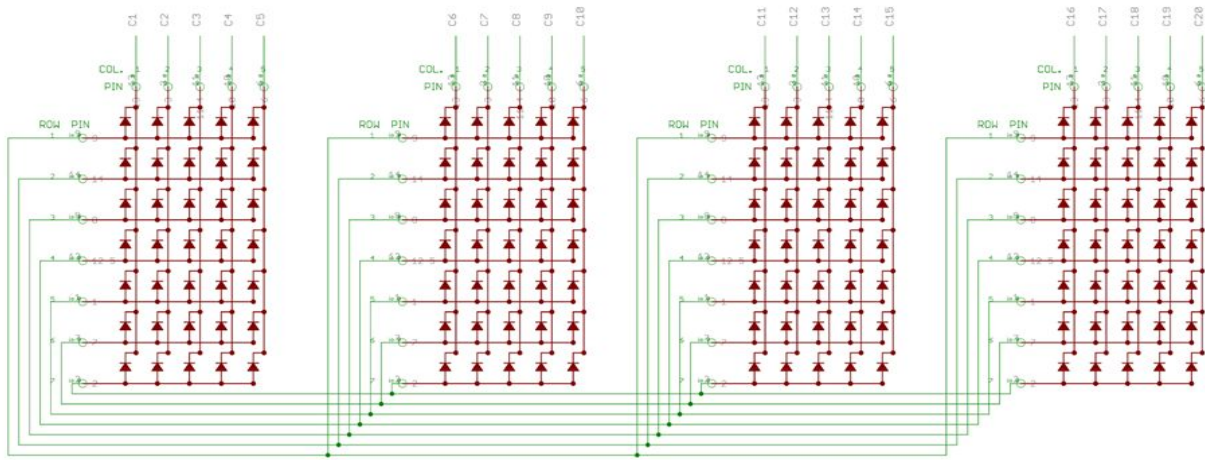


Figure 8: LED Matrix Schematic

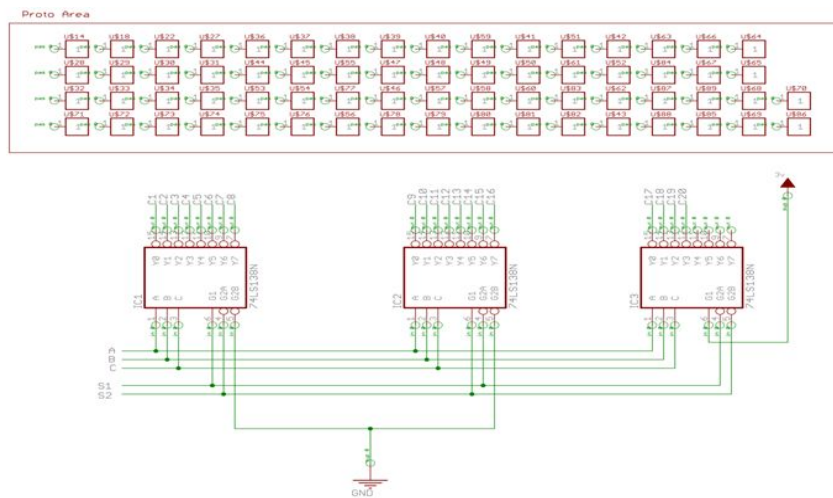
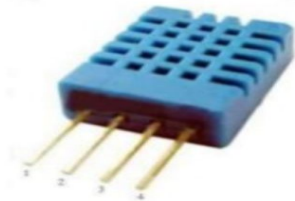


Figure 9: EPROM Decoder/Demultiplexer

6.1 Datasheets

DHT11 sensor and pin description



Features

Pin	Name	Description
1	VDD	Power supply 3 - 5.5 V DC
2	DATA	Serial data output
3	NC	Not connected
4	GND	Ground

Figure 10: DHT-11

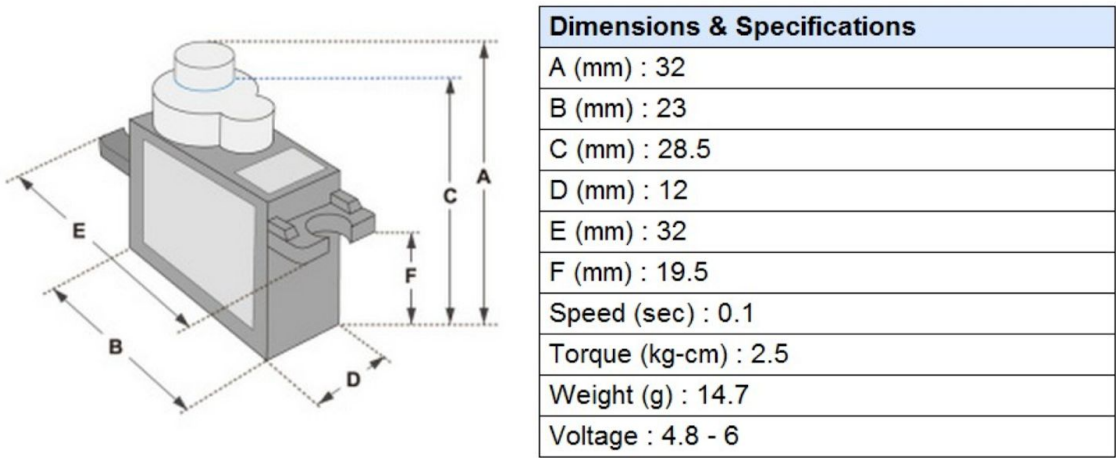


Figure 11: Servo Motor

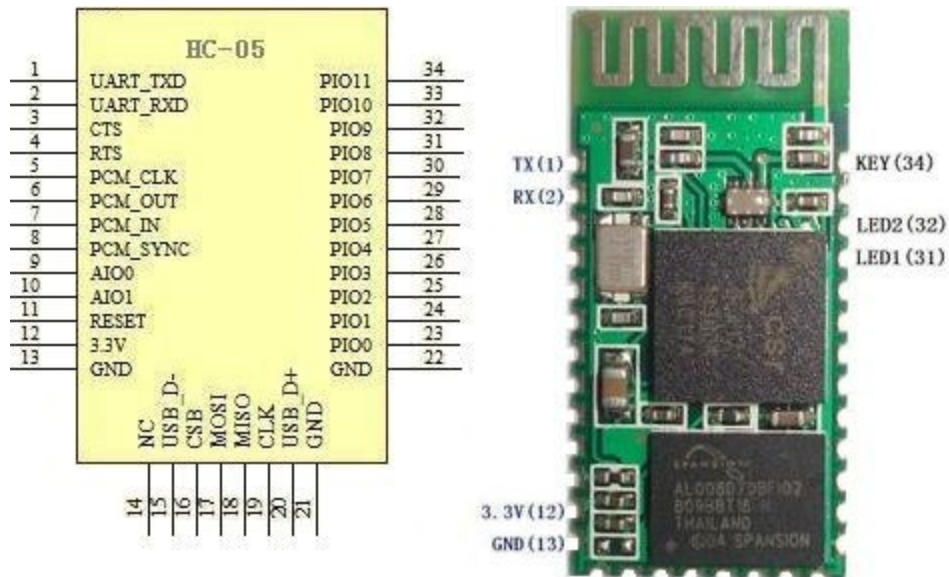


Figure 12: HC-05

7. Project Management

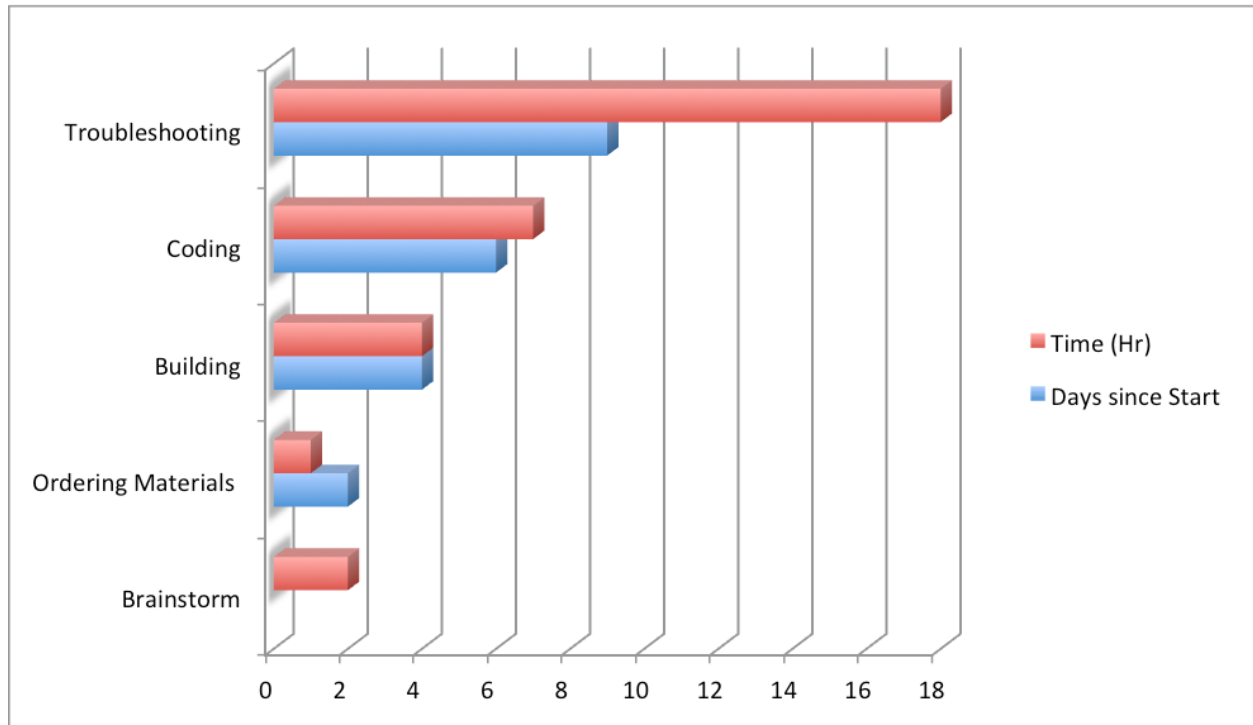


Figure 13: Progress Report

6. Conclusion

The overall result of the project ended being semi satisfactory the device did not display the full range of data that we expected. We could not apply all the components together the code wouldn't compiled with the vast amount of libraries in the code. To get it to work with additional parts made a mess of the whole program which we couldn't troubleshoot to identify the reason the code wouldn't compile. The most logical reason is the way C++ works in which the code must be in order but even when the code was arranged properly it wouldn't compile and work with the device. There was also the issue with the microcontroller which could be another reason the code didn't compile even though it was fixed. Our hopes are to find a way to go around these issues such as the PCB limitations which doesn't have enough space for other

components to be added even if we add other I/O ports it would still cause issues with the other components since it would collide together.

Reference

Solder:Time Desk Clock LTE Product website

https://www.spikenzielabs.com/Catalog/index.php?main_page=product_info&cPath=44&products_id=1275&zenid=ebc2a0790ccff985d0065291fa4503a8

From Arduino to a Microcontroller on a Breadboard

<https://www.arduino.cc/en/Tutorial/ArduinoToBreadboard>

Health Kit: Humidity and Temperature Control include bluetooth and app

https://create.arduino.cc/projecthub/dianakhalipina/health-kit-humidity-and-temperature-control-7d9297?ref=tag&ref_id=bluetooth&offset=7

Temp humidity tutorial

<https://www.brainy-bits.com/dht11-tutorial/>

Arduino I/O Expander

<https://www.youtube.com/watch?v=IwL275gjw04>

Code library

<https://drive.google.com/open?id=17MZ8mizuvJFBcdPJQoDhy5v4EaXr8mZh>