Self-Monitoring Stress Device

Submitted To

Dr. Emily Porter

Prepared By

Matthew E. Barondeau
Harini M. Cherupalla
Matthew Q. Davis
Angelos Koulouras
Sneha S. Pendharkar
Neha J. Shah

EE464 Senior Design Project
Electrical and Computer Engineering Department
The University of Texas at Austin

Spring 2020

CONTENTS

CONTENTS (Continued)

TABLES

FIGURES

EXECUTIVE SUMMARY

1.0 INTRODUCTION

Stress is becoming an increasingly common challenge in the day-to-day lives of college students. If students are more aware of their stress, the destructive consequences of stress can be reduced. Our senior design project is a self-monitoring stress system using a network of sensors accompanied by a mobile application. The sensors measure heart rate and Galvanic Skin Response (GSR), since these are shown by physiological indicators of stress. These sensors are interfaced with the Texas Instruments MSP432 microcontroller, which collects sensor data. The data collected is compiled as a Comma Separated Value (CSV) file and sent to the mobile application. We use a machine learning model integrated into the mobile application to evaluate the data and make an assessment of the stress level. This data is also plotted as a line graph on the mobile application so users can view their trends over time. Through this graph, users can see the effectiveness of different stress-reducing activities such as meditation and mindfulness.

2.0 DESIGN PROBLEM

Stress in college students is often overlooked and can lead to potentially serious issues. With changes in lifestyle and new responsibilities, college students experience high levels of stress that cause feelings of frustration and pain. The purpose of our project is to help students monitor their stress levels and the impact of stress-reducing activities. To ensure that our device can be useful in helping students, our functional requirements include the design of a portable network of sensors that measure multiple physiological indicators of stress. The data from these sensors should be extracted to evaluate the stress levels, and this information should be recorded for the user to view. The budget allocated towards the design, construction, and testing of the project was $1000.

3.0 DESIGN SOLUTION

Our project consists of three subsystems: hardware, embedded system, and software. The hardware subsystem is comprised of a network of sensors. These sensors are interfaced with the embedded system which collects sensor readings and communicates this data to the software subsystem. Our software subsystem, which is an iOS mobile application, forms the user interface of our project.

Hardware Subsystem

Our network of sensors includes two sensors: a heart rate sensor and a galvanic skin response sensor. Both increased heart rate and increased skin conductance are indicators of stress. To measure heart rate, we are using a MAX30102 Reflective Optical Sensor (ROS) which shines an infrared Light Emitting Diode (LED) onto the skin and measures the reflected light on a photodiode. The amount of reflected light changes with changes in oxygen saturation in blood vessels. Oxygen content increases immediately following a heartbeat and decreases after. To measure GSR, we are using a Grove GSR sensor which applies a voltage across a voltage divider through electrodes placed on two fingers. The resistance across the fingers changes as the user becomes more or less stressed, which modifies the voltage output of the divider and the value read by the sensor. With excessive motion, the readings from these sensors may be inaccurate. To address this issue, we use an MMA8451 accelerometer to store X, Y, and Z positions in a hardware queue.

Embedded Subsystem
Our device uses an energy efficient Texas Instruments MSP432 embedded processor to collect data from the sensors and communicate the measurements to the mobile application. The MSP432 reads GSR sensor values through a high precision Analog-to-Digital Converter (ADC). The MSP432 also reads the values of reflected light from the ROS and if there is a pattern of increasing infrared light followed by a decrease, the software marks it as a heartbeat and begins to look for the next beat. Along with this, the MSP432 reads the MMA8451 hardware queue through an I2C read access and calculates the X, Y, and Z coordinates. We use the Universal Asynchronous Receiver Transmitter (UART) to transmit the sensor data to a serial console and convert it to a CSV file to send to the mobile application.

Software Subsystem
Our user interface for the project is an iOS mobile application so that users can view the data read from the sensors. The embedded system sends the data read from the sensors to the mobile application as a CSV file. A machine learning model is integrated into the mobile application to classify each data point as stressed or not stressed. In iOS development, kNN classifiers and neural networks can be integrated into an application, and we chose the kNN classifier since its training time is generally faster. The mobile application includes a navigational menu to view the data as a line graph, to view average values for a day as a color-coded calendar, and to view stress-reducing activities.

4.0 DESIGN IMPLEMENTATION
The design and construction of our project involved three stages of hardware development and simultaneous iterative software development. COVID-19 presented some unforeseen challenges, but we were still able to build a functioning device and accompanying mobile application.

Hardware
In the first stage of the hardware development, we focused on a breadboard implementation, so the sensors were placed on a breadboard and connected to the MSP432 using jumper wires. The goal of the first stage was to develop fully functional driver interfaces for the sensors. The ROS and accelerometer drivers use a repeated start I2C interface that we wrote, and the GSR sensor uses analog-to-digital conversion from an existing library.

For our second stage, we used a Printed Circuit Board (PCB) since I2C connections can be unreliable on a breadboard. We soldered components directly on the PCB to reduce device size and increase portability. As part of this stage, we focused on Bluetooth transmission of data from the embedded system to the mobile application using a Texas Instruments CC2650 Bluetooth module. Every second, the MSP432 microcontroller would communicate values read from the sensors to the CC2650, which would send packets to the mobile application specifying the characteristic ID and this new data.

Soon after our data transmission was successful, we had to suddenly change our plans due to the pandemic since labs on campus were shut down. Completing our PCB required liquid solder which we did not have access to, and components were spread out across team members who were no longer in the same city. We decided to focus on a functional, albeit not as portable, device to demonstrate the overall purpose of our device.

Our third stage of development involved building a new PCB that used headers to place components on the PCB and used a jumper wire connection for the GSR sensor. Since our implementation used a PCB, we had reliable I2C connections. Due to components being in different locations, we could no longer use Bluetooth transmission and instead compiled sensor data into a CSV file that would be imported by the mobile application.

Software

To classify the data received, we implemented and integrated a kNN classifier into an iOS mobile application. This model was first built in Python and then integrated using Apple's Core ML toolkit. For training, we used a publicly available stress dataset. As part of the mobile application, we also implemented five tabs, which include options to view the data as a line graph and color-coded calendar, and options for stress-reducing activities. The libraries we used to implement these parts are the Charts library, the JTCalendar library, the SwiftCSV library, and an open-source codebase on GitHub by Oleh Zayats.

5.0 TESTING AND EVALUATION

To assess the success of our design, we conducted extensive acceptance, integration, and system testing, focused on individual components, multiple subsystems, and the entire device respectively. The testing was incremental and was done continuously as we implemented new features.

Acceptance Testing

To test the GSR sensor, one team member wore the finger gloves that are part of the sensor and performed breathing exercises to view the output of the sensor as a graph. While breathing normally, the GSR output was a periodic signal similar to a sine wave and while holding the breath, the GSR output showed a rising signal. The graphical output of the GSR sensor corresponded to the user's input breathing, so we concluded that the sensor was functioning correctly.

To test the ROS component, we compared the sensor value with manual heart rate readings. The manual readings were taken by a team member who placed two fingers on their neck and counted the number of pulses from the carotid artery for one minute. The beats per minute were compared with the sensor readings and this test was repeated nine additional times. We averaged the results and found the sensor value was within 5% of the measured rate. This test showed that the ROS properly represented heart rate.

Our tests to determine the functionality of the accelerometer involved moving the sensor in the x, y, and z directions to verify a change in the x, y, and z positions respectively. Once we confirmed that the coordinate changes corresponded to the sensor movement, we concluded that all of our sensors worked as desired.

To verify the correctness of our machine learning model, we plotted graphs of the accuracy and Area Under the Curve (AUC) in a Python notebook. For a k-value equivalent to 8, we found the accuracy for the kNN classifier was 71% and the AUC was 77%, which showed that the kNN classifier performed well compared to other models. To test the overall functionality of the mobile application, we designed unit tests to check individual components. These unit tests were

created to confirm our application met our functional requirements. The unit test feature on Xcode allowed us to track timing performance and record button inputs to verify functionality.

Integration Testing
For our integration testing, we combined our hardware and embedded subsystem to check that all the sensors, when connected to the embedded system at the same time, were outputting the correct data. We viewed this data on a PuTTY terminal window and saw that data was collected every second. This data was exported to a CSV file and sent to the mobile application. Next, we combined our embedded subsystem and software subsystem to ensure that the data collected as a CSV file could properly be read by the mobile application and shown as a line graph.

System Testing
After combining the hardware and software deliverables, we tested the entire flow of data from collection to output on the mobile application to verify overall system functionality. We collected data from the sensors for an extended period of time, printed this data to a UART console, and compiled it into a CSV file. The file was sent to another team member who imported it into the mobile application. The data from the file was run through our machine learning model and we viewed the outputs on the Graphs page of the mobile application.

Ideally, we would have completed more testing in environments of varying stress levels to further verify the functionality of the device. We would have also extended testing to all the team members to see how data and results differ from person to person. Despite the challenges we faced, we were able to confirm that all the components of the device worked as desired both separately and when integrated together in the final system.

6.0 TIME AND COST CONSIDERATIONS

Over the course of two semesters, our team encountered shipping and development delays that slowed down our progress. In terms of cost, we remained within the budget for device design, construction, and testing.

The delay in receiving the PCB from Hong Kong had the most impact on our project. International shipping was affected due to COVID-19 and the PCB was delivered three weeks after the expected date. Soon after, campus labs were shut down, so we had to redesign our PCB and order a new version. In addition, our sensor drivers were more complicated to write than we anticipated. Contrary to what we thought, the existing Texas Instruments library did not include routines necessary for our sensor interfaces.

Our team met budget constraints well and we spent $327.10 out of our $1000.00 budget for total development. The assembly of one device unit costs $91.46, and the most expensive part of this is the $23.59 MSP432 launchpad. If we replaced the launchpad with the MSP432 chip as we had intended in our second stage of hardware development, the cost of this part would be $6.27. Our additional expenses included spare parts and testing equipment.

7.0 SAFETY AND ETHICAL ASPECTS OF DESIGN

Since our device is a wearable that includes the collection and transmission of private medical data, we designed our system such that users are not endangered in any way. Human testing has

many regulations, so all our team members completed the CITI Human Research Certification course. To mitigate the risk of injury from the PCB, we designed a Computer Aided Design (CAD) model for a case with beveled edges to hold the PCB and provide a layer of protection. Since our device outputs the medical data collected from the sensors into a CSV file, it is not vulnerable to hacks presented on wireless networks.

8.0 RECOMMENDATIONS

Given the chance to alter some of our implementation decisions and given more time, we would modify and add components to increase the accuracy, portability, security and accessibility of our project. For convenience, we would use the interrupt capability of the MMA8451 accelerometer so that an interrupt would be triggered on excessive movement instead of the accelerometer reporting X, Y, and Z coordinates to the MSP432. In future development, we recommend adding blood pressure and heart rate variability measurement to increase the accuracy of our machine learning model. Both of these can be measured using the ROS in our device. We also suggest transmitting sensor data using Bluetooth communication to increase the portability of the device. This is a viable option since we had it working in the second stage of our hardware development. Another step in increasing the portability of the design is soldering all components to the PCB so that it can be sewn into a glove and placed on the wrist. We have developed the prototype required for this, so completing it is a natural extension to our project. To improve the security of the device, we recommend implementing data encryption, which can be done using the AES hardware accelerometer included in the MSP432 microcontroller. With this approach, all the medical data being sent from the device to the mobile application will be securely encrypted instead of in clear text. Finally, to increase the accessibility of our project, we would develop an Android application based on our iOS application blueprint.

9.0 CONCLUSION

Over the course of two semesters, our team has worked on providing a method for students to monitor their stress levels through a network of sensors and an accompanying mobile application. To ensure that the device causes minimal disturbance to everyday life, we designed a portable glove that contains the PCB enclosed in a case so it is convenient for users to use the device. We hope that with data recorded over time and stored in the mobile application, users will be able to view the effectiveness and impact of stress-reducing activities such as mindfulness and meditation. With this device, our goal is to increase self-awareness so students perform better both academically and outside school.

1.0 INTRODUCTION

College students regularly experience stress but do not have a comprehensive way to track and reduce it. High stress levels have a negative impact on emotional and physical health, resulting in feelings of frustration and difficulty sleeping [1]. Our senior design project looks to provide a unique solution to stress management in college students by building a self-monitoring stress detection device with an accompanying mobile application.

Our device uses a network of sensors that measure heart rate and skin conductance, which are physiological indicators of stress. These sensors cause minimal interference to the user's everyday life by taking measurements from the wrist and fingers. We use the MSP432 microcontroller to collect data from the sensors. Data gathered by the microcontroller is transmitted to a mobile application, run through a kNN classifier, and plotted as a line graph on the mobile application for the user to easily interpret. Users are able to monitor their stress levels on the mobile application, which records and displays data over time. This enables comparison of the effectiveness of various stress-reducing activities such as mindfulness and meditation.

This document details our work on the design, construction, and testing of the device. Our development process involved setting the requirements according to the design problem, devising a solution to address the problem, implementing the solution, and testing the individual components as well as the entire system. We also highlight the time, cost, and safety considerations that factored into this process. Since stress management is a prevalent issue, we believe our device will continue to be useful and have included recommendations to add to and improve the existing design.

2.0 DESIGN PROBLEM

Stress can contribute to mental health issues, even if people do not realize they are experiencing stress. According to the American College Health Association Fall 2018 National College Health Assessment, 63% of college students in the United States experienced overwhelming stress during the school year [2]. With changes in lifestyle, increased workloads, and new responsibilities, college students experience high levels of stress that cause feelings of frustration, difficulty relaxing and sleeping, and physical pain. Unfortunately, long-term stress is

a risk factor for cardiovascular disease, depression, anxiety, and gastrointestinal problems. Although universities offer mental health counseling, there still are not enough resources to meet the needs of all students [3]. In our senior design project, we constructed a device to help students track and monitor their stress levels. The purpose of our device is to help users detect stress so they can be aware of it and actively try to reduce it in order to minimize the risk associated with acute stress.

To ensure that our device will fulfill the aforementioned purpose, our functional requirements include a network of sensors that measure multiple physiological indicators of stress. The outputs of the sensors should be extracted to make a single decision-based assessment of the stress level. Additionally, there should be a way for a user to record and monitor stress levels over time so users can see their changes over long periods of time. As part of our specifications, we required our device to be small and non-intrusive in order to cause the least disturbance to a user's everyday life. The budget allocated to the design, construction, and testing of the device was $1000.

3.0 DESIGN SOLUTION

Our device measures Galvanic Skin Response (GSR), heart rate, and motion using several sensors and sends this information to an embedded processor. The embedded system outputs the data to a serial console where it is then saved as a Comma Separated Value (CSV) file. The smartphone application imports the CSV file, runs the data points through a kNN classification model, and plots the results as a line graph so it can be easily interpreted by the user. Users can monitor their stress levels through the mobile application, which records and displays data over time. Average stress levels for one day will be reported using a color-coded calendar. This enables comparison of the effectiveness of various stress-reducing activities such as meditation or mindfulness. The sensors, embedded system, and mobile application shown in Figure 1 on the next page will be discussed in detail in this section.

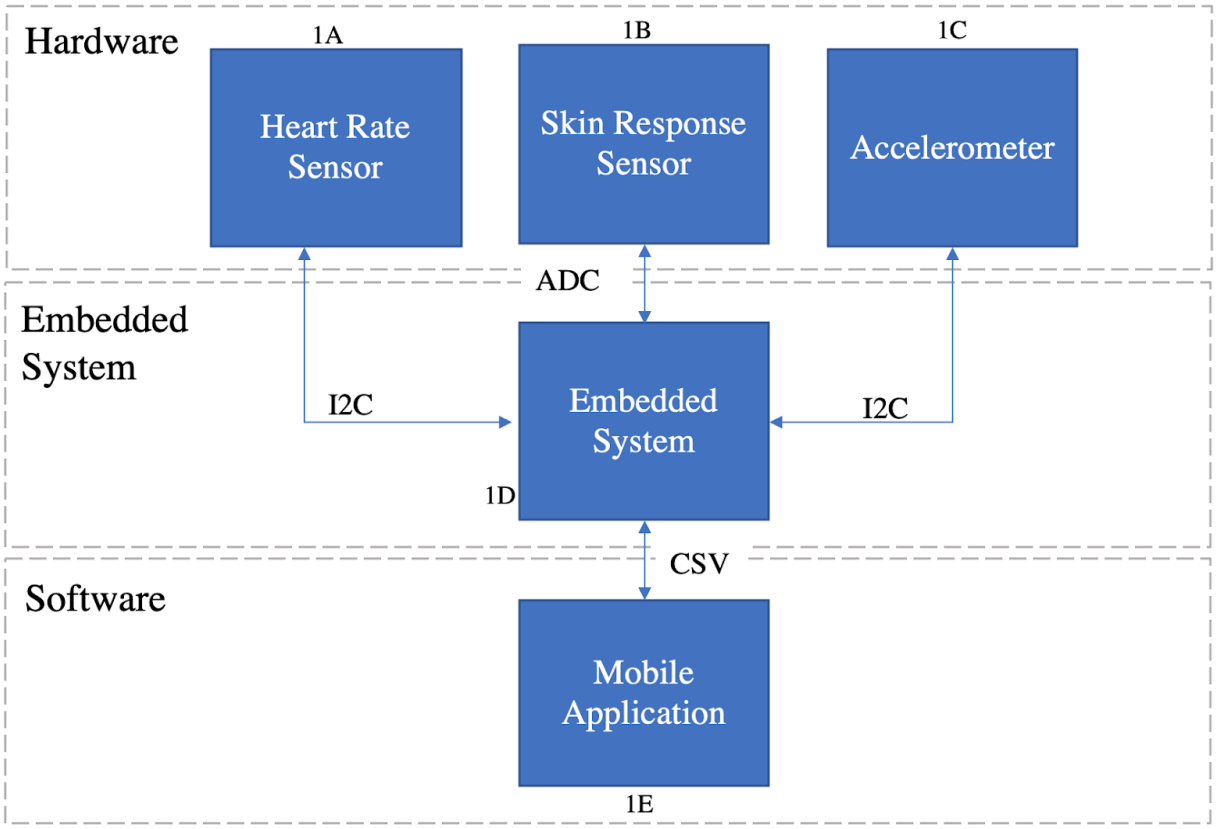


Figure 1. Block Diagram of Overall System

3.1 Subsystems

As shown in Figure 1, our project is composed of three subsystems: hardware, embedded system, and software. The hardware consists of a system of sensors to measure GSR and heart rate as these are physiological parameters correlated to stress. Additionally, the sensor system contains an accelerometer to provide movement data to the mobile application. The motion sensor is a necessary component since the GSR and heart rate sensors can provide unreliable data when subject to excessive movement. All the sensors are shown on Figure 1 as components 1A - 1C. The second subsystem is the embedded system which collects data from the sensors through communication protocols such as I2C and analog voltages. The embedded system, labeled as 1D, and all the communication protocols are shown in Figure 1. Our third subsystem consists of a mobile application that receives data from the embedded system via a CSV file and classifies the user's stress level. The mobile application is shown as 1E in Figure 1.

3.1.1 Hardware

The network of sensors encompasses three components: a GSR sensor, a Reflective Optical Sensor (ROS), and an accelerometer. The GSR used in our device detects increased skin conductance and the ROS measures SpO2 levels. Increased skin conductance and decreased SpO2 are physiological signals that indicate acute stress. To detect excessive motion that may disturb the sensor readings, an accelerometer is placed on the wrist. These sensors and their input/output interfaces with the embedded system are detailed below in Figure 2.

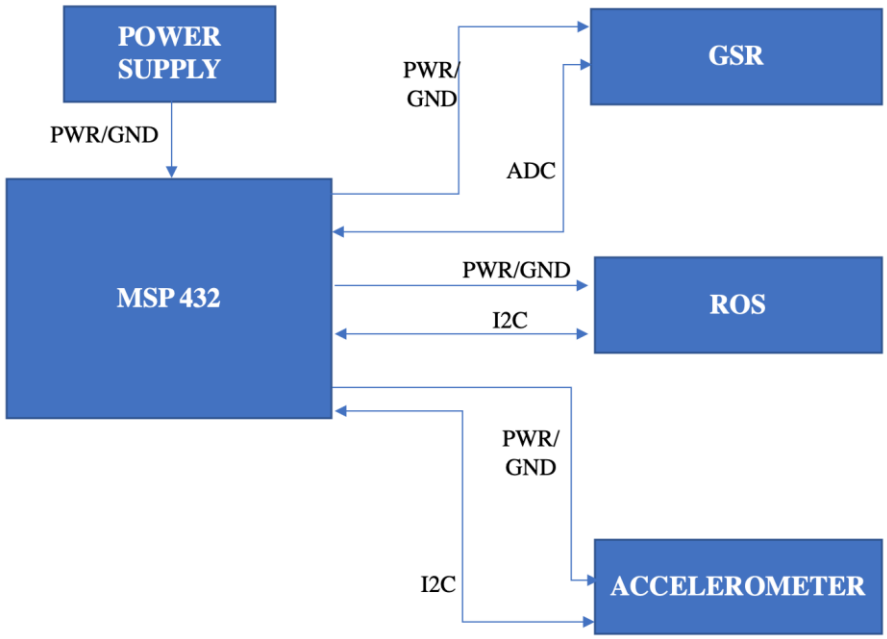


Figure 2. MSP432 I/O Connection Diagram

3.1.2 Embedded System

Our system contains an embedded processor to read sensor values and communicate the data to the mobile application. We chose the MSP432 microcontroller from Texas Instruments, shown in Figure 3, since our team has previous experience with Texas Instruments microcontrollers. Therefore, we were able to leverage our existing knowledge to develop drivers for the sensors. We also chose the MSP432 because it is a low-power, energy efficient processor with useful peripherals such as a high precision Analog-to-Digital Converter (ADC) and I2C interface. The high precision ADC is able to detect very small changes in GSR, which leads to a more accurate

model for stress classification. The I2C interface on the MSP432 allows the embedded system to connect to a widespread range of sensors. We are using the Universal Asynchronous Receiver Transmitter (UART) to transmit the sensor data to a serial console and convert it to a CSV file to send to the mobile application.

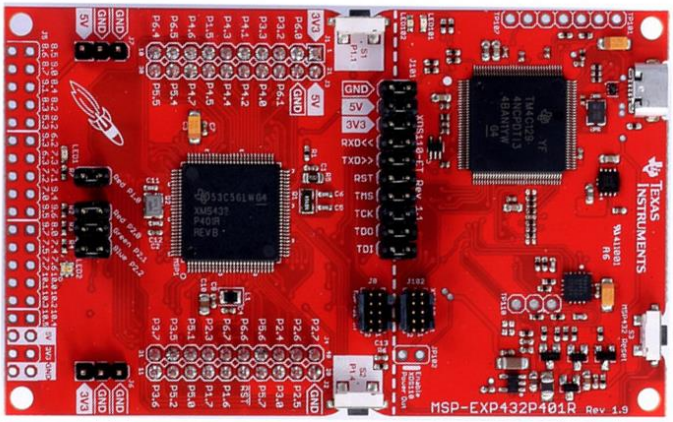


Figure 3. MSP432 Microcontroller

3.1.3 Software

The mobile application, currently developed for the iOS platform, forms the user interface for our project. We created an iOS application since the majority of our team uses iOS devices. Additionally, iOS has more restrictions than other operating systems so our working proof-of-concept can be readily adapted to any other device type. The embedded system sends the data read from the sensors to the mobile application as a CSV file. On the mobile application, we use a machine learning model to classify each data point as stressed or not stressed. The machine learning approach helps make the app personalized (as each person has different biological characteristics) and is retrainable to improve stress detection accuracy. The data read from the sensors is also plotted as a line graph so that it can be easily interpreted by the user. There are stress-reducing activities, such as meditation, yoga, breathing circles, breathing exercises, and calming music, provided in the app. The goal is for students to be able to see how certain activities affect their stress levels. Screenshots of the mobile application can be found in Appendix A.

3.2 Sensor Operation

Our hardware design consists of three sensors: Grove galvanic skin response sensor, MAX30102 reflective optical sensor, and MMA8451 triple axis accelerometer.

We chose the Grove GSR sensor, shown below in Figure 4, to measure skin conductance because it is cost-effective and widely used in wearable devices. The Grove GSR sensor works by applying a voltage across a voltage divider. The divider consists of a tunable reference resistor nominally set to 160kΩ and two electrodes connected to the user's fingertips. The resistance across the fingers will change as the user becomes more or less stressed, which in turn will modify the voltage output of the divider. The output is run through a low and high pass filter on the sensor before appearing as a voltage that can be read by an external device. The MSP432 microcontroller uses a successive approximation analog-to-digital converter to read in the voltage with 14 bits of precision and store the value in a software variable. This variable is then printed out to the screen along with heart rate and acceleration.

Figure 4. Grove Galvanic Skin Response Sensor

We selected the MAX30102 reflective optical sensor developed by Maxim Integrated, shown in Figure 5, to measure heart rate because it is cost-effective, accurate, and well-documented. The ROS works by shining an infrared Light Emitting Diode (LED) into the finger or wrist and measuring the reflected light on a photodiode. The amount of reflected light changes due to oxygen saturation in the blood vessels in the major arteries. Oxygen content increases immediately following a heartbeat and decreases after. The ROS records the amount of reflected light through the photodiode and stores the values in a queue. The values in the queue can be accessed via an I2C read access. Our MSP432 microcontroller reads the values of reflected light

from the ROS and then interprets the values to determine if there is a heartbeat. The routine scans through the reflected light values and if there is a pattern of increasing infrared light followed by a decrease, the software marks it as a heartbeat and begins to look for the next beat.

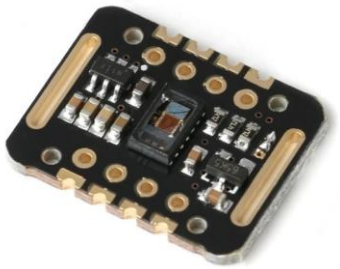


Figure 5. MAX30102 Reflective Optical Sensor

We selected the MMA8451 accelerometer, shown below in Figure 6, since it is affordable and uses the same I2C protocol as the reflective optical sensor. The MMA8451 accelerometer uses a triple-access pressure sensor to record pressure in the X, Y, and Z positions. The pressure values are converted to a 14-bit value corresponding to the degree of pressure, and then stored in a hardware queue. This hardware queue is readable through an I2C read access. Our MSP432 launchpad performs a 6-byte read to obtain the X, Y, and Z coordinates from the accelerometer and then scales them to conform to the standard meters per second (m/s) format in fixed point notation. These directional values are then printed to the UART.

Figure 6. MMA8451 Accelerometer

3.3 Model Operation

We chose to develop an iOS mobile application to display the data from the sensors. Currently, iOS supports only two types of machine learning models for on-device training: kNN classifiers and neural networks. All other models have to be trained before their integration into the

application and cannot be re-trained or changed once they are in the application. As a result, only kNN classifiers and neural networks can be trained on the application to give personalized predictions. We chose to use the kNN classifier since its training time is fast and it requires minimal effort from the user to re-train the model. A kNN classifier makes decisions based on the "k" nearest neighbors for a sample. The sample consists of the physiological parameters of stress that are measured by the device. The class of the input data sample will be the most frequent class in the "k" nearest neighbors. The proximity of a sample to other samples is calculated using the Euclidean distance between two points, as shown below in Equation 1,

$$\textit{Distance between } x_a \textit{ and } x_b = \sqrt{((x_{a1} - x_{b1})^2 + (x_{a2} - x_{b2})^2 + \ldots + (x_{an} - x_{bn})^2)}, \quad (1)$$

where x_a and x_b are the two samples. For a given sample, once we have the distance between that sample and all other samples, we can easily find the "k" closest ones. The output defines the class of the sample as "stressed" or "not stressed". The aggregated results are then displayed on the mobile application where the user can view the results of the model.

4.0 DESIGN IMPLEMENTATION

We methodically began to implement building our device with three stages of hardware development followed by iterative software development. Despite the unforeseen circumstances of COVID-19, we were able to adjust our design to successfully build a functioning device, model, and mobile application.

4.1 Hardware

In the first stage of development, we placed the ROS and accelerometer on a breadboard and ran wires from the MSP432 launchpad pins to the sensors, as shown in Figure 7. We wanted to create a proof-of-concept and develop driver interfaces between the sensors and microcontroller, so we relied on the breadboard implementation. The base driver for the MSP432 did not accommodate the repeated-start method of I2C, which is why we developed our own driver with an abstracted interface to read and write to the I2C peripherals. Using an analog-to-digital conversion in an existing MSP432 driver library, we could quickly read values from the GSR.

Completing the breadboard implementation enabled us to have functional drivers for all our sensors.

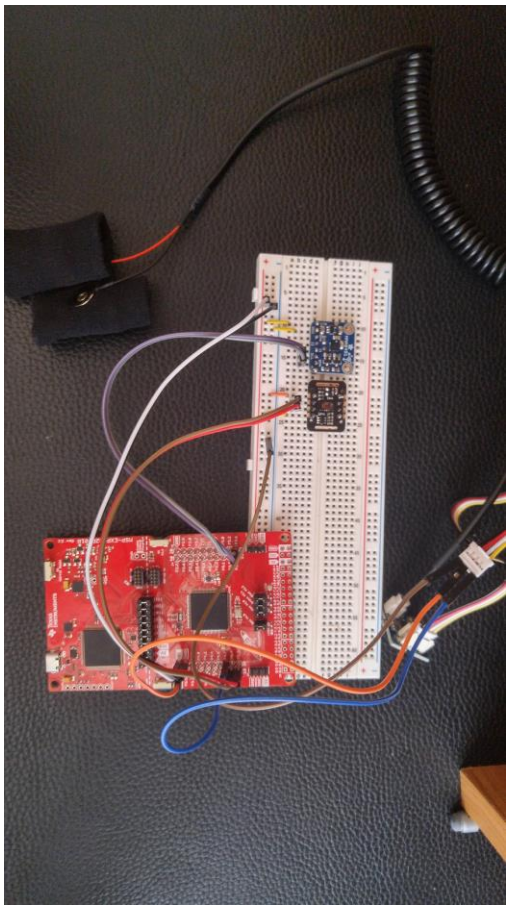


Figure 7. First-Stage Breadboard Implementation

Our second stage of development used a Printed Circuit Board (PCB) with the MSP432 chip and all relevant components placed directly on it, shown in Figure 8. I2C connections can be unreliable due to excess capacitance on a breadboard, and thus prevents long periods of data collection from the ROS and accelerometer. Therefore, we shifted our design to a PCB configuration which enables more stable connections by using copper traces rather than jumper wires. We powered the PCB with a rechargeable battery. Additionally, using a PCB significantly reduced the size of our design and made it small enough to place on the wrist. Since we already confirmed that the sensors could read data and send it to the microcontroller from the breadboard implementation, we wanted to focus on transmitting the data to the mobile app in our second prototype. For wireless communication, we decided to explore Bluetooth using a Texas

Instruments CC2650 Bluetooth module soldered to the PCB. First, we connected the CC2650 to our mobile application using Bluetooth. Every second, the MSP432 microcontroller would read the values from the sensors and communicate their new values to the CC2650. The CC2650 would then send packets to the mobile application specifying the characteristic ID and the new value. With this approach, the sensor values are updated automatically from the view of the application.

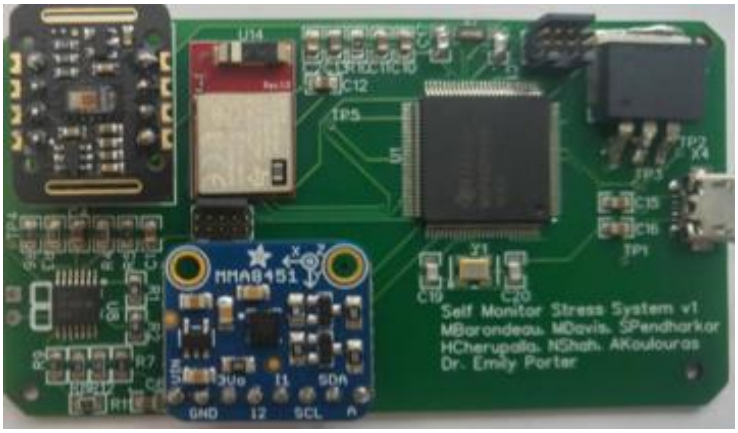


Figure 8. Second-Stage PCB Implementation

Once our device data collection and data transmission were successful, we were abruptly forced to change our plans due to the pandemic. Once the labs on campus shut down, we could no longer use the reflow oven required to solder the oscillator and other essential components onto the PCB. We also could not use the 3D printers to print the PCB encasing that would be sewn into the wearable glove. Additionally, our team members returned to homes across the country, so many components were divided among team members and we could not complete testing together. For example, the rechargeable battery was left in Austin and not with the teammate who left campus. We also could no longer pursue Bluetooth communication as our data transmission method. As a result, we decided to stop developing our second-stage PCB and shift towards a practical third-stage design that could still demonstrate the overall functionality of our device.

Our third and final stage of development consists of a new PCB combining design elements from the breadboard implementation and the second-stage PCB. Similar to the breadboard implementation, our final design uses an MSP432 launchpad connected directly to the I2C

devices and a jumper wire connection for the GSR. Similar to our second-stage PCB, our final design's PCB traces for the I2C sensors solved the reliability problem. Since we lost access to the reflow oven, we used headers to place components easily without requiring soldering, shown in Figure 9. We connected the MSP432 launchpad to the PCB using the headers as depicted in Figure 10, which made our overall design too large to be placed comfortably on the wrist. However, our team decided to prioritize device functionality over device aesthetic. With this final PCB design, we can reliably collect data from the sensors and compile the data points into a CSV file to be transmitted to the mobile application. Completing the final hardware configuration allowed us to begin developing the model built into the mobile application.

Figure 9. Headers on Third-Stage PCB

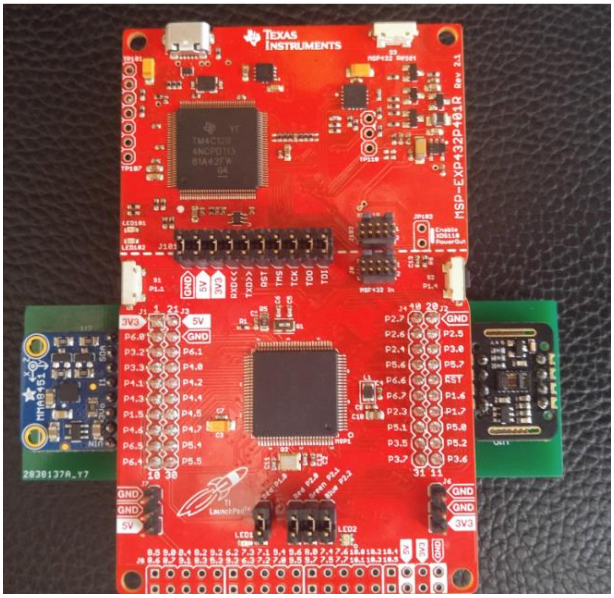


Figure 10. Third-Stage PCB Implementation

4.2 Software

The user interface for our project is an iOS mobile application which receives data from the embedded system in the form of a CSV file. On the mobile application, we use a machine learning model to classify each data point as “stressed” or “not stressed” with a kNN classifier. We implemented our kNN classifier using samples consisting of four features: heart rate, electrocardiogram, skin response, and electromyography. The sample is associated with one output specifying the class of the sample. The model implementation process is shown in Figure 11. The model was first built in Python and integrated into the mobile application using Apple’s CoreML toolkit. For training, we used a publicly available stress dataset used in a PhD thesis [4]. With the classifier integrated into the app, training can be done infinitely many times on the device.

Figure 11. Model Implementation Process

In addition to classifying the data on the mobile device, we implemented five tabs on the mobile application, as shown in Figure 12. The data read from the sensors is plotted as a line graph and also shown as a color-coded calendar, so it is in a more readable format than a CSV file. To plot the graphs, we used the Charts library, to draw the calendar we used the JTCalendar library, and to manage CSV files we used the SwiftCSV library. For the stress-reducing activities, we created buttons that link to various external websites such as calming music and guided meditation. We used an open-source codebase on GitHub by Oleh Zayats to implement a breathing exercise where a circle has an increasing and decreasing radius for the user to synchronize their breathing with (source code available at https://github.com/oleh-zayats/BreatheView).

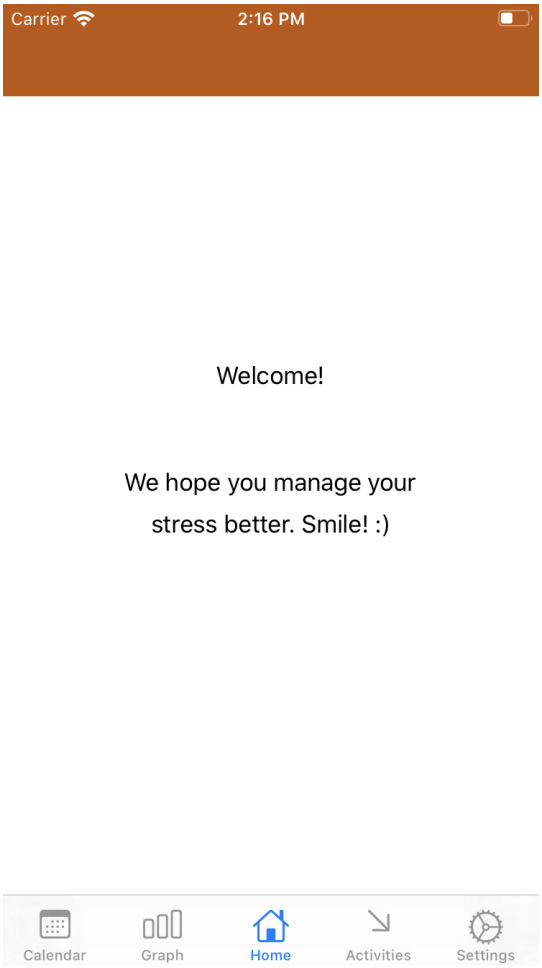


Figure 12. Five Tab Navigation Pane on Mobile Application

5.0 TEST AND EVALUATION

To assess the success of our design, we conducted tests on both the hardware and software sections of our project. We first ran acceptance tests on the power system, battery life, GSR, ROS, accelerometer, model, and mobile application. When all the components successfully met our expectations, we combined the subsystems and ran integration tests. Lastly, we ran system tests to verify functionality of the final system. Despite limitations in testing caused by the COVID-19 pandemic, which forced us to modify our testing strategies to accommodate the challenges of components no longer being shared amongst our team, the tests presented here fully demonstrate functionality of all aspects of the project.

5.1 Acceptance Testing

To verify that our individual components behaved as expected, we first completed acceptance testing for our power system, sensors, model, and mobile application. Once the components worked accurately and successfully passed our tests, we put them together to build our system.

5.1.1 Hardware Tests

As explained in Section 4.0, the second version of our device design used a rechargeable battery to power the system. To test this power system, we input common voltages into the voltage regulator circuit and observed the output waveform on an oscilloscope. The typical voltages tested included 3.7V (common LiPo battery), 5V (common USB input), and 7.4V (two LiPo batteries in series). To power our device, our system requires a minimum voltage of 3.3V. We first tested the 3.7V input and realized that since it only produced an output of 2.66V, as shown in Figure 13, it would be unable to meet our needs. Then, we increased the input voltage to 5V and observed the steady 3.3V output as shown in Figure 14. From the tests conducted, we determined that in order to receive a steady output of 3.3V, we must input at least 5V. However, we could not find a rechargeable 5V battery available in the market. Therefore, we decided to use two 3.7V LiPo batteries in series, resulting in an input supply of 7.4, since one would not supply an adequate amount of power. With this configuration, we tested an input supply of 7.4V into the voltage regulator and observed the 3.3V output, as shown in Figure 15. This result passed our tests for the power system and allowed us to move on to test battery life.

Figure 13. Regulator for 3.7V Input for 1 LiPo Cell, 2.66V Output

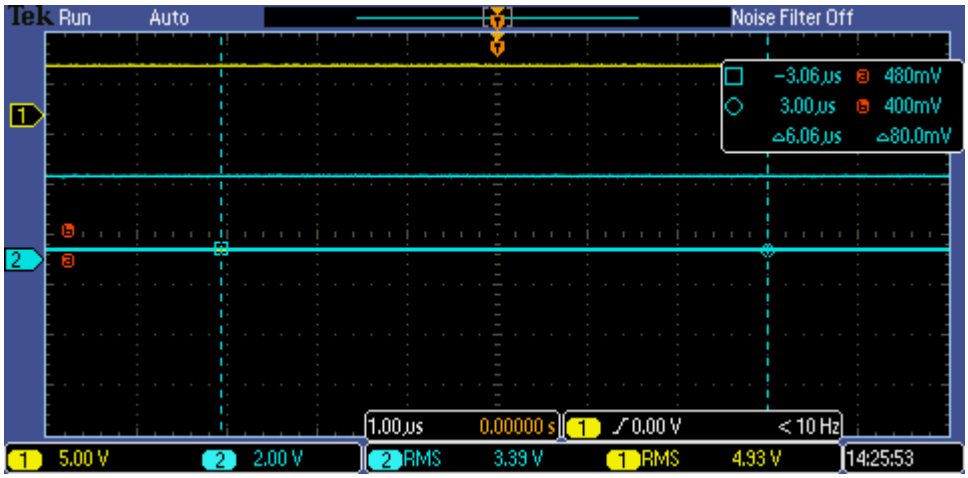


Figure 14. Regulator for 5V Input, 3.3V Output

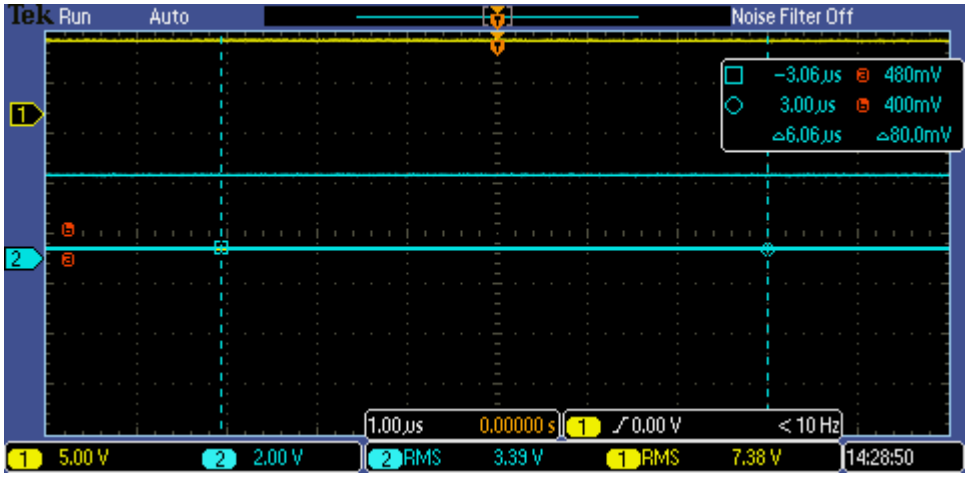


Figure 15. Regulator for 7.4V Input for 2 LiPo Cells, 3.3V Output

Based on our voltage regulator test, we decided to use two LiPo batteries in series as our voltage supply. To ensure the battery charging and discharging behavior met our expectations, we completed calculations for expected battery life. To compute battery life, we used the LiPo battery datasheet and component specifications detailed in Appendix B. With a battery capacity of 1200 milliamp-hours [mAh], MSP432 peak current draw of 12 milliamps [mA], and total sensor current draw of 7 milliamps [mA], we can use Equation 2 given below,

$$\textit{Battery Life [h]} = \textit{(Battery Capacity [mAh])/(Load Current [mA])}, \qquad (2)$$

to calculate the battery life to be 63 hours if our device ran on peak performance at all times.

This battery life exceeds the common daily usage metric of 8 hours, so the battery meets our expectations.

To test the GSR sensor, one team member wore the finger gloves and experimented with various breathing rates to observe the GSR's output as a graph. The finger gloves were worn on the ring and middle fingers, as shown in Figure 16. The breathing rates tested include breathing normally for 40 seconds, holding the breath for 20 seconds, and taking deep breaths for 40 seconds. While breathing normally, the GSR output resulted in a periodic signal mimicking a sine wave. While holding the breath, the GSR output showed a rising signal, as shown in Figure 17. During deep breathing, the GSR graph showed a sine waveform with a larger period than normal breathing. Since the graphical output of the GSR sensor corresponded to the user's input breathing, we concluded the GSR functioned accordingly and met our expectations.

Figure 16. GSR Test Setup

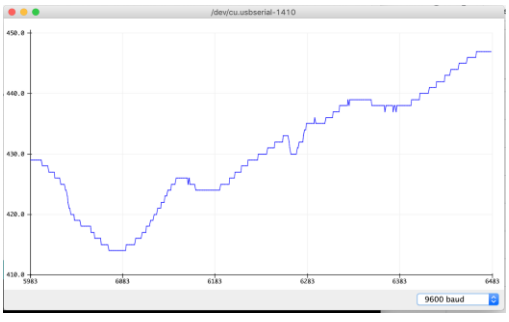


Figure 17. GSR Sensor Output for Holding Breath Test

To test the ROS component, we compared the readings for heart rate manually and from the sensor for ten cases. To manually check heart pulses, one member placed two fingers on their neck and counted the number of pulses from the carotid artery for one minute. The number of beats per minute counted were then compared to the results of the fingertip measurement from the ROS device. This experiment was repeated nine additional times. In the ten cases, the carotid-measured heart rate ranged from 72 to 80 beats per minute. Similarly, the ROS-calculated heart rate ranged from 74 to 84 beats per minute. To confirm the accuracy of the sensor results, we averaged the heart rate values from the ten cases and determined that the sensor value was within 5% of the measured rate. This rate was within the acceptable range, so we determined that the ROS worked accurately and properly represented the heart rate. In the future, we would improve our testing method by verifying the heart rate with a finger pulse oximeter or an Apple Watch to check real-time readings over a period of time.

To test the accelerometer, we confirmed accurate values for all three directional axes (x, y, z). Through a hands-on test, as shown in Figure 18, we verified that the motion sensor responded with values that matched the physical behavior. To test the x-axis, we moved the sensor from the left to the right and verified an increase in x-axis readings. Then we moved the sensor from backwards to forwards and verified an increase in y-axis readings. Finally, we moved the sensor from bottom to top and verified an increase in z-axis readings. After these three basic tests, we moved the sensor in random order and verified that all the axes values had changed, shown in Figure 19. Once we confirmed that the changes in the accelerometer values corresponded with the behavior, we concluded that the accelerometer worked as expected.

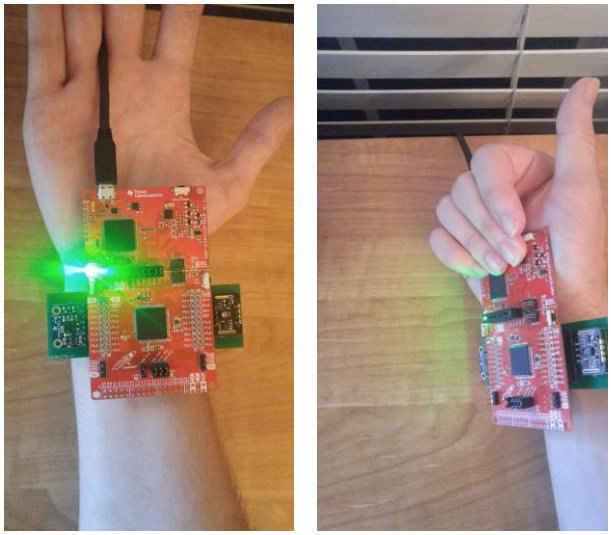


Figure 18. Hands-On Accelerometer Test

```
X:      -1026   Y:      168     Z:      3768
X:      -2.57   Y:      0.41    Z:      9.15    m/s^2
Portrait Up Front

X:      -2580   Y:      368     Z:      3136
X:      -6.12   Y:      0.79    Z:      7.79    m/s^2
Landscape Left Front

X:      -2614   Y:      -2474   Z:      3522
X:      -0.11   Y:      -16.75  Z:      -8.94   m/s^2
Landscape Left Front

X:      -286    Y:      -3578   Z:      1846
X:      -0.57   Y:      -8.77   Z:      4.44    m/s^2
Portrait Up Front

X:      68      Y:      328     Z:      4062
X:      0.17    Y:      0.75    Z:      9.72    m/s^2
Portrait Down Front
```

Figure 19. Accelerometer Test Results

5.1.2 Software Tests

The model was trained using features selected from a publicly available stress dataset [4]. After this, to verify the correctness of the model, we plotted graphs of the accuracy and Area Under the Curve (AUC) in a Python notebook. Accuracy is defined as the ratio of the points we classified correctly over all the points that were classified. The AUC is the area under the Receiver Operating Characteristic (ROC) curve. An ROC curve is the plot of the true positive rate against

the false positive rate for different classification thresholds. With the number of neighbors k equal to 8 (common design choice), we saw that the accuracy for the kNN classifier was 71% and the AUC was 77%, which showed that the kNN classifier performed well. To compare these values with other machine learning models, we used the scikit-learn library in Python to see the performance of logistic regression, random forest, XGBoost, and decision tree models on the stress dataset. We also used the Keras library to implement a neural network with a single hidden layer and sigmoid activation. As seen from Table 1 given below, the XGBoost model performs best on the dataset, but it cannot be trained on the device. The neural network, which can be trained on the device, does not perform better than the kNN classifier. Since iOS only allows us to train kNN classifiers and neural networks on the device, we decided to use the kNN classifier based on the results of these tests shown in Table 1.

Table 1. Model Test Results

Model	**Accuracy**	**AUC**
kNN (k=8)	0.71	0.78
Logistic Regression	0.62	0.70
Decision Tree	0.76	0.83
Random Forest	0.79	0.87
XGBoost	0.82	0.88
Neural Network (Keras)	0.69	0.77

To test the functionality of the mobile application, we designed unit tests to check different individual components. The unit tests, listed in Appendix C, were created based on our predetermined app specifications and the desired app functionality. Then, we compared all the capabilities of our app and wrote a test to confirm that each component was behaving as

expected. To verify these tests, we used the simulator available on Xcode. The unit test feature on Xcode allowed us to track performance, such as the time for a page to load, and record button inputs to verify functionality. This feature also allowed us to automate the test cases. If the proper page loaded or button press corresponded with the correct behavior, we counted a successful test. After using the simulator to verify success, we double-checked the results on an iPhone to make sure the app worked as expected on a real mobile device.

5.2 Integration Testing

We combined our hardware and embedded subsystems to ensure that all sensors were outputting the correct data. First, we checked that all three sensors could send data to the MSP432 microcontroller individually. Then, we ensured that the MSP432 could capture all the sensor values at the same time. Using UART, we viewed the sensor data on a PuTTY terminal window. This is shown below in Figure 20, where each row has the heart rate, X coordinate, Y coordinate, Z coordinate, and GSR value. A set of values is collected every second. After collecting data for a specified amount of time, the values are exported to a CSV file.

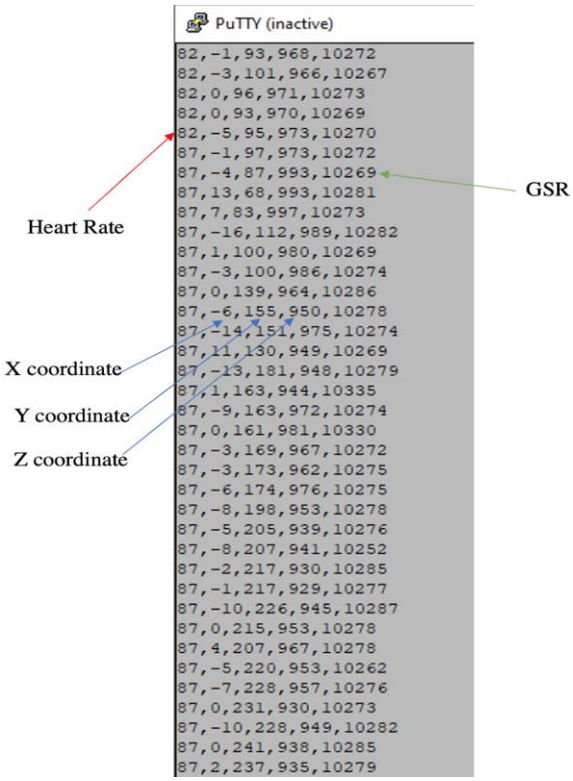


Figure 20. Serial Console Output

Next, we combined our embedded and software subsystems to ensure that the software system could import the CSV file containing values collected by the embedded system. This process also involved running the data through the machine learning model and viewing the results as a graph on our mobile application, as shown below in Figure 21. We made sure the graph could display the heart rate, GSR, and accelerometer data correctly.

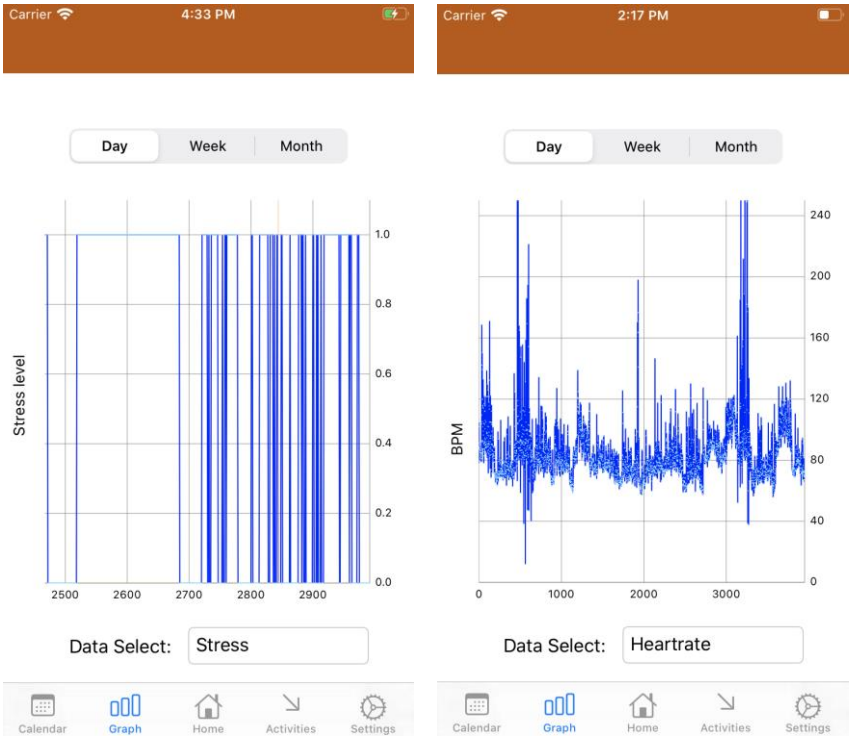


Figure 21. Graph Data

5.3 System Testing

To test the full system, we combined the software and hardware deliverables. We linked the UART console output which listed the GSR, ROS, and accelerometer data values to a CSV file. This file contained all the sensor output data over a given period of time. This file was sent to another team member who imported the file into the mobile application. From here, a team member ran the data through the model and observed the output results on the Graphs page of the mobile application.

Ideally, we could have improved our final testing method by transmitting data in real-time via Bluetooth communication, which was our original plan. If we had time for improvements, we

would have routed the GSR, ROS, and accelerometer data to the mobile application over a secure Bluetooth link. After the smartphone received the data, we would have verified that the model accurately classified the data. Finally, we would have displayed the results graphically in the mobile application and checked that the data was represented accurately. This process would have been done in various environments of varying stress levels to verify the functionality of the device. Despite the challenges that we faced, through the testing procedures, we were able to confirm that all implemented elements of both the hardware and software designs are working as planned and meet the design constraints.

6.0 TIME AND COST CONSIDERATIONS

Over the course of EE364 and EE464, our team encountered several timing issues that slowed our progress such as shipping delays and developing delays. In terms of cost, we remained within the budget for our device design, construction, and testing phases.

The shipping delay that had the most impact on our project was the PCB ordered from JLC PCB in Hong Kong. We decided to order from this manufacturer because they had our components in stock and could solder for us within an extremely reasonable price. However, international shipping was affected due to COVID-19 and the PCB was delivered three weeks after the expected arrival date. This delay reduced our testing time significantly. While working on this PCB version, the Engineering Education and Research Center's Undergraduate Soldering Lab closed due to the pandemic. Consequently, we could not correctly solder the surface mount components without the reflow oven in the lab. This further delayed our overall progress since we had to completely redesign our PCB and order another version from JLC PCB. For the second PCB order, we paid for priority shipping to ensure we would have time to receive the board and test it well before the due date.

In addition to issues regarding the PCB, the sensor drivers were more complicated to develop than we anticipated. When we selected the MSP432 microcontroller as our system's embedded processor, we expected to use Texas Instrument's existing driver library. However, while working on the drivers for our sensors, we found out that TI's library did not include the routines

necessary for interacting with our sensors. As a result, we spent extra time writing the I2C communication drivers, which was not what we originally planned.

In terms of cost, our team met budget constraints extremely well. In total, we spent $327.10 out of our $1000.00 budget for total development costs (see Appendix D). The assembly of one device unit costs $91.46, as shown in the bill of materials in Table 2. The largest expenditure is the MSP432 Launchpad, which costs $23.59. We believe that this cost could be reduced in the future if we replaced the launchpad with the MSP432 chip, which only costs $6.27. The additional expenses in our budget were spent on spare parts such as extra sensors and testing equipment such as a pulse oximeter.

Table 2. Bill of Materials

Quantity	Description	Part	Unit Cost	Total Cost	Source
1	GSR	Grove GSR Sensor	$14.90	$14.90	Amazon
2	Battery	Lithium Ion Polymer Battery	$9.95	$19.90	Mouser
1	Charger	USB Rechargeable LiPo Battery Charger	$12.50	$12.50	Mouser
1	Glove	Fingerless Glove	$1.37	$1.37	Amazon
1	Circuit	PCB	$2.00	$2.00	JLCPCB
2	Headers	67996-220HLF	$0.83	$1.66	Digikey
1	Launchpad	MSP432EXP432P401R	$23.59	$23.59	Texas Instruments
1	Accelerometer	MMA8451	$6.95	$6.95	Amazon
1	ROS	MAX30102	$8.59	$8.59	Amazon
			TOTAL	$91.46	

7.0 SAFETY AND ETHICAL ASPECTS OF DESIGN

Our device is a wearable that includes the collection and transmission of private medical data. Therefore, we designed our system keeping safety and ethical considerations in mind such that users are not endangered and data is not breached in any way. Our main safety concerns involve human testing and placing electronics near the skin. Our main ethical concerns involve data privacy and security.

Since our wearable is a medical device worn by people, we have to make sure our protocols respect the safety of the subjects that wear the device. Human testing has many regulations in order to protect the participants in research, so all members of our team completed the CITI Human Research Certification course (see Appendix E). For example, the specified guidelines that we reviewed in training included asking for permission prior to collecting data and allowing the participant to refuse participation at any point in the process. However, we could not test the device on multiple people due to the social distancing guidelines imposed by COVID-19.

Another safety concern with our device was the potential for shocks and burns from the PCB if it was shorted when placed directly on skin. To mitigate this risk, we designed a Computer-Aided Design (CAD) model of a case to hold the PCB and provide a layer of protection. This model, shown in Figure 22, included beveled edges to prevent injuries from sharp corners. We intended to 3D print the CAD model and ensure the safety of our device for users by designing packaging that keeps safety as a priority.

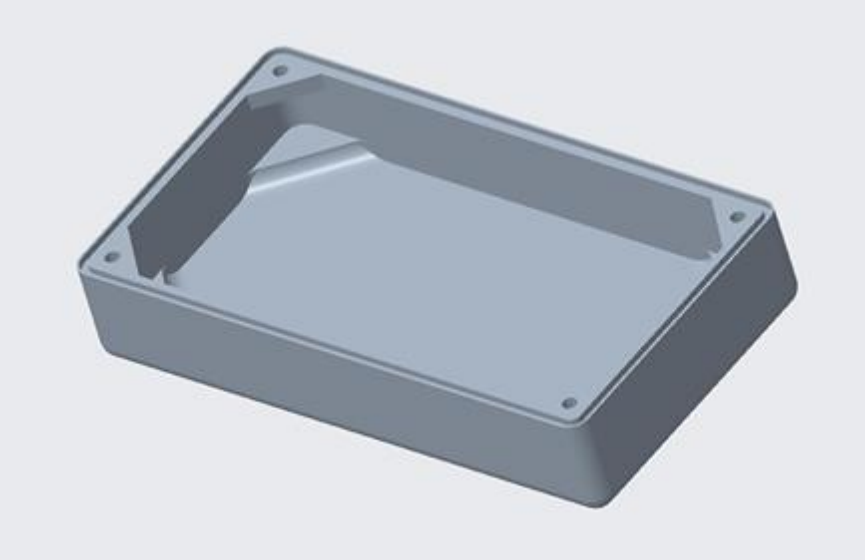


Figure 22. CAD Model for PCB Case with Beveled Edges

Our device collects personal medical data such as heart rate and GSR, which must be kept private to limit mishandling of medical data. Since our device outputs data into a CSV file, it is not vulnerable to hacks presented on wireless networks like WiFi. Furthermore, the CSV file itself does not contain any identifying information about the user. As a result, the data is secure since we communicate anonymous data to the mobile application through a file upload instead of a continuous transmission.

8.0 RECOMMENDATIONS

Given the chance to alter some of our implementation decisions and given more time, we would modify and add components to increase the convenience, accuracy, portability, security, and accessibility of our project.

8.1 Implementation Modification

Looking back, we recommend implementing an alternative configuration for detecting excessive motion using the MMA8451 accelerometer. Currently, the accelerometer calculates the X, Y, and Z coordinates and communicates these values to the microcontrontroller. The MMA8451 has the capability to trigger an interrupt on excessive movement which is more useful for the model than the raw movement values. Rather than comparing the acceleration values to determine if there has been excessive movement, we suggest configuring the interrupts in the accelerometer

to set a flag if there is excessive movement. The device would then report the flag rather than the acceleration values.

8.2 Additional Components

In the future, we suggest improving our device by also monitoring blood pressure and heart rate variability, which are other physiological indicators correlated with stress. Blood pressure and heart rate variability can both be derived from SpO2, which is measured by the reflective optical sensor in our device. By expanding the capabilities of a sensor in our current design, it is a practical next step to measure other physiological indicators so that we can increase the accuracy of our prediction model.

We also recommend incorporating Bluetooth communication into our device. It enables wireless communication between our microcontroller and mobile application. Since we already had Bluetooth working in our second-stage implementation, we believe that it is a viable next step that can be easily implemented in the future. Bluetooth further increases the portability of our device by allowing for a greater distance between the transmitting and receiving devices.
In tandem with bluetooth communication, we suggest implementing data encryption. The MSP432 microcontroller has an AES hardware accelerator included, which can be used to quickly and securely encrypt data being transmitted to the mobile application. The key for the encryption can be set by the application, so each user can encrypt their data using a separate key. The advantage to this approach is that all medical data being sent from the device to the mobile application will have security encryption, so that it will not be easily hacked. In addition, even if the wrong phone connects to the device, it will not be able to access this data. Encrypting the data would increase the security of the user's medical data without affecting performance.

Packaging could be improved by soldering all the sensors and components to the PCB so that the size of the device is small enough to be placed on the wrist. We began this process during our first two stages of implementation, so it would be straightforward to finish the process with access to the labs. We also already designed and prototyped a non-functioning version of our final packaging as a glove shown in Figure 23. Combining a working PCB with the glove would be an intuitive development in extending the project.

Figure 23. Glove Prototype

Finally, to increase the accessibility of our mobile application, we would develop an Android application for our project. We currently have a functioning iOS application, so we have the blueprint for the Android application. Creating an Android application would expand the accessibility of our interface to a wider user base.

9.0 CONCLUSION

Over the course of two semesters, our team has worked on providing a method for students to monitor their stress levels through a network of sensors and an accompanying mobile application. We have integrated heart rate and galvanic skin response sensors in our device to detect increased heart rate and increased skin conductance, which are indicative of high stress levels. The iOS application that our team has developed gives users the opportunity to view their data and the trends over time. To ensure that this device causes minimal disturbance to everyday life, we designed a portable glove that contains both a PCB and a rechargeable battery so it is convenient for users to use.

We hope that our device will help students view the effectiveness and impact of stress-reducing activities such as mindfulness and meditation. While the unforeseen challenges brought about by the pandemic prevented us from having many students interact with the device, we see its potential in helping students manage and decrease stress. It is common for students to go on autopilot in high stress situations, and self-awareness is essential to battle the detrimental effects of stress. The more self-aware students are, the more likely they are to achieve their goals both academically and outside school.

REFERENCES

[1] N. LeBlanc and L. Marques, "Anxiety in college: What we know and how to cope", Harvard Health Blog, 2019. [Online]. Available: https://www.health.harvard.edu/blog/anxiety-in-college-what-we-know-and-how-to-cope-2019052816729.

[2] American College Health Association, “American College Health Association-National College Health Assessment II: Reference Group Executive Summary Spring 2018”, Silver Spring, MD: American College Health Association, 2018. [Online]. Available: https://www.acha.org/documents/ncha/NCHA-II_Spring_2018_Reference_Group_Executive_Summary.pdf

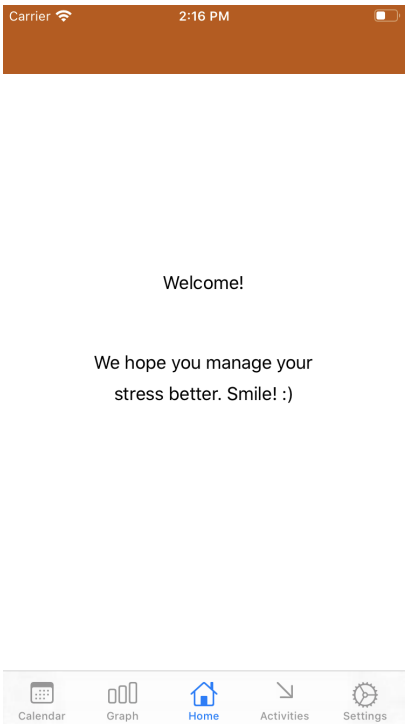
[3] Stephanie Adeline (2018, March 9). “Demand for CHMC services spikes after subsidized counseling charges” [Online] Available: https://www.dailytexanonline.com/2018/03/09/demand-for-cmhc-services-spikes-after-subsidized-counseling-charges

[4] J. A. Healey, "Wearable and automotive systems for affect recognition from physiology", Ph.D. dissertation, Dept. of Electrical Engineering and Computer Science, MIT, Boston, 2000. [Online]. Available: https://dspace.mit.edu/handle/1721.1/9067.

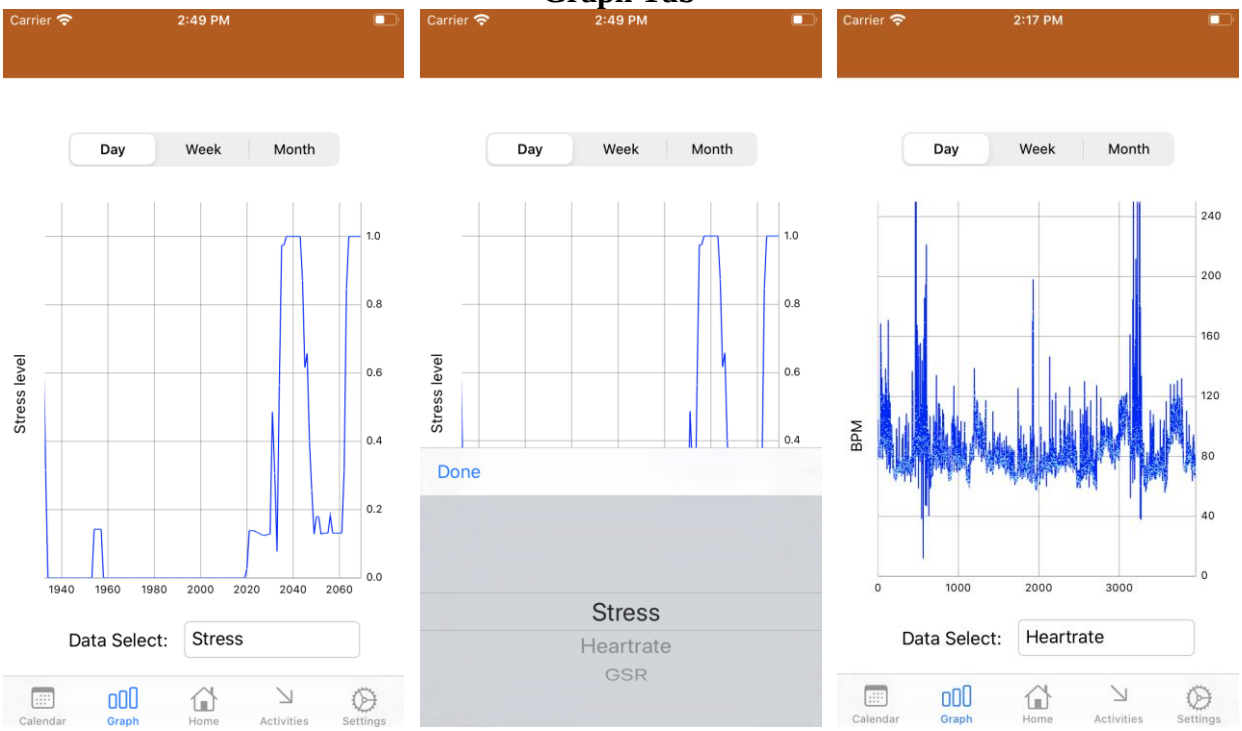
APPENDIX A – MOBILE APPLICATION

APPENDIX A – MOBILE APPLICATION

Home Screen

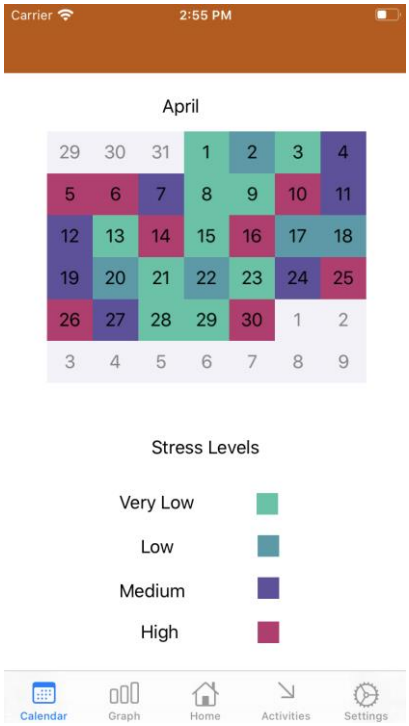


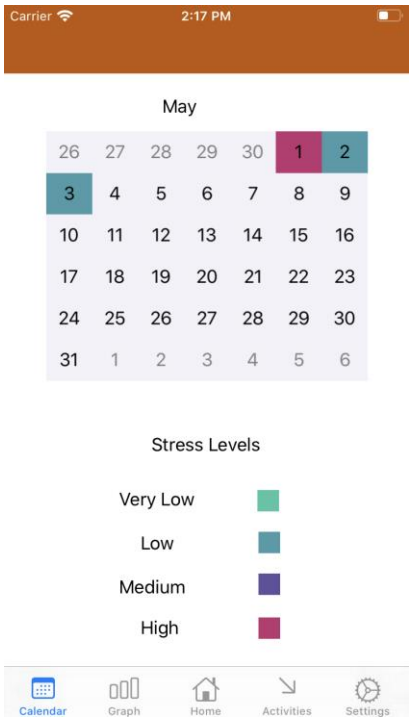
Graph Tab



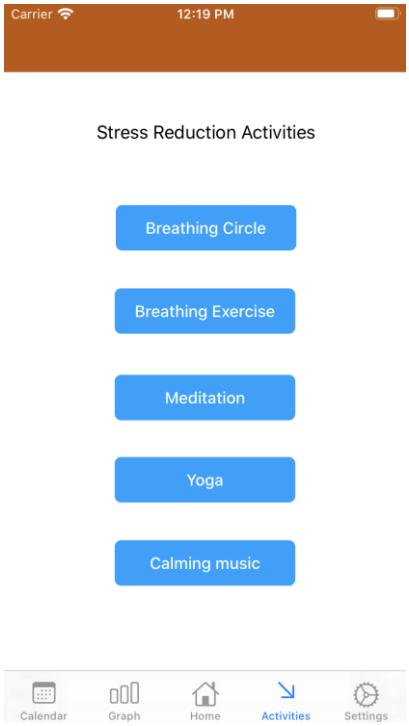
APPENDIX A – MOBILE APPLICATION

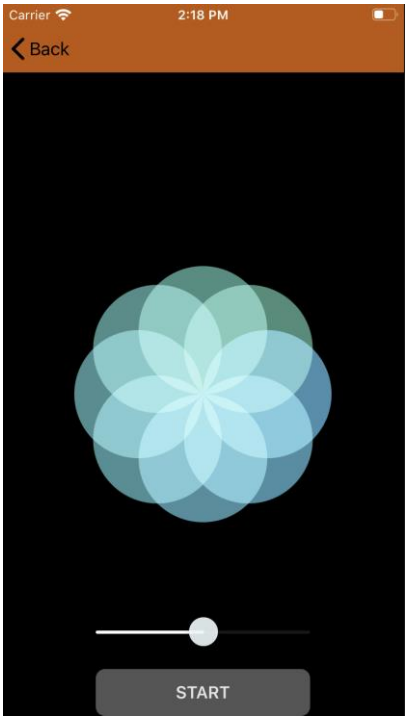
Calendar Tab





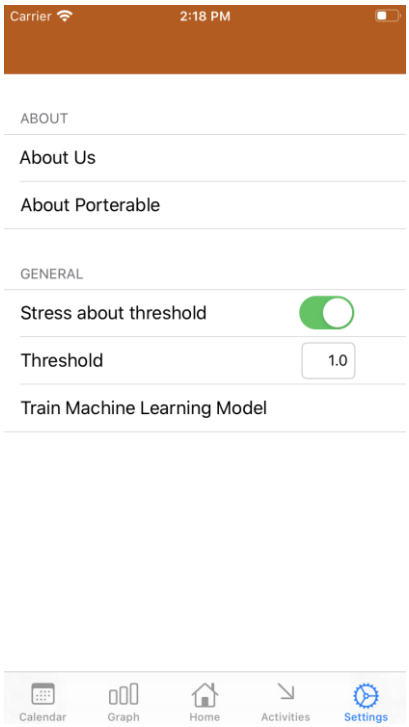
Stress Reduction Tab

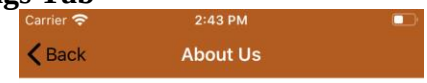




APPENDIX A – MOBILE APPLICATION

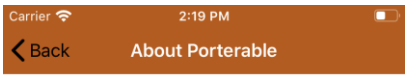
Settings Tab





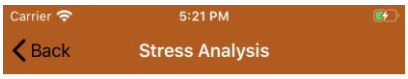
About Us:

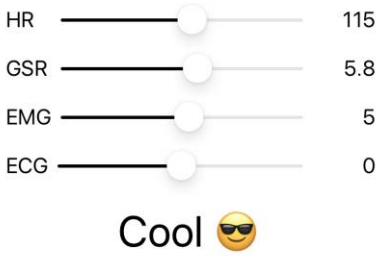
We are electrical and computer engineering students at The University of Texas at Austin. Our team members are Matthew Barondeau, Harini Cherupalla, Matthew Davis, Angelos Koulouras, Sneha Pendharkar, and Neha Shah. Our faculty mentor is Dr. Emily Porter.



About Porterable:

Porterable was created as a part of our capstone design project to help students manage their stress better. By using sensors to measure heart rate, GSR, and motion, our device collects physiological data correlated with stress. Our app uses this data to classify stress and display results in a user-friendly format.

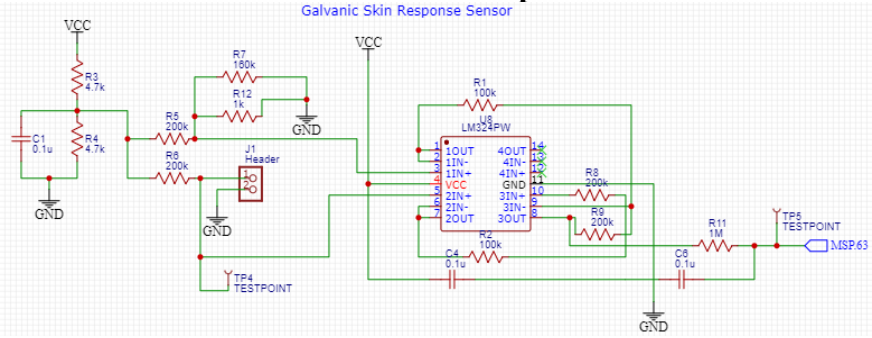




APPENDIX B – COMPONENT SPECIFICATIONS

APPENDIX B – COMPONENT SPECIFICATIONS

Grove Galvanic Skin Response Sensor

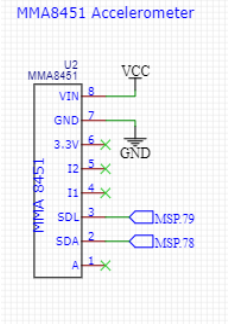


Specification

Parameter	Value/Range
Operating voltage	3.3V/5V
Sensitivity	Adjustable via a potentiometer
Input Signal	Resistance, NOT Conductivity
Output Signal	Voltage, analog reading
Finger contact material	Nickel

APPENDIX B – COMPONENT SPECIFICATIONS

MMA8451 Accelerometer



2.2 Electrical Characteristics

Table 3. Electrical Characteristics @ VDD = 2.5V, VDDIO = 1.8V, T = 25°C unless otherwise noted.

Parameter	Test Conditions	Symbol	Min	Typ	Max	Unit
Supply Voltage		VDD(1)	1.95	2.5	3.6	V
Interface Supply Voltage		VDDIO(1)	1.62	1.8	3.6	V
Low Power Mode	ODR = 1.56 Hz	I_{dd}LP		6		µA
	ODR = 6.25 Hz			6		
	ODR = 12.5 Hz			6		
	ODR = 50 Hz			14		
	ODR = 100 Hz			24		
	ODR = 200 Hz			44		
	ODR = 400 Hz			85		
	ODR = 800 Hz			165		
Normal Mode	ODR = 1.56 Hz	I_{dd}		24		µA
	ODR = 6.25 Hz			24		
	ODR = 12.5 Hz			24		
	ODR = 50 Hz			24		
	ODR = 100 Hz			44		
	ODR = 200 Hz			85		
	ODR = 400 Hz			165		
	ODR = 800 Hz			165		
Current during Boot Sequence, 0.5 mSec max duration using recommended Bypass Cap	VDD = 2.5V	Idd Boot			1	mA
Value of Capacitor on BYP Pin	-40°C 85°C	Cap	75	100	470	nF
STANDBY Mode Current @25°C	VDD = 2.5V, VDDIO = 1.8V STANDBY Mode	I_{dd}Stby		1.8	5	µA
Digital High Level Input Voltage SCL, SDA, SA0		VIH	0.75*VDDIO			V
Digital Low Level Input Voltage SCL, SDA, SA0		VIL			0.3*VDDIO	V
High Level Output Voltage INT1, INT2	I_O = 500 µA	VOH	0.9*VDDIO			V
Low Level Output Voltage INT1, INT2	I_O = 500 µA	VOL			0.1*VDDIO	V
Low Level Output Voltage SDA	I_O = 500 µA	VOLS			0.1*VDDIO	V
Power on Ramp Time			0.001		1000	ms
Time from VDDIO on and VDD > Vmin until I^2C ready for operation	Cbyp = 100 nF	BT	—	350	500	µs
Turn-on time (STANDBY to ACTIVE)		Ton		2/ODR + 1 ms		s
Turn-on time (Power Down to ACTIVE Mode)		Ton		2/ODR + 2 ms		s
Operating Temperature Range		Top	-40		+85	°C

1. There is no requirement for power supply sequencing. The VDDIO input voltage can be higher than the VDD input voltage.

APPENDIX B – COMPONENT SPECIFICATIONS

MMA8451 Accelerometer

2.3 I^2C interface characteristics

Table 4. I^2C slave timing values(1)

Parameter	Symbol	I^2C Fast Mode		Unit
		Min	Max	
SCL clock frequency	f_{SCL}	0	400	kHz
Bus-free time between STOP and START condition	t_{BUF}	1.3		µs
(Repeated) START hold time	$t_{HD;STA}$	0.6		µs
Repeated START setup time	$t_{SU;STA}$	0.6		µs
STOP condition setup time	$t_{SU;STO}$	0.6		µs
SDA data hold time	$t_{HD;DAT}$	0.05	0.9 (2)	µs
SDA setup time	$t_{SU;DAT}$	100		ns
SCL clock low time	t_{LOW}	1.3		µs
SCL clock high time	t_{HIGH}	0.6		µs
SDA and SCL rise time	t_r	20 + 0.1 C_b(3)	300	ns
SDA and SCL fall time	t_f	20 + 0.1 C_b(3)	300	ns
SDA valid time (4)	$t_{VD;DAT}$		0.9 (2)	µs
SDA valid acknowledge time (5)	$t_{VD;ACK}$		0.9 (2)	µs
Pulse width of spikes on SDA and SCL that must be suppressed by internal input filter	t_{SP}	0	50	ns
Capacitive load for each bus line	Cb		400	pF

1.All values referred to $V_{IH(min)}$ ($0.3V_{DD}$) and $V_{IL(max)}$ ($0.7V_{DD}$) levels.
2.This device does not stretch the LOW period (t_{LOW}) of the SCL signal.
3.C_b = total capacitance of one bus line in pF.
4.$t_{VD;DAT}$ = time for data signal from SCL LOW to SDA output (HIGH or LOW, depending on which one is worse).
5.$t_{VD;ACK}$ = time for Acknowledgement signal from SCL LOW to SDA output (HIGH or LOW, depending on which one is worse).

2.4 Absolute Maximum Ratings

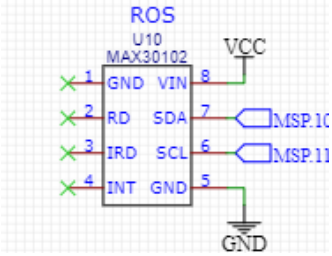
Stresses above those listed as “absolute maximum ratings” may cause permanent damage to the device. Exposure to maximum rating conditions for extended periods may affect device reliability.

Table 5. Maximum Ratings

Rating	Symbol	Value	Unit
Maximum Acceleration (all axes, 100 µs)	g_{max}	5,000	g
Supply Voltage	VDD	-0.3 to + 3.6	V
Input voltage on any control pin (SA0, SCL, SDA)	Vin	-0.3 to VDDIO + 0.3	V
Drop Test	D_{drop}	1.8	m
Operating Temperature Range	T_{OP}	-40 to +85	°C
Storage Temperature Range	T_{STG}	-40 to +125	°C

APPENDIX B – COMPONENT SPECIFICATIONS

MAX30102 Reflective Optical Sensor



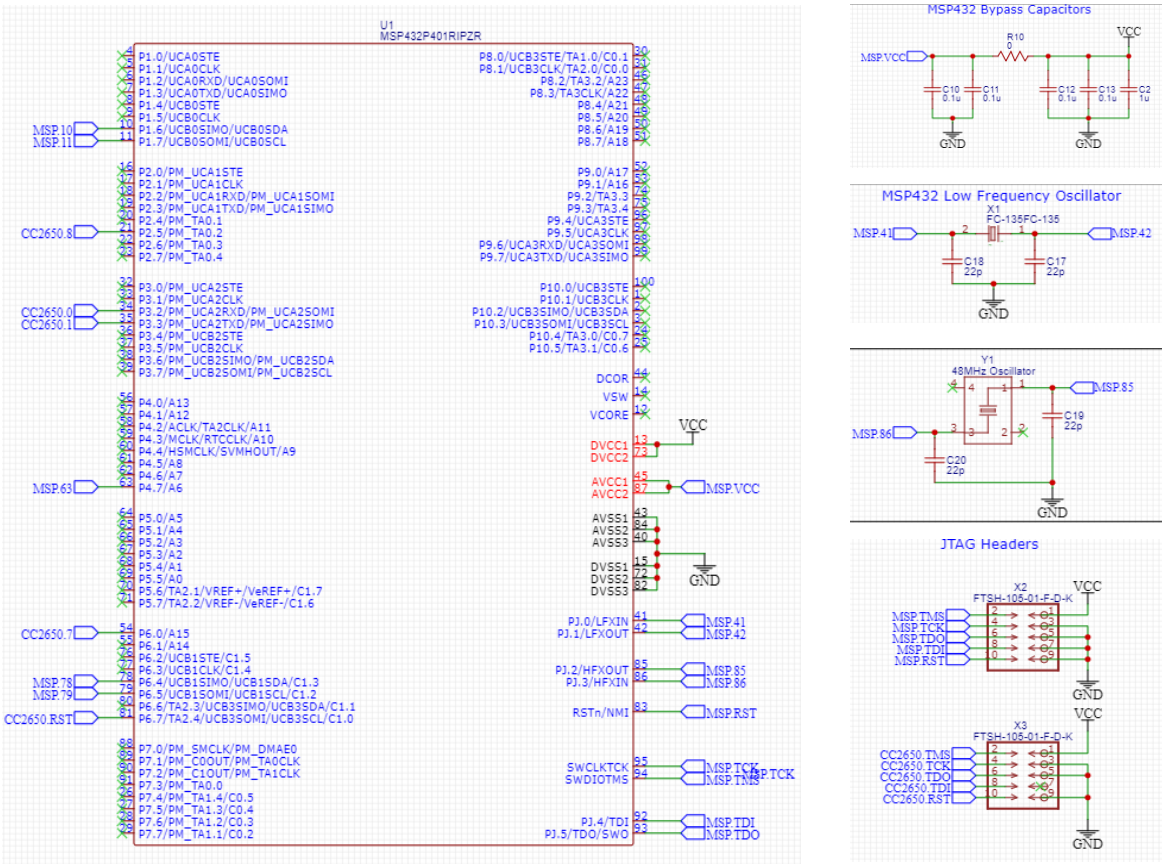
Absolute Maximum Ratings

V_{DD} to GND .....-0.3V to +2.2V
GND to PGND.....-0.3V to +0.3V
V_{LED+} to PGND.....-0.3V to +6.0V
All Other Pins to GND .....-0.3V to +6.0V
Output Short-Circuit Current Duration.....Continuous
Continuous Input Current into Any Terminal .....±20mA
ESD, Human Body Model (HBM).....2.5kV
Latchup Immunity.....±250mA

Continuous Power Dissipation (T_A = +70°C)
OESIP (derate 5.5mW/°C above +70°C) 440mW
Operating Temperature Range..... -40°C to +85°C
Junction Temperature.....+90°C
Soldering Temperature (reflow) .....+260°C
Storage Temperature Range..... -40°C to +105°C

APPENDIX B – COMPONENT SPECIFICATIONS

MSP432 Microcontroller



5.1 Absolute Maximum Ratings[1]

over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)

	MIN	MAX	UNIT
Voltage applied at DVCC and AVCC pins to V_{SS}	–0.3	4.17	V
Voltage difference between DVCC and AVCC pins[2]		±0.3	V
Voltage applied to any pin[3]	–0.3	V_{CC} + 0.3 V (4.17 V MAX)	V
Diode current at any device pin		±2	mA
Storage temperature, T_{stg} [4]	–40	125	°C
Maximum junction temperature, T_J		95	°C

5.3 Recommended Operating Conditions

Typical data are based on V_{CC} = 3.0 V, T_A = 25°C (unless otherwise noted)

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage range at all DVCC and AVCC pins [1] [2] [3]	At power-up (with internal V_{CC} supervision)	1.71		3.7	V
		Normal operation with internal V_{CC} supervision	1.71		3.7	
		Normal operation without internal V_{CC} supervision	1.62		3.7	
V_{SS}	Supply voltage on all DVSS and AVSS pins			0		V
I_{INRUSH}	Inrush current into the V_{CC} pins[4]				100	mA
f_{MCLK}	Frequency of the CPU and AHB clock in the system[5]		0		48	MHz
T_A	Operating free-air temperature		–40		85	°C
T_J	Operating junction temperature		–40		85	°C

APPENDIX B – COMPONENT SPECIFICATIONS

Lithium Polymer Battery

3、 Specification

Item		Specifications	Remark
Nominal Capacity		1200mAh	0.2C_5A discharge，25℃
Nominal Voltage		3.75V	Average Voltage at 0.2C_5A discharge
Standard Charge Current		0.2 C_5A	Working temperature：0～40℃
Max Charge Current		1C_5A	Working temperature：0～40℃
Charge cut-off Voltage		4.2V	CC/CV
Standard Discharge Current		0.5C_5A	Working temperature：25℃
Discharge cut-off Voltage		2.75V	
Cell Voltage		3.7-3.9V	When leave factory
Impedance		≤50mΩ	AC 1KHz after 50% charge,25℃
Weight		Approx:22g	
Storage temperature	≤1month	-10～45℃	Best 20±5℃ for long-time storage
	≤3month	0～30℃	
	≤6month	20±5℃	
Storage humidity		65±20% RH	

APPENDIX C – UNIT TESTS FOR MOBILE APPLICATION

APPENDIX C – UNIT TESTS FOR MOBILE APPLICATION

Test Number	Unit Test	Verification Date
1	Home page loads within 3 seconds	March 25, 2020
2	Calendar page loads within 3 seconds	March 25, 2020
3	Graph page loads within 3 seconds	March 25, 2020
4	Activities page loads within 3 seconds	March 25, 2020
5	Settings page loads within 3 seconds	March 25, 2020
6	“Breathing Circle” button links to breathing circle	April 17, 2020
7	“Breathing Exercise” button navigates to desired external page on Safari	April 17, 2020
8	“Meditation” button links to desired external page on Safari	April 24, 2020
9	“Yoga” button links to desired external page on Safari	April 24, 2020
10	“Calming music” button links to desired video URL on Safari	April 24, 2020
11	Graph appears with default “Stress” data within 5 seconds	April 26, 2020
12	Graph options bar provides choices for Stress, Heart Rate, GSR, and SpO2	April 26, 2020
13	Selecting “Heart Rate” populates graph with heart rate data within 5 seconds	April 26, 2020
14	Selecting “GSR” populates graph with GSR data within 5 seconds	April 26, 2020
15	Selecting “SpO2” populates graph with SpO2 data within 5 seconds	April 26, 2020
16	Selecting “Stress” populates graph with stress data within 5 seconds	April 26, 2020
17	Color-coded calendar appears with data	April 4, 2020
18	Toggle on settings page works	April 8, 2020
19	Selecting “About Us” from settings page loads corresponding description	April 20, 2020
20	Selecting “About Porterable” from settings page loads corresponding description	April 20, 2020

APPENDIX D – BUDGET

APPENDIX D – BUDGET

Quantity	Description	Item	Unit Cost	Total Cost	Source
4	GSR	Grove GSR Sensor	$14.90	$59.60	Amazon
3	ROS	MAX30102	$8.59	$25.77	Amazon
3	Microcontroller	MSP432	$23.00	$69.00	Texas Instruments
1	Bluetooth Module	CC2650	$29.00	$29.00	Texas Instruments
2	Accelerometer	MMA8451	$6.95	$13.90	Adafruit
2	Battery	LiPo Battery	$9.95	$19.90	Amazon
1	Charger	Battery Charger	$12.50	$12.50	Amazon
8	Glove	Fingerless Glove	$1.37	$10.96	Amazon
1	Oximeter	Finger Pulse Ox	$19.99	$19.99	Amazon
1	Encasing	Battery Holder	$9.99	$9.99	Amazon
1	Circuit	PCB v1	$2.00	$2.00	JLCPCB
2	JTAG Headers	FTSH-105-01-F-D-K	$2.78	$5.56	Digikey
1	Microcontroller	MSP432P401RIPZR	$6.27	$6.27	Texas Instruments
1	Op-Amp for GSR	LM324PW	$0.44	$0.44	Mouser
1	High Speed Oscillator	48MHz Oscillator	$0.33	$0.33	Mouser
1	Low Speed Oscillator	32kHz Oscillator	$0.59	$0.59	Digikey
7	Capacitors	0.1u	$0.10	$0.70	Mouser
3	Capacitors	1u	$0.10	$0.30	Digikey
4	Capacitors	22p	$0.10	$0.40	Mouser
1	USB Header	ZX62D-B-5PA8	$0.74	$0.74	Mouser
1	LDO Regulator	LM2937ES-3.3	$1.87	$1.87	Mouser
1	Bluetooth Chip	CC2650MODA	$8.84	$8.84	Texas Instruments
4	Resistor	200k	$0.10	$0.40	Mouser
1	Resistor	1k	$0.10	$0.10	Mouser
1	Resistor	1M	$0.10	$0.10	Mouser
2	Resistor	100k	$0.10	$0.20	Mouser
1	Resistor	0R	$0.10	$0.10	Mouser
2	Resistor	4.7k	$0.10	$0.20	Mouser
1	Resistor	160k	$0.10	$0.10	Mouser
1	Circuit	PCB v2	$2.00	$2.00	JLCPCB
2	Headers	67996-220HLF	$0.83	$1.66	Digikey
1	Launchpad	MSP432EXP432P401R	$23.59	$23.59	Texas Instruments
			TOTAL	$327.10	

APPENDIX E – CITI CERTIFICATES FOR HUMAN TESTING

APPENDIX E – CITI CERTIFICATES FOR HUMAN TESTING

CITI PROGRAM

Completion Date 02-Oct-2019
Expiration Date 01-Oct-2022
Record ID 33581686

This is to certify that:

MATTHEW EDWARD BARONDEAU

Has completed the following CITI Program course:

Not valid for renewal of certification through CME. Do not use for TransCelerate mutual recognition (see Completion Report).

Human Research (Curriculum Group)
Biomedical Researchers (Course Learner Group)
1 - Basic Course (Stage)

Under requirements set by:

University of Texas at Austin

CITI
Collaborative Institutional Training Initiative

Verify at www.citiprogram.org/verify/?w7d4cc185-0c27-4778-86be-4875eec87df5-33581686

CITI PROGRAM

Completion Date 07-Oct-2019
Expiration Date 06-Oct-2022
Record ID 33661677

This is to certify that:

HARINI M CHERUPALLA

Has completed the following CITI Program course:

Not valid for renewal of certification through CME. Do not use for TransCelerate mutual recognition (see Completion Report).

Human Research (Curriculum Group)
Biomedical Researchers (Course Learner Group)
1 - Basic Course (Stage)

Under requirements set by:

University of Texas at Austin

CITI
Collaborative Institutional Training Initiative

Verify at www.citiprogram.org/verify/?w2e1a581f-0a51-43f0-bcc9-58d0beb038f9-33661677

APPENDIX E – CITI CERTIFICATES FOR HUMAN TESTING

CITI PROGRAM

Completion Date 03-May-2020
Expiration Date 03-May-2023
Record ID 36507537

This is to certify that:

MATTHEW QUINCY DAVIS

Has completed the following CITI Program course:

Not valid for renewal of certification through CME. Do not use for TransCelerate mutual recognition (see Completion Report).

Human Research (Curriculum Group)
Social/Behavioral Researchers (Course Learner Group)
1 - Basic Course (Stage)

Under requirements set by:

University of Texas at Austin

CITI
Collaborative Institutional Training Initiative

Verify at www.citiprogram.org/verify/?w08fa1fcd-6fa2-4dcf-b7ab-82b73dddfeff-36507537

CITI PROGRAM

Completion Date 07-Oct-2019
Expiration Date 06-Oct-2022
Record ID 33675829

This is to certify that:

ANGELOS GEOR KOULOURAS

Has completed the following CITI Program course:

Not valid for renewal of certification through CME. Do not use for TransCelerate mutual recognition (see Completion Report).

Human Research (Curriculum Group)
Biomedical Researchers (Course Learner Group)
1 - Basic Course (Stage)

Under requirements set by:

University of Texas at Austin

CITI
Collaborative Institutional Training Initiative

Verify at www.citiprogram.org/verify/?wa3561951-4248-42a9-a0f0-87a2d69daa3d-33675829

APPENDIX E – CITI CERTIFICATES FOR HUMAN TESTING

CITI PROGRAM

Completion Date 07-Oct-2019
Expiration Date 06-Oct-2022
Record ID 33675780

This is to certify that:

SNEHA S PENDHARKAR

Has completed the following CITI Program course:

Not valid for renewal of certification through CME. Do not use for TransCelerate mutual recognition (see Completion Report).

Human Research (Curriculum Group)
Biomedical Researchers (Course Learner Group)
1 - Basic Course (Stage)

Under requirements set by:

University of Texas at Austin

CITI
Collaborative Institutional Training Initiative

Verify at www.citiprogram.org/verify/?w03159fac-f61c-46f9-aee4-c7ac22002e87-33675780

CITI PROGRAM

Completion Date 04-Oct-2019
Expiration Date 03-Oct-2022
Record ID 33648158

This is to certify that:

NEHA SHAH

Has completed the following CITI Program course:

Not valid for renewal of certification through CME. Do not use for TransCelerate mutual recognition (see Completion Report).

Human Research (Curriculum Group)
Social/Behavioral Researchers (Course Learner Group)
1 - Basic Course (Stage)

Under requirements set by:

University of Texas at Austin

CITI
Collaborative Institutional Training Initiative

Verify at www.citiprogram.org/verify/?w48517be3-7d5b-4018-ab68-671105f92dd5-33648158