WORCESTER POLYTECHNIC INSTITUTE

ECE 2799 - Habit Helper

Team 14

Homework 4 - Define Product Interfaces

April 9, 2019

Responsible Engineers

Fivos Kavassalis

Faith Kurtz

Catherine Roberts

Course Managers

William R. Michalson

Jennifer Stander

Table of Contents

Table of Figures

1. Introduction

In this report we will be describing the structure and implementation strategy for our prototype. Using information gathered from datasheets and our prior electronic knowledge we have created a thorough description of how the device will be assembled and operated. The following sections will give a clear picture of what we imagine the finished product will look like and the processes we believe will make it work. The components for this project were chosen after careful consideration of all the options. Such justifications are available in the proceeding document and in the second homework assignment. Additionally, in this report, technical details will be revealed as to how we plan to use our carefully chosen parts to their fullest potential. Overall, the conclusions made below are based on extensive research and are our best estimates as to how the overall system will behave.

2. Architectural Description

In this section, we will provide an architectural description of the device. This will include block diagrams of the device as well as detailed description of each module. The basic and specific top-level block diagrams for the device are shown in figures 1 and 2 below.

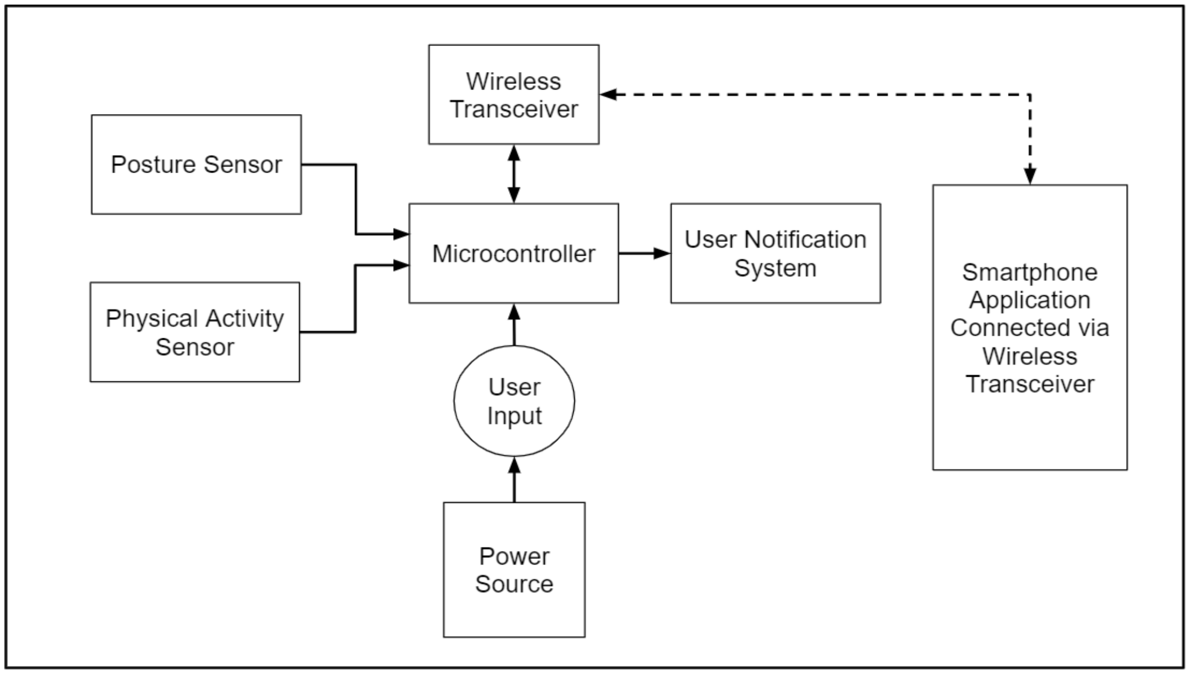


Figure 1: Top-Level Block Diagram

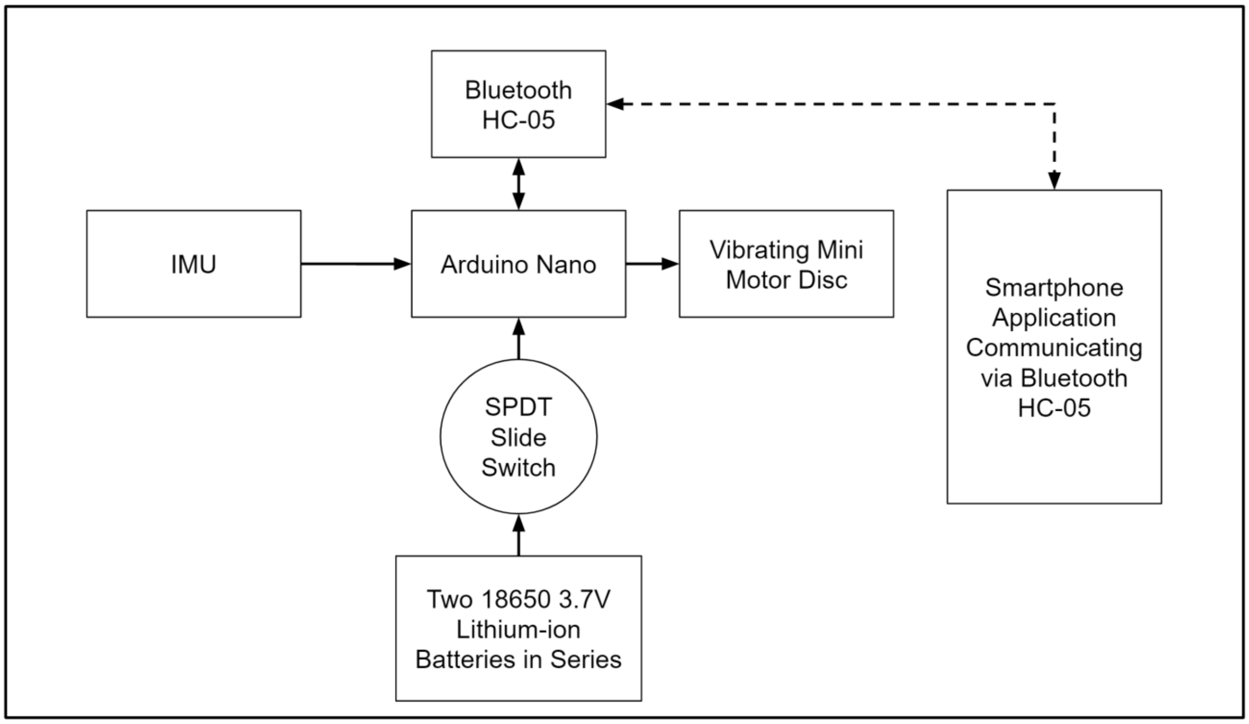

Bluetooth HC-05
IMU
Arduino Nano
Vibrating Mini Motor Disc
Smartphone Application Communicating via Bluetooth HC-05
SPDT Slide Switch
Two 18650 3.7V Lithium-ion Batteries in Series

Figure 2: Specific Block Diagram

2.1 Power Source

The method we will be using to power our device is through two 3.7V Lithium-ion batteries connected in series with a battery capacity of 2600mAh. In addition, these batteries are rechargeable and will hold a charge for many days (i.e. approximately 80 hours), which will make our device more convenient and user-friendly. These batteries come in a long cylindrical shape that will be a good fit for our product because the ideal shape is a long and thin device that fits comfortably over the shoulders. Lastly, two batteries instead of one will evenly distribute the batteries' weight and make the device more likely to stay on and be more comfortable.

2.2 User Input

In order to turn the device on and off there will be a SPDT slide switch connecting the batteries to the microcontroller. This will allow the user to conserve power when the device is not in use.

2.3 Microcontroller

The microcontroller model we chose is the Arduino Nano. Out of all the microcontrollers on the market, the Nano is the most feasible model for our application because it is small but it still contains all of the pins we need. We were originally going to purchase a microcontroller with a built-in wireless transceiver, but these boards put the Bluetooth into the peripheral. This seemed like it could cause problems, so we decided to get a separate Bluetooth module.

2.4 Posture and Physical Activity Sensor

In order to measure posture and physical activity we are using an IMU sensor. The IMU measures posture by sending the value of the Z-axis to the microcontroller. Bad posture is detected by calculating the difference between the original value when calibrated and the value at latest data point. Physical Activity will be measured using linear acceleration data. Since sitting and participating in non-physical activity will not register as linear acceleration, we can determine if the user is inactive by past linear acceleration data.

2.5 User Notification System

In order to notify the user that they are slouching or have not moved in a while we are implementing a small vibrating motor. The motor will be located on the user's back/shoulder area and will provide a brief vibration to remind the user to sit up straight or to go for a walk. This notification method is instantaneous and does not require the user to interrupt any activities in order to interact with the device. The motor will be controlled by the Arduino Nano and will be triggered using data collected by the IMU.

2.6 Wireless Transceiver

In order to transmit data to our application's database, we will be using a Bluetooth transceiver. The transceiver will be a HC-05 Bluetooth module in our design because we would like fast transmission speeds without having high power consumption. Additionally, we decided to choose Bluetooth SPP (Serial Port Profile) over BLE, because the former is designed for continuous two-way data transfer with high application throughput (up to 2.1 Mbps). Overall, the transceiver will facilitate the communication between the app and the rest of the physical parts.

2.7 Smartphone Application

The smartphone application makes up the software side of our project. The application will store history of the user's posture and physical activity data and display them in a user-friendly graph. Additionally, the application will provide user preferences including turning the physical notification system (the vibrator) on and off. When the user first puts on the device the app will also prompt them to calibrate the IMU to their unique sitting and moving habits. The application will be programmed using Android Studio and will communicate with the physical device via Bluetooth.

3. Module Detailed Descriptions

The full hardware schematic of the device is shown in figure 3 below.

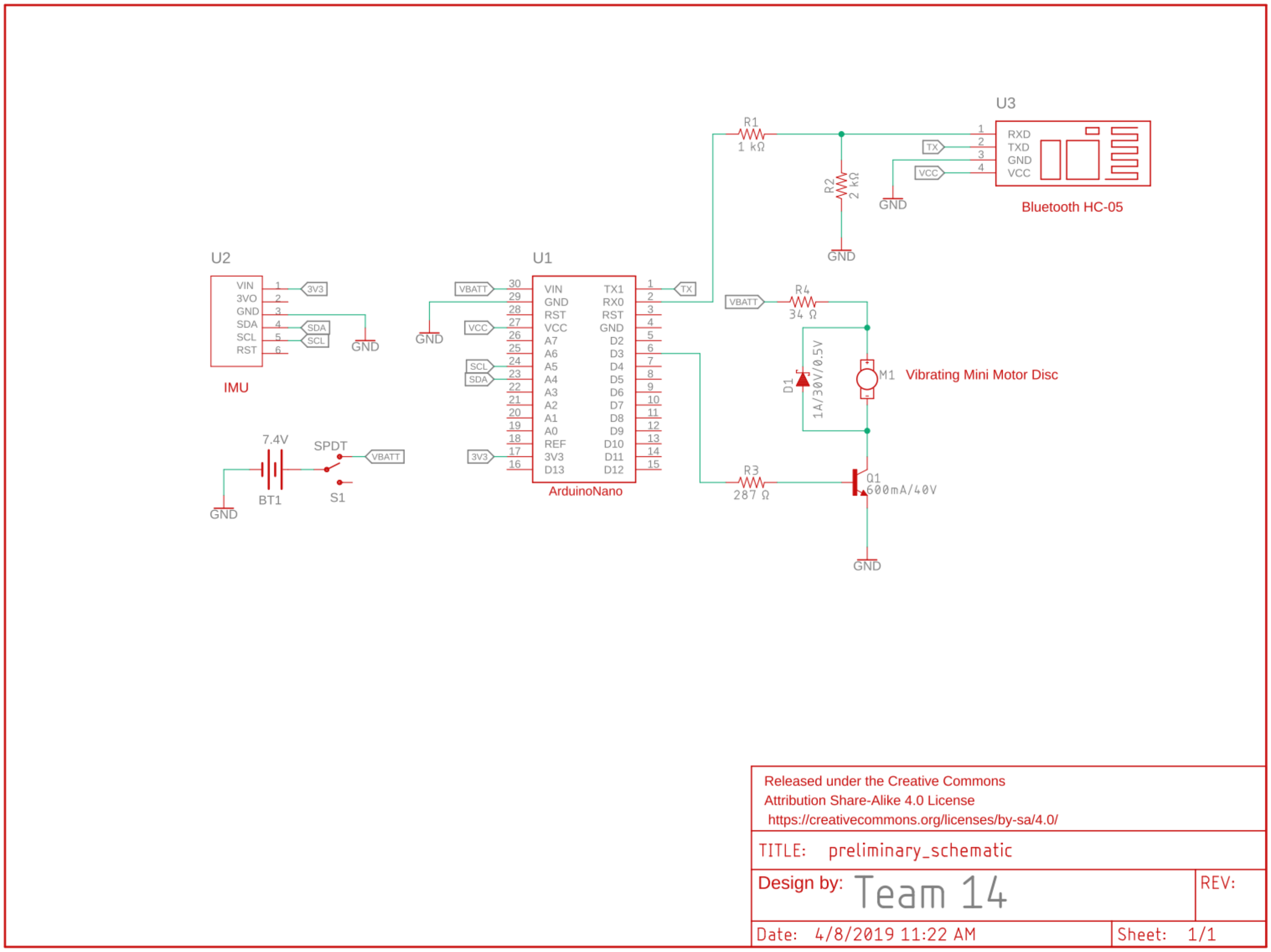


Figure 3: Schematic Diagram of the Hardware Device

In this section, each of the modules of the device will be discussed individually. For each module, we will describe the module functionality, the specifications and calculations used to reach the current design and ensure functionality, the inputs and outputs of the module, and the plan for testing the module individually (before integration).

The outer case of the device will not be discussed here because although we intend to make a prototype of the case for demonstration purposes, actually integrating the hardware with the case is outside of our project constraints. We will create a model of the proposed case and display it separately from the working prototype as a demonstration of what the final product would look like.

3.1 Arduino Nano

The microcontroller we have chosen to use is the Arduino Nano. This model is the best for our project because it combines an abundance of digital and analog data pins, data and clock lines, and a variety of power and ground pins with a very small size. The size for our device is a very important factor because it needs to be small enough to be discreetly worn beneath clothing. On the other hand, we have many modules that require their own data pin (sometimes more than one), power, and ground pins. As a result, we concluded that the Arduino Nano is the ideal microcontroller for this product.

3.1.1 Module Functionality

The Arduino Nano is our central microcontroller which includes the central processing unit that will be used for our system. More analytically, the microcontroller's functionality is to instruct the peripherals, like the IMU, the Bluetooth HC-05 and the vibrating motor to execute an individual set of commands. These commands could be to either request for sensor data gathered, to instruct transmission of data, or to enable a peripheral output. Furthermore, it is the microcontroller's responsibility to effectively process the data received from its peripherals using its own resources. For our device, we will program the microcontroller to initialize each of the other modules, along with variables and data structures that will be used, to check periodically if the user is slouching or if they are physically inactive, and to process this information. Moreover, whenever the Bluetooth HC-05 module receives a valid request through our mobile application, the microcontroller will attempt to satisfy this request by instructing the HC-05 to transmit data coming from the program's data structures. Lastly, the microcontroller will command the Bluetooth module to automatically transmit the daily posture and physical activity data to the user's phone app at 8 pm each day. The device's functionality, which will be executed through the Arduino Nano microcontroller, is also explained in the flowchart below (figure 4).

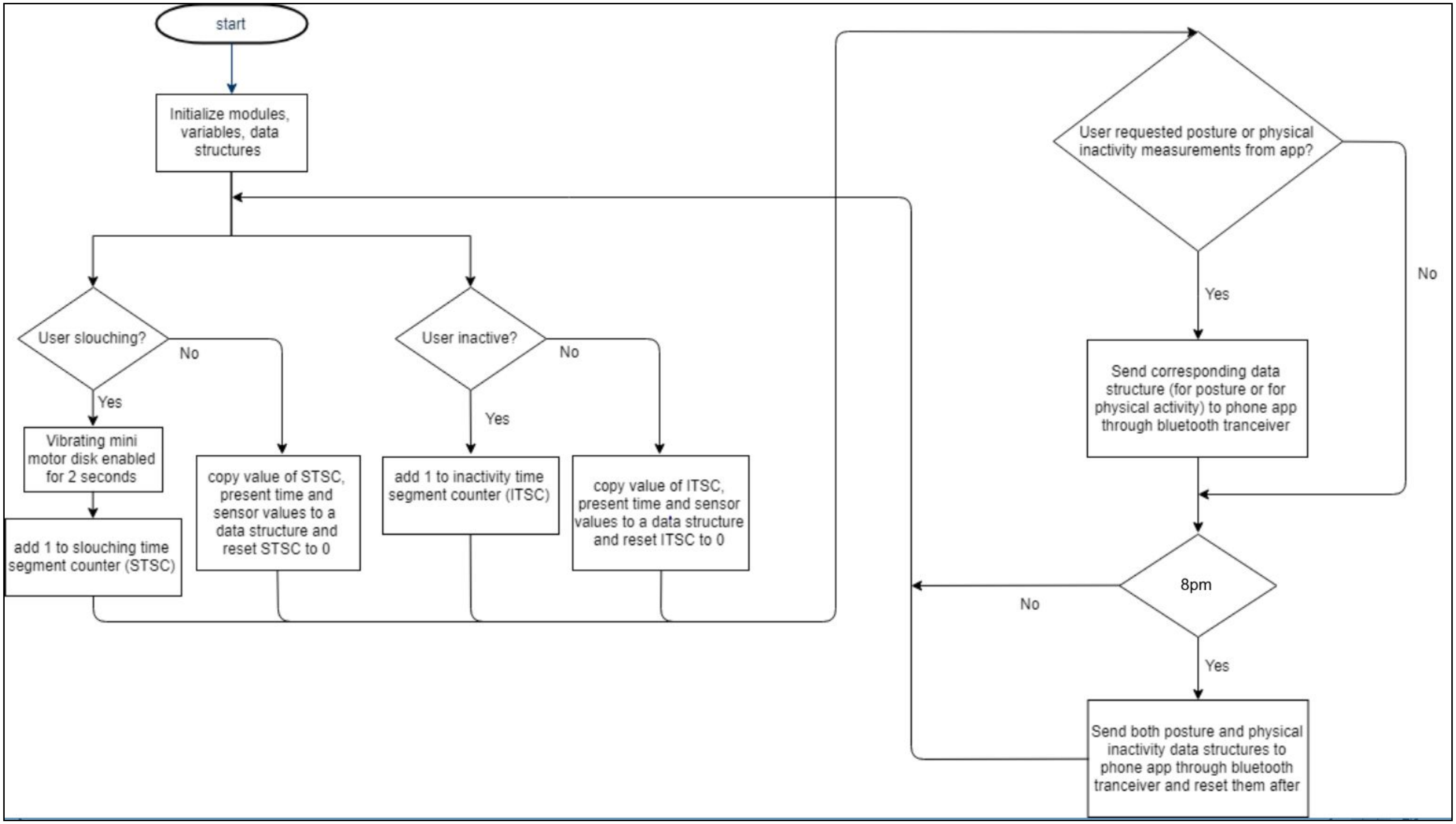

start
Initialize modules, variables, data structures
User slouching?
No
Yes
Vibrating mini motor disk enabled for 2 seconds
add 1 to slouching time segment counter (STSC)
copy value of STSC, present time and sensor values to a data structure and reset STSC to 0
User inactive?
No
Yes
add 1 to inactivity time segment counter (ITSC)
copy value of ITSC, present time and sensor values to a data structure and reset ITSC to 0
User requested posture or physical inactivity measurements from app?
Yes
No
Send corresponding data structure (for posture or for physical activity) to phone app through bluetooth tranceiver
8pm
No
Yes
Send both posture and physical inactivity data structures to phone app through bluetooth tranceiver and reset them after

Figure 4: Microcontroller Function Flow Chart

3.1.2 Specs and Calculations

The following specifications and calculations were used to ensure (to a reasonable degree of certainty) that the Arduino Nano module would function as planned. Many of these specifications were also used to make key design decisions for the module, particularly with regard to the inputs and outputs.

To verify that the Nano could be powered properly as well as supply power to its peripherals during implementation and testing, the following specifications were found in the data sheets. First, the recommended input voltage limit of the Nano is 7–12V. This means that the Nano can be successfully powered by the battery module we intend to use. Another power option which will be used during the programming and testing phase, is USB power via a computer. According to the data sheet, the total maximum current draw from the Arduino when powered from a USB port or power supply is 500mA. This means that even when powering the Nano by USB, it can supply more than enough current for all of the connected modules.

Next, the input/output (I/O) pins were examined to ensure that they will be capable of the functionality we need. We found the following specs.

- The voltage limits on I/O pins are -0.5 to +5.5V max
- The total max current per I/O pin is 40mA
- The sum of output currents of all I/O pins combined must not exceed 200mA

Since none of the peripherals connected to the Nano module will exceed these specs, the module should work as expected. The calculations used to confirm this are explained in the paragraphs below.

The Arduino Nano voltage values on I/O pins can span between -0.5V to 5.5V. Consequently, there is the possibility of pairing the Bluetooth module with the microcontroller and controlling the vibrating mini motor disc if fitting configurations are applied. More analysis on these topics will be provided in the sections below. For the case of the motor, it should be noted that one of the limitations of the Arduino Nano is that the maximum current that an I/O pin can provide is 40mA. Unfortunately, our vibrating motor draws from 40mA to 100mA when working from 2V up to 5V. This means that the startup current will be larger than the motor’s rated current, which could severely damage if not permanently destroy the pin. To avoid this issue, we included a transistor to limit the current coming out of the Nano I/O pin. We also added a few other components to complement this change (please refer to section 3.5 and the schematic in figure 3 of this report).

Another important consideration regarding the Arduino Nano is that the sum of all currents out of all the I/O pins combined cannot exceed 200mA. Despite the fact that the Nano voltage regulator may allow up to 500mA current draw across the “5V” and the “GND” pins, all analog and digital I/O pins must not amount to more than that limit. Therefore, we had to keep this specification in mind in order to not damage our microcontroller. The maximum current draw that is reached when adding up all of the current values in our design is 67.3mA. This calculation is shown in equation (1) below.

$$\text{IMU Max} + \text{Bluetooth Max} + \text{Motor Max} = 12.3\text{mA} + 40\text{mA} + 15\text{mA} = 67.3\text{ mA} \qquad (1)$$

Please note that the equation above is taking into account the absolute maximum current draw possible at any given time. The actual draw will be a much lower value (for example, the average draw for the Bluetooth module is only 8mA).

In conclusion, the Nano’s limitation with respect to the current sum of its I/O pins being below the 200mA limit is fulfilled in our circuit design.

3.1.3 Inputs and Outputs

The recommended input voltage for the Arduino Nano is 7-12VDC. We will supply the microcontroller with 7.4VDC through its “Vin” pin using the battery arrangement explained in section 3.2. However, the operating voltage of this microcontroller is 5V, meaning that it has a built-in voltage regulator. Furthermore, since the Nano is powered, it must also be grounded, which is done through the “GND” pin. The microcontroller is connected to the IMU through I2C. More specifically, the SCL (Serial Clock) and SDA (Serial Data) busses are connected to two of its analog pins that were designed for the I2C communication protocol, namely “A5” for SCL and “A4” for SDA. The Nano also powers the IMU with 3.3VDC through its “3V3” output pin (detailed characteristics of IMU are discussed in section 3.4). It is also connected to the Bluetooth HC-05 module through UART in order to transmit and receive serial data. The microcontroller powers the HC-05 using its “5V” output pin and uses its “Tx1” pin to receive data from the HC-05. The transmit and receive logic signal of the HC-05 uses 3.3V. Therefore, receiving signals from the HC-05 to the Arduino Nano microcontroller is safe, since the I/O pins can receive signals from -0.5V to +5.5V (in actuality, it is “Vcc+0.5V”, which for a 5V Arduino is +5.5V). However, the Arduino will transmit signals to the receiving pin of the HC-05 which has 3.3V logic, using 5V logic through the “Rx0” pin. To combat this discrepancy, which could potentially destroy our Bluetooth module, we created a voltage divider which will be described in section 3.3 along with other details concerning the Bluetooth HC-05 module. Finally, the Arduino Nano controls a vibrating motor through the “D3” digital I/O pin which also has PWM functionality. The configuration surrounding the vibrating mini motor disc is explained in section 3.5 of this report.

3.1.4 Individual Module Testing

We will test the Arduino Nano module individually before officially integrating it with any other modules. For individual testing, we will focus on the software side of the module. To do this, we will connect it to a computer via USB. We will then work within Arduino IDE to test that the Nano is programmed correctly. Test inputs will be given within the program, and the resulting output of the Nano will be checked for accuracy. For example, a certain range of inputs from the IMU module will be considered to denote that the user is slouching. We will artificially input values within these ranges into the Nano and check that it recognizes these measurements as slouching. We will also check that it processes these inputs correctly, and that the resulting outputs have the expected values.

3.2 Battery and Switch

The battery and switch are the most top-level controlling forces within our product. We are using a SPDT slide switch to control whether or not the device will be powered on or off. The batteries we decided were the most applicable to our product are Lithium-Ion batteries with specifications of 3.7V, 2600mAh. We will be using two batteries in series, which will increase the voltage output to 7.4V. These batteries will power each of the physical modules and electricity will be shared using the microcontroller and will be distributed with a series of resistors, transistors, and diodes at each module.

3.2.1 Module Functionality

The batteries coupled with the switch will control the power output to the circuit. The batteries were chosen for their long life and their ability to be recharged. Additionally, we wanted Lithium-Ion batteries because they are able to provide both a relatively high voltage and current. The other batteries we considered either had the battery capacity we needed, but lacked a high enough voltage (e.g. NiMH, AA batteries) or vice-versa. The switch we decided to use is a SPDT slide switch which will disconnect the power from the circuit when not in use. SPDT stands for Single Pole Double Throw, but in our case we are using it as an SPST switch by using the common terminal, which is connected to the battery's positive lead and either one of the other two terminals, which is connected to the rest of the circuit. The remaining terminal is not connected to anything.

3.2.2 Specs and Calculations

Our microcontroller (Arduino Nano) requires an input voltage of 7V to 12V, but its built-in regulator lowers the operating voltage to 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 maximum 12.3mA. The Bluetooth transceiver (HC-05) will be powered through the 5V pin and will require maximum 30-40 mA only while the module is attempting to pair with another device. Finally, our vibrating motor (vibrating mini motor disc from Adafruit) will be powered directly from the battery and controlled through a digital I/O pin (with PWM) of the Nano. When the vibrating motor is enabled, the battery will provide it with about 87mA under 4.4V with the help of a voltage divider (87 mA is the current draw it requires when it is working under 4.4V - the calculations and the circuit regarding the vibrating motor will be explained in section 3.5.2). However, on average, the current draw of our system is estimated to be around 25mA (approximately 15mA for the Arduino Nano while running, around 2mA for the IMU and about 8mA for our Bluetooth transceiver). The current draw of the vibrating motor is not included in this estimation, since it will be off for most of the time. Moreover, we would like to note that we will measure current draw more accurately using a Digital Multimeter when we receive our materials. On the other hand, our goal for battery life is to last about 5 days, with 16 hours of battery life allocated to each day. Therefore, our goal for total battery life is about 80 hours. From the equation below we will be able to determine the capacity of the battery. 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.

$$\frac{Battery\ Capacity\ (Amphours)}{Current\ Draw\ (Amps)} \cdot 0.8 = Battery\ Life\ (Hours) \quad (2)$$

After taking duty cycle for each of the modules into account and using this equation, the resulting battery capacity we get based on our requirements is 2500mAh. Therefore, we decided to use two 3.7V lithium-ion batteries with a battery capacity of 2600mAh connected in series for powering our system.

3.2.3 Inputs and Outputs

Our design uses two batteries that will be connected in series. The power source is controlled by a SPDT slide switch, which is connected to the vibrating motor, the microcontroller and all other modules by extension. The switch has a common terminal that is connected to the positive terminal of the batter. With respect to the other two terminals of our switch, one is connected to the the rest of our circuit through the “Vin” pin of the Arduino Nano and the positive terminal of the vibrating motor, whereas the other is not connected to anything. The side whose pin is connected the microcontroller will be the ‘on’ position and the side whose pin is not connected to anything will be the ‘off’ position.

3.2.4 Individual Module Testing

Testing for the switch will be as simple as seeing if power is applied to the circuit when in the ‘on’ position and no power is applied in the ‘off’ position. There are many tests that can be done to see if this is the case, these include simply watching to see if the microcontroller turns on when the switch is in the ‘on’ position or using a Digital Multimeter (DMM) to measure the output voltage at the switch terminals. We will be able to test the batteries using a DMM as well to see what outputs the batteries are averaging, which will determine if any schematic changes need to be made to accommodate the batteries.

3.3 Bluetooth HC-05 Module

The wireless transceiver that we will using for this project is a HC-05 Bluetooth module. The Bluetooth will act as a gateway to the software side of the device by sending the data collected by the IMU to the database of our smartphone application.

3.3.1 Module Functionality

The Bluetooth HC-05 module of our device will provide the means for connection to the user's smartphone. It will send data from the Arduino Nano microcontroller to the phone application module and vice versa. The data being sent from the HC-05 to the phone will include information on the user's posture, physical activity level, and the time at which these measurements were taken. When the phone transmits to the HC-05, it will be a request for the aforementioned data to be sent.

3.3.2 Specs and Calculations

The following specifications and calculations were used to ensure that the Bluetooth module would function as planned. Many of these specifications were also used to make key design decisions for the module, particularly with regard to the resistors connected to the HC-05.

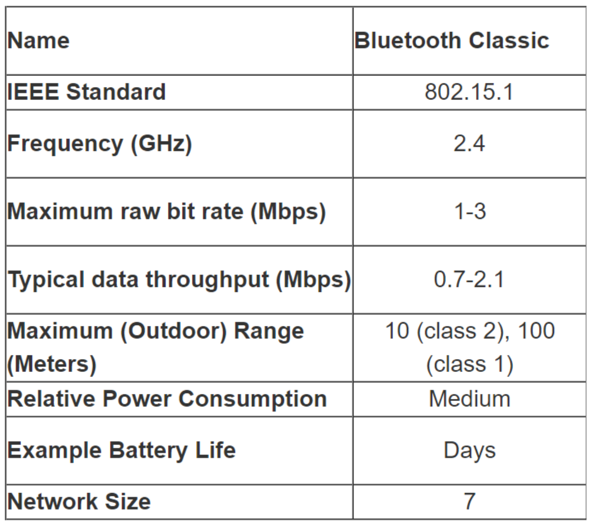
The main concern with this module was making sure the correct logic voltage level was being sent to the HC-05 from the Arduino Nano and vice versa. The logic level of the Nano is approximately 5V and the logic level of the HC-05 is approximately 3.3V. To accommodate this difference, we will implement a simple voltage divider between the RX pin of the HC-05 and the RX0 pin of the Nano. This voltage divider needs to deliver only 3.3V to the HC-05. We found that we will need proportional resistor values of R1 = ½*R2 using equation (3).

$$3.3V = 5V*R2/(R1 + R2) \rightarrow R1 = R2*(5V/3.3V - 1) \simeq ½*R2 \quad (3)$$

Based on this, we chose R1 to be 1kΩ and R2 to be 2kΩ (Both 1% tolerance). No additional components are needed on the TX pin of the receiver because the Nano will accept any voltage between -0.5V and 5.5V at the TX1 pin.

Another concern was the current draw of the HC-05 on the Nano. The HC-05 will be connected to the 5V output pin (VCC) of the Nano. We had to be sure that the current draw would not exceed the limit for the connected Nano pin. According to the HC-05 specs, the current fluctuates in the range of 30-40mA during pairing. After pairing, regardless of whether or not the HC-05 is processing data, the current is 8mA. Based on this, the maximum amount of current draw on the VCC pin in 40mA. This will not cause any issues because the VCC pin on the Nano can handle up to 500mA.

We decided to choose Bluetooth SPP (Serial Port Profile) because it is designed for continuous two-way data transfer with high application throughput (up to 2.1 Mbps). Consequently, it is ideal for sending bursts of data between two devices. During prototyping, we would like to make sure that performance is guaranteed for short distances by having a continuous broadband link (multiple data will be received/transmitted). Figure 5 below shows some of the basic characteristics of the Bluetooth profile being used in our device.



Name	**Bluetooth Classic**
IEEE Standard	802.15.1
Frequency (GHz)	2.4
Maximum raw bit rate (Mbps)	1-3
Typical data throughput (Mbps)	0.7-2.1
Maximum (Outdoor) Range (Meters)	10 (class 2), 100 (class 1)
Relative Power Consumption	Medium
Example Battery Life	Days
Network Size	7

Figure 5: Bluetooth Classic Basics

Because of these characteristics and the standardized nature of Bluetooth SPP, there should be very few issues related to the transceiving of data between the HC-05 and the user's smartphone.

3.3.3 Inputs and Outputs

This section will cover each of the inputs and outputs for the Bluetooth module (as shown in the schematic in figure 3). As explained in the Specs and Calculations section, the inputs for the HC-05 module are the 5V input, which powers the HC-05 itself, and the 5V logic signal from pin RX0 of the Nano. This logic level is reduced to 3.3V at the RX pin of the HC-05. The HC-05 module has only one output, a 3.3V logic signal at the TX pin. In addition to these hardware inputs and outputs, the HC-05 also has the input and output associated with receiving and transmitting wireless signals, but these will not be considered because they do not affect the other modules.

3.3.4 Individual Module Testing

In order to test this module, it will need to be connected to some form of processor. Because of this, we will conduct testing of the module once it is connected to the Arduino Nano. Before connecting it to any modules besides the Nano, we will perform some basic testing. First, we will manually measure the voltages and currents at the inputs and outputs to ensure that the components are behaving as expected. Next, we will try pairing the HC-05 with a smartphone. The finished phone application is not needed for this. We intend to use an existing Bluetooth terminal phone application to communicate with the HC-05 and test its functionality.

3.4 BNO055 Absolute Orientation Sensor (IMU)

The IMU sensor is the method we are using to track posture and physical activity. This sensor collects orientation data using 9-axes and includes absolute orientation, angular velocity vector, acceleration vector, magnetic field strength vector, linear acceleration vector, gravity vector, and a temperature sensor.

3.4.1 Module Functionality

We will be utilizing absolute orientation to measure posture because it can detect subtle movements like slouching. Additionally, we will be using the linear acceleration vector to measure physical activity. The linear acceleration vector will work well for this application because if the value increases we will know the user is moving around, but if it has been zero or close to zero for a while we will tell the user that they should go for a walk. The IMU will be controlled using the Arduino Nano and the information it gathers for both absolute orientation and the linear accelerator vector will be transmitted to an application via the Bluetooth module.

3.4.2 Specs and Calculations

According to the specifications, the IMU can be powered between 3.3V to 5V. We chose to power it with 3.3V in order to reduce power consumption. Another important aspect of the IMU which had to be calculated was the duty cycle—in other words, how much of the time it would actually be running. Taking this into account drastically reduced our required battery size, because the IMU only needs to take measurements periodically, not constantly. The BNO055 Absolute Orientation Sensor has multiple power modes. We are hoping to use both normal mode and suspend mode. In normal mode, the maximum current draw is 12.3mA. In suspend mode, the maximum current draw is 0.04mA. If the device takes a measurement for 10 seconds per minute, it will only be in normal mode 16.67% of the time. This means that the approximate average current draw is 2.08mA. Based on these calculations, the battery size needed for the device is significantly smaller than if the IMU was in normal mode all the time.

3.4.3 Inputs and Outputs

The IMU is connected to the microcontroller by three pins: VIN, SDA, SCL. The IMU is powered by its VIN pin connected to the 3.3V pin on the Nano. The data collected by the IMU is transmitted through the SDA and SCL pins that connect to the A4 and A5 pins on the microcontroller, respectively. The SDA pin is the primary data pin that transmits important information to the data line on the Nano (pin A4). The SCL pin is the IMU's clock that transmits when the data was collected in milliseconds to the microcontroller clock line (A5). Lastly, the GND pin on the IMU is connected to ground.

3.4.4 Individual Module Testing

Testing for the IMU will consist of collecting data from the IMU to identify patterns in the outputs. We will put the IMU in similar situations to what it will have to do in the final prototype like wearing it on our backs to see what values it generates. After testing the IMU in these situations multiple times we will extract the range of values that represent the user movements that we are trying to identify.

3.5 Vibrating Mini Motor Disc

The mini vibration motor module will be used to give an instant notification to the user that they are slouching and/or have been sedentary for too long. It is important that the motor disk be small in order to keep the prototype as tiny as possible, so we bought the Vibrating Mini Motor Disc from Adafruit because it is smaller than a dime. While the motor would appear easy to work with (it only has on/off modes and a positive and negative input/output), it requires a fairly complicated circuit to make sure stress is not placed on the rest of the circuit. In addition to the motor, this module will include a Schottky diode, a NPN transistor, and various resistors.

3.5.1 Module Functionality

The vibrating mini motor disc is used as a direct user notification system for our application. More precisely, the vibrating motor is enabled through our microcontroller for a couple of seconds whenever our device senses that the user is slouching. If the user keeps slouching even after they are notified from the vibrating motor once, the motor will be activated again after a longer period of time as long as bad posture position is being sustained throughout that time. The amount of time till the vibrating motor could potentially be enabled again will be a multiple of the time interval between two consecutive checks of the user’s posture. We decided to apply this implementation because we want to make sure we directly remind users on their posture position, without overly disturbing them since the device will periodically check the user’s posture in short time fragments.

3.5.2 Specs and Calculations

The following specifications and calculations were used to ensure that the vibrating motor module would function as planned. Many of these specifications were also used to make key design decisions for the module, particularly with regard to the circuit elements connected to the motor itself.

In our application, we will control the vibrating motor through a PWM pin of the Arduino Nano. However, for this to happen without burning the PWM pin, we have to add 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 circuit shown in the schematic (low-side switch circuit). 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 PWM 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.

First, we calculated the value of the resistor connected between the Arduino Nano and the transistor. This resistor is used to limit the current the digital output must source and the transistor base must handle. We wanted a resistor that would draw a current of approximately 15mA from the I/O pin of the Nano with a 5V digital output. We found a value of 287Ω using equation (4) below.

$$R = (V_{in} - V_{be})/I = (5V - 0.7V)/15mA \simeq 287\Omega \qquad (4)$$

This resistor (1% tolerance) is shown as R3 in the Multisim circuit simulation displayed in figure 6.

Next, the resistor between the battery and motor was calculated. The purpose of this resistor is to adjust the current draw of the motor on the batteries. It is also necessary because the motor is meant to function between 2V–5V and the batteries supply 7.4V. It acts as a voltage divider with the resistive load of the motor. Since the motor has a linear V-I characteristic (5V current draw: 100mA, 4V current draw: 80mA, 3V current draw: 60mA, 2V current draw: 40mA), it can be easily modeled as a resistive load. By Ohm’s law, the motor can be modeled as a 50Ω resistor, as shown in equation (5).

$$R = V/I = 5V/100mA = 50\Omega \quad (5)$$

Once the motor resistance was known, the voltage divider resistor value was chosen. In order to stay safely within the 2-5V range across the motor, we chose a resistor value (resistor with 1% tolerance) that would leave just below 4.5V across the motor. For simplicity, we used 4.4V. The calculation is shown in equation (6) below.

$$4.4V = 7.4V*50\Omega /(50\Omega +R) \rightarrow R = 7.4V*50\Omega/4.4V - 50\Omega \simeq 34\Omega \quad (6)$$

Finally we had to integrate a flyback diode in our design. The reason for adding this component to our circuit, is because the motor is partially an inductor. Consequently, without a diode across the motor, if the transistor shuts off quickly, then the voltage across the motor will get larger to ensure the continuity of current flow. This will probably result in destroying the transistor. However, if a flyback diode is used when the transistor shuts off rapidly, the current that is still needed to flow through the motor (inductor) for some time will flow through the diode instead. For our application, we decided to use a Schottky diode, namely the 1N5818. We decided to use this type of diode because, when the motor is controlled through a PWM pin, we need a diode with fast reverse recovery at this low voltage. Schottky diodes have fast twitching speeds, meaning that they are great with higher frequencies since they still succeed in rectification (i.e. preventing reverse flow of current). Therefore a Schottky diode suits our application better than general purpose diodes, because we will be turning the motor on and off rapidly, as implied by "PWM", whereas general purpose diodes are power rectifiers intended for normal power line frequencies (e.g. 50-60 Hz). The lower forward voltage of the 1N5818 will also cause less backwards EMF on the inductance during off-time and thereby, will make the system more efficient. To conclude, the 1N5818 fulfills all of our requirements regarding maximum reverse DC voltage, average current rectified and maximum forward voltage of our application, based on our calculations and the results from the Multisim simulation.

The Multisim simulation which demonstrates the calculations above is shown in figure 6.

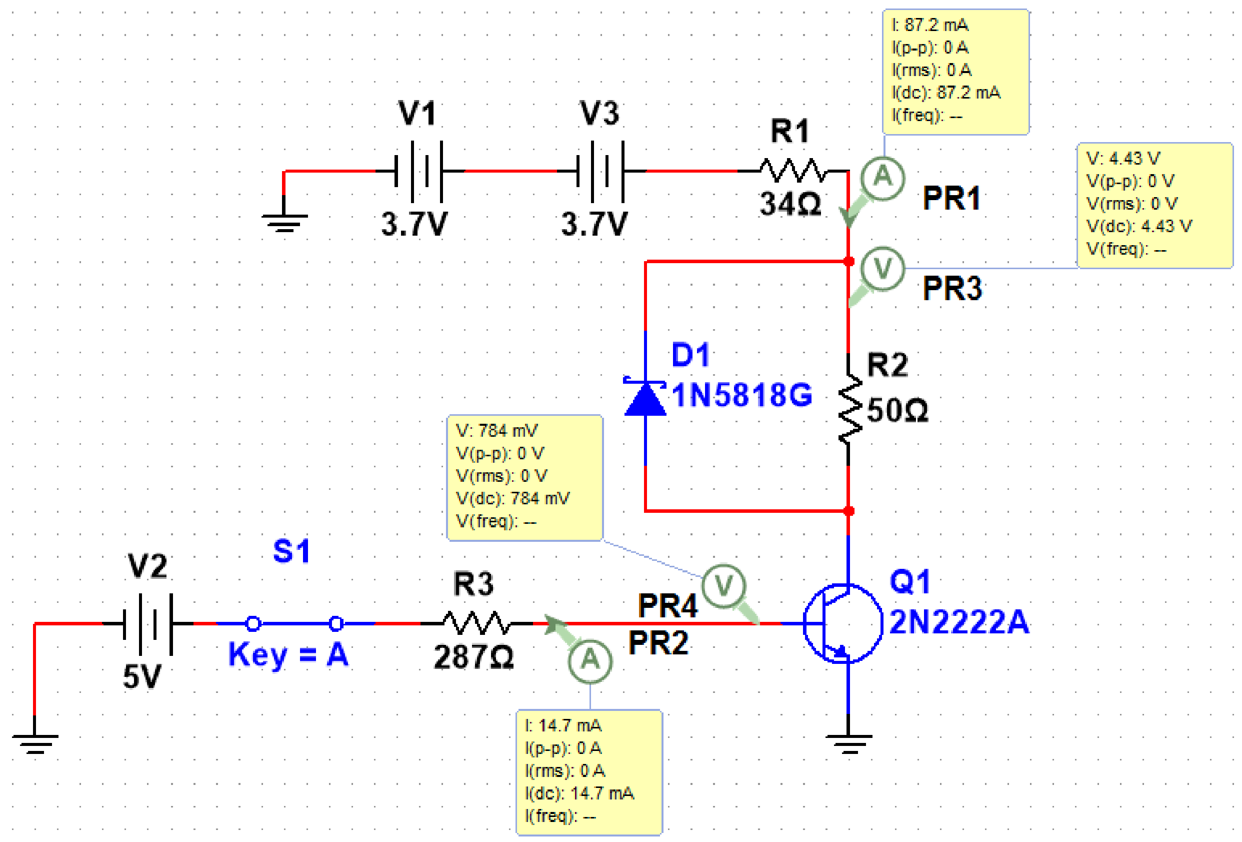


Figure 6: Vibrating Motor Module Simulation

3.5.3 Inputs and Outputs

The vibrating motor will be controlled through the “D3” PWM pin of the Arduino Nano microcontroller. Therefore, based on our calculations and the specs of the modules used, we ended up with the following configuration. The “D3” pin will be connected to the base side of the P2N2222A (NPN) transistor with a 287Ω resistor in between the two, to limit the current sourced by the pin. The negative terminal of the vibrating motor will be connected to the collector side of the transistor, while its positive terminal will be connected to the positive leads of the battery with a 34Ω resistor in between to decrease the voltage of the battery to an acceptable value with respect to the vibrating motor. Moreover, the 1N5818 Schottky diode is connected across the vibrating motors as a flyback diode to eliminate the voltage spike created when the field in the motor collapses and thereby, avoid burning the Nano or the transistor. Lastly the emitter voltage of the P2N2222A transistor is connected to ground.

3.5.4 Individual Module Testing

In order to test this module before integrating it with any other modules, we will use a voltage supply and a Digital Multimeter (DMM) available in the Atwater Kent Labs. By supplying the input and output voltages which will be supplied by the other modules after integration, we can simulate the function without risking damage to any other components. We will check the functionality of the module, i.e. that the supply voltage is sufficient, the motor turns on and off when the logic voltage is changed, it vibrates properly, and the response time is appropriate for the application.

3.6 Software Application

Our product prototype will also include a phone application that will interact with our device. The purpose of this phone app is to inform the user about their habits regarding posture and physical inactivity. The user will be able to "chat" with the device to receive the processed sensor data from our microcontroller, via the Bluetooth module. The processed sensor data are constructed by programming our microcontroller to judge whether the user has been slouching or has been physically inactive based on the sensor values received from the IMU. While executing this, the microcontroller will also do time-keeping in order to keep track of when and for how long the previously mentioned events take place. The device will also output these data without any user input, once a day. Finally, these data will be used to display graphs in order to provide visual feedback to the user, concerning their habits that are sensed by our device.

3.6.1 Module Functionality

The smartphone application provided along with our device will assist the user in receiving useful feedback with respect to their posture and physical activity on a daily manner. The phone app will communicate with our device through a Bluetooth terminal. The data that will be transmitted and received from the app will be displayed on the app's screen in a chat box format. The user will be able to "ask" the device on how much time they have been slouching or have been physically inactive on that day till the time they requested for that information. At 8pm each day, the data of one day will be received from the phone app and will also be shown in the Bluetooth terminal chat box. Any data received from our device will be processed, resulting to the display of graphs regarding posture and physical activity. More precisely, they will show the amount of time the user has had a good vs a bad posture and for how much time a user has been physically active vs inactive during the day. Furthermore, the application will be intuitive to navigate and the data will be presented in a way that they will be able to give a clear message and provide tips to enhance health based on the sensors' readings. The phone app is being developed in Android Studio and the graphs will be created with the help of the "GraphView" library. The app's functionality is also displayed in the flowchart below (figure 7).

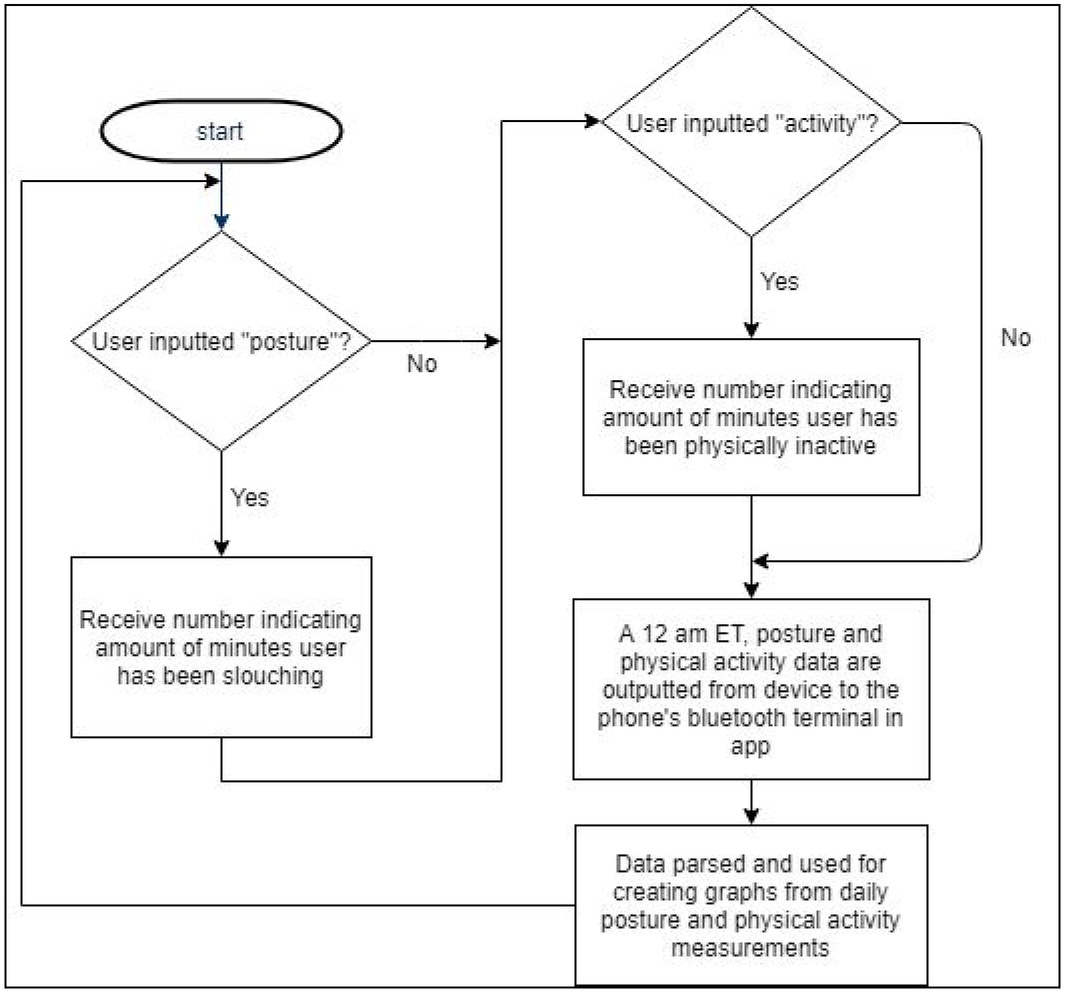

start
User inputted "posture"?
No
Yes
Receive number indicating amount of minutes user has been slouching
User inputted "activity"?
Yes
No
Receive number indicating amount of minutes user has been physically inactive
A 12 am ET, posture and physical activity data are outputted from device to the phone's bluetooth terminal in app
Data parsed and used for creating graphs from daily posture and physical activity measurements

Figure 7: Application Flow Chart

3.6.2 Inputs and Outputs

The data collected by the IMU will be transferred and processed by the microcontroller. Then, the Bluetooth module will transmit the processed data to the application, either upon the user's request (output from app to device) or automatically to show the daily posture and physical activity data of the user. These input data will be shown in the Bluetooth terminal chat box screen and will also be parsed in order to output user friendly graphs that will depict the user's data for the day. Additionally, the app will have other navigation and user interface graphics that will allow the customer to choose the information they are viewing and allow them to request more up to date data from the device.

3.6.3 Individual Module Testing

To test our phone app, we will need to see if all its individual functionalities are working as expected. First, we need to check whether the correct data is displayed on the Bluetooth terminal screen and whether the data transmitted and received from the app are as anticipated. To test this feature, we will transmit and receive multiple values and see the response caused by our microcontroller and app programming. Furthermore, we have to test if our data is parsed correctly in order to display graphs based on the input data received from our device. To do that, we will manually send some dummy data values from the microcontroller to the app to check if the graphs that were created are corresponding to the input data.

4. System Integration

Once each module has been tested individually, we will begin integrating each module into a complete system. This section will describe our plan for completing the integration and testing processes for both hardware and software modules. Some modules are considered both hardware and software. For example, the Arduino Nano must be physically connected to other modules, but it also contains a significant amount of software functionality. Because of this, at the beginning of the hardware and software sections, each module being considered in the section will be listed.

4.1 Hardware Module Integration and Testing

The hardware module integration process will consist primarily of creating the physical connections between modules and making sure that each module has the correct voltage and current supply in order to work properly. It will be done in parallel with software integration.

The modules will be physically connected together in the following order, beginning with the Arduino Nano, which acts as the center of our building process.

- Arduino Nano
- IMU
- Vibrating Motor
- Bluetooth HC-05
- Batteries
- Switch

The IMU is being connected first because it is arguably the most complex part to integrate, and we are trying to avoid complications from any other modules while integrating it with the Arduino Nano. The vibrating motor and HC-05 can be implemented in any order as there is no particular reason to integrate one before the other. The batteries must be integrated after each of the other modules which draw current. This will give us a more accurate depiction of the batteries' function in the final device. Finally, the switch will be added last, once each of the other hardware components and the batteries are functioning correctly.

4.1.1 IMU Integration and Testing

The first step is connecting the IMU to the Arduino Nano. As shown in the schematic (figure 3 shown in section 3), the IMU will receive its power directly from the Arduino Nano. The voltage input pin of the IMU (VIN) will be connected to the 3.3V output of the Nano (3V3). In addition, the I2C clock pin (SCL) and I2C data pin (SDA) will be connected to the I2C-capable pins of the Nano (A4 and A5). The IMU will also be connected to ground.

To test that the hardware is configured correctly, a multimeter will be used to measure the voltages at the pins of each module. We expect to see a voltage of 3.3V at the input of the IMU.

We will also check the current flow to make sure the current draw of the IMU does not exceed the maximum current draw from the Nano 3V3 pin (50 mA).

4.1.2 Vibrating Motor Integration and Testing

Next, the vibrating motor module will be connected to the Nano. Before attempting to turn the motor module on using the Nano, we will have first tested the module individually to make sure the voltage and current is properly regulated. Once we are sure that running the motor will not damage the Nano, we will test its integration with the Nano. To test the integration of the vibrating motor before the software of the device is fully implemented, we will give simple commands through the Nano to turn the motor on and off.

4.1.3 Bluetooth HC-05 Integration and Testing

Next, the Bluetooth module will be connected to the Arduino Nano. The HC-05 wireless Bluetooth transceiver will be powered by the Nano's 5V output pin. This pin was chosen because the HC-05 can work on any voltage between 3.6V and 6V. In addition, the receiver (RX) and transmitter (TX) pins of the HC-05 will be connected to the RX0 and TX1 pins of the Nano.

Because the logic level of the Nano is approximately 5V and the logic level of the HC-05 is approximately 3.3V, a simple voltage divider will be used between the RX pin of the HC-05 and the RX0 pin of the Nano. This voltage divider will deliver only ⅔ of the voltage to the HC-05, which is approximately 3.3V. This will lower the voltage being sent to the HC-03 receiver. No additional components are needed on the TX pin of the receiver because the Nano will accept any voltage between -0.5V and 5.5V at the TX1 pin. Please note that the RX0 pin of the Nano sends data, and the TX1 pin receives data.

To test the hardware integration of the HC-05 with the Nano, we will measure the voltages and current draw. This will include ensuring that the voltage divider is working correctly. The actual functionality testing of the Bluetooth module integration will be covered in the Software Module Integration and Testing section of this report.

4.1.4 Battery Integration and Testing

Until this point, the Arduino Nano will have been powered by USB plugged into a computer. The next step will be to connect the two 3.7V lithium ion batteries which will be placed in series to the Nano. This is a simple but crucial step, as the batteries must be able to drive not only the Nano but also all of the connected components.

To test that the batteries integrate properly with the rest of the modules, we will perform various tests. These will include running the device with all of the modules at the lowest power possible, and again with each module on at the same time to make sure the maximum draw can be handled. Eventually, we will also run the device as a whole (in the way we expect it to be used) for an extended period of time to test the duration of the battery life.

4.1.5 Switch Integration and Testing

The final hardware integration will be the switch/user input module. For this, we will simply connect the hardware switch between the battery module and the Arduino Nano module. This will allow the user to manually turn the entire device on and off.

In order to test that the switch works as expected, we will measure the voltage on either side of the switch when it is on, and again when it is off.

4.2 Software Module Integration and Testing

The software module integration process will consist primarily of making sure each module of the device communicates smoothly with the other modules. It will also include extensive testing of the data processing performed within the Arduino Nano itself as it relates to other modules.

The modules will be integrated together in the following order, beginning with the Arduino Nano, which is the main controller of all software functionality.

- Arduino Nano
- IMU
- Bluetooth HC-05
- Phone Application

This integration order is based on what is needed to test each modules functionality. First, we will need to program the Arduino and the IMU to communicate with each other. This is how we will gather the posture and physical activity data for the device’s main functionality. This will need to work properly before we can send the collected data to the phone application via the HC-05. Once the IMU and Nano are integrated in terms of software, the Bluetooth HC-05 will be integrated via programming with the Nano. Finally, once the Bluetooth connection is working smoothly, we can begin to improve the phone application's functionality and appearance, as well as how it interacts with the data it receives.

The testing process for the software integration will involve the continuous testing and debugging of our code as we write it. As we integrate new modules, we will continuously check that each of the modules is communicating with the others and processing the data correctly. The final test will simply be to make sure each of the functions of the device are successfully carried out.

4.3 System Testing

In order to test the system as a whole, we will position the device on a test user's body where it would be placed during operation. We will test each aspect of the functionality. This will include posture detection, physical activity detection, the vibration functionality, the on/off switch functionality, the Bluetooth syncing functionality, and the useability of the phone application. During the final testing phase, we will be using the product specifications and requirements as a benchmark to compare the functionality to. This will ensure that our final prototype meets the requirements and we do not overlook any critical features.

5. Conclusion

The planning outlined above shows that we are ready to start building our prototype, but our parts have not arrived yet. Once they arrive we will be able start building each module. Nevertheless, we have started developing our mobile application in Android Studio. We have managed to make a Bluetooth terminal that has the ability to pair with other Bluetooth devices; we will test its functionality further when we set up the Bluetooth module and program the microcontroller to process the data that the module will receive. The current status of every module is that planning is complete, but building has yet to begin. Since we have not started building it is not yet fully apparent what is going to cause problems and take more time than expected. Some modules that we foresee taking longer are the communication between the software and hardware and the calibration of the IMU sensor. Moreover, optimizing the detection of bad posture or physical inactivity consistently may also require additional time and effort. Overall, we have exhausted the planning phase and now it is time to start building!

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.