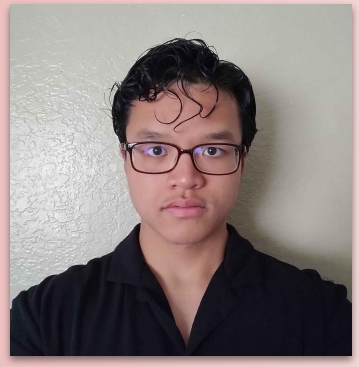
EASE : Notifying Feminine Hygiene Pad

Senior Design Group 29

The Team



Alexander Nguyen

Electrical Engineer

Matthew Poole

Electrical Engineer

David Garzon

Computer Engineer

Dillon Sardarsingh

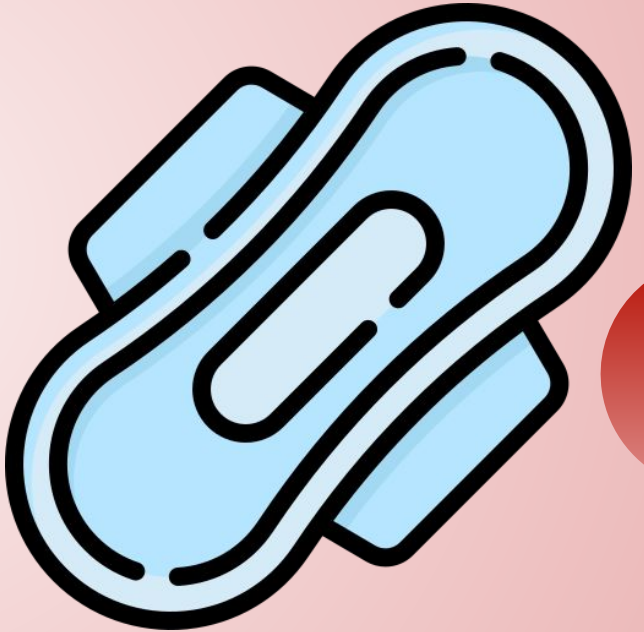
Electrical Engineer

Motivation and Background

- Some menstrual cycles can be unpredictable which makes it difficult to determine
  - **When** to replace a fully used hygiene product
  - **How often** to replace such products
- Can lead to leakage
  - Embarrassment
  - Inconvenience
  - Messy
  - Ruins clothing
- Presented to us by **Gabriela Mercado** , a finance student at the University of Central Florida
  - Developed the concept for this project when she was 16
  - Resulted from having multiple leakages herself, leading to embarrassment and stained clothing
  - Saw the lack of products currently on the market that addressed these problems
- Our team is motivated to positively impact and directly improve the daily lives of everyday people through the development of this product

Goals and Objectives

- Accurately detect when a menstrual product is at capacity or nearing capacity.
  - Perform the aforementioned detection with an accuracy of 75% or higher
  - Comfortable and discreet form factor
- Successfully notify its user
  - Reliably send data between sensor and alert system
  - Threshold notification at 75% and 95% capacity
  - Immediate notification of perimeter leakage
  - Fast response times
- Battery life long enough to last a typical period cycle of approximately 7 days



Engineering Specifications

EASE Notifying Feminine Hygiene Pad	
Quantitative Metrics	
Max. Surface Area	125 cm^3
Max. Weight	10 g
Temperature Limitations	Up to 46℃ [3]
Environmental Rating	IP 35 [4]
Volume Accuracy	2 mL
Response Time	Transmits within ~1 s of product reaching ~75% and 95% capacity
Battery Life	7 Days
Performance Reliability	Provides an accurate notification 90% of the time

Engineering Specifications

Qualitative Metrics	
User Interface	Data should be displayable via an app on a mobile device
Communication	Transmitter should be capable of communicating real-time data to the user's mobile device, including current blood volume within the hygiene pad when prompted.
App Specifications	Sends a notification when pad is nearing maximum capacity
Physical Specifications	Compatible with the most common types of external feminine hygiene products
	Product is reusable
	Comfortable and lightweight such that it is virtually unnoticeable when in use
	Pliable and able to conform to reasonable shapes as necessary for the user
	Able to withstand contact from bodily fluids including but not limited to blood, water, sweat, and discharge
	Device is safe for its expected application

Hardware Selection

MCU Part Selection

- ESP32-WROOM-32 Module for prototyping
- Surface Mount QFN and VQFN Packaging

	TI CC2640R2L	**STM** STM32WBA54KGU6	**Infineon** CYW207
Cost	$4.17	$7.39	$6.56
Packages	VQFN, QFN	UFQFPN	QFN
Size	5.1x5.1 mm	5x5 mm	5x5 mm
Total Pin and GPIO Count	32, 15	32, 18	40, 16
Development Environment	Code Composer Studio IDE	STM32CubeIDE 3rd Party IDEs	Support Discontinued

Sensor Technology

Feature	Time of Flight Sensors	Capacitance Sensors	Pressure Sensors	Float Level Sensors	Solid State Sensors
Accuracy	High in controlled environments	Less accurate, needs calibration	Versatile and reliable	Needs defined, static fluid container	Discrete data, less continuous accuracy
Simplicity	Complex setup required	Simple, easy to use unconventionally	Varies by type (e.g., piston, capacitive, piezoelectric)	Simple but needs auxiliary sensor	Very simple implementation
Environmental Dependency	Sensitive to environment	Needs dielectric constant of liquid	Can work with varied environments	Limited to specific container setups	Requires conductive liquid
Best Use Case	Controlled environments	Small, defined environments	Versatile applications	Large, defined fluid containers	Flexible applications
Drawbacks	Needs controlled environment, large	Less accurate, needs calibration	Can be large, difficult in small spaces	Not intuitive for small scale	Discrete data, susceptible to wear

Capacitance to Digital Converter

Parameter	TI FDC1004	AD AD7746	Microchip CAP1298	TI FDC2214	ADI AD7156	Maxim MAX11200	NXP PCF8883
Resolution	16-bit	24-bit	8-bit	28-bit	12-bit	24-bit	N/A (Touch)
Range	±15 pF	±4 pF	Touch-Based	Up to 250 pF	Automatic Calibration	External	Touch-Based
Baseline	0 - 100 pF	0 - 17 pF	N/A	N/A		N/A	N/A
Interface	I^2C	I^2C	I^2C	I^2C	I^2C	SPI	GPIO (Touch)
Conversion Rate	100 samples/sec	45 Hz	N/A	High-Speed	Fast	Medium	Fast
Package	TSSOP-16	TSSOP-16	Various	QFN-16	MSOP-8	QFN, TSSOP	SOT-23-6
Temperature Compensation	✓	✓	X	✓	✓	X	X

UART to USB Bridge IC

	Microchip MCP2221A	Silicon Labs CP2102	FTDI FT230X	FTDI FT260
Baud Range	300-460800bps	300-100000bps	300-3000000bps	1200-12000000bps
Voltage Range	3.0 to 5.5V	3.0 to 5.25V	3.3 to 5.25V	1.8 to 3.3V
Package	TSSOP-14	QFN-28	SSOP-16	TSSOP-28
Price	$2.53	$5.59	$2.26	$1.99

Battery Selection

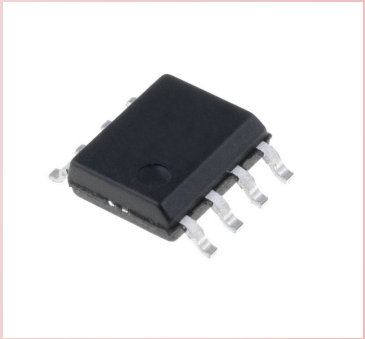
	Attribute			
Type of Battery	Size	Safety	Rechargeable	Output Voltage
Lithium Ion	Varies	Prone to exploding if not handled correctly	Yes	3.7V / cell
NiMH	Varies	Generally safe	Yes	1.2V
Coin Cell	⌀ 20 mm x 3.2 mm	Safe	No	3-3.6V
Alkaline	Varies	Safe	No	1.5V

Voltage Regulator Technology

	Linear	Switching Regulators		
Specifications	**LDO**	**Buck**	**Boost**	**Buck-Boost**
Efficiency	Moderate (60-90%)	High (up to 95%)	High (up to 95%)	High (up to 95%)
Heat	Moderate	Low	Low	Low
Step-up or Step-down	Step-down	Step-down	Step-up	Both
Complexity	Simple	Complex	Complex	Complex

Voltage Regulator Part Selection

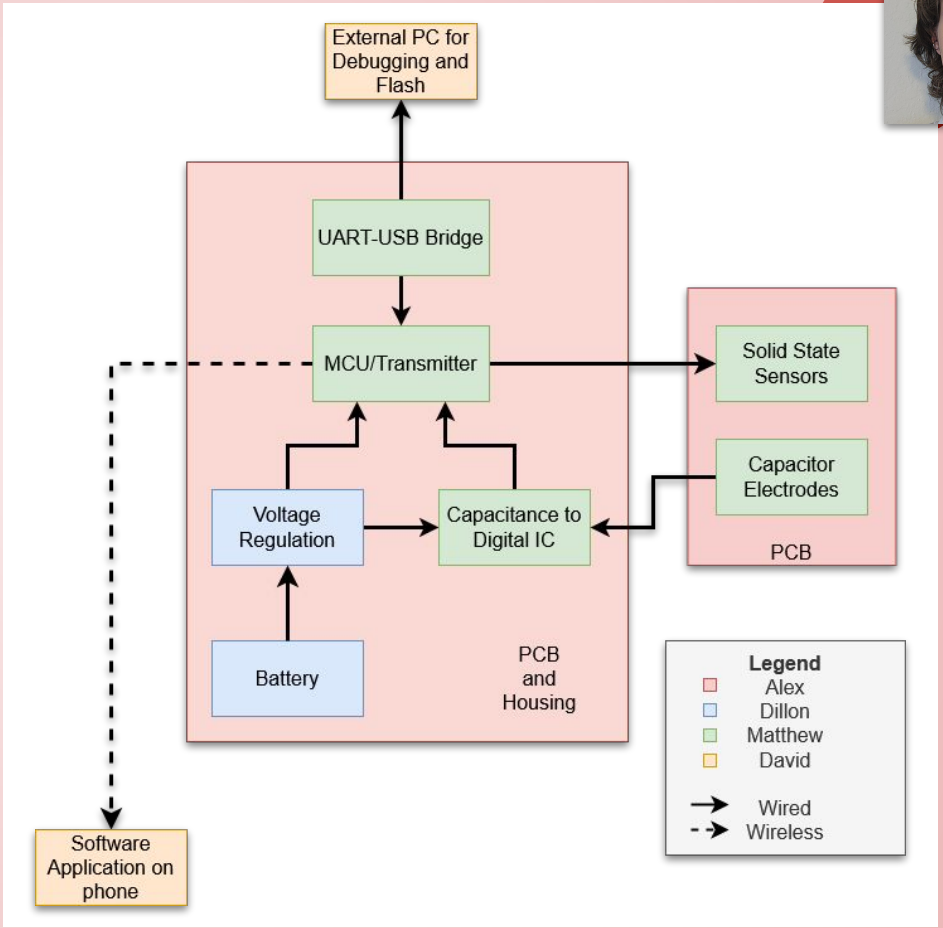
	LDO		Boost		
Voltage Regulator	**MCP1700**	**TPS 783**	**TPS61023**	**Comidox Boost Converter**	**LTC1682-3.3**
Voltage Output	1.8, 2.5, 3.0, 3.3, 5.0V options	1.8, 1.9, 2.6, 3, 4.2V options	5V at 1 amp	5V at 480 mA	3.3V at 50 mA
Quiescent Current	1.6 microamps	0.5 microamps	N/A	N/A	N/A
Dropout Voltage	178 mV at full load	175 mV at full load	N/A	N/A	75-120 mV
Size	4.62 x 4.71 x 3.62 mm	8.12 mm² 2.9 x 2.8	17.8 x 11.3 x 5.6 mm	11 x 10.5 x 7.5 mm	5.01 x 6.18 x 1.44 mm



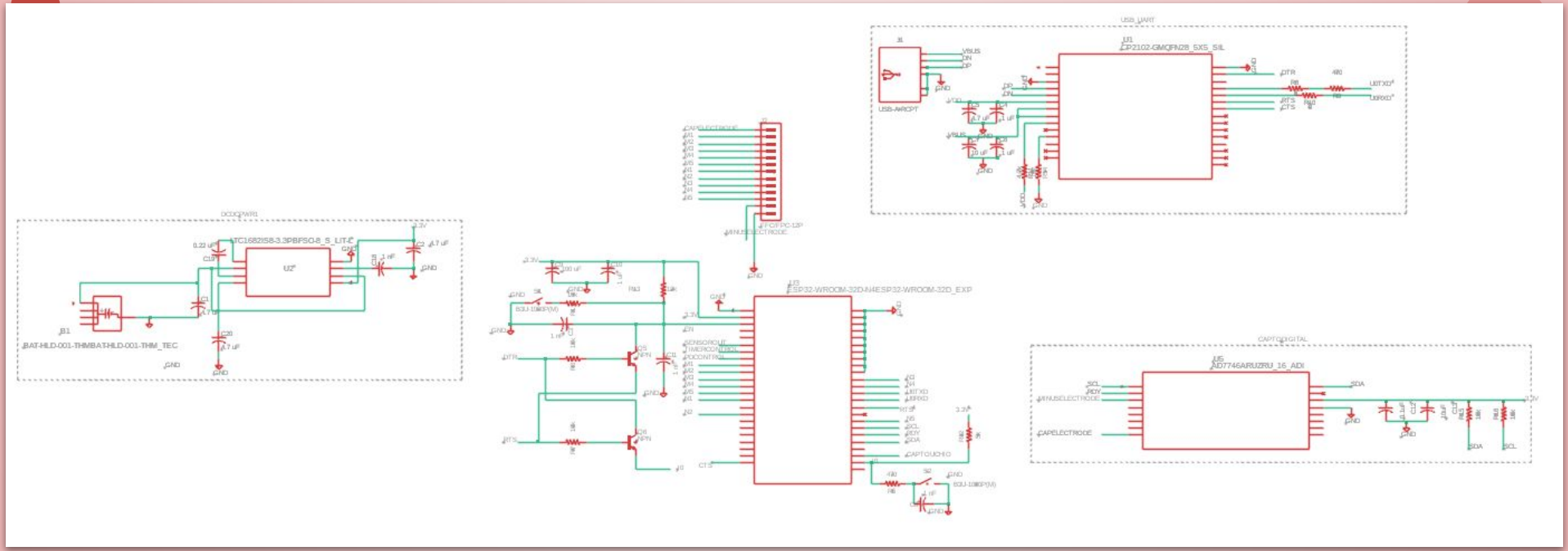
Hardware Design

Block Diagram

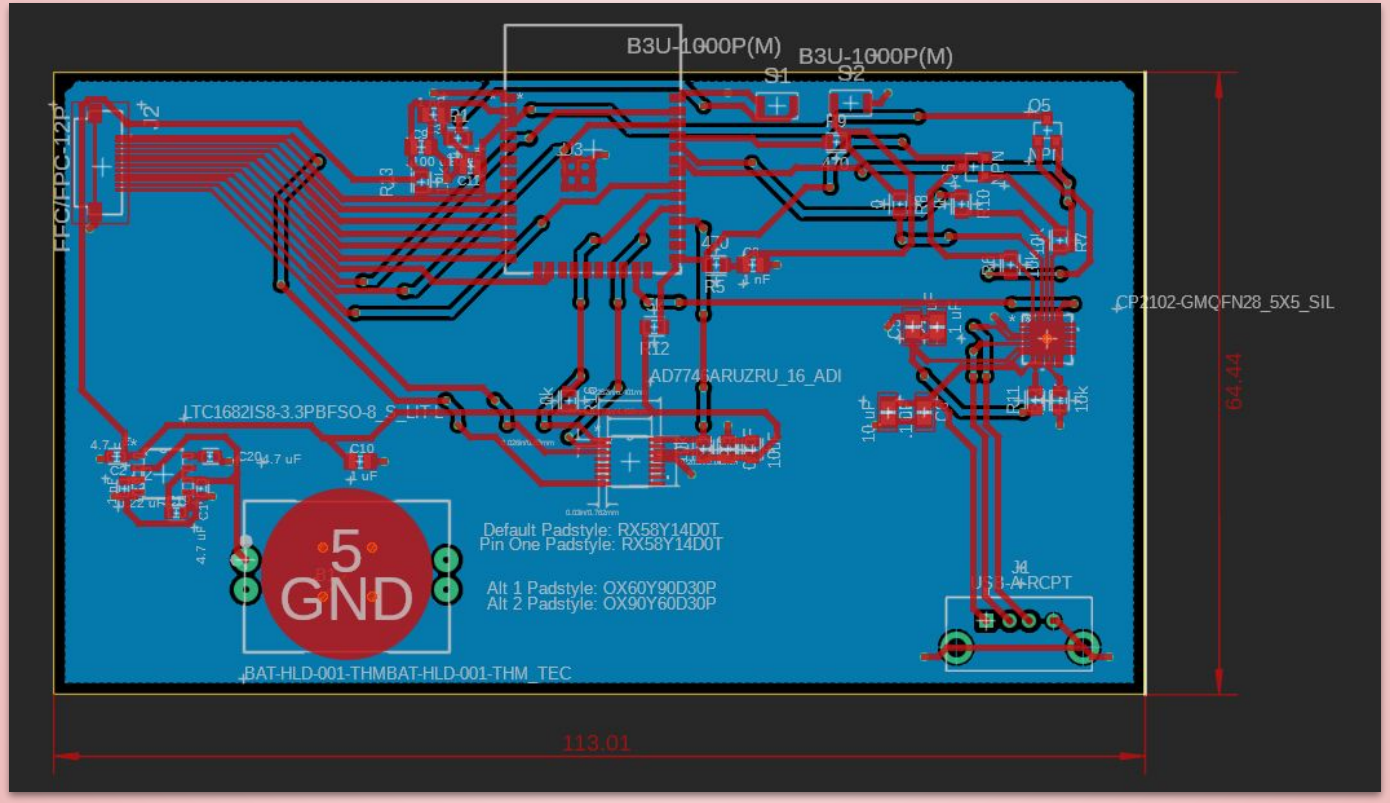
- Small subsystem count for miniaturization purposes
- Separation of sensor and data processing PCBs to achieve size specifications



Main PCB Schematic



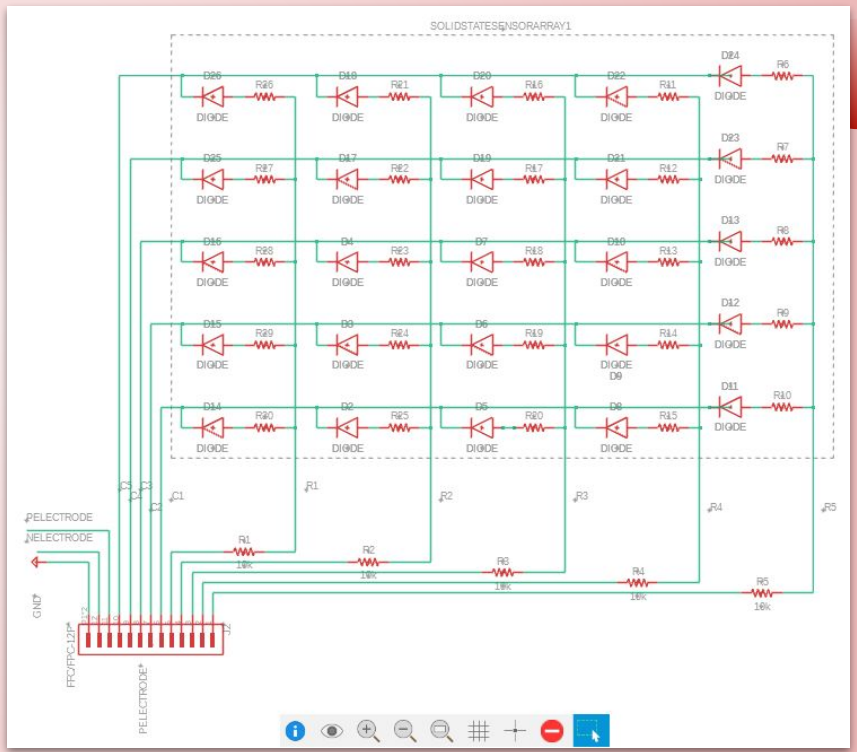
Main PCB Layout



Electrode PCB

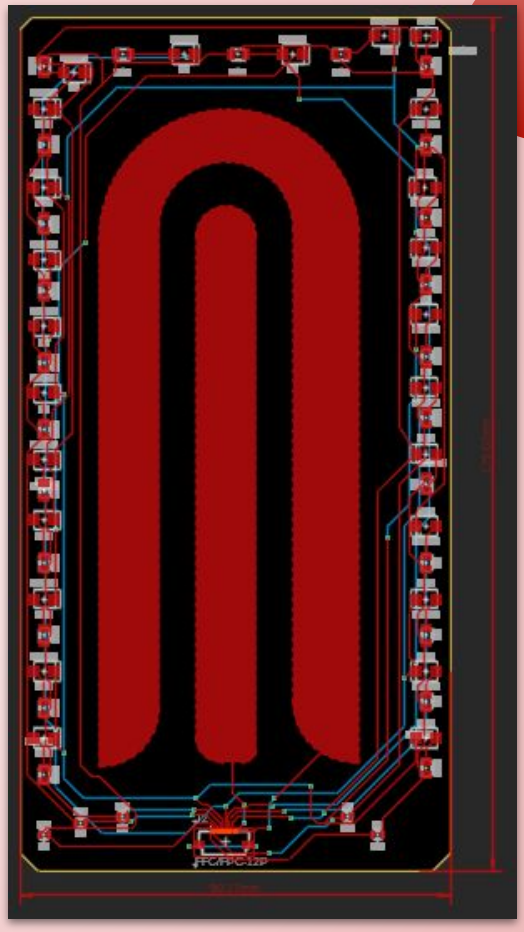
- The solid state sensors are setup in an array similar to a keyboard matrix. This way we can maximize our GPIO usage.

- The solid state array is accompanied by the fringing field capacitor electrodes.



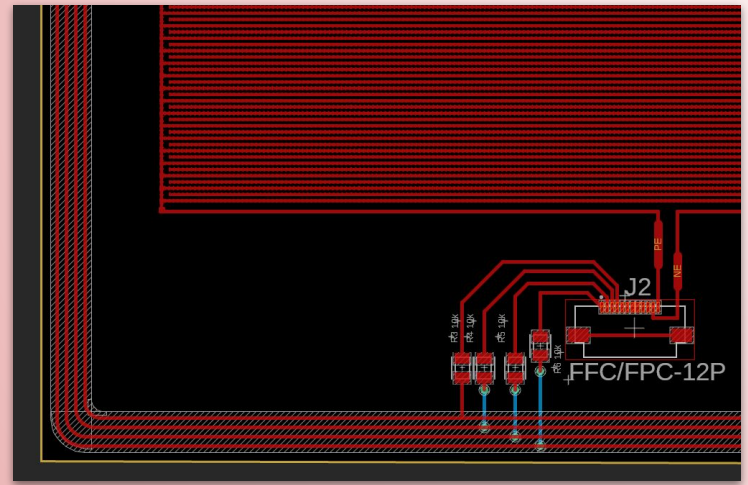
Electrode PCB Layout

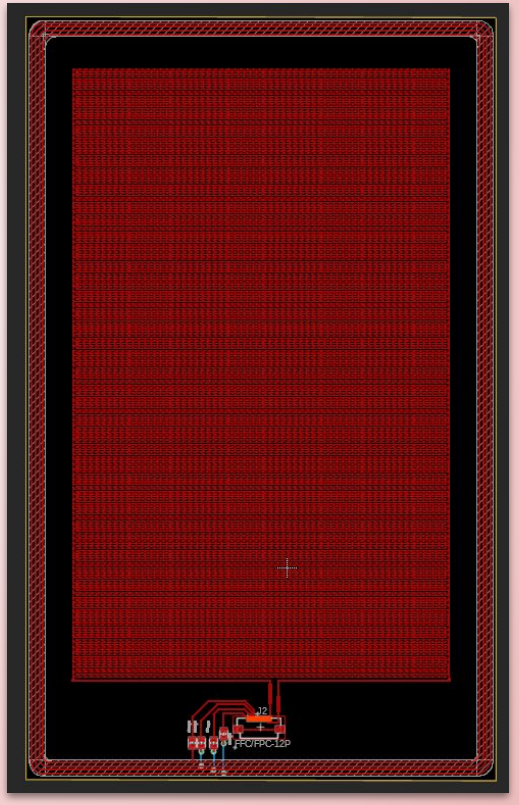
- The matrix array sits in a ring along the outside of the capacitor electrodes. This way the solid state sensors provide supplemental data to the capacitive sensor.
- The capacitor electrodes use the fringing field for contactless sensing. A copper pour polygon is used here and effort is made to keep bottom layer traces way from the electrodes to reduce noise.
- The capacitor electrodes and diodes all need to be waterproofed while the resistor pads need to stay exposed.



Supplemental Electrode PCB layout

- Solid state array removed in favor of concentric rings.
- Capacitor electrode design can utilize the extra space and achieve more precision with an interdigitated design.





Software Selection

Development Environment Selection

In order to select our development environment for the ESP32, we must compare our options:

IDE	Arduino IDE	VS Code + PlatformIO Extension	Espressif IDE (Eclipse based)	Eclipse + ESP-IDF Plugin
Ease of Use	Beginner-friendly	Intermediate	Advanced	Advanced
Setup Complexity	Minimal	Moderate	High	High
Supported Languages	C, C++	C, C++, Python	C, C++	C, C++
Debugging Support	Limited	Advanced (JTAG, GDB)	Full Debugging (JTAG, OpenOCD)	Full Debugging (JTAG, OpenOCD)
Code Intellisense	Basic	Advanced	Advanced	Advanced
RTOS Support	Limited (via libraries)	Yes (ESP-IDF, FreeRTOS)	Yes (ESP-IDF, FreeRTOS)	Yes (ESP-IDF, FreeRTOS)
Integration	Basic (via board support)	Full (via PlatformIO)	Native	Native
Best For	Quick prototyping	More flexible with multi-board support and debugging	Official ESP32 IDE with deep integration of ESP-IDF	Similar to Espressif IDE but with manual setup

Development Environment Selection

In order to select our app related development environment, we must understand our needs:

- We want to build a mobile app for both iOS and Android devices
  - Ideally, our development environment should natively allow cross-platform development
- We want to be able to test our app easily and with multiple different devices
  - If local simulation is possible, then this brings a huge advantage instead of needing a local device at all times
- We want to be able to take in the BLE communication given by the ESP32 and perform actions based on that communication
- We want to be able to send a push notification to the device after certain actions are met

Development Environment Selection

In order to select our app related development environment, we must compare our options:

Environment	Flutter	React Native	Swift	Kotlin
Primary Language	Dart	JavaScript / TypeScript	Swift	Kotlin
Target Platforms	Android, iOS, Web, Desktop	Android, iOS, Web	iOS only	Android only
UI Framework	Flutter Widgets	React Native Components	SwiftUI / UIKit	Jetpack Compose / XML
Performance	Near-native	Good	Native	Native
Live Preview	Yes	Yes	Yes	Yes
IDE Options	VS Code, Android Studio, etc	VS Code, WebStorm	Xcode	Android Studio
Ease of Learning	Moderate	Easy (if familiar with JS)	Moderate	Moderate
Best For	Cross-platform UI, High Performance	JavaScript developers, hybrid applications	iOS-only applications	Android-only applications

Programming Language Selection

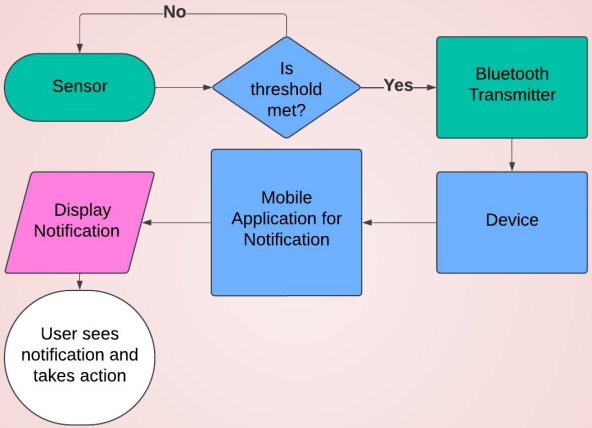
We had several programming language options to select from based on our Development Environment but the chosen programming languages are below:

- Embedded Development
  - C++ as our programming language
    - Has the most native support for Arduino IDE and most code examples are given in C++.

- App Development
  - Flutter framework
    - Chosen for cross-platform development and chosen over React Native for BLE testing simplicity
  - Dart as our programming language
    - Native supported language for Flutter development.

Software Design

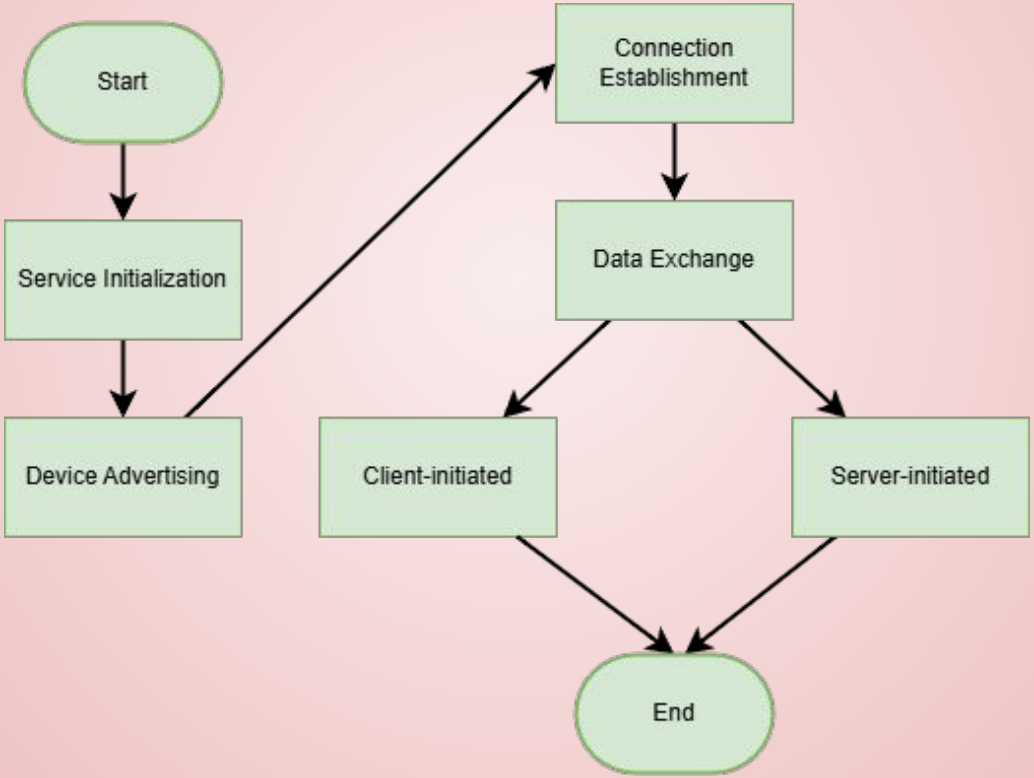
System-level Flowchart



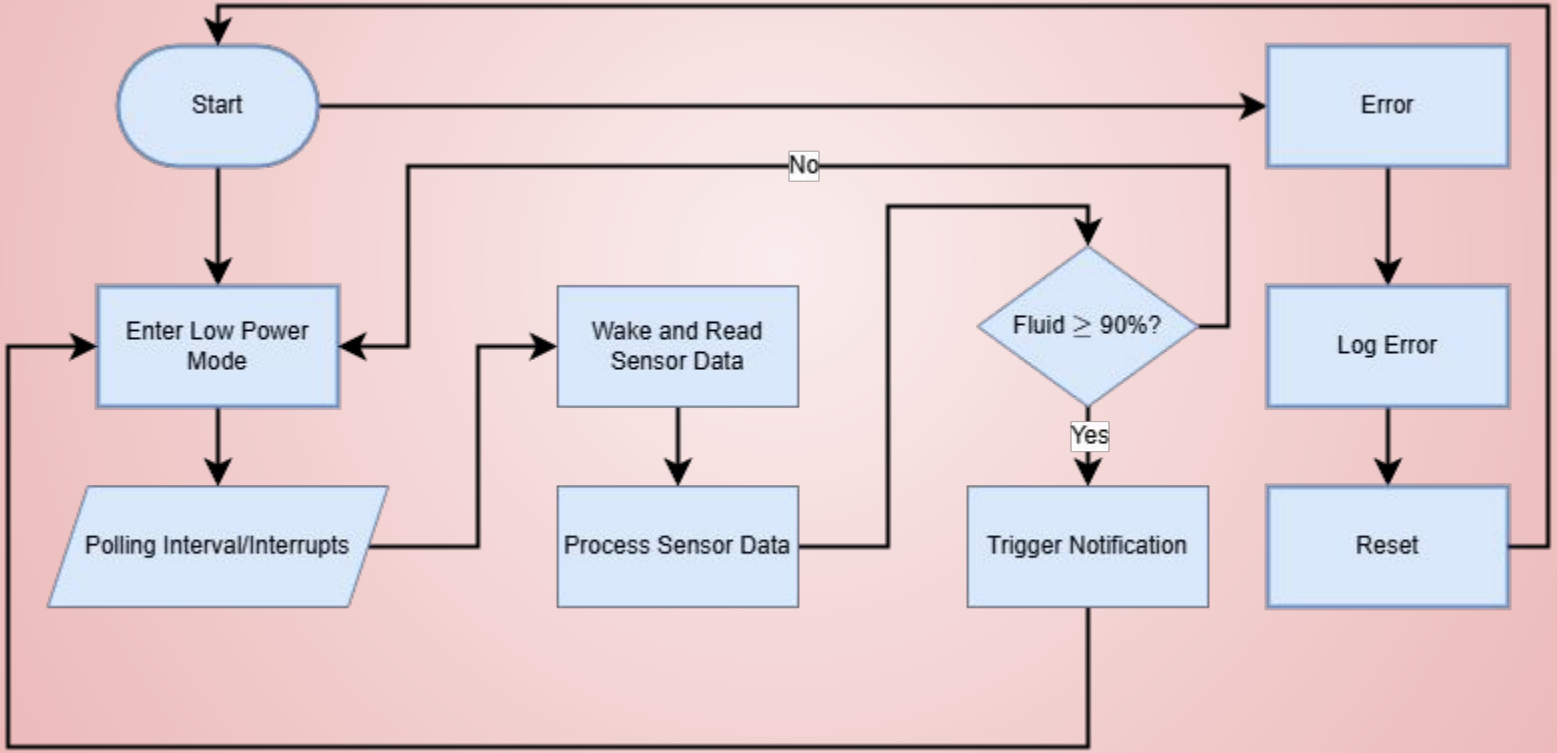
Flowchart Legend

- Hardware (Alex, Dillon, Matthew)
- Backend Software (David)
- User Facing Software (David)

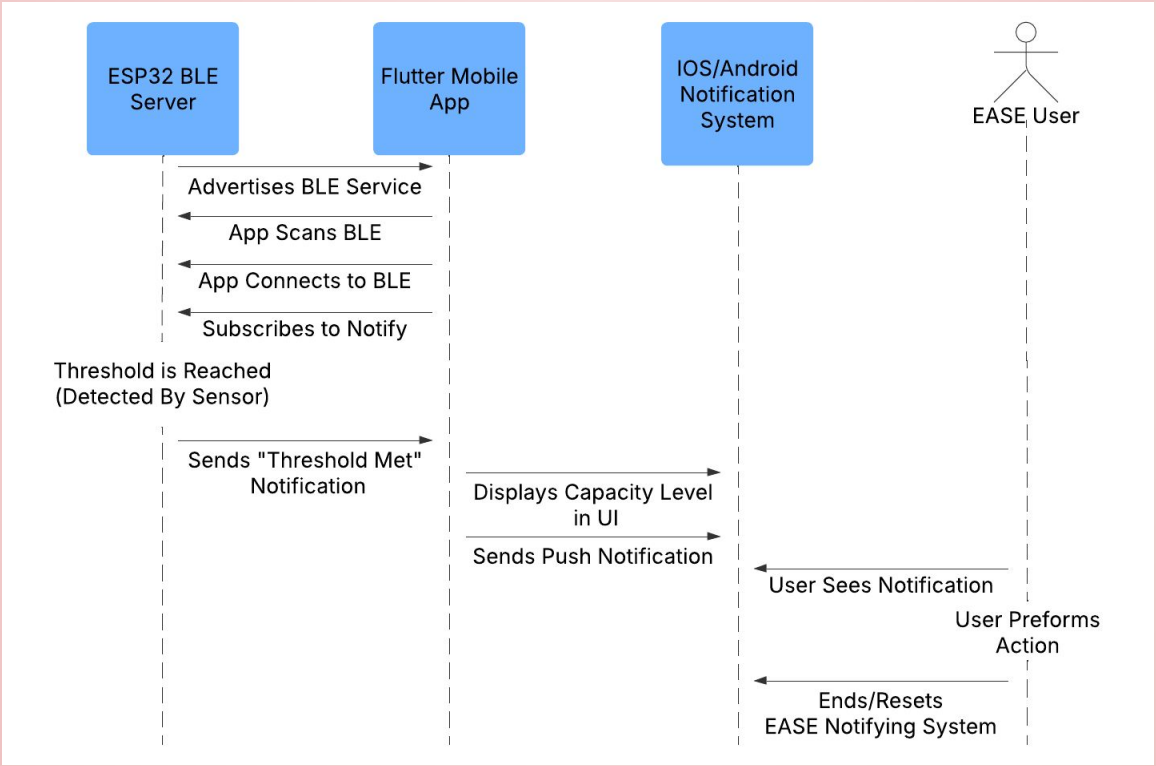
Bluetooth (BLE) Flowchart



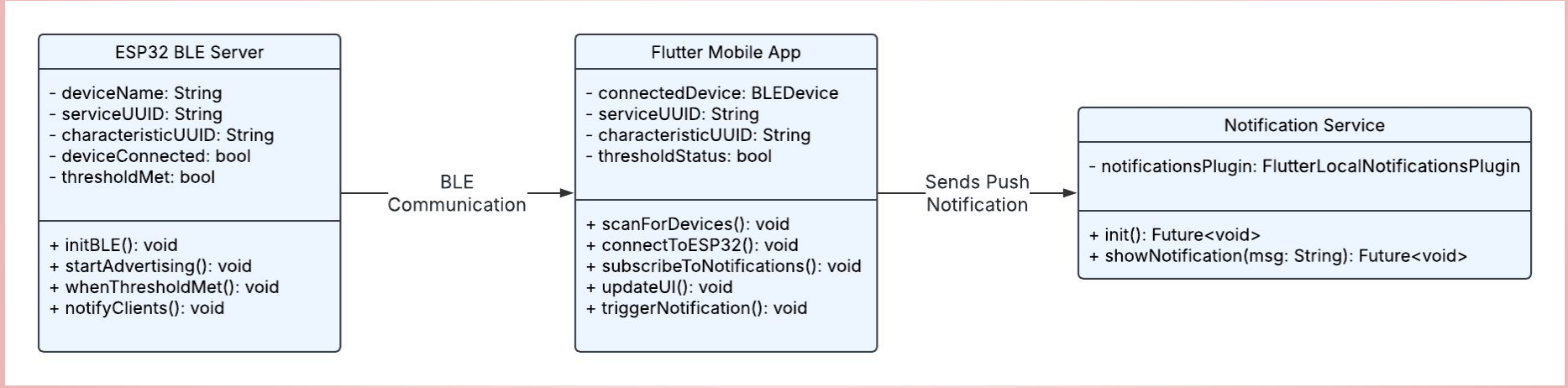
Sensor Code Flowchart



Notification Sequence Diagram

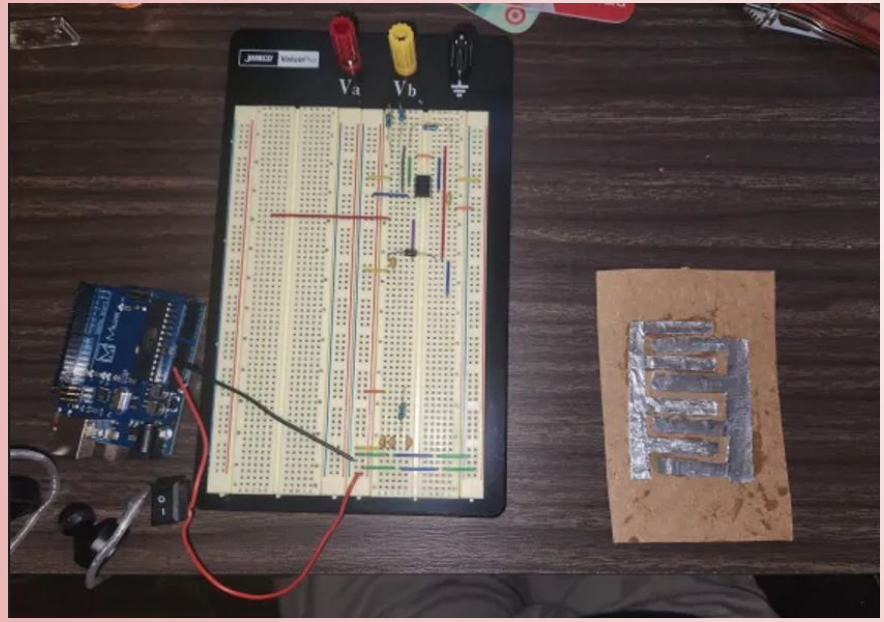


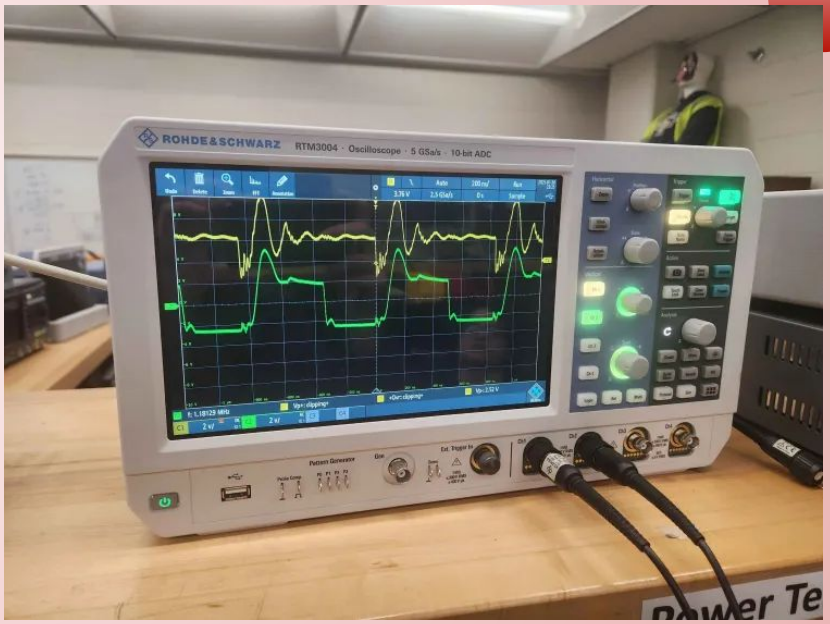
Notification Class Diagram



Prototyping & Testing

Initial Breadboarding





Hardware Testing

- Capacitive Sensing
  - Range should be defined and then translated across a 0% and 100% capacity of dielectric
  - Sensitivity of dielectric must be tested to show variability
  - Values of the capacitance to digital converter must be verified to be reflected of the defined range
  - Consistency of applied blood volume at 75% and 95% capacity
- Solid State Sensing
  - Verify that sensors trip upon immediate contact with fluid

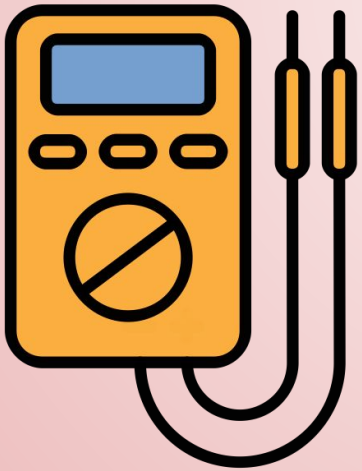
Software Testing

- Embedded Development
  - We need to ensure the ESP32 can be found as a Bluetooth server for pairing on different platform devices. This can be tested locally with different devices like iOS and Android mobile devices.
  - We need to ensure the threshold sensor notifier works consistently and accurately by repeated local testing.

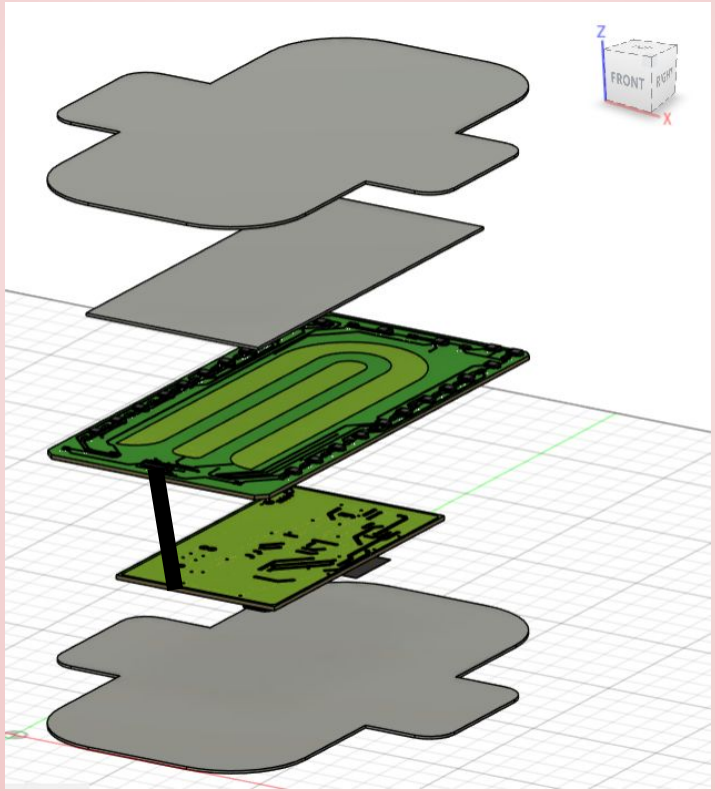
- App Development
  - We need to ensure cross-platform front-end styling is consistent among devices by simulation testing and live demo on different local devices.
  - We need to ensure BLE (Bluetooth Low Energy) features work on different devices with a smooth set-up.
  - We need to ensure post-notifications appear on different platform which can be tested via simulation and locally with different devices.

System Testing

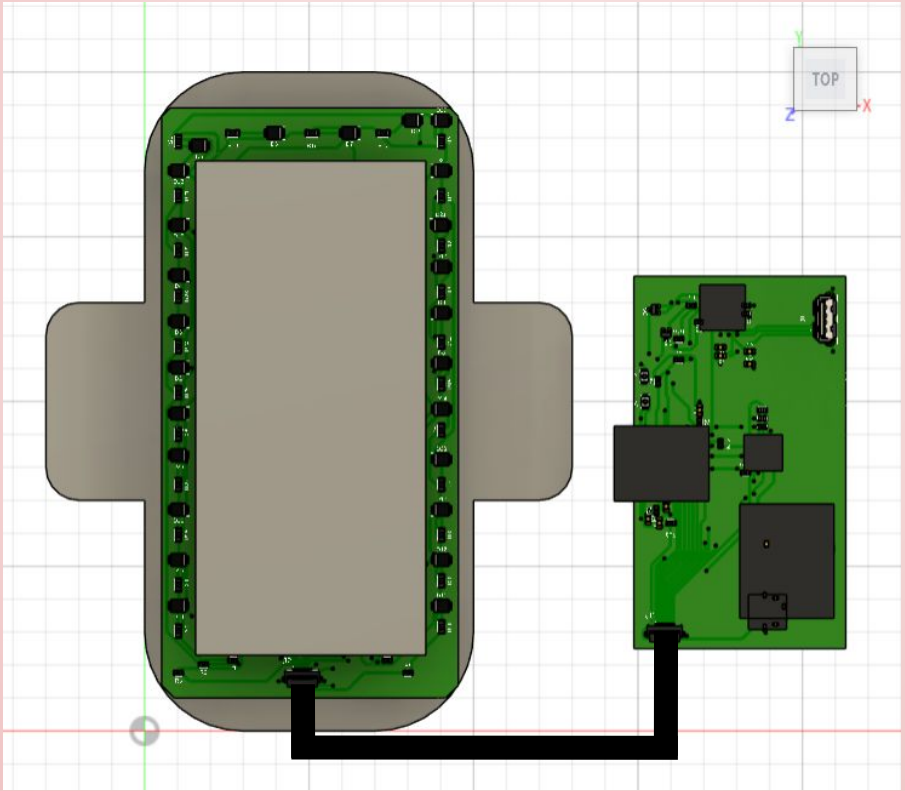
- Applied synthetic blood in mL increments
- Consistency of app notification at the specified thresholds
- Repeated testing to verify 90% reliability
- Testing of physical integration as a feminine hygiene product



Ideal Product Integration



Testing Product Integration



Administrative Content

Budget

	Part	Cost	Quantity	Subtotal
1	Voltage Regulator	$6.98	1	$6.98
2	Buttons	$0.70	2	$1.40
3	Battery Holder	$0.36	1	$0.36
4	Flex Cable Connector	$0.53	2	$1.06
5	USB Connector	$1.30	1	$1.30
6	Cap to Dig Converter	$14.58	1	$14.58
7	NPN Transistor	$0.03	2	$0.06
8	6 Inch Ribbon Cable	$3.84	1	$3.84
9	Diodes	$0.26	24	$6.28
10	SMD Resistors and Capacitors	$0.03	1	$0.03
11	Rigid PCB	$1.60	1	$1.60
12	Flexible PCB	$6.78	1	$6.78
				$44.27

Work Distribution

Component	Primary Responsibility	Secondary Responsibility
Sensor	Matthew Poole	Alexander Nguyen
Capacitance to Digital Converter	Alexander Nguyen	Matthew Poole
Software Application	David Garzon	Dillon Sardarsingh
MCU	Matthew Poole	David Garzon
Power Supply	Dillon Sardarsingh	Alexander Nguyen
Housing Unit / PCB	Alexander Nguyen	Matthew Poole
Administrative	Alexander Nguyen	

Progress & Plan for Completion

1-2 Weeks | 3-4 Weeks | 5-6 Weeks | 7-8 Weeks

Present → **Midterm Demo** → **Live Demo**

Test Capacitive Attributes of PCB

Verify system is capable of meeting the specified engineering specifications and metrics

Determine which padding integration optimally supports real-life application of the product

* Troubleshoot as necessary

Thank you!

Are there any questions?