

JnJ's Clockwork

Software and Hardware Specification Sheet

Juan Sebastian Rodriguez, Jordan Pulido, Johnson Dinh

January 24, 2019

Declaration of Joint Authorship

We, Juan Rodriguez, Jordan Pulido, and Johnson Dinh, confirm that this work submitted for the project is the joint work of ourselves, and does not infringe upon anyone's copyright nor violate any proprietary rights. Any uses made with other authors' ideas, equations, figures, techniques, or any other material from the work of other people included in the project are in accordance with the standard referencing practices. Johnson Dinh has handled the database connection along with the app integration for the hardware, Juan Sebastian Rodriguez created the foundation of the app, along with its features and functionality. Juan also designed the overall structure of the hardware's appearance. Jordan Dave Pulido was responsible for the assembly and functionality of the Amplifier component of the hardware.

Proposal

January 17th 2019

Proposal for the development of JnJ's Clockwork

Prepared by Johnson Dinh, Juan Rodriguez, Jordan Pulido

Computer Engineering Technology Student

<https://github.com/JuanRodriguez19/JnJ-s-Clockwork>

Executive Summary

As a student in the Computer Engineering Technology program, I will be integrating the knowledge and skills I have learned from our program into this Internet of Things themed capstone project. This proposal requests the approval to build the hardware portion that will connect to a database as well as to a mobile device application. The internet connected hardware will include a custom PCB with the following sensors and actuators: HTU21D-F Humidity/Temp Sensor (0x40), I2S 3W Class D Amplifier Breakout MAX98357, 0.56" 4-Digit 7-Segment display. The database will store: Username, Password, Timestamp, Temperature Reading, Alarms Saved By Users. The mobile device functionality will include: Alarm Clock, Time Zones, Timers, Stopwatch, Temperature readings, User Information, Customization Features and will be further detailed in the mobile application proposal. I will be collaborating with the following company/department: Humber Prototype Lab. In the winter semester I plan to form a group with the following students, who are also building similar hardware this term and working on the mobile application with me. These are the following group members: Juan Rodriguez, Johnson Dinh, Jordan Pulido. The hardware will be completed in CENG 317 Hardware Production Techniques independently and the application will be completed in CENG

319 Software Project. These will be integrated together in the subsequent term in CENG 355 Computer Systems Project as a member of a 2 or 3 student group.

Background

The problem solved by this project is: As a youth, it becomes increasingly difficult to manage and maintain a proper sleeping schedule. The snooze button is used to give the user 5 more minutes to relax and properly wake up, however this is oftenly abused and the user ends up repeatedly hitting the snooze button, which often leads to time wasting. A bit of background about this topic is: This project will consist of an alarm clock application which will link up to a physical hardware element via bluetooth. The hardware being developed would contain a display where the current time, alarm settings, and local temperature readings would appear. The app is where the user would be able to customize and select what they want to appear on the display. Each sensor in the hardware portion of the project serve their own purpose in conjunction with one another. The Humidity/Temp sensor would give the current readings of the temperature and store them in the database, the Haptic sensor will vibrate the device with an alarm goes off as a time of notification. The display screen will be responsible for displaying the core information requested by the user.

Existing products on the market include Google Home. I have searched for prior art via Humber's IEEE subscription selecting Institute of Electrical and Electronics Engineers and have found and read A DIY approach to pervasive computing for the Internet of Things: a smart alarm clock which provides insight into similar efforts.

In the Computer Engineering Technology program we have learned about the following topics from the respective relevant courses:

- Java Docs from CENG 212 Programming Techniques In Java,

- Construction of circuits from CENG 215 Digital And Interfacing Systems,
- Rapid application development and Gantt charts from CENG 216 Intro to Software Engineering,
- Micro computing from CENG 252 Embedded Systems,
- SQL from CENG 254 Database With Java,
- Web access of databases from CENG 256 Internet Scripting; and,
- Wireless protocols such as 802.11 from TECH152 Telecom Networks.

This knowledge and skill set will enable me to build the subsystems and integrate them together as my capstone project.

Methodology

This proposal is assigned in the first week of class and is due at the beginning of class in the second week of the fall semester. My coursework will focus on the first two of the 3 phases of this project:

Phase 1 Hardware build.

Phase 2 System integration.

Phase 3 Demonstration to future employers.

Phase 1 Hardware build

The hardware build will be completed in the fall term. It will fit within the CENG Project maximum dimensions of 12 13/16" x 6" x 2 7/8" (32.5cm x 15.25cm x 7.25cm) which represents the space below the tray in the parts kit. The highest AC voltage that will be used is 16Vrms from a wall adaptor from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will be 20 Watts.

Phase 2 System integration

The system integration will be completed in the fall term.

Phase 3 Demonstration to future employers

This project will showcase the knowledge and skills that I have learned to potential employers.

The brief description below provides rough effort and non-labour estimates respectively for each phase. A Gantt chart will be added by week 3 to provide more project schedule details and a more complete budget will be added by week 4. It is important to start tasks as soon as possible to be able to meet deadlines.

Display screen for hardware element required. Materials for creation of the device. Casing for sensors and Raspberry Pi. Additional connectors to link up sensors to one another.

Concluding remarks

This proposal presents a plan for providing an IoT solution. The hardware device is a convenient option for those that want to maintain a solid time schedule all while being able to view current temperature readings of the area around them without having to open up external applications. This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative IoT capstone project demonstrating my ability to learn how to support projects such as the initiative described by [3]. I request approval of this project.

References

- [1]Google Home. (2016). Retrieved from https://store.google.com/ca/product/google_home
- [2]Institute of Electrical and Electronics Engineers. (2015, August 28). IEEE Xplore Digital Library [Online]. Available: <https://ieeexplore.ieee.org/search/advsearch.jsp>
- [3]Scott, G. and Chin, J. (2013). A DIY approach to pervasive computing for the Internet of Things: A smart alarm clock - IEEE Conference Publication. [online] ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/6659445> [Accessed 15 Jan. 2019].

Abstract

The purpose of this project is to create a device which alarms the user more effectively for those wanting to be notified on time with ease. It will consist of an alarm clock application which will link up to a physical hardware element via bluetooth. This technical report will give a thorough analysis of the development process regarding every aspect of the project, ranging from app structure to hardware assembly. Explanations revolving around certain ideals and decisions will be given to provide readers insight on why certain features are present along with their purpose.

Test cases will be examined during development to acknowledge mistakes that were made along the road along with the solutions for said mistakes. Any hardships and difficulties will also be documented for prevention purposes in the future of similar circumstances. Hardware explanation and breakdowns can be located in the Requirement Specification portion of the report.

Table of Contents

Declaration of Joint Authorship	1
Proposal	2
Abstract	5
Table of Contents	6
Illustrations	7
Introduction	8
Project Description	9
Requirement Specifications	9
Hardware	9
Software	9
Database	9
Build Instructions	10
HTU21D-F Temperature/Humidity Sensor (0x40)	10
0.56" 4-Digit 7-Segment display	10
I2S 3W Class D Amplifier Breakout MAX98357	10
Introduction	10
Budget for Materials Required	11
Time Schedule	12
Assembly of Pi	13
Wiring	14
PCB Design Files	16
PCB Soldering	18
Power Up	19
Case Design	20
Assembly for Hardware	20
Installing CircuitPython	20
Code for Sensors	22

Create library object using our Bus I2C port	22
Database Design	26
Mobile Application	26
Reproduction of Project	27
Background	27
Problem	27
Solution	27
Conclusion	27
Recommendations	27
Bibliography	28

Illustrations

Introduction

JnJ's Clockwork is an android based alarm mobile application where users are able to set and customize alarms of their choice for daily use. The user will be able to create multiple alarm profiles along with being able to set a timer and utilize a stopwatch. When connected to its corresponding hardware component, the app furthers its capabilities by allowing users the ability to read local temperatures via the sensor included. The project is designed to give users ease of access to anything time related in a simple and clean formfactor.

Students can benefit greatly from this product for those that struggle waking up at certain times of day and require an external source for assistance such as the app. It can also be used to set reminders for certain time periods such as deadlines for assignments etc. The temperature readings are an added bonus as it readily displays the current temperature in real time without having the need to open other respective applications for similar purposes.

In order for the app to function as intended, it is linked up with a database known as Firebase. This allows user information such as their own personal accounts, alarm templates and temperature values to be stored in a cloud which is ultimately located in Firebase. A Unique approach we are taking for this project would be the exclusion of a snooze button on the hardware itself which must be manually turned off in the app when alarms are triggered. This would make it harder for users to simply ignore the alarm they set up initially with a simple button press and motivate them to wake up.

Project Description

Requirement Specifications

Hardware

The hardware portion of this project will be a joint effort between each member of the group as there are many responsibilities in order for everything to function as intended. In terms of the hardware design and enclosure, it will be handled by Juan. The functionality of each sensor will be tested and operational mainly by Jordan with help from Johnson when required. Connections between sensors and the Raspberry Pi will be accomplished by Johnson. The Integration of components may require additional help from every member due to problems that may occur during development.

The project utilizes many hardware components such as the Raspberry Pi 3B+, HTU21D-F Humidity/Temp Sensor (0x40), and a Display screen. These sensors will be inclosed using 3D printed materials with a maximum dimension size of: 12 13/16" x 6" x 2 7/8 ". The Humidity and Haptic sensor are ready for integration with one another as they were already completed last semester. The new inclusion for the project is the Display Screen as it is crucial to display the time and temperature from the application. The android smartphone's role will act as the device's remote as it can communicate with the clockwork through bluetooth. A 8GB micro SD card will be used as storage as it can store the installation of the Raspbian OS, and reading and writing values from the Clockwork. The PCB (printed circuit board) acts as the structure and support of the system for sensor connections.

Software

The android application will be developed and maintained by Juan and Johnson. Add ons and additional functionality will be incorporated with the help from Jordan. The app is mostly complete in its current state. The only things that are left to work on is bluetooth functionality and debugging. The app needs to respond to the hardware in order to display desired information from the application.

The project utilizes a smartphone capable of running Android API 21 or higher. An up to date version of Android Studio was used to build the mobile application. A Raspberry Pi 3 was implemented with connection between the hardware and application. Updating the Raspbian OS to its newest version was used throughout the project. The mobile application will be used to work alongside the hardware components. Firebase real-time readings is going to be used for communication such as storing user and temperature readings.

Database

The database will be designed, created and upheld by Juan and Johnson. The database connection is established and connected to the mobile application. Reading and writing from the sensor to the database are also required. The database utilizes user-authentication to allow maximum security and protection for the users information. In order to read and write temperature, the user must be registered using a username and password through authentication processing. Offline mode allows access to the app, without the need to register and login. Offline mode skips user-authentication, and moves the user to the actual app. In offline mode, there will be no form of communications to the database. Therefore the user is unable to read/write temperature to the database.

Build Instructions

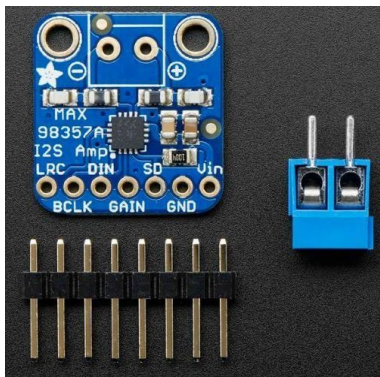
HTU21D-F Temperature/Humidity Sensor (0x40)



0.56" 4-Digit 7-Segment display



I2S 3W Class D Amplifier Breakout MAX98357



Introduction

JnJ's Clockwork is an android based alarm mobile application where users are able to set and customize alarms of their choice for daily use. The user will be able to create alarm profiles along with being able to set timers and stopwatches. When connected to its corresponding

hardware component, the app furthers its capabilities by allowing users the ability to read local temperatures via the sensor included. The project is designed to give users ease of access to anything time related in a simple and clean formfactor.

This project consists of the HTU21D-F Temperature/Humidity Sensor for reading local temperature values, a 1.2" 4-Digit 7-Segment display for displaying current time and temperature readings, and a I2S 3W Class D Amplifier Breakout MAX98357 for audio when an alarm goes off.

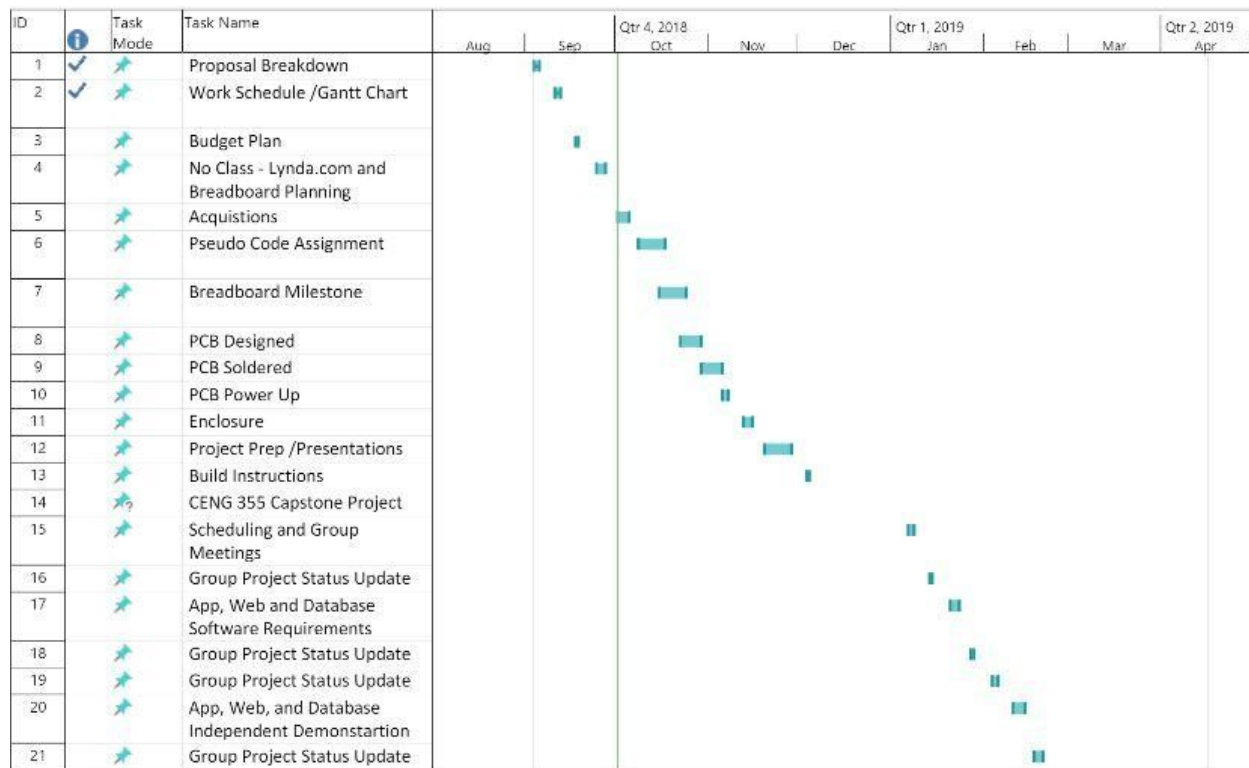
Budget for Materials Required

The required materials and budget for this project can be found in the documentation folder in the repository.

Budget For JnJ's Clockwork							
Description	Quantity	Source	Unit Cost	Shipping Cost	Subtotal	Status	Link
I2S 3W Class D Amplifier Breakout MAX98357	1	Adafruit.com	\$7.83 CAD	\$13.62 CAD	\$21.45 CAD	Acquired	https://www.adafruit.com/product/3006
HTU21D-F Temperature & Humidity Sensor	1	Adafruit.com	\$19.35 CAD	\$26.40 CAD	\$48.26 CAD	Acquired	https://www.adafruit.com/product/1899
Speaker 3" Diameter 8 Ohm 1 Watt	1	Adafruit.com	\$2.57 CAD	Included with Amplifier	\$2.57 CAD	Acquired	https://www.adafruit.com/product/1313
1.2" 4-Digit 7-Segment display	2	Adafruit.com	\$24.93 Per Unit	Included with Amplifier	\$24.93 CAD	Acquired	https://www.adafruit.com/product/1268
CanaKit Raspberry Pi 3 B+ (B Plus) Starter Kit (32 GB, Premium Black Case)	1	Amazon.ca	\$99.99 CAD	Free because of amazon prime, or else \$6.99 CAD	\$107.34 CAD	Acquired	https://amzn.to/2xKwXIH
Cooper Foil Tape with Conductive Adhesive 6mm x 15 meter roll	1	Adafruit.com	\$5.95 USD or \$7.95 CAD	\$20.39 USD or \$26.40 CAD	\$27.11 USD or \$35.10 CAD	Acquired	https://bit.ly/2xxYsVF
120pcs Multicolored Dupont Wire Kit 40pin Male to Female	1	Amazon.ca	\$10.99 CAD	Free because of amazon prime, or else \$6.99 CAD	\$17.98 CAD	Acquired	https://amzn.to/2O9jvr2
Electronic Parts Kit	1	Humber Store	\$120.00 CAD	Free	\$135.60 CAD	Acquired	Not available Online
GPIO Stacking Header for A+/B+/Pi2	5	Humber Prototype Lab	Free	Free	Free	Acquired	Not available Online
Total Cost for Aquired Items					\$393.23 CAD		
Created by Juan Rodriguez							
5-Mar-18							

Time Schedule

Realistically, this project should take around 3-4 weeks to complete if all materials and facilities are available to you. The materials themselves might take a week to arrive due to shipping, but the actual process of assembling and programming should not take longer than a week if proper time is given . A couple of hours each day can be dedicated towards the different aspects of the project to make time usage more efficient and effective. For me, this project took around 2 whole semester (8 months) to finish along with an average work time of around 2.5 hours a week. Here is the time schedule we followed:



5. Once installation is completed, you should be brought to the desktop. Connect yourself to either Wifi or wired connection in order to perform the next few steps.
6. Open the terminal in the top left corner of the screen and input the following lines (this takes quite a long time):
Shell wget
`https://raw.githubusercontent.com/six0four/StudentSenseHat/master/firmware/hshcribv01.sh \`
`-O /home/pi/hshcribv01.sh` `chmod u+x /home/pi/hshcribv01.sh`
`/home/pi/hshcribv01.sh`
7. Now it is time to set up a VNC connection so that you can access your Pi on any computer screen. From the Start Menu, go -> Preferences->Raspberry Pi Configuration->Interfaces, then set VNC to Enabled. Now on the desktop in the top right corner, you should see a VNC logo. When you click it you should see an IP address for your Pi which will be used to connect it via the VNC software. Download the software on any computer you wish to communicate with the Pi: <https://www.realvnc.com/en/connect/download/vnc/>
8. Once the software is installed, connect the ethernet cable from the Pi to your computer of choice to have a direct connection. Now you can simply input the same address you found in the Pi in the VNC software and it should connect.
9. To turn off the Pi, type `sudo power down` in the terminal.

If you are still unsure or struggling with a part in particular, this video provides a step by step explanation for everything required: https://www.youtube.com/watch?v=xBlxuf_LSCM

Wiring

Before wiring each sensor to the breadboard, it is important to solder the pins that come included to the sensors corresponding pin layouts. Additionally, make sure to cut the excess pins that come included that will not be required for each sensor.

When soldering, make sure you have safety glasses equipped along with having proper ventilation that contains a extractor arm for the fumes. A soldering toolkit is also required which is available in most labs. Here is a great soldering tutorial to help with those unsure.

<https://www.youtube.com/watch?v=3230nCz3XQA>

You can wire the sensors to the Raspberry Pi using the following charts:

HTU21D-F Temperature/Humidity Sensor (0x40).

Device Pin	Pi
1 (VIN)	[5.0v]
2 (3.3v)	[3.3v]
3 (GND)	[GND]
4 (SDA)	[GPIO 2]
5 (SCI)	[GPIO 3]

0.56" 4-Digit 7-Segment display.

The IO pin only needs to be connected if you are working with the 1.2" display which has the same pinouts as the 0.56" one.

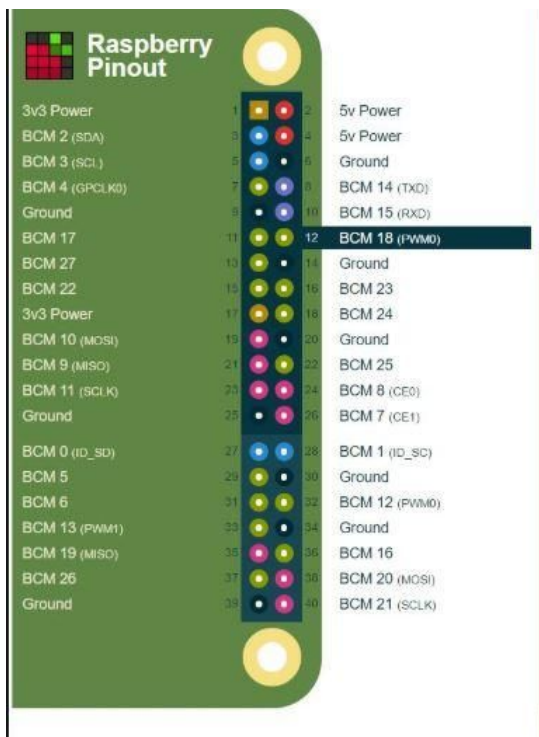
Device Pin	Pi
1 (D)	[SDA]
2 (C)	[SCL]

3 (+)	[5.0v]
4 (-)	[GND]
5 (IO)	[3.3v]

I2S 3W Class D Amplifier Breakout MAX98357.

Device Pin	Pi
1 (VIN)	[5.0v]
2 (GND)	[GND]
3 (DIN)	[GPIO21]
4 (BCLK)	[GPIO18]
5 (LRCLK)	[GPIO19]

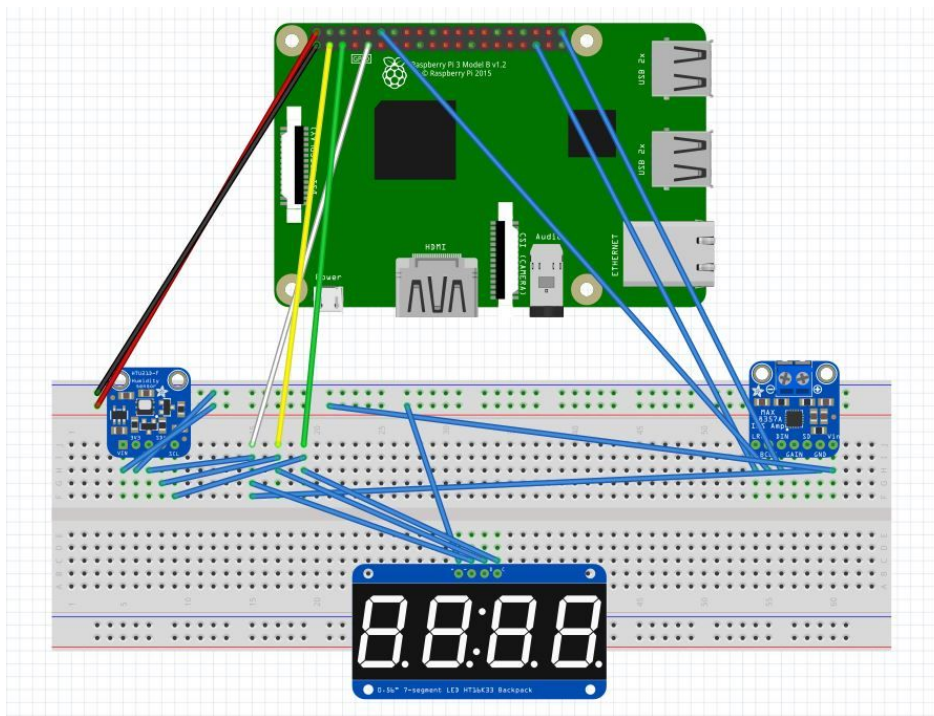
This is the layout for the pins of the Raspberry pi for guidance on where certain pins are located:



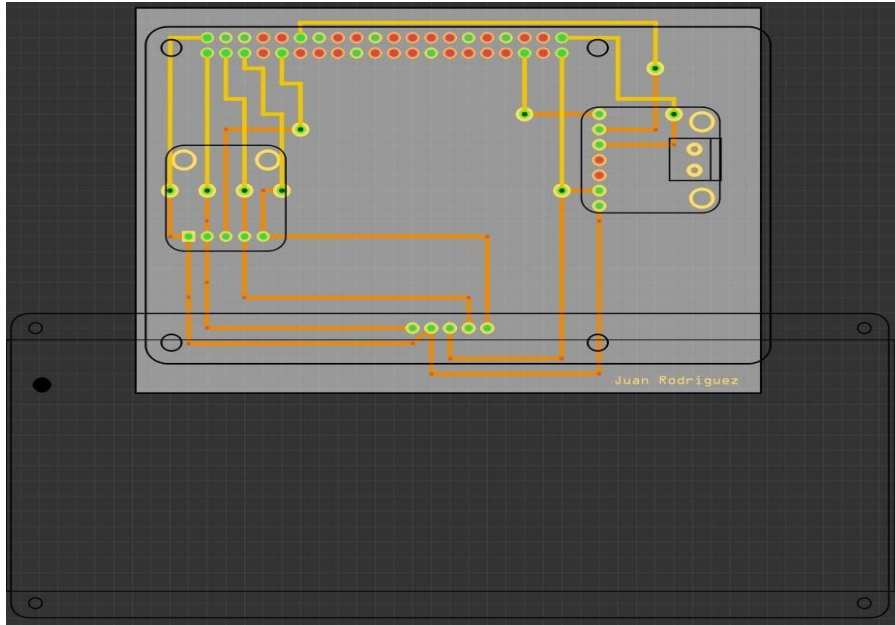
PCB Design Files

In order to develop the PCB design files, the application Fritzing is required along with the Corresponding sensors file which must be added to the application under MyParts. The files can be located here: <https://github.com/adafruit/Fritzing-Library/tree/master/parts>

Once the Sensor is added to parts, you can create a fritzing diagram for the wiring of the pi and sensor. It should look similar to this.



From here you can create the PCB design from the wiring you just designed. The PCB layout should look similar to this. Reminder, the IO pin is only connected to 3.3v for the 1.2" screen.

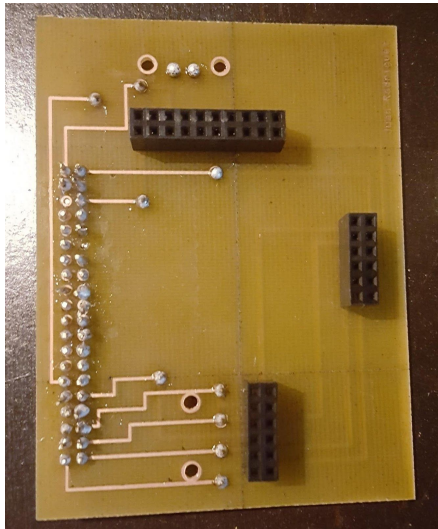


With these now ready, you can put together your Gerber files and create your PCB using a laser cutter machine. The Gerber files are located and can be downloaded [Here](#)

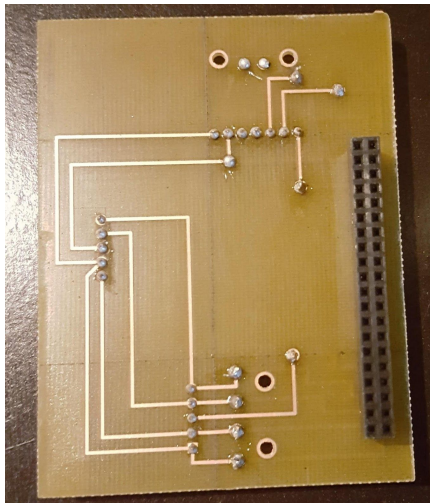
PCB Soldering

Using the same rules as when soldering the Sensors, solder pieces of wire in between the vias on the PCB board. Once that's done, solder the 20 pin socket to the PCB board to the corresponding holes for where the pi would connect. For the remaining pin sockets you have, solder in the respective headers for each sensor in the appropriate locations. Your final board should look similar to this.

Top view:



Bottom view:



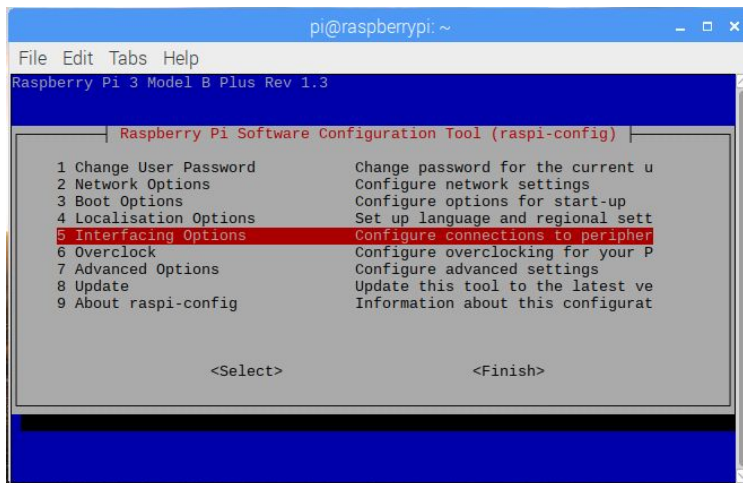
Power Up

In this section, we will now see if everything works, this works on whether you have soldered your PCB or you normally wired it onto your circuit board. Once connected boot up the Raspberry Pi, open the terminal window and follow these steps:

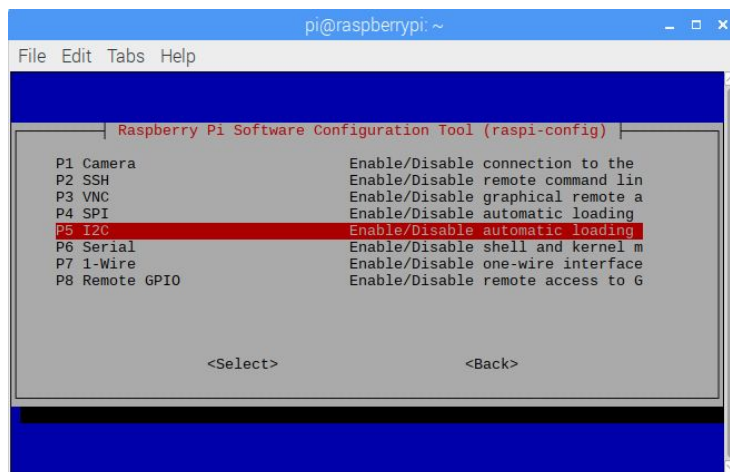
1. This will bring you into the configuration tool

`sudo raspi-config`

1. Use your arrows keys to go down and select "Interfacing Options"

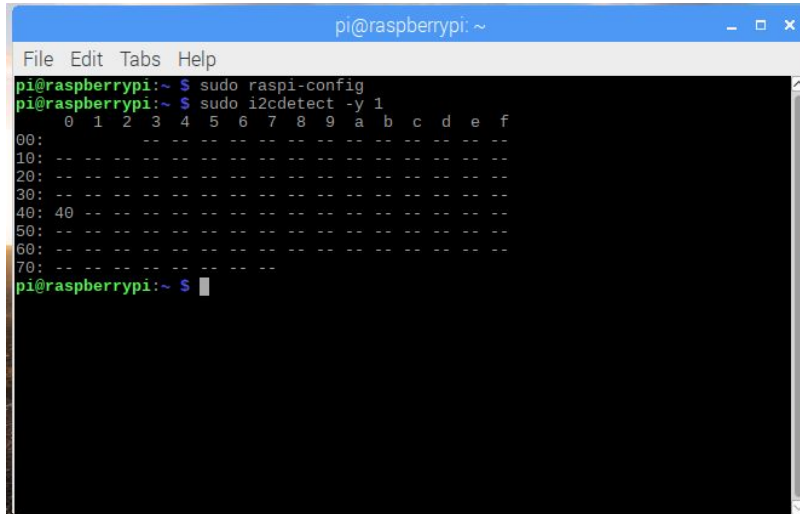


2. Select I2C and submit yes. It should display ARM I2C is enabled.



- Exit by selecting the finish option. By using the command below it should your address for the sensors being (0x40) and (0x70).

`sudo i2cdetect -y 1`



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$ sudo raspi-config  
pi@raspberrypi:~$ sudo i2cdetect -y 1  
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
40: 40  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
pi@raspberrypi:~$
```

Case Design

Still in development. The maximum dimensions the final product can be is around 13/16" x 6" x 2 7/8" = (32.5cm x 15.25cm x 7.25cm).

Assembly for Hardware

This can only be worked on once case is complete.

Installing CircuitPython

For this project a lot of code will be used with Circuit Python so here is the installation process for it.

The following tests were experimented using Adafruit:

1. Run the update commands for the Raspberry Pi.


```
sudo apt-get upgrade
```

2. When done installing, run the command line for the python tools

```
sudo pip3 install --upgrade setuptools
```

3. Verify you have I2C Enabled

```
ls /dev/i2c*
```

A terminal window screenshot showing the command 'ls /dev/i2c* /dev/spi*' being executed. The output is '/dev/i2c-1 /dev/spidev0.0 /dev/spidev0.1'. The command and the first part of the output are highlighted with a blue circle.

```
pi@devpi:~ $  
pi@devpi:~ $  
pi@devpi:~ $ ls /dev/i2c* /dev/spi*  
/dev/i2c-1 /dev/spidev0.0 /dev/spidev0.1  
pi@devpi:~ $
```

4. Begin to install the Python Libraries

```
pip3 install RPi.GPIO
```

5. Use the following command to install adafruit-blinka:

```
pip3 install adafruit-blinka
```

To test if Python works, open python in the Raspberry Pi (it should be installed at this point), and write an example file to sample output.

```
import board

import digitalio

import busio

print("Hello blinka!")


# Try to great a Digital input
pin = digitalio.DigitalInOut(board.D4)

print("Digital IO ok!")


# Try to create an I2C device
i2c = busio.I2C(board.SCL, board.SDA)

print("I2C ok!")


# Try to create an SPI device
spi = busio.SPI(board.SCLK, board.MOSI, board.MISO)

print("SPI ok!")


print("done!")
```

Save it, then run it on the command line by typing

```
python3 blinkatest.py
```

The following should be seen

```
(py) pi@devpi:~/py $ python blinkatest.py
Hello blinka!
Digital IO ok!
I2C ok!
SPI ok!
done!
(py) pi@devpi:~/py $
```

Code for Sensors

HTU21D-F Temperature/Humidity Sensor:

With the code provided in this repository, this test code should get your sensor to read and write temperature/humidity.

Firstly though assuming you installed circuit python, all that is required is to run the command line to install HTU21D libraries.

```
sudo pip3 install adafruit-circuitpython-htu21d
```

You can copy this code and run it.

```
```import time
import board
import busio
from adafruit_htu21d import HTU21D
```

### Create library object using our Bus I2C port

```
i2c = busio.I2C(board.SCL, board.SDA)
sensor = HTU21D(i2c)
```

```
while True:
 print("Temperature: %0.1f C" % sensor.temperature)
 print("Humidity: %0.1f %% " % sensor.relative_humidity)
 time.sleep(2)
```

Your output should look like this.

```
pi@raspberrypi:~ $ sudo python3 htu21d_simpletest.py
Temperature: 27.9 C
Humidity: 14.1 %

Temperature: 27.9 C
Humidity: 14.2 %

Temperature: 27.9 C
Humidity: 14.2 %
```

### **0.56" 4-Digit 7-Segment display:**

For this sensor we want to be able to display the current time. With that python and its libraries need to be installed:

```
sudo apt-get update sudo apt-get install build-essential python-dev
```

You would also need python smbus and python-imaging library:

```
sudo apt-get install python-smbus python-imaging
```

Clone the url onto your pi and move into it:

```
git clone https://github.com/adafruit/Adafruit_Python_LED_Backpack cd
Adafruit_Python_LED_Backpack
```

This is the last library you need to install:

```
sudo python setup.py install
```

Now go into your file named examples:

```
cd examples
```

There are a lot of test codes we can use here, but in case we just need our sensor to display the time:

```
sudo python ex_7segment_clock.py
```

### **I2S 3W Class D Amplifier Breakout MAX98357:**

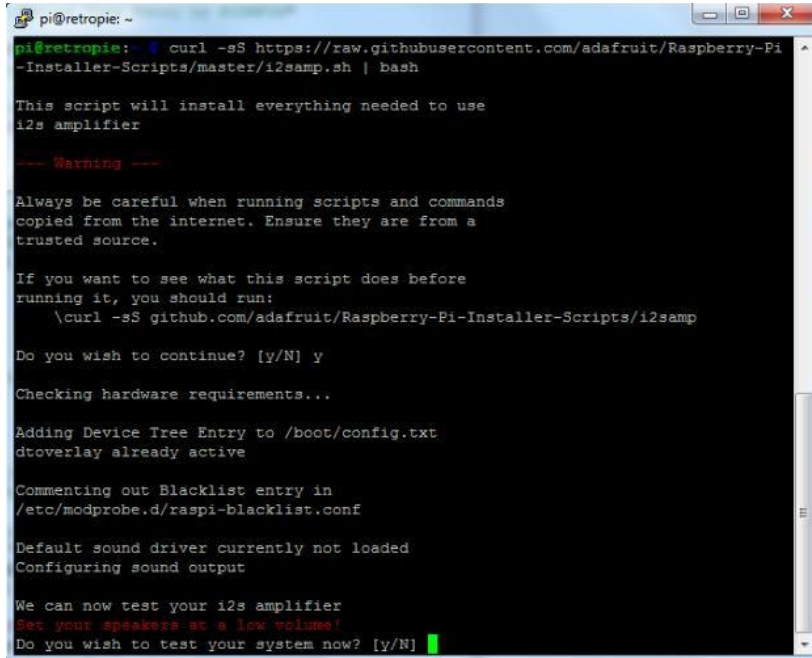
For testing the sensor, the following line was used with the help of internet connectivity within the Raspberry Pi.

```
curl -sS
```

```
https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/i2samp.sh |
```

```
bash
```

Select yes for the following questions the file asks by typing "y".



```
pi@retropie: ~
pi@retropie:~$ curl -sS https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/i2samp.sh | bash

This script will install everything needed to use
i2s amplifier

--- Warning ---

Always be careful when running scripts and commands
copied from the internet. Ensure they are from a
trusted source.

If you want to see what this script does before
running it, you should run:
 \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp

Do you wish to continue? [y/N] y

Checking hardware requirements...

Adding Device Tree Entry to /boot/config.txt
dtoverlay already active

Commenting out Blacklist entry in
/etc/modprobe.d/raspi-blacklist.conf

Default sound driver currently not loaded
Configuring sound output

We can now test your i2s amplifier
Set your speakers at a low volume!
Do you wish to test your system now? [y/N] █
```

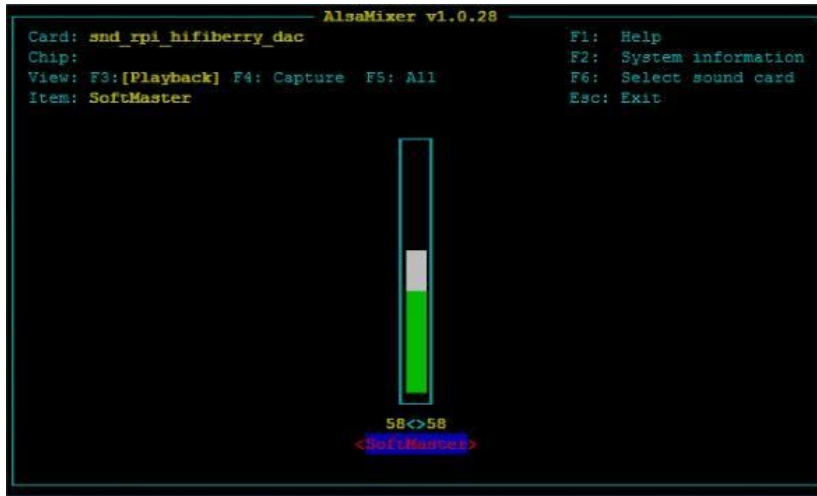
The file will ask to reboot. Type "y" again to reboot the Raspberry Pi. When the device is rebooted, type in the same command

`curl -sS`

`https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/i2samp.sh |`

`bash`

The script should recognize the device at this point and noise should be coming out of the speaker. To adjust the volume, Alsamixer was used within the terminal. The recommended volume is 50 within Alsamixer.



Press "ESC" to exit Alsamixer. To actually save the volume settings THE PI MUST BE REBOOTED TWICE, type in the following to reboot the system.

```
sudo reboot
```

Start up the Raspberry Pi, type in the following to generate a white noise coming out of the speaker

```
speaker-test -c2
```

If real sound wants to be heard, here is a demo

```
sudo apt-get install -y mpg123 mpg123 http://ice1.somafm.com/u80s-128-mp3 ``
```

Open another terminal window to access Alsamixer to adjust volume of the speaker.



```
pi@raspberrypi:~ $ speaker-test -c2
speaker-test 1.1.3
Playback device is default
Stream parameters are 48000Hz, S16_LE, 2 channels
Using 16 octaves of pink noise
Rate set to 48000Hz (requested 48000Hz)
Buffer size range from 2229 to 8916
Period size range from 1114 to 1115
Using max buffer size 8916
Periods = 4
was set period_size = 1114
was set buffer_size = 8916
 0 - Front Left
 1 - Front Right
Time per period = 5.822878
 0 - Front Left
 1 - Front Right
Time per period = 5.989974
 0 - Front Left
```

If errors occur, refer to Adafruit's manual setup by clicking [HERE](#)

## Database Design

The database connection is established and connected to the mobile application. Reading and writing from the sensor to the database are also required. The database utilizes user-authentication to allow maximum security and protection for the users information. In order to read and write temperature, the user must be registered using a username and password through authentication processing. Offline mode allows access to the app, without the need to register and login. Offline mode skips user-authentication, and moves the user to the actual app. In offline mode, there will be no form of communications to the database. Therefore the user is unable to read/write temperature to the database. The database connection works in conjunction with firebase.

## Mobile Application

The mobile application upon startup will provide the login screen. Through user authentication, the application will transfer to the main menu. We designed the user interface to be clean and

simple so that it would feel inviting for the user rather than being complex and intimidating. Having the 3 main features of the app (Clock, Alarm, Timer) on the homepage via tab fragments would make it quick and easy for the user to traverse without having to flip through various menus and options. The data that the app uses such as different time zones, alarms, and stopwatch count would be provided by user input so that they could choose whatever they desire. The mobile app also includes a section where it connects to the firebase where it has the temperature readings. In terms of hardware connectivity, the user will be able to set an alarm from the app and that information will then be sent to the database. The alarm data will then be sent to the hardware component, which will ultimately trigger the hardware to go off.

## **Test Cases**

### Authentication:

This is important as it gives the user access to the database. The first thing they will see is if they can login, for new users they would have to register. If the user attempts to login without registering, a message will appear noting that the account is not registered. The register page consists of an email and password. When the user inputs a incorrect email address such as @email.com, it would not work due to the authentication recognizing that this is not a proper email address. Therefore, the user would have to use either gmail, hotmail, yahoo, etc. Once everything is met, the account is now registered and saved onto the firebase authentication. They can now log in, if any mistakes are made a warning will appear prompting the user that there is a mistake. A successful login message will appear, if they meet all the necessary credentials and they will move on the to the clock screen.

### Firebase Read/Write:

In order to read and write to the database, the user first has to be logged in. If the user is in offline mode they will not be able to read or write to the database. The database is structured so that it will always use the UID as the primary key and then the temperature and timestamp under it. So if a new user saves a temperature, a new structure of their UID, temperature readings, and timestamp will be displayed on the database. This in turn can always have accurate reads of timestamp and temperature for different selected users. When the user creates an alarm profile, it will be saved to the database under their user id along with their temperature values. This then allows the amplifier to receive the database data and act accordingly, in this case having audio play with the alarm condition is met.

#### Clock Screen Test:

This fragment is shown right after someone logs in, or go in offline mode. You can access this page anytime by clicking the Tab that has a label “Clock” on it. To test the app there are textViews displaying the current time, and the time of the selected time zone. Select a time zone by pressing the spinner, it will show a list of all the available time zones in standard time. Once you pick a time zone, the textView for the selected time should change to the appropriate time. Changing to landscape mode changes the layout design to fit appropriately.

#### Alarm Screen Test:

You can get to this Fragment by clicking the second tab labeled “Alarm”. At first it should show the text “No Alarm Set” clicking the set alarm button will open up a TimePickerDialog that, on default, should be on the exact time you click the button, you can set the time by picking the hour then selecting the minutes. Or you can change to a different mode to change the alarm manually. After selecting the time, the text view should show “alarm set for \_\_\_\_” and will send a

notification once it reaches the time selected. To cancel the alarm, click the cancel button and the text should change to “alarm canceled”. Changing to landscape mode will retain an appropriate layout design. There is a button at the bottom of the page called hardware which sets an alarm that will communicate with the database and the hardware component, not solely the mobile phone. This will make so the audio that hits when the alarm goes off comes from the amplifier and speaker from the raspberry pi, not the phone.

### Stopwatch Screen Test:

The stopwatch can be accessed by clicking the third tab Fragment labeled Stopwatch. In this screen it displays a chronometer for minutes and seconds. If you press the start button it will start to count up. If you press the pause button it will pause the stopwatch at the time, pressing start again will resume the stopwatch. Reset button will reset the time displayed back to 0. landscape has an appropriate design layout.

## **Android Components**

Many android components and libraries are used during development of the app, each with their own specific purpose.

Major methods used for general overall use:

- **Intent (this, MainActivity.class)** - this method is often called when switching between screens (Often used with buttons or TextView links)
- **findViewById (int id)** - this method is used frequently to locate and interact with views found from layout resource files that are attached to the current activity

- **toastmakeTest(applicationContext, text, duration)** - this method is used often to send feedback to the user when they do a certain task like logging in.

Main methods for user authentication:

- **signInWithEmailAndPassword** - this method is called when the user gains access to their respective account once the proper email and password are entered
- **getCurrentUser()** - this method retrieves the data from the current user logged into the server, thereby accumulating further input they enter and save to the database

Main methods of Clock Fragment:

- **SimpleDateFormat** - allows you to set and use your own custom date format. Allows you to set the timezone format.
- **Calendar** - It's a abstract class with methods that allow you to set calendar fields such as Year, Month, Day of month, Hour.
- **TimeZone** - represents a timezone offset, you can use `getAvailableIDs` to set the spinner widget to display all the time zones.

Main methods of Alarm Fragment:

- **TimePicker** - A widget that allows you to pick a date and time 24h AM/PM, using `TimePickerDialog` that displays a dialog to the user to set the time.
- **AlarmManager** - This allows you to schedule your application to be run at some point in the future, using intents to broadcast it. It retains the application when the device is asleep.

- **NotificationManager** - Class to notify the user of events that happen. This is how you tell the user that an alarm has been triggered in the background. You can set vibration, lights, sound, descriptions and the icon. You can also put default intensity settings for vibration or sounds.

Main methods of Stopwatch Fragment:

- **Chronometer** - A class that implements a timer. You can set when the timer should start counting and also change it to a countdown timer.
- **SystemClock** - has access to all timekeeping facilities. The use of this method was to set the chronometer to 0 when you start it, so it started on the system clock time.

## Reproduction of Project

If you are to follow these steps exactly as explained in this guide, you should be able to reproduce this project without any difficulties considering you have the necessary equipment and tools.

Background

Problem

Solution

## Conclusion

In Conclusion, JnJ's Clockwork is a android based project designed to give users ease of access to anything time related in a simple and clean formfactor. The project consists of mainly layers in order to function as designed. It utilizes 3 distinct sensors being the 4-digit, 7 segment display, Humidity/Temperature sensor, and audio amplifier. All sensors communicate with our database in firebase which allow interactivity with the android app. Users have the ability to create alarm profiles, set different time zones, create stopwatches, and view/save local temperature readings all in one condensed formfactor.

## Recommendations

## Bibliography