University of Arkansas – CSCE Department
Capstone II – Final Report– Fall 2021

Care-Mate: Wheelchair Pressure Distribution Mapping System

Benjamin Allen, Clay Griscom, Hugo Serrano,
Kira Threlfall, David Whelan

Abstract

One risk faced by wheelchair-bound patients is the development of pressure ulcers, which reduce the patient's quality of life and increase the risk of infection. This can lead to major complications. This project is designed to reduce this risk by constructing a pressure sensor array and application to enable visualization of high-pressure areas that may form pressure ulcers.

There are several devices that can measure the pressure distribution on a wheelchair, but they are expensive and may be inaccessible. The application interfaces with the pressure sensor array via Bluetooth and then displays this information as a heatmap and allows users to calibrate the sensitivity of the pressure pad.

1.0 Problem

The problem we are trying to address significantly impacts people with disabilities that are bound to wheelchairs. Their placement in a wheelchair is vitally important to their quality of life. If not monitored carefully, those who spend considerable time in a wheelchair can develop pressure ulcers. Pressure ulcers in wheelchair users are caused by restrictions of blood flow due to sitting for extended periods of time. Not all patients in wheelchairs can feel when there is low blood flow. Additionally, it is impossible for caregivers to know where pressure ulcers might form, complicated by the inability to see pressure ulcers forming. Caregivers can adjust the patient in the wheelchair throughout the day, but this does not guarantee the prevention of pressure ulcers.

These pressure ulcers are painful and make the patient more susceptible to infection, and there is a clear negative impact on the patient's quality of life. Currently there are a few products that offer solutions to this problem, which give a map of pressure distribution on a wheelchair seat. This distribution information can reveal points of high pressure that can lead to the formation of pressure ulcers. This can lead to better patient positioning and preventative measures.

While these solutions exist, they lack accessibility in ease-of-use and cost. Current solutions are often expensive and directed at research instead of patient care. These difficulties often prevent caregivers from using such tools. Some also require a dedicated computer, making them non-portable. A portable solution would be easier to use for both caregivers and patients.

2.0 Objective

This is a joint project by students from the ELEG, BMEG, and CSCE departments in the College of Engineering, and a student from the Walton College of Business. The objective of this project is to design, build, and test a system to take pressure readings on a wheelchair and display this data. Our project is targeted at accessibility — instead of being designed to be used by researchers, the target audience is the caregiver. We, the CSCE team, will build a mobile application to interface with the hardware created by the Electrical and Biomedical Engineering teams. This app will provide real-time data displayed using a color blind-friendly format. The app will show pressure maps from both the seat and the back of the wheelchair. Bluetooth will be used to increase portability and enable communication with mobile phones running the Android operating system.

By the end of the project, we will have an Android application that will allow the user to interface with a pressure sensor array placed on the wheelchair. The information provided by the sensor array and application will enable caregivers to better assist their patients.

3.0 Background

3.1 Key Concepts

Heatmaps are graphical representations of data usually using a gradient of color to represent lower to higher values of data. Heatmaps can be used in a variety of ways to help visualize data collected, usually from an area of a screen, image, or sensors. It is commonly used to show temperature gradients.

Pressure mapping is a visualization of the pressure exerted by a surface on a measurement device. The map can be viewed both live and statically depending on how the system and user decide on displaying the data. Pressure mapping is commonly used in medical examinations and in improving ergonomics in everyday objects. A pressure map is a heatmap that displays values of pressure.

Bluetooth transmitters are electronics that allow devices to communicate with each other via the Bluetooth protocol. Transmitters can be connected to consumer devices by audio output or USB/Type-C ports, or electronically integrated into small-form-factor devices. They are commonly used to enable wireless communication without a Wi-Fi network.

A Pulse Oximeter is a small device that measures a patient’s oxygen saturation level. It is a painless device to use since it only needs to be clipped onto a thin part of the body, like a finger, and uses light to measure how much oxygen is in the blood. Given the simplicity of the device it is used in many cases where the patient has a condition that may affect their blood oxygen levels like asthma or anemia.

3.2 Related Work

The Body Pressure Measurement System (BPMS) by Tekscan is a similar system to the one we have created. The system is a mat containing a grid of pressure sensors that allows for a pressure distribution map displayed in software with the resolution equal to the number of sensors on the mat. The Tekscan BPMS system also promises to allow for multiple mats to be used to cover larger surface areas such as beds. The BPMS system is marketed for several different uses such as furniture and bed design, material testing, and seating or positioning research. The sensors all allow for a 5-psi pressure range with multiple sensor configurations that allow for different seating arrangements [1]. Our design will allow for wireless Bluetooth connection between the sensors and an android phone as opposed to the wired connection that the Tekscan system requires. This will allow for more convenient usage of our device by allowing freedom of movement when monitoring and adjusting the patient in their wheelchair. Another sensor by Xsensor called the ForSite SS is a pressure system designed specifically for assisting in wheelchair seating adjustment. The mat can be set on the wheelchair and monitor how the patient is sitting to aid in adjusting the patient’s seating. The Xsensor pressure system also comes with an app that connects via Bluetooth and allows the user to monitor the pressure map of the patient’s seat on the physician's tablet. It has a pressure range of 0.1 - 3.87 psi as well as 5 frame/second refresh rate [2].

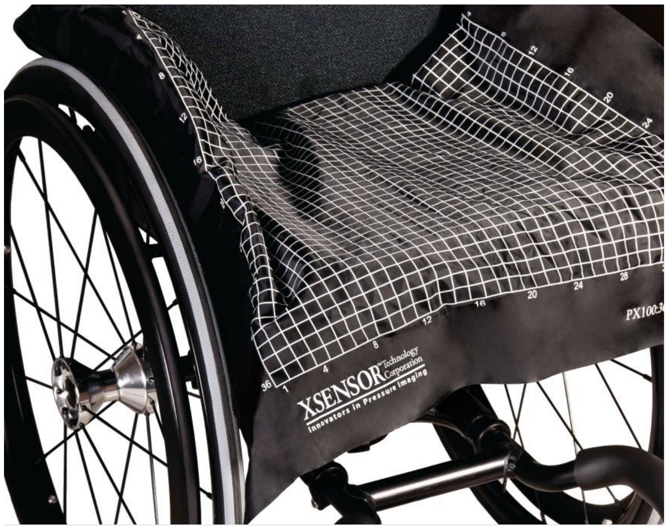

XSENSOR Technology Corporation
Innovators in Pressure Imaging
PX100:3

Figure 1: Xsensor ForSite SS Pressure Mat

Blue Chip Medical Products, Inc. has a wheelchair pressure mapping system called the MeasureX Pressure Mapping system. This is a system also designed specifically for assisting in adjusting wheelchair bound patients based on pressure point analysis. Like the other sensors there is also compatible software that comes along with the sensor pad. The MeasureX system has wired and wireless options for communication between the software and sensors. The MeasureX has a larger range of up to 30 psi [3]. The BodiTrak2 Wireless IoT Pressure Mapping System is another general-purpose pressure mapping pad and software. They offer both a BodiTrak2 Pro and Lite app for PC or MacBooks and Android or iOS devices respectively. The pad is marketed for multiple different applications, one of them being wheelchair users. The BodiTrak offers two resolutions of sensors, one having 256 sensors and the other 1024 sensors [4]. The software displays a 2-Dimensional array of a pressure heat map similar to all the other devices.

Each of the designs listed are very similar to our proposed design. They all use a mat or pad with an array of pressure sensors that sit on the seat of a wheelchair to aid the physician in making seating adjustments for the patient. This will help prevent pressure ulcers from forming because of the patient not being seated properly. Each of the systems also has some sort of application that shows the pressure distribution map of the patient seated in the wheelchair. Our project differs in the following aspects. The first is the inclusion of a continuous pressure calibration system designed by the BMEG team which makes use of a Pulse Oximeter. Another difference is the price of our system in comparison to the other products listed. Our design has a bill of materials with a cost under $1000. With the addition of the Pulse Oximeter calibration tool and the price point we will be able to design a device that has an edge over the products already offered.

4.0 Design

4.1 Design Goals

Wireless connectivity: A Bluetooth transmitter was used to communicate readings from the sensor array. Wired connectivity was not an option, as physical wires can become entangled in the wheelchair's mechanisms. Constant connectivity between the application and the sensor array is required to enable continuous data updates.

Companion Application: A mobile application was created to display readings from the sensor array. This application was developed to run Android operating system. It enables the user to switch between viewing data from the seat pressure sensor array, the back pressure sensor array, or both simultaneously. The data received from the sensor array is displayed as a heatmap, with a color palette designed to be usable by color blind individuals.

Sensor Array: A single prototype sensor array of 64 pressure sensors was constructed by the ELEG team with assistance from Clay Griscom on the CSCE team. It is used to gather pressure readings from the wheelchair user; these are processed by a microcontroller then sent through a Bluetooth transmitter. Originally, we planned to construct two such arrays, but due to supply constraints, only a single array was constructed. This array can be moved between the seat and the back of the wheelchair.

Calibration: The companion application provides a method for calibrating the pressure map display. To calibrate the application display, a caretaker first uses the continuous pressure calibration system to press the patient into the wheelchair. The continuous pressure calibration system is a device created by the BMEG team to assist in accurate calibration. When the Pulse Oximeter attached to the patient reports hypoxic blood flow, a caretaker presses the calibration button on the companion application. The application reads the pressure from the sensor array to calibrate the display and stores this information until the next calibration event.

Accessibility: Heat maps that use all colors in the rainbow, while common, are not accessible to color blind users (see Figure 2). The pressure map visualizations in the companion application were made accessible to most color blind users by using the Viridis color scale [5]. Viridis is specifically designed for legibility by users with the most common forms of color blindness (see Figure 5). To ensure that users can properly interpret the map, we will provide a color scale to the side of each pressure map. As a stretch goal, we may allow the user to choose from a selection of color gradients.

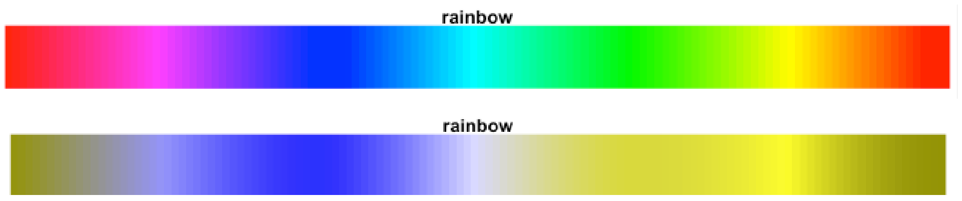


Figure 2: The rainbow color scale, and the same scale as seen by green-blind users [5].

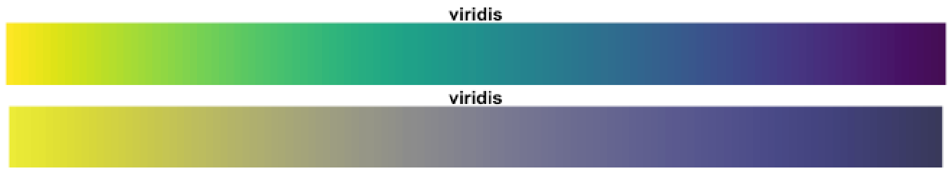


Figure 3: The Viridis color scale, and the same scale as seen by green-blind users [5].

Device Independence: The companion application was built on the Ionic framework [6], which provides a cross-platform development system based on Typescript. Ionic allows the application to run on both Android and iOS devices, although we limit our testing to Android devices due to availability of development hardware.

4.2 Detailed Architecture

Hardware

There are two hardware components that have been designed by the ELEG team for this project: the pressure sensor to gather pressure information from the patient and the microcontroller that receives the raw pressure data from the sensors, organizes it, and transmits it over Bluetooth to the companion app on an Android phone. The sensor pad is a 12-inch x 12-inch pad consisting of 64 sensors arranged in an 8 x 8 square and is designed to have each sensor read individually. The circuit board utilizes the PIC16F877A microcontroller as the control unit. The board consists of the microcontroller, an external quartz crystal clock, an HC-06 Bluetooth module, an AD5206 digital potentiometer, and headers for each sensor with multiplexers to select the sensors.

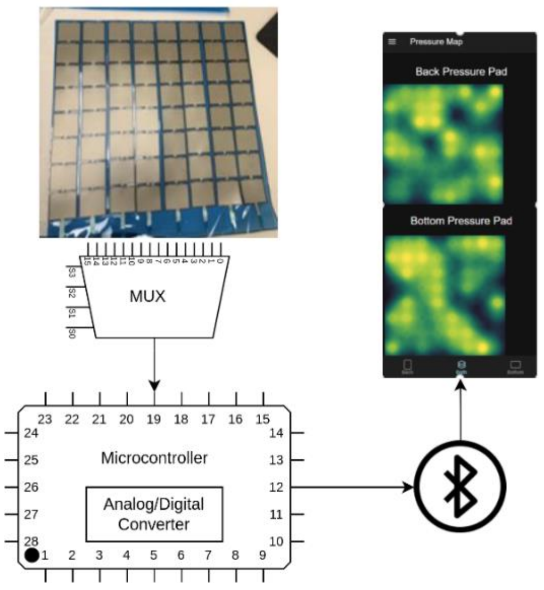


Figure 4: Hardware Data Flow Chart

The board has two 16 to 1 multiplexers that are used to select each sensor. One multiplexer is used as a high side voltage input to each sensor while the other is used as a data input to the microcontroller. The sensor array is set to have eight sensors share one multiplexer input going into the microcontroller with each of the eight sensors on that row having a separate voltage select from the other multiplexer. Having the two multiplexers in this orientation makes the selection of each sensor similar to accessing a 2-dimensional array with the voltage in multiplexer acting as the vertical axis and the data input multiplexer as the horizonal axis. This allows for all 64 sensors to be selected using only half of the inputs for each multiplexer. The board is capable of handling up to two sensor pads simultaneously with each sensor pad having 256 sensors. The data from each sensor is sent to the microcontroller through one of the microcontroller's analog-to-digital converter ports. This signal is scaled using the potentiometer to keep the signal within the desired range. Communication between the microcontroller and potentiometer is achieved through SPI communication from the microcontroller. Two configuration settings are sent that set the resistance range and channel select. Each multiplexer has 4 select ports that are mapped to 8 separate output ports on the microcontroller. The microcontroller selects each sensor one at a time and converts the sensor data to a discrete value. The microcontroller then converts that number to an integer value from 33 to 125 that represents how much relative force is being put on each sensor.

Once the microcontroller collects a data entry from the sensor it transmits the reading over Bluetooth to the companion application. The microcontroller can send one 8-bit value at a time though its USART transmission port. The microcontroller operates at a frequency of 8 MHz and is configured to a baud rate of 9600 for asynchronous serial communication. The integer sensor value gets cast as an 8-bit character and is loaded into the transmit register of the microcontroller. The microcontroller waits for the transmit interrupt to occur signifying that the character has been sent and loads the next value. The Bluetooth module is connected to the USART transmit pin on the microcontroller and acts as a slave device that broadcasts the sensor data being sent from the microcontroller. For correct communication, the Bluetooth module is set to 9600 baud as well. Inside the code, an integer variable is set that acts as a counter. Every time a sensor is read the counter iterates to keep track of how many sensors have currently been read for that set of data. Once every sensor has been read the counter resets and the microcontroller sends either 255 or 254 to the phone application to signify the completion of that data packet. If 255 is sent, then the microcontroller has sent data from the seat sensor; if 254, then a set of back sensor data has been sent.

Frameworks and Libraries

To make our application as accessible as possible we decided to make it as platform agnostic as possible. We chose to develop our application with the Ionic Framework [6] to meet this goal. Ionic allows development using the React, Vue, or Angular frameworks. Our project uses Angular as this had the best documentation and some team members had Angular experience. Ionic applications are developed as web applications in TypeScript, an extension of JavaScript. This allows us to prototype the application locally without needing to use emulators or actual hardware. When the application is ready to be tested in hardware, Ionic can be used to generate native applications for both IOS and Android. Since these are native applications, they can be distributed through the relevant app store and do not require online hosting services like a traditional web app. The native applications are more friendly to use since it is placed on the users' home screen for easy access.

Additionally, a JavaScript library called heatmap.js [7] was used to handle the task of drawing pressure map data onto the page and a module called capacitor-bluetooth-serial was used to interface with Android's Bluetooth capabilities [8]. Heatmap.js was used without modification, except for a necessary change to one script file to fix a bug caused by an update to Chrome-based browsers. The capacitor-bluetooth-serial module also required another single line change due to browser updates.

User Interface and User Experience

Falling in the category of health-related applications, the companion application must be neat and simple to navigate. It uses the sans-serif font family, primarily the Roboto font, since it allows for more natural reading rhythm with its natural width lettering. Key sections in the application have a larger title font size that will allow the user to swiftly see what they are looking at. The application uses a dark theme to reduce eye strain, while the color scheme for the application is a light blue hue that will evoke feelings of safety and relaxation from the user while they navigate the application.

The companion application is divided into three pages accessing different functionalities. The three pages contain the Pressure Map, Bluetooth, and Calibration page. Navigation of the application is controlled by the side menu that shows the three different pages that are accessible. For smoother navigation, the current page will be highlighted blue from the list of pages in the side menu (see Figure 5). Once a page is selected, the side menu will close, and the selected page will be displayed.

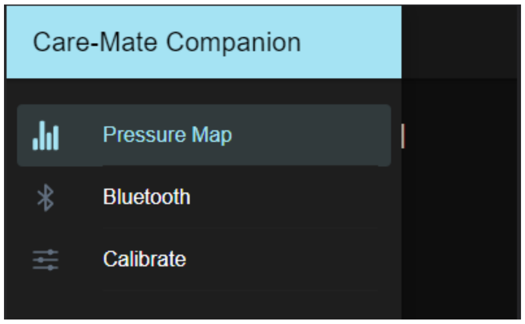


Figure 5: Care-Mate Companion side-menu navigation

On the Pressure Map page (see Figure 6), you will see a content view and tab bar with three buttons on the bottom. The content view holds the pressure map and a legend, using the Viridis color scale [5], with a title describing the map directly below. The tab bar will allow the user to quickly switch which pressure map is being displayed. One of the buttons on the tab bar will be highlighted to show which pressure map is currently being displayed. The buttons on the tab bar have an icon and description pair. The "Back" button has a portrait icon that will display the pressure map of the back pad, the "Seat" button has a landscape icon that will display the pressure map of the seat pad, and the "Both" button has a stack icon that will display the pressure map of both back and seat pads.

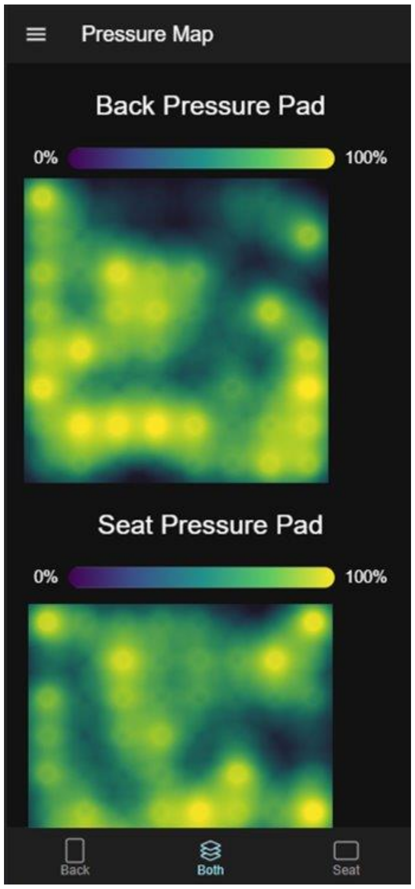
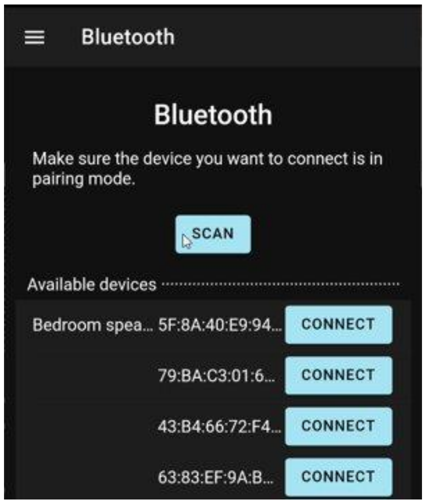


Figure 6: Care-Mate Companion Pressure Map page.

The Bluetooth page (see Figure 7) displays a title, instructions, a button, and a list. The title allows for a simple reminder that they are on the Bluetooth page, while the instructions tell the user to put the other device into pairing mode. The button is colored blue to stand out and has the word “scan”. Once the application scans for devices, all the available devices will populate in a list below the “Available devices” line. From there the user can select the pressure pad from the list to connect to the application.


Bluetooth

Bluetooth

Make sure the device you want to connect is in pairing mode.

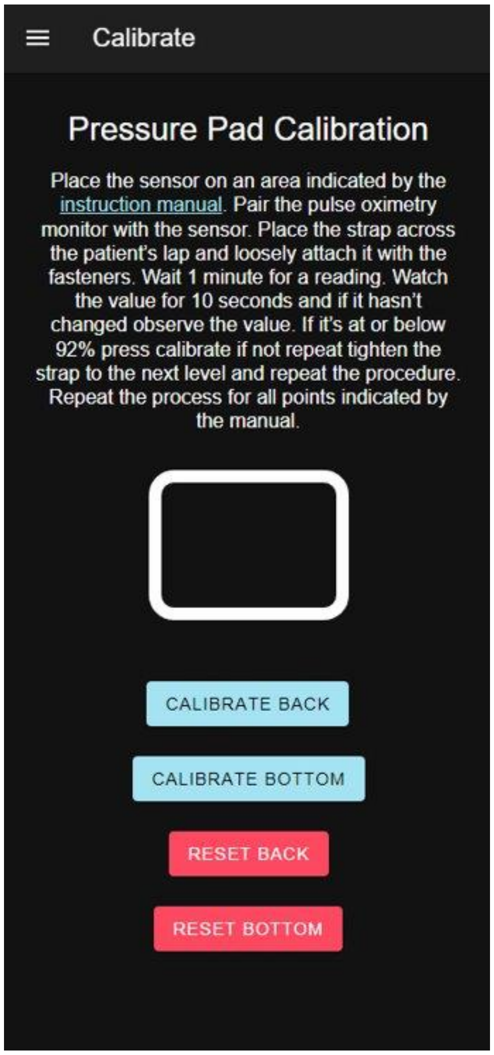
SCAN

Available devices

Bedroom spea...	5F:8A:40:E9:94...	CONNECT
	79:BA:C3:01:6...	CONNECT
	43:B4:66:72:F4...	CONNECT
	63:83:EF:9A:B...	CONNECT

Figure 7: Care-Mate Companion Bluetooth page.

The Calibrate page (see Figure 8) will display a title, instructions, and buttons. The page starts with the title to tell the user that this is where the pressure pad is being calibrated, followed with some instructions on how to calibrate the pressure pad. Directly below the instructions there is a landscape icon to give the user an image to refer to the pressure pad. All of this leads the user down to a pair of blue calibrate buttons and red reset buttons.



Pressure Pad Calibration

Place the sensor on an area indicated by the instruction manual. Pair the pulse oximetry monitor with the sensor. Place the strap across the patient's lap and loosely attach it with the fasteners. Wait 1 minute for a reading. Watch the value for 10 seconds and if it hasn't changed observe the value. If it's at or below 92% press calibrate if not repeat tighten the strap to the next level and repeat the procedure. Repeat the process for all points indicated by the manual.

Figure 8: Care-Mate Companion Calibrate page.

Application Logic and Backend

The backend logic for the companion application is modularized into multiple services, each with a single concern. These services are the BluetoothService, CalibrationService, and HeatmapService. These services are provided to the application views by dependency injection, and each service provides an interface detailing the methods that views can call. All services are singleton services, which means there is only one instance of each service shared across the application; this was necessary to maintain consistency within the application. We also developed a callback system to enable data updates to trigger view updates. The figure below illustrates the dependencies between the pages and services in the application. Note that every module depends on the Ionic framework as well.

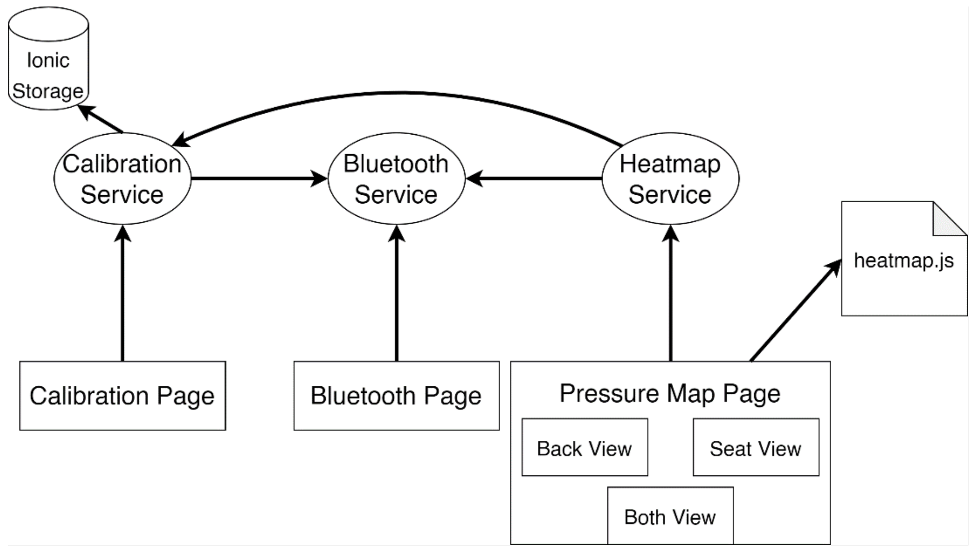


Figure 9: An arrow pointing to a component is a dependency on that component, i.e., the calibration page depends on CalibrationService. Ionic storage and heatmap.js are external modules not written by our team.

We needed to develop a small event communication system to enable automatic refreshing of the pressure map displays whenever new data was received from the Bluetooth transmitter. Ionic previously provided a publisher-subscriber event system but this has been deprecated. Our solution was to have views and services register callback methods that can be called to communicate the receipt of new data.

The BluetoothService is responsible for connecting and communicating data between the pressure pad and the mobile device. To enable the testing scheme that is described below, the BluetoothService was implemented using Contracts. There is an abstract class that defines the functions and data used by a BluetoothService. This definition allows for several Bluetooth implementations that are interchangeable. Currently there are two different BluetoothService implementations. The first implementation is the LocalBluetoothService which generates random data without any hardware dependencies. This allows for concurrent development when the hardware resources are limited. The second Bluetooth service is the HC06BluetoothService. This service interfaces with a Bluetooth module to retrieve the pressure data.

The CalibrationService is responsible for storing calibration data and using the current calibration data to calibrate input data. On initialization of the application, the CalibrationService is constructed with any previously stored calibration data. If there is no calibration data from persistent storage, it assumes a default array. The CalibrationService has methods to set calibration array and scale input data with a stored calibration array. When setting the calibration array, we store the array on the device, using the Ionic storage-angular module, and locally. The calibration methods call the BluetoothService to receive the most recent input from the sensor array. These methods are asynchronous to ensure that data is being received and stored successfully. The methods to scale input data use the calibration arrays stored locally to decrease wait time when updating the pressure map view. Separate methods exist for both back and seat sensor arrays, so we can easily extend this application to accommodate two input arrays.

The HeatmapService is responsible for properly scaling and formatting received data so the pressure map can be optimally displayed. This data is then provided to the heatmap.js plugin for drawing onto the screen. HeatmapService exposes methods to scale the displayed pressure map to match the user's screen size, generate a configuration profile for the heatmap based on the current selected page, and package data received from the CalibrationService and BluetoothService for heatmap.js. Upon creation the service registers itself with the BluetoothService's callback for received data. After initialization, whenever the BluetoothService receives data, it calls the method registered by the HeatmapService. This method adds scaled coordinate information to the data and then notifies the view that new data has been received. We did not originally plan to need a separate service just for displaying pressure maps, but due to the limitations imposed by Ionic we had to build this service to manage the visual configuration without duplication of code.

Code Structure

Ionic structures the source code as a collection of nested modules; the organization of these modules is similar to an Angular project. Project-specific source code is stored in the "src/app" directory. Each service (HeatmapService, BluetoothService, CalibrationService) is stored in a directory called heatmap, bluetooth, and calibration, respectively. Ionic stores pages under a subdirectory named pages, and each page is able to store subpages as subdirectories. Within these directories, HTML files provide page structure, SCSS files provide styling information, and Typescript files provide the logic for the view and request routing.

Development Testing

Our project is dependent on the hardware designed by the Electrical Engineering team. This could have caused bottlenecks in our development process if they ran into any issues. To combat this problem, we structured our project to handle different types of input and planned several different testing methods. There is a local Bluetooth service that generates random data that can be displayed. This allows us to test the rest of the application without needing to rely on hardware. That still did not allow us to test any of the required Bluetooth connectivity. When parts were ordered, an additional Bluetooth module was ordered for us to use. This module was hooked up to an Arduino microcontroller. This microcontroller was programmed to send mock data like the local Bluetooth service. The mock hardware allowed us to test connection and data processing well ahead of the completion of the sensor pad.

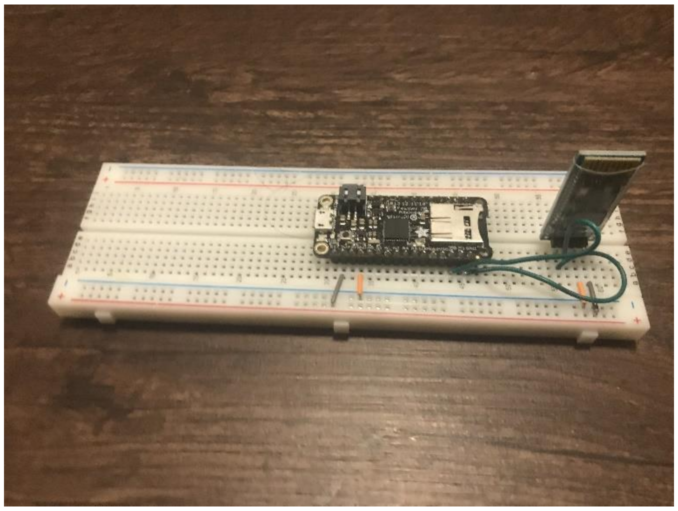


Figure 10: The Arduino microcontroller attached to the Bluetooth module used for development testing

Hardware Testing

The Hardware was tested at multiple stages throughout development. The first testing stage was to confirm that the microcontroller was operating as expected and communication between the MPLAB IDE and the microcontroller was correct. The Electrical Engineering team put eight LED lights on the multiplexer select lanes that allowed for this testing. Simple programs were created that toggled the LEDs to confirm that each of the I/O ports and the microcontroller were working correctly. These LEDs would also aid in knowing which sensor signal we were receiving at any given time. The next set of testing was the Bluetooth communication from the microcontroller. To do this a program was created that continuously sent signals from the microcontroller to the Bluetooth module. The module would broadcast the data and get picked up by the Bluetooth receiver on an android phone. The data would get displayed and verified on a Bluetooth terminal on the phone. When the signal was not correct an oscilloscope would be used on the leads going into the Bluetooth module from the microcontroller to see the signal and make adjustments. Once the Bluetooth module was verified the analog to digital converter could be tested by the Bluetooth module. The sensor pad was plugged into the PCB and the multiplexers were hardcoded to read only one sensor at a time. The potentiometer and A to D converter were configured and the sensor data was read into the microcontroller then sent out to the phone and read on the phone's Bluetooth terminal. To test different sensors, code was implemented to walk through the pad selecting one sensor at a time to confirm each sensor behaved correctly.

Integration Testing

At this point end-to-end testing with the app could begin. The full microcontroller code was implemented then tested using the application on a test phone. The phone was connected to a laptop while running the Chrome inspect tool from the DevTool suite. This allowed us to monitor what the app was picking up from the microcontroller. Instead of seeing one sensor value at a time we could see all 64 sensors at once every iteration. This configuration was used to test the full program on the microcontroller and monitor the sensor display. We were able to test the interaction between the app and microcontroller as well. By seeing the functioning application, we were able to make hardware and software adjustments to improve the readability of the sensor heatmap. Changing the scaling of the internal hardware cleaned up the sensor array and revealed new problems with the hardware and software. Most of these problems pertained to data formatting. We continued this testing process to ensure that the whole system was working properly. To see an extensive list of what we fixed, refer to the tasks that took place between 3/31/2022 and 4/14/2022 in schedule in section 4.5.

4.3 Risks

Risk	Risk Reduction
HIPAA compliance introduces legal complexity	With our architecture above the only information that may be saved are the calibration values and the connection information, which are not specific to any patient.
Sensor Array design is dependent on other teams	We developed mock sensor array data and a non-hardware dependent Bluetooth service to test the application while the prototype sensor array was being produced. We also tested our Bluetooth connectivity with an Arduino that sends mock data to a Bluetooth transmitter.
Creation of an iOS-compatible application would hinder progress on the prototype	We decided to only develop the application for the Android operating system to avoid duplicating effort and increasing costs on acquisition of Apple development hardware. Android was chosen because the development ecosystem is freely available, test hardware is simpler and cheaper to acquire, and because some team members already have experience developing Android applications.

4.4 Tasks

Below is a brief outline of the tasks we completed for this project. Please refer to section 4.5 Schedule for a more detailed task list.

1. Verify requirements with BMEG and ELEG teams to ensure understanding across teams.
2. Write the final proposal report based on feedback given to draft proposal report.
3. *(Beginning of Capstone II)* Create the team website to track progress.
4. Collaborate with the ELEG team throughout the semester to build the pressure pad system.
5. Create mock data of the sensor arrays to simulate the application without requiring the assembled sensor array.
6. Set up our development environments.
7. Create the BluetoothService to allow for connectivity through Bluetooth.
8. Create the CalibrationService to handle all computations related to calibration.
9. Create the HeatmapService to handle the heatmap display logic.
10. Create the Bluetooth page to allow users to connect to devices.
11. Create the Calibration page to allow users to calibrate the system with the current input from the sensor array.
12. Create the Pressure Map pages to allow users to view the current input from the sensor array adjusted for calibration.
13. Add events to update the Pressure Map pages
14. Continuously test the application with a Bluetooth module, Arduino, and mock data.
15. Test the application with the sensor array system.
16. Solicit feedback from BMEG and ELEG teams regarding application functionality and any required changes.
17. Implement required changes (if any).
18. Create the Final Report, Final Presentations, and poster.

4.5 Detailed Schedule

Tasks	Start by	Complete by
Verify requirements	11/1/2021	11/15/2021
Write final proposal report	11/15/2021	11/28/2021
Create the team website	11/28/2021	12/7/2021
Create individual pages	11/28/2021	12/7/2021
Install Ionic CLI and Android Studio	1/20/2021	1/27/2022
Define the Bluetooth interface	1/27/2022	2/3/2022
Create calibration service to set calibration array	1/27/2022	2/3/2022
Clear demo ionic screen	1/27/2022	2/3/2022
Add UI elements to screen	1/27/2022	2/3/2022
Update calibration data service to new requirements	2/3/2022	2/10/2022
Calibration Page: add calibration UI elements and connect to service	2/3/2022	2/10/2022
Read Bluetooth library docs and Bluetooth module docs	2/3/2022	2/10/2022
Research heatmap.js or find alternative	2/3/2022	2/10/2022
Create static mock data of the sensor arrays for internal testing	1/27/2021	2/14/2022
Collaborate with BMEG to improve UI	2/10/2022	2/17/2022
Create the heat map display logic	2/10/2022	2/17/2022
Add calibration service for separate top/bottom sensor arrays	2/3/2022	2/17/2022
Hook up Bluetooth module to an Arduino	2/10/2022	2/22/2022
Create the heat map display	2/17/2022	2/24/2022
Add persistent calibration	2/17/2022	2/24/2022
Change UI based on BMEG feedback	2/17/2022	2/24/2022
Add Bluetooth page	2/24/2022	3/03/2022
Create events to update data	2/24/2022	3/03/2022
Connect Calibration to Bluetooth	2/24/2022	3/03/2022
Create scaling logic for heatmap	2/24/2022	3/03/2022
Read documentation (hardware data sheets)	2/1/2022	3/03/2022
Create mock data of sensor arrays to be sent through Bluetooth	2/24/2022	3/03/2022
Test PCB with ELEG	2/14/2022	3/01/2022

Complete Preliminary Report and Presentation	3/08/2022	3/13/2022
Add temperature scale next to maps	3/03/2022	3/17/2022
Create calibration confirmation popup	3/03/2022	3/17/2022
Test sending data through Bluetooth module to phone	2/17/2022	3/17/2022
Add reset calibration method	3/03/2022	3/17/2022
Make pressure map colorblind friendly	3/03/2022	3/17/2022
Program Microcontroller	2/1/2022	3/14/2022
Add a color scale so users can easily read the pressure map	3/17/2022	3/24/2022
Add reset calibration button	3/17/2022	3/24/2022
Assemble the Sensor Array	3/01/2022	3/24/2022
Test the phone with the sensor array	3/24/2022	4/07/2022
Fix bug in capacitor-bluetooth-serial library	3/31/2022	4/14/2022
Fix how HC06 Bluetooth service formats data	3/31/2022	4/14/2022
Fix error in calibration service with HC06BluetoothService	3/31/2022	4/14/2022
Fix double click on calibrate button causes error	3/31/2022	4/14/2022
Give confirmation of Bluetooth connection	3/31/2022	4/14/2022
Fix ascii error in micro controller	3/31/2022	4/14/2022
Switch expected values from 0-100 to 33-125	3/31/2022	4/14/2022
Create Final Report, Presentation, and Poster	4/14/2022	4/25/2022

4.6 Deliverables

- **Design document:** Describes and diagrams the architecture of the completed application. Also describes and diagrams the hardware components of the system.
- **Project web site:** Archived project web site completed during Capstone II.
- **Android application source code:** Complete source code, build files, and layout files necessary to build and install the companion mobile application. The source code will be organized as an Android Studio project for easiest compilation and installation/emulation.
- **Final Report and Presentation:** Description of work completed, additional goals contained, and an outline of possible future work.

5.0 Key Personnel

Benjamin Allen – Allen is a senior Computer Science/Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed courses in mobile application development and software engineering, along with a longstanding internship as a software engineering consultant where he has developed web servers and mobile applications. His responsibilities are centered on the development of the mobile application with a focus on the pressure map display logic and backend organization.

Clay Griscom – Griscom is a Senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed System Synthesis and modeling and Circuits and Electronics which will help in the hardware development of the project. Responsible for aiding in the development of the communication hardware between the application and sensor array.

Hugo Serrano – Serrano is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Mobile Programming and Information Retrieval which applies to this project. He is responsible for application development and UI/UX.

Kira Threlfall – Threlfall is a senior Computer Science and Pure Math major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed Algorithms and Information Retrieval which are relevant to application development. She has been a Mobile Application Development Intern for J.B. Hunt and a Software Engineering Intern for Arvest where she learned about mobile application development and API development. Threlfall is responsible for application design and data manipulation.

David Whelan – Whelan is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses. He has taken Ubiquitous and Wearable Computing and Mobile programming which directly applies to this project. He has also interned at Marshalltown Tools and Ayoka LLC where he gained experience in business process and working with large code bases. He is responsible for team coordination and Bluetooth connectivity.

Jennifer Steinauer, PTA and Nathan Jowers, PT – Steinauer and Jowers work for the UAMS outpatient clinic and provided our team with technical support and used their in-the-field experience to guide us.

Mike and Josh Fohner – The Fohners have donated a wheelchair for use during the hardware development and testing of this project.

6.0 Facilities and Equipment

Members of the CSCE used their personal computers to develop the Android application. We checked out Android phones from the department in order to test the Bluetooth interface. For testing before the sensor array system was complete, we used an HC-06 Bluetooth module and an Arduino.

7.0 References

[1] Tekscan Pressure Measurement System

https://www.tekscan.com/products-solutions/systems/body-pressure-measurement-system-bpms-research

[2] Xsensor ForSite SS

https://www.xsensor.com/solutions-and-platform/csm/wheelchair-seating

[3] Blue Chip Medical Products inc. MeasureX Pressure Mapping system

https://www.bluechipmedical.com/seating-positioning/pressure-mapping-for-seating-positioning/

[4] BodiTrak2 Wireless IoT Pressure Mapping System

https://www.boditrak.com/products/medical/wheelchair.php

[5] Introduction to the viridis color maps.

https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html

[6] Ionic Framework, https://ionicframework.com/

[7] Heatmap.js, Patrick Wied, https://www.patrick-wied.at/static/heatmapjs/

[8] capacitor-bluetooth-serial, https://github.com/agro1desenvolvimento/capacitor-bluetooth-serial