WORCESTER POLYTECHNIC INSTITUTE

ECE 2799 - Habit Helper

Team 14

Homework 5 - Define Product Interfaces

April 23, 2019

Responsible Engineers

Fivos Kavassalis

Faith Kurtz

Catherine Roberts

Course Managers

William R. Michalson

Jennifer Stander

Table of Contents

Table of Figures

1. Introduction

This report will describe the changes that have been made to our device since the original design. Changes have been made to the battery module, vibrating motor circuit, physical case architecture, microcontroller programming approach, circuit components in bluetooth module, and pinout of the microcontroller. The effects of these changes on the project schedule will also be discussed.

2. Original Design

Based on the value analysis for our specific module design options, we originally planned on using the following components in our design:

- Case: Crescent shaped flexible case placed around the neck from behind
- Sensors (Posture and physical activity): Adafruit BNO055 Absolute-Orientation Sensor (IMU)
- User input (on/off): Switch
- User notification system: Vibrating mini motor disc
- Microcontroller: Arduino Nano Microcontroller
- Wireless Transceiver: HC-05 Wireless Bluetooth Serial Transceiver Module
- Power source: Two 3.7V 4800mAh rechargeable lithium battery cells connected in series (80 hour battery life-5 days at 16 hours per day)

We did not have specific components chosen for the switch or the vibrating mini motor disc. We also did not have any additional circuitry designed to drive the motor safely or account for integration of the different modules. The aspects of the design which have been changed or added for our final design are discussed in section 3.

3. Discussion of Errors that Required Modification

3.1 Battery

Our initial estimate of the battery life needed for our design was almost twice as much as what we are currently using. We calculated the number of mAhrs required incorrectly because we did not take duty cycle into account, and thus were calculating for every component being fully on the entire time. Once we took duty cycle into account and studied the data sheets more closely, we achieved a much lower estimate of about 2600mAhrs. This should easily keep the device running for 5 days at 16 hours a day. Our actual need could be even lower than this, but for this prototype we chose to leave a wide margin since we do not have time to implement low power modes or perform detailed analysis of power consumption during run time. A more detailed description of our calculations for the battery is described below.

For our device's power source, we initially decided to connect two 3.7V, 6000mAh Li-ion batteries in series. We based this number on our device's customer requirements (i.e. battery life of approximately 5 days, rechargeable) and our first estimation of the average current draw that the device would need to operate. However, this initial approximation was not very accurate as we either used the components' maximum current consumption from the datasheet for our calculations, or we hypothesized that they would consume the same amount of current continuously during the entire time the device was turned on, not taking each component's duty cycle into account.

More precisely, in our initial design we stated that our microcontroller (Arduino Nano) requires an input voltage of 7V to 12V and then uses its regulator to lower it to its operating voltage (5V). The maximum total current we can draw from an Arduino Nano when it is powered via an external power supply is 500mA. The IMU will be powered through the Nano's 3.3V pin and will require a maximum of 12.3mA. We plugged in a value close to that in our initial estimation (10mA). However, we saw that the IMU will draw about 2mA on average, and thereby we replaced the first value (12.3mA) with 2mA in our calculations for our revised design. The Bluetooth transceiver (HC-05) will be powered through the 5V pin and we said that it will require about 30mA, but after looking more closely into its datasheet, we realized that it will consume a maximum of 30-40 mA only while the module is attempting to pair with another device. Under normal circumstances (i.e. when Bluetooth module is connected to the user's phone), the Bluetooth module's current consumption falls to 8mA. Furthermore, in our first estimation we approximated that the Arduino Nano would consume about 20mA while running, but after taking a deeper look at the datasheet we substituted that value with 15mA on our second estimation. Finally, the current draw from our vibrating mini motor disc was not included in the circuit's average current consumption, since it is enabled for a relatively short period of time whenever it is required. The current draw of our system is estimated to be around 60mA (approximately 20mA for the Arduino Nano while running, around 10mA for the IMU and about 30mA for our Bluetooth transceiver).

After implementing a Value Analysis based on the initial estimation data, we decided to pick two 3.7V, 6000mAh Li-ion batteries (rechargeable) in series to power our system, not only due to the Nano's recommended input voltage (i.e. recommended voltage is 7-12V and by connecting two 3.7V batteries in series, we get 7.4V), but also on our estimation of average current consumption which we figured to be around 60mA (approximately 20mA for the Arduino Nano while running, around 10mA for the IMU and about 30mA for our Bluetooth transceiver). Then, according to equation (1) below, for 80 hours of battery life and 60mA of current draw, the resulting battery capacity is 6000mAh. The reason behind multiplying the division of battery capacity with current draw with 0.80 is so we can receive a more realistic number of the battery life provided to our project and not base our solution off the ideal battery life.

$$\frac{Battery\ Capacity\ (Amphours)}{Current\ Draw\ (Amps)} \cdot 0.8 = Battery\ Life\ (Hours) \quad (1)$$

Following our second estimation, the battery capacity decreased dramatically as the average current consumption was found to be significantly less than the initial estimation. On average, the current draw of our system is estimated to actually be around 25mA (approximately 15mA for the Arduino Nano while running, around 2mA for the IMU and about 8mA for our Bluetooth transceiver). Consequently, with 80 hours of battery life still as a requirement, equation (1) above gives us a battery capacity of 2500mAh. Therefore, with these revised calculations, we derived that the batteries that were the most applicable to our product are Lithium-Ion batteries with specifications of 3.7V, 2600mAh (rechargeable). We will be using two batteries in series, which will increase the voltage output to 7.4V in order to satisfy the recommended input voltage of the Nano (7-12V). These batteries will power each of the physical modules. With the exception of the vibrating motor module, current will be sourced using the microcontroller and will be distributed with a series of resistors, transistors, and diodes at each module.

3.2 Motor Circuit

The original design of our vibrating motor module was driven directly from the Arduino Nano and only included a 287Ω resistor to limit the current draw of the motor to 40mA. However, the maximum current a digital I/O pin can source in the Nano is 40mA. Therefore, by just having a resistor between the motor and the pin we may burn the corresponding digital I/O pin. Consequently, we realized that it would be better to drive the motor directly from the battery and control it from the Nano using a transistor circuit (low-side switch). The design we implemented for this is shown in figure x, where R3 (287Ω) is connected to a digital I/O pin on the Nano. Further details on the circuit are described below.

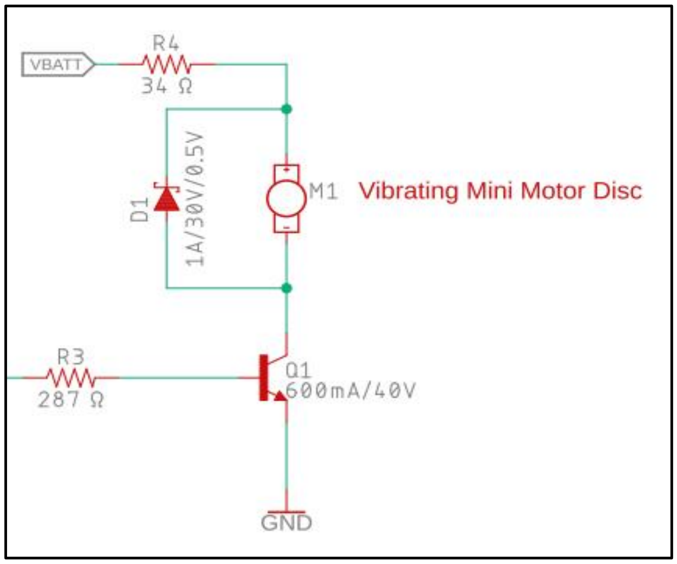

Figure 1: Vibrating Motor Circuit

We will control the vibrating motor through a digital I/O pin of the Arduino Nano. To avoid burning the digital I/O pin, we added some extra components to our configuration since the motor itself draws 100mA at 5V (minimum current draw is 40mA at 2V) while the maximum current an Arduino Nano digital I/O pin can source is 40mA. To remedy this, we used an NPN transistor and voltage divider. More specifically, we selected the P2N2222A amplifier transistor with a max reverse DC voltage of 30V, 1A of average current rectified and a forward voltage of 500mV, which fits the standards of our application based on our calculations below. The transistor will function as an electrical switch which will be controlled through the microcontroller digital I/O pin. A resistor must be added between the two to enforce current limitation. Furthermore, a flyback diode was placed across the motor to provide a safe path for the inductive kickback of the motor. Instead of powering the motor through the Arduino Nano, this circuit powers it directly from the battery, with the Nano turning it only on and off. To find the component values for this circuit, we calculated them using basic circuit analysis, then tested them in Multisim.

3.3 Case Architecture

The original design included a fairly stiff band connecting the back module of the case to the side modules. There was also high friction silicon on the back module in order to keep the device in place. However, the issue with this design was that the batteries on the left and right modules of the device were fairly heavy. This meant that leaning forward would cause them to hang forward instead of sitting flat against the user. In addition, they could pull the back module out of place. To fix this issue, we altered the design to replace the fairly stiff band with connectors between the back and side modules which are flexible but hold their shape. This design allows the device to fit comfortably on a wide range of users. In addition, rather than having the back module of the device sit at the very top of the spine, we lowered it slightly to allow the connectors to grip the users shoulders more securely. These changes are shown in figures 2 and 3 below.

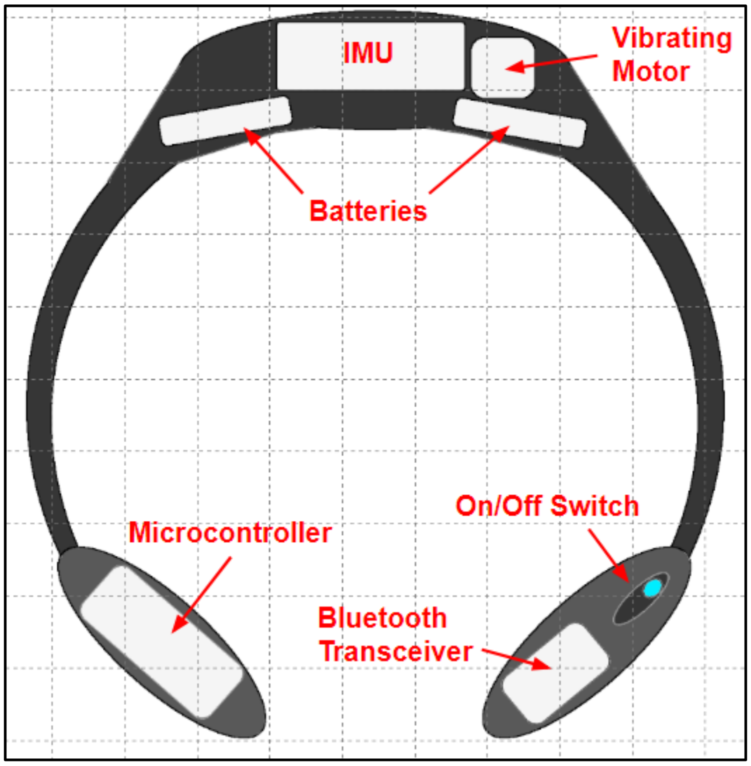

Figure 2: Previous Case Design

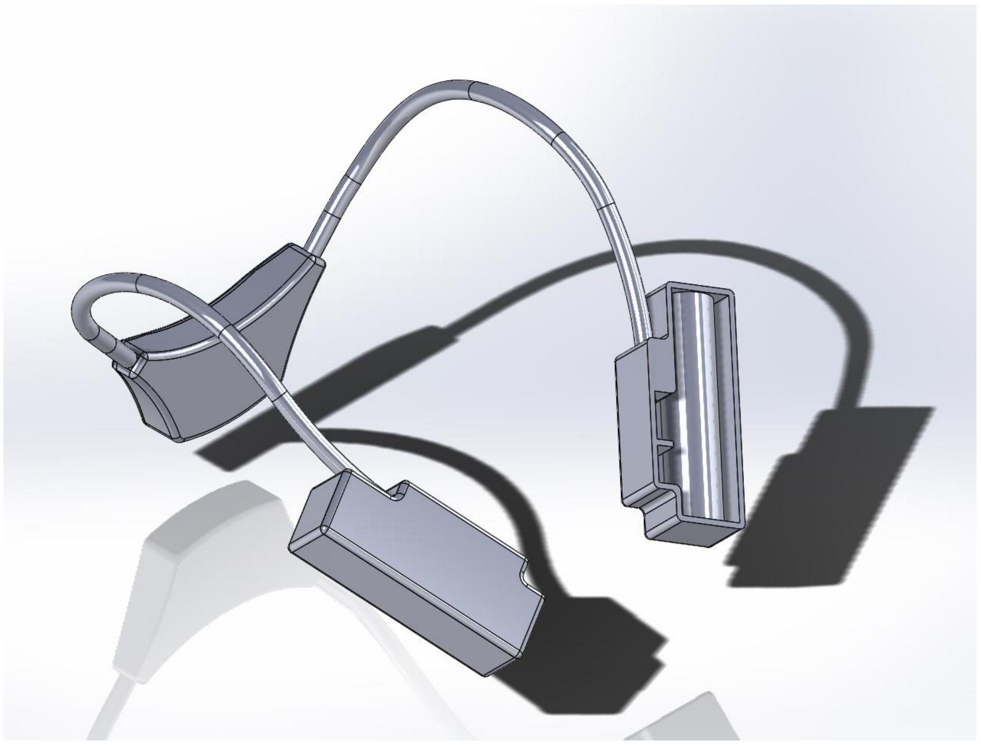

Figure 3: Current Case Design

3.4 Microcontroller Programming

We have made three major changes to our microcontroller programming approach since our original design. First, during the measurement process, we are no longer collecting data every second. Inσtead, we are spacing out our data collection. Second, we are no longer identifying the data using timestamps. We have implemented code which eliminates the need for them and saves storage space. Finally, when storing the collected data, we have decided against using a database for our prototype and are instead storing the data on the Arduino Nano and the user’s phone.

The current program for the measurement process only collects data during the first 10 seconds of each minute. We realized that taking measurements every second for an entire minute was redundant since a user will typically be slouching or inactive for long periods at a time. Therefore, the device takes measurements once per second for 10 seconds each minute. It then checks that the majority of these measurements suggest slouching/inactivity (to account for small variations). Finally, it assigns either slouching/not slouching and inactivity/activity to that minute.

After taking the measurements for a given minute, the data for this minute must be stored. Originally, we were going to store it using arrays and timestamps. However, in order to conserve the limited storage space of the Nano, we have implemented an alternative method of storage. The data is stored in two arrays, one for slouching data and one for inactivity data. Rather than storing each minute in an element of the array, minutes spent consecutively slouching or being inactive are stored together. In each array, the data is stored as blocks in an alternating pattern. For example, the first element in the slouching data array stores a number of consecutive minutes spent not slouching, the next element contains the number of consecutive minutes spent slouching, and so on. To access the data, the elements of the array are accessed from last to first. This means that the time can be counted backward from the time of access, eliminating the need for timestamps. Due to the long stretches of time between changes (correcting posture, etc.), this method will save a great amount of data storage space.

This data is stored directly on the Arduino Nano until the device is synced with the phone, then the data is moved to the phone and cleared from the device. We decided to use this storage method instead of a database for both ease of implementation and security reasons. Because of the limited time available to complete the prototype, setting up a database for the device is outside of the scope of the project. In addition, since we do not have time to research the security/privacy implications of storing a user's habits in a database, it is safer for the time being to store the data locally.

3.5 Bluetooth Voltage Divider Resistors

The original design for the bluetooth module included a voltage divider circuit at the RX pin of the HC-05 using 1kΩ and 2kΩ resistors. In our current design these have been changed to 10kΩ and 20kΩ resistors.

This voltage divider was implemented to ensure that the correct logic voltage level was being sent to the HC-05 from the Arduino Nano and vice versa. Since the logic level of the Nano is approximately 5V and the logic level of the HC-05 is approximately 3.3V, we found that the voltage divider required proportional resistor values of $R1 = ½*R2$. However, although the 1kΩ and 2kΩ resistors we originally chose would theoretically provide the correct voltage, the small resistance values meant that a lot of current would flow to ground and be wasted. To remedy this, we changed the values to 10kΩ and 20kΩ to reduce the current draw to ground.

3.6 Microcontroller Pinouts

Initially, the Arduino Nano pins that we used for communicating with the HC-05 Bluetooth module in our design where the serial 0 (RX) and 1 (TX) pins, which are used to receive and transmit TTL serial data. The Nano's "RX0" pin was connected to the receiving pin of the Bluetooth HC-05 through a voltage divider since the Nano transmits/receives 5V logic signals, whereas the HC-05 transmits/receives 3.3V logic signals (The Nano's pin is labeled as "RX" to signify that it is the receive pin for the ATmega 328, therefore it is the pin that should be connected to an "RX" pin of another module). Similarly, the Nano's "TX1" pin is connected to the HC-05's "TX" pin. In this bus, the HC-05 is the only one transmitting data, which means that the Arduino (5V logic) can accept signals from the Bluetooth module without the help of any additional circuitry (i.e. voltage divider). However, these pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip. Consequently, if we have those pins connected to our Bluetooth module we must always disconnect the Tx and Rx pins of the Bluetooth module from the Arduino because those are the same pins that the Arduino uses to burn the code to the chip. If we have the module connected to those pins, there will always be a pullup so the code will not be uploaded and an error will occur. Therefore, to make things more comfortable for ourselves while programming our device, we decided to just use two digital I/O pins which, along with the help of the SoftwareSerial library, have the ability of transmitting and receiving data. More specifically, digital I/O pins D4 (role of "RX0") and D3 (role of "TX1") are used to establish communication between the Nano and the HC-05. Finally, in our initial design the "D3" pin was "occupied" with the responsibility of controlling the vibrating mini motor disc. Thereby, in our revised design we use the "D5" pin to control the motor instead.

4. Revised Design

With the exception of the changes described in Section 2, the current design is the same as described in Homework 4. This section will briefly summarize our revised design.

The block diagram for our device is shown in figure 4 below.

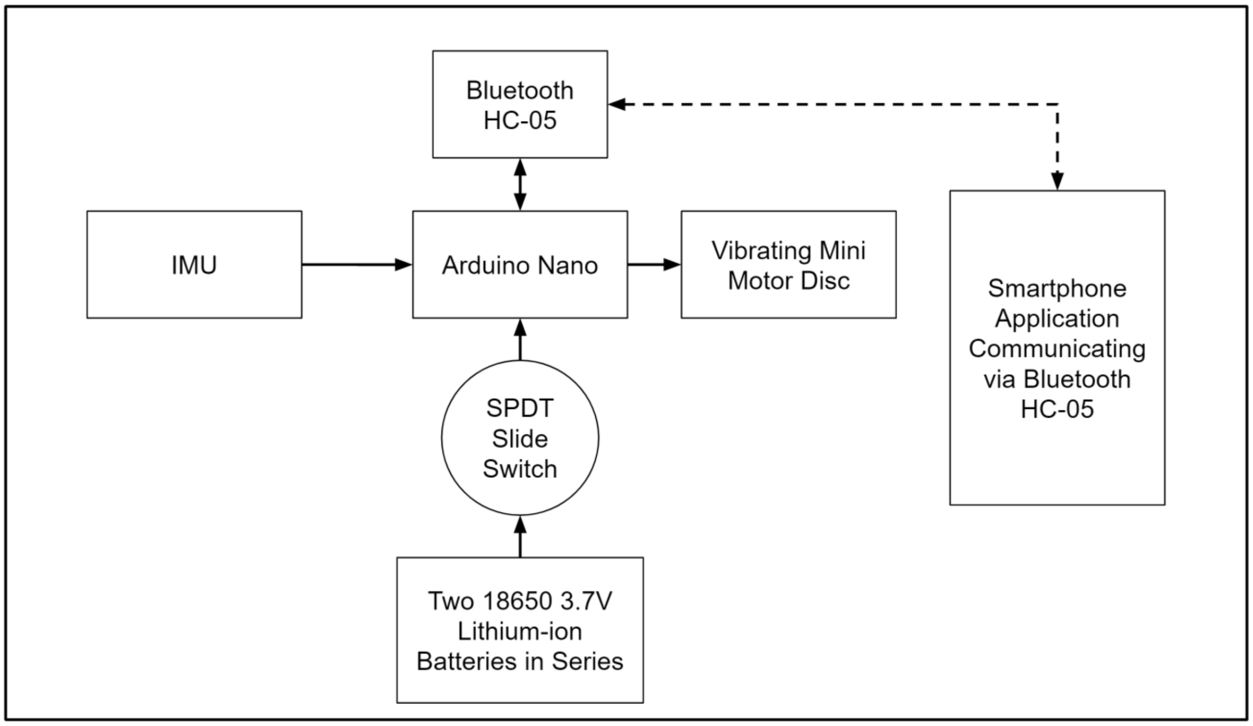
Bluetooth HC-05
IMU
Arduino Nano
Vibrating Mini Motor Disc
Smartphone Application Communicating via Bluetooth HC-05
SPDT Slide Switch
Two 18650 3.7V Lithium-ion Batteries in Series

Figure 4: Device Block Diagram

The final design shown in Figure 5 below illustrates the most recent version of the design with all technical aspects worked out and documented.

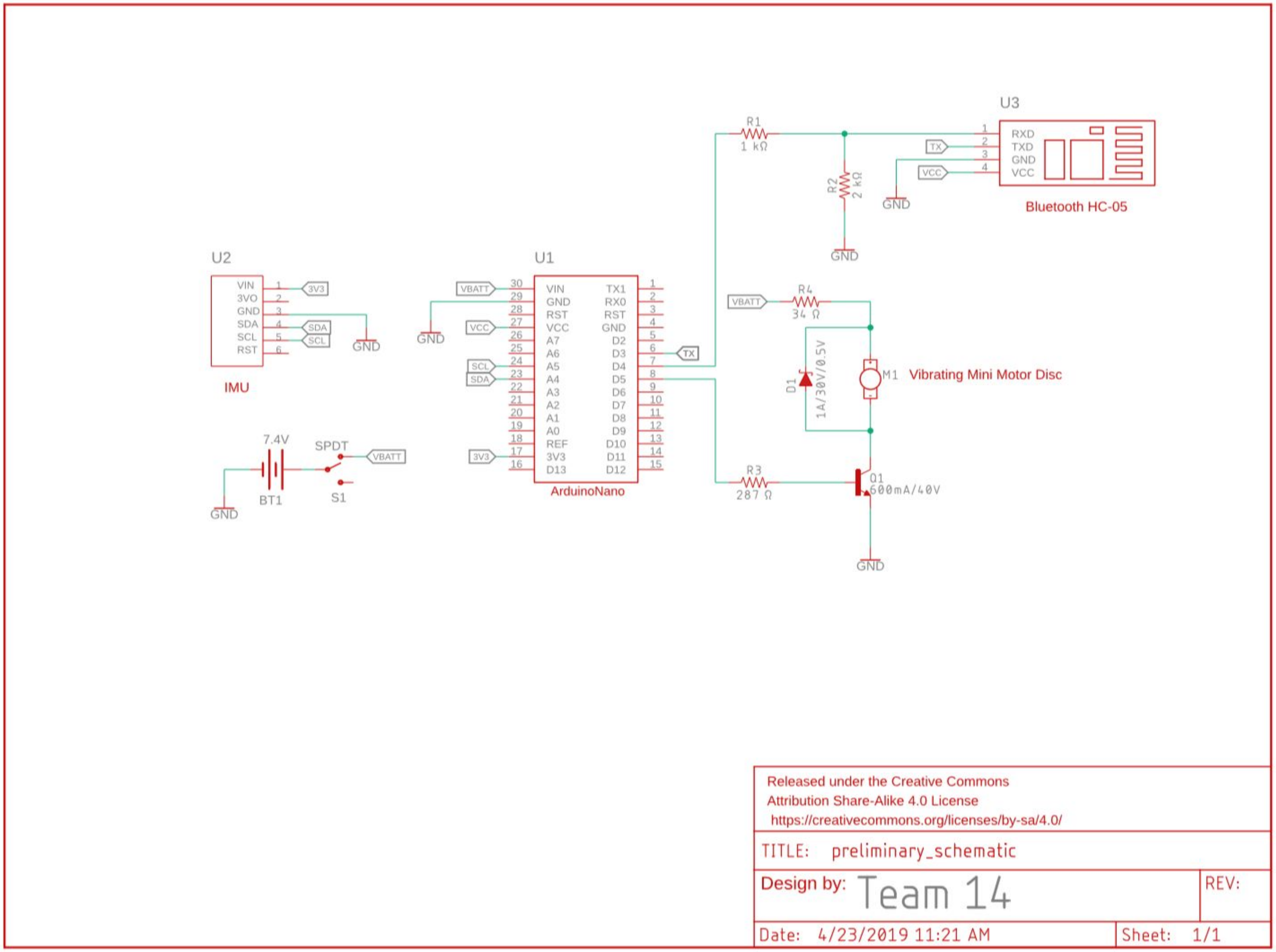

Figure 5: Schematic of our Revised Design

In our final schematic, the two Li-ion batteries (3.7V, 2600mAh) that are connected in series provide 7.4V and 2600mAh to the system. These batteries are then connected to an SPDT switch which enables the user to turn the device on or off. Excluding the common terminal which is connected to the batteries, one of the two is connected to the Arduino Nano and the vibrating motor circuit, whereas the other is grounded. The Arduino Nano, which is our central microcontroller, does all the processing required for our device to be functional, sources current to the IMU and the Bluetooth HC-05 and commands all the peripherals in our circuit to implement a specific, unique set of instructions. The Nano is connected with the IMU through the I2C communication protocol, with the microcontroller as the master and the IMU as a slave. Furthemore, our microcontroller is connected to the Bluetooth HC-05 through UART. Finally, the part of the circuit surrounding the vibrating motor is implemented so we can control the vibrating motor through a digital I/O pin of the Arduino Nano. The P2N2222A transistor is used as an electronic switch to turn the motor on or off. The resistor in between the transistor and the pin (287Ω) is used to limit the current sourced by the pin. Moreover, a flyback diode (Schottky) is connected across the motor since the latter is used as an inductor when the transistor shuts off. The purpose of the resistor that is connected to the positive terminal of the motor is to implement a voltage divider circuit along with the resistance of the motor in order to limit the voltage across the motor since the motor accepts voltages in the 2V-5V range and the battery provides 7.4V.

5. New Schedule

Despite some changes to our design, we are still on track for our overall schedule. We have extended some building and testing tasks by a few days, and the build milestone was moved out by 4 days. We also extended testing by a day. However, since we left some room in our original schedule for these changes, we should have no trouble meeting our deadlines. Our updated Gantt chart is shown in figure 6.

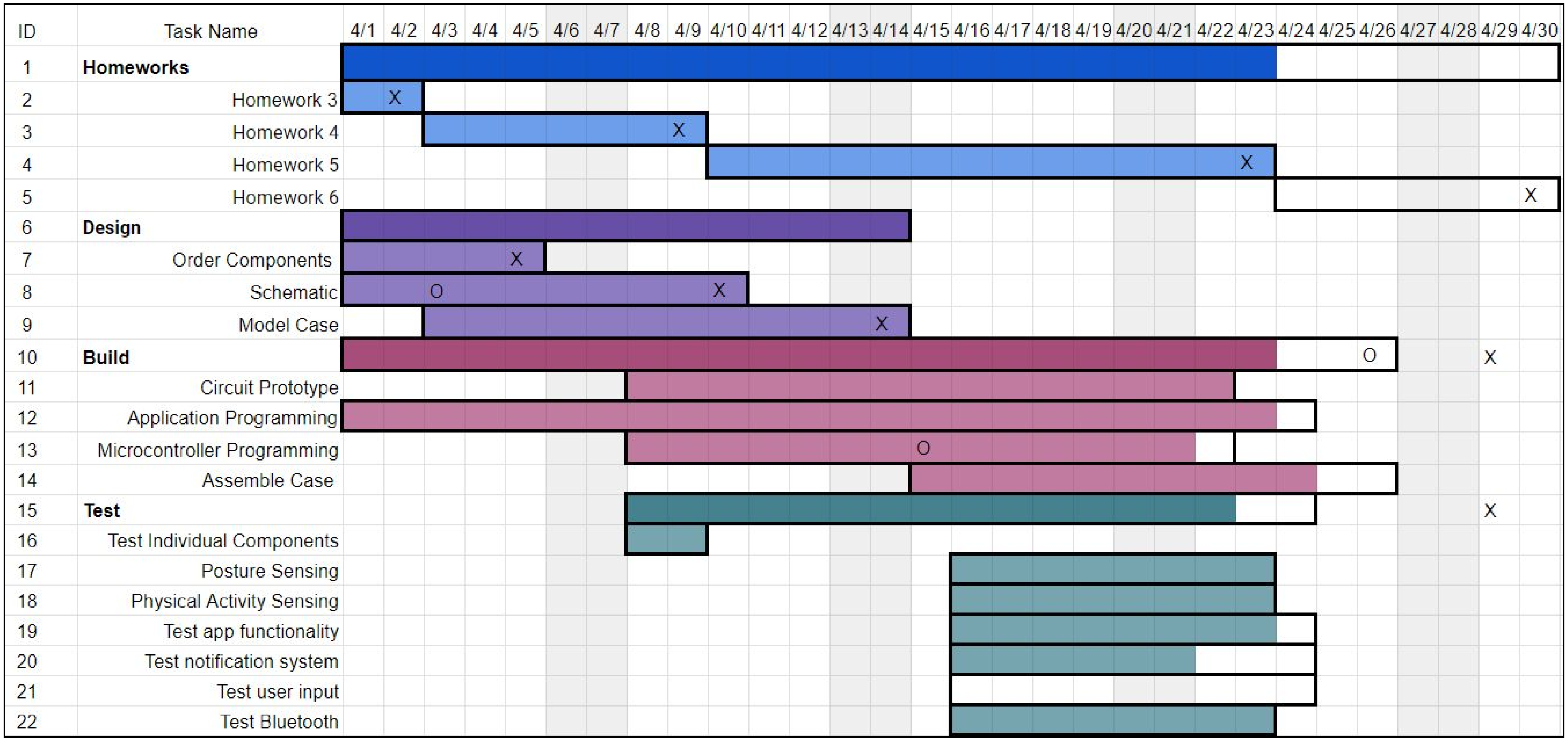

Figure 6: Updated Gantt Chart

6. Conclusion

Overall, the ultimate goal of the prototype has stayed constant throughout the revision process. From the very beginning we have maintained that our device will measure physical activity and posture which will help the user create better habits. The way in which we decided to execute this goal has shifted throughout the process as we were faced with challenges and inspired with better ideas. The final design is very similar to the very first design with only minor bug fixes. This can be attributed to the large amount of time we dedicated to thoroughly considering every detail of the preliminary design. There were some areas that were overlooked at the beginning, like the vibrating motor circuit, but were ultimately corrected with the help of our senior engineers. The final design presented in this report exemplifies almost an entire term’s worth of research and testing that culminated into a design bringing to life our original goals.

References

B. Recny, “Bluetooth UART - SPP vs. BLE implementations,” *Rigado*, Feb-2019. [Accessed: 26-Mar-2019]

BOSCH, “BNO055 Intelligent 9-axis absolute orientation sensor,” BNO055 datasheet.

D. Kliszowski, “Bluetooth Classic vs. Bluetooth Low Energy on Android - Hints & Implementation Steps,” *Droids or Roids* , 14-Aug-2018. [Accessed: 26-Mar-2019].

G. Staples, “Arduino Power, Current, and Voltage Limitations,” *Electric RC Aircraft Guy, LLC.* [Online]. [Accessed: 5-Apr-2019].

Sparkfun, “CR18650 2600mAH datasheet” CR18650 datasheet [18-Jul-2012].

ITead Studio, “HC-05 -Bluetooth to Serial Port Module,” HC-05 datasheet.

J. Blom, “Bluetooth Basics,” *SparkFun*. [Online]. [Accessed: 26-Mar-2019].

J. Dee, “How to Power a Project,” *SparkFun*. [Online]. [Accessed: 26-Mar-2019].

L. Ada, “All About Batteries,” *Adafruit*. [Online]. [Accessed: 26-Mar-2019].

L. Ada, “Vibrating Mini Motor Disc,” *Adafruit*. [Online]. [Accessed: 27-Mar-2019].

Microchip, “megaAVR datasheet,” ATmega48A/PA/88A/PA/168A/PA/328/P datasheet.

Mouser, “Low drop power Schottky rectifier ,” 1N5818 datasheet.

N/A, “NPN Transistor,” *Electronics Tutorials.* [Online]. [Accessed: 6-Apr-2019].

N/A, “SPST, SPDT, and DPDT Switches Demystified,” *Music from Outer Space.* [Online]. [Accessed: 6-Apr-2019].

N/A, “Toggle Slide Switch Breadboard & Perfboard Compatible: SPDT,” *Tinkersphere.* [Online]. [Accessed: 28-Mar-2019].

Nate, “Battery Technologies,” *SparkFun*. [Online]. [Accessed: 26-Mar-2019].

O. Lathrop, “Purpose of the diode and capacitor in this motor circuit,” *Electrical Engineering Stack Exchange*. [Online]. [Accessed: 7-April-2019].

ON Semiconductor, “Amplifier Transistors NPN Silicon,” P2N2222A datasheet.