3. DESIGN APPROACH

The SafePaws vest is a smarter way to track a dog's vitals remotely. The vest is integrated with a heart rate sensor and a body temperature thermometer to monitor one's dog while on-the-go. The Safe Paws vest is designed to collect data and measurements autonomously, while also allowing the ability to control it wirelessly via a mobile application. This section explores the final design choices, including considerations of multiple design options, an overview of the system, and detailed descriptions of each subsystem. Our design approach emphasizes data accuracy, a friendly user-experience, and pet safety. All of which is guided by the project requirements, constraints, and engineering standards.

3.1. Design Options

The objective of this section is to provide a detailed overview of the design choices considered. Two major design options were explored, each with its unique advantages and limitations. Background is provided for each option, explaining the decision-making process and design justifications. By providing an outline for the benefits and drawbacks of each design, we further elaborate on why the chosen components will provide optimal solutions for achieving the project's objectives. The design options were evaluated within the boundaries of our project requirements, constraints, and engineering standards, to ensure that they are both practical and satisfy the required technical goals.

3.1.1. Design Option 1

A low-profile collar was considered in the original design approach. This design involves a collar with the sensors built into it, along with pockets for the controller and battery. A collar has the advantage of being smaller than a vest and can be secured to a dog to provide consistent conditions for sensors to make their readings.

The collar design has its advantages reduced when the other required components are factored into its design, particularly the microcontroller and battery. The degree of durability required for the components could not be properly addressed. As the battery, microcontroller, and sensors must be secured to its substrate (collar and pockets), and stand up against the elements, normal wear-and-tear, and from dogs themselves.

3.1.2. Design Option 2

Another design option that was considered was an integrated vest. This approach incorporates a similar design. Where the vest stores the technical components, including the sensors, controller, and battery. A vest would allow for the technical components to be more spaced out throughout the design and allow the microcontroller and battery to be better secured to the substrate (inside of the vest) to minimize their movement. The advantages to this design include pet comfort and more space for components to be more easily integrated. A vest also allows for the components to be not compacted into a weight around its neck and be more comfortable for the dog. The limitations are bounded by the vest's size, but incorporating straps will allow it to be adjusted for a secure fit for a range of sizes. For these reasons, the vest design was ultimately chosen.

3.2. System Overview

This section provides an overview of the SafePaws vest, exploring the system's functionality, how it connects with the subsystems, and how each subsystem interacts with one another.

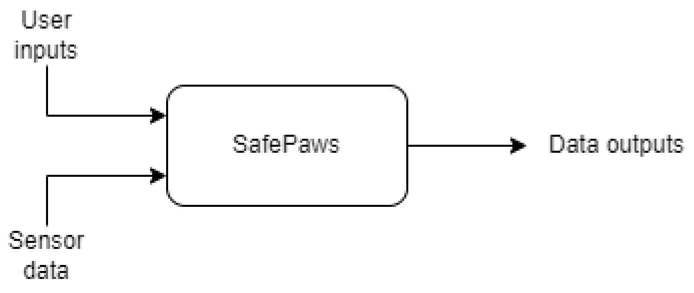

Figure 3-1: SafePaws System at a glance (Level 0)

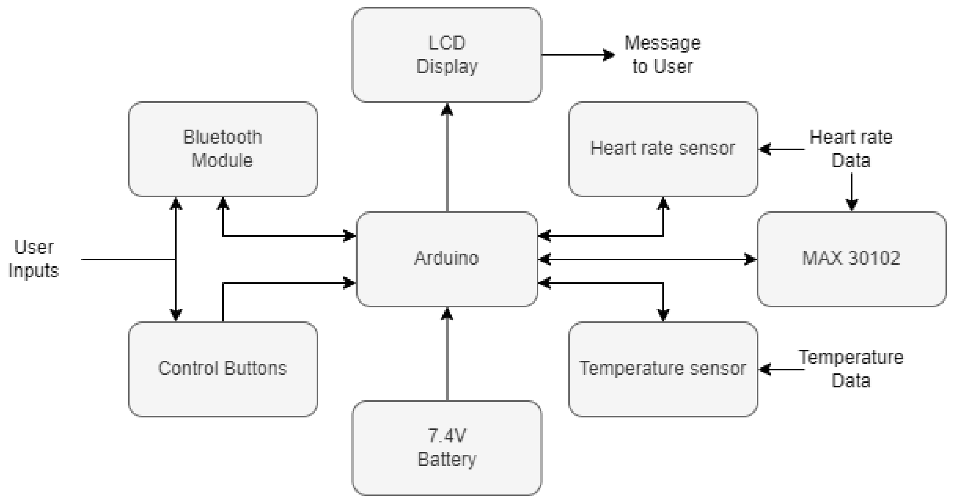

Fig. Error! No text of specified style in document.**-2: SafePaws Functionality (Level 1)**

In this section, we list the components that will make up the primary system and each of its subsystems. Along with the functionalities that they will provide for the project and their most important technical specifications. In section 3.2.1, the other microcontrollers considered for the project are also listed.

3.21. Microcontroller

The microcontroller that was chosen for the project was the Arduino Uno REV3. It has several features that allow for integration of sensors, peripherals, and a well-supported software library. It comes with several connectivity options for plug-and-play sensors, including support for both analog and digital devices. Off-the-shelf peripherals are readily available and will allow for easy installation implementation of 802.11-based devices for Bluetooth and/or Wi-Fi connectivity. A myriad of software libraries is available for implementing the peripherals and allow for customization to suit the needs of the project. In Table 3-1, devices considered for the project are listed, along with the advantages and drawbacks that they could each provide.

Table 3-1: Microcontroller Considerations

Device	Advantages	Drawbacks
Arduino Uno REV3	• Provides support for both analog and digital sensors, which allows for more sensors to be considered. • C++/C-based, native code can be run directly on the hardware (no pre-compiled executables are required). • Wide range of hardware peripherals are available. • Large, well-supported software library is available for use by its developer community. • Low power consumption.	• Slightly larger profile versus other microcontroller options. • Low storage available; requires additional SD card peripheral to store files. • Low memory (32 KB) [1].
Basys Board	• Lowest power consumption of all microcontrollers considered. • HDL-based assembly language used to program the Basys Board requires less hardware resources to run [2].	• HDL-based programming (more difficult and time-consuming to write and debug) • Hardware sensor support (fewer options available when compared to Arduino) • Less support for their software library when compared to other options. • Best suited as a field programmable gate array (FPGA) rather than as a microcontroller [3].
Raspberry Pi	• Most powerful processor out of all the microcontrollers considered (1.4 GHz). Large built-in memory (1GB+) [4]. • Native SD-card slot. • Multiple programming languages are supported via compiled executables (C, C++, Python, Java, etc.). • Large, well-supported software libraries available by its developer community. • Native USB, Bluetooth, Wi-Fi, and network support [4]. • Low-power consumption.	• Software development-focused device. Thus, fewer hardware/sensor options are available [5]. • Does not include built-in support for analog sensors.

The table above, Table 3-1, describes the advantages and disadvantages from exploring options with multiple microcontrollers.

3.3. Subsystems

The prototype for SafePaws includes three subsystems. The three subsystems are the following: sensors, power systems, and software.

3.3.1. Subsystem 1

For the heart rate sensor, a photoplethysmography sensor was chosen. Other types of sensors looked at were Piezoelectric which uses vibration generated by heartbeat to determine heartrate, and Electrocardiogram which involves placing electrodes on skin contact to detect heartrate. A photoplethysmography, ppg, uses light to monitor blood within the body. Choosing photoplethysmography (PPG), the sensor utilizes an LED as its light source, emitting light at precise wavelengths to detect changes in blood volume within the body. Pulse Sensor and MAX30102 were chosen for this project. MAX30102 was chosen for its high LED output, which is used for thicker areas of fur. The Pulse Sensor is chosen for its average LED output, which is used on areas of the animal with less fur.

Table 3-2: Heart Rate Sensor Options

Sensor	Price	Dimensions	Voltage (in V)	Amp (in A)	LED output wavelength
Pulse Sensor (Photoplethysmography) [7]	$24.99	4 x 0.5 x 6 in	3.0 - 5.5 V	4 mA	565 nm
MAX30102 (Pulse Detection ppg) [8]	$12.99	5.6 x 3.3 x 1.55 mm	3.3 - 5.5v	20 mA	660/880 nm
AST1041 [9]	$19.95	5.6 x 3.3 x 1.55 mm	1.8 - 5.0v	20 mA	870/900 nm

Pulse Sensor and MAX30102 were chosen for this project. The Pulse Sensor is chosen for its average LED output, which is used on areas of the animal with less fur. MAX30102 was chosen for its high LED output, which is used for thicker areas of fur.

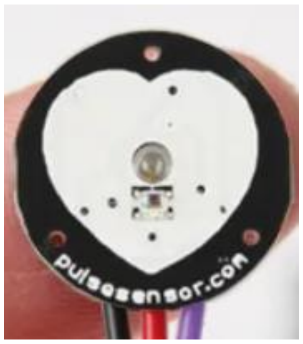

Fig. 3-3: Pulse Sensor

Fig. 3-4: MAX30102

For the body temperature sensor, DS18B20 was chosen. The DFRobot I2C Temperature and Humidity Sensor, while cheaper did not have the same features as DS18B20. The DS18B20 has a feature which allows multiple use of the same sensor. This feature is called the one-wire interface which uses only one digital pin for communication. DS18B20 has a temperature range of -55°C **-** 125°C.

Table 3-3: Body Temperature Sensor Options

Sensor	**Price**	**Dimensions**	**Voltage**	**Amp**	**Temperature Range**
DS18B20 (Digital thermometer) [10]	$13.66	• Length 91cm • Diameter 4mm	3.0 - 5.0 V	1 - 1.5 mA	-55°C – 125°C
DFRobot I2C Temperature & Humidity Sensor [11]	$7.43	• DuPont length 5cm • Wire length 1.5m	2.5- 5.5 V	3.0 µA	-40°C – 125°C

The choice of sensors was carefully considered during the design process. The heart rate sensor selected is a photoplethysmography (PPG) sensor, which utilizes LED light sources to monitor changes in the blood volume within the body. The Pulse Sensor MAX3012 is chosen to cater to different types of fur thickness. MAX3012's high LED output will cater to animals with thicker fur while the Pulse Sensor's Average LED output for areas with less fur.

Fig 3–4: DS18B20

For the body temperature sensor, the DS18B20 was chosen due to its versatile 1-wire interface, allowing the use of multiple sensors with just one digital pin for communication. The sensor also detects a temperature range of –55°C to 125°C. The sensors were chosen for accurate and reliable data collection for the project requirements.

3.3.2. Subsystem 2

The design team compared the various battery types, and the most practical option was the lithium-based batteries. Both lithium-ion and lithium-polymer batteries provide enough adequate power and battery life in the required space, and both at a reasonable price within the budget. The power requirements of each component are small enough that the system can continue operating beyond our battery life goal set for the component.

Component Name	Power draw (mA)
Arduino Uno [1]	42 mA
Temperature Sensor [10]	2 mA
Pulse Sensor [7]	4 mA
Max 3012 Pulse Sensor [8]	20 mA
LCD Screen	1.5 mA
Total power required	69.5 mA

We are using the Arduino’s built in voltage regulator to convert our battery input voltage of 7.4 volts, to the required 5 volts. 7.4 volts was chosen because it is the closest commonly available battery voltage to 5 volts that we could find. All calculations are done with the Arduino's efficiency of 67.6% in mind. [6]

$$5\,V \cdot 0.0695\,Ah = 0.3475\,W$$

$$\left(\frac{24\ hours}{0.676}\right) \cdot 0.3475\,W = 12.34\,Wh$$

$$\frac{12.34\,Wh}{7.4\,V} = 1.67\,Ah$$

To meet our goal of 24-hour battery life we require a battery with 12.34-Watt hours, or 1.67 Amp hours, of battery life. The battery we chose to go with has an amp hour rating of 2.6, this measurement exceeds our required battery life by 13.43 hours.

$$7.4\,V \cdot 2.6\,Ah = 19.24\,Wh$$

$$\frac{19.24\,Wh}{0.3475\,W} \cdot 0.676 = 37.43\,h$$

We chose this battery because it gives us plenty of extra room for margin just in case the specification sheets are wrong about the required amperage draw of some of the components, all while still being within budget and size constraints.

3.3.3. Subsystem 3

Below are sections explaining various software implementations that utilize the hardware to ensure proper function of the controller. The software design can be broken into two different systems: controller software and app software. A high-level software design can be seen in both Figure 3-5 and Figure 3-6.

3.3.3.1. Controller Software

The diagram below shows what the general controller software will do once it is turned on.

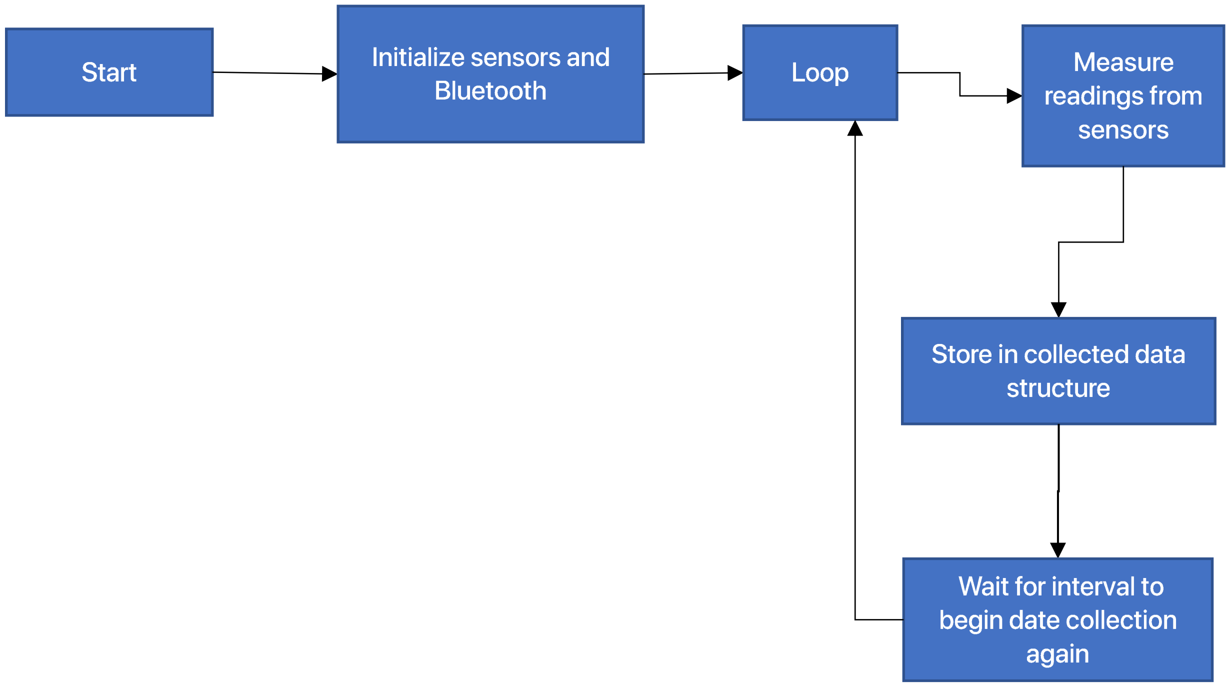
Start
Initialize sensors and Bluetooth
Loop
Measure readings from sensors
Store in collected data structure
Wait for interval to begin date collection again

Fig. Error! No text of specified style in document.**-5: SafePaws controller flowchart**

The flowchart above describes how the components and controller board will work. The sensor components are initialized along with the Bluetooth communication module. Then, the components are set to operate remotely and autonomously measure readings and storing said readings into a data structure.

3.3.3.2. Mobile Application Software

The app software serves as the user interface for the vest. The software program is designed to allow the user to set base parameters and view data readings. The app also triggers user alerts if the sensor data exceeds those set parameters at any moment. The design ensures user-friendliness and data accuracy.

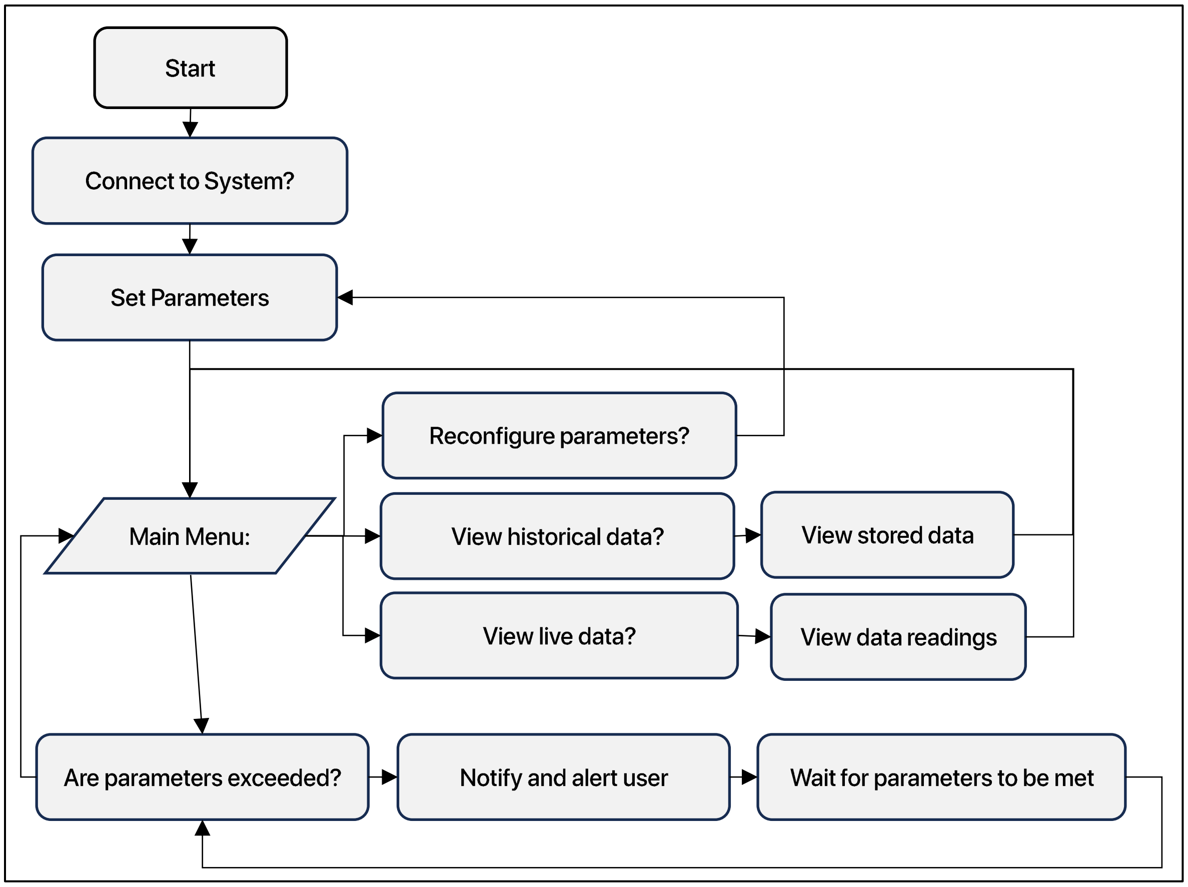

Fig. Error! No text of specified style in document.**-6: SafePaws mobile application flowchart**

The application software subsystem of the SafePaws system is designed to offer a user-friendly experience while ensuring strong data management. The primary user interface takes the form of a mobile application with a user-friendly graphical user interface (GUI). This interface allows users to connect to the system, configure monitoring parameters, access live data readings, review historical data readings, and receive real-time notifications and alerts. The data exchange format between the system and the mobile app is standardized to JSON, ensuring readability and interpretability for both ends [12]. This format includes fields for vital signs data, timestamps, and system status. To handle simultaneous events without errors, the implementation considers a protothreads approach, where different software components run almost as separate threads. For example, one thread manages data collection and analysis, while another handles user commands and GUI updates. This parallel processing approach effectively prevents deadlock and livelock situations, ensuring that the system remains responsive to user input while efficiently collecting and processing data. This software design ensures a balance between user interaction and data processing.

3.4. Level 2 Prototype

In the Design II semester, our primary objective is to test and improve upon a fully functional prototype of the SafePaws vest, having all subsystems and components integrated. The prototype should represent the design values set of data accuracy, user-friendliness, and safety. Our intention in finalizing our prototype is to have the prototype be available to a user of a random testing subject group and allow that user to utilize our prototype on their own understanding. The prototype should operate remotely upon activation, enabling the user to access the mobile application via the Bluetooth communication module.

Within the mobile application, the user should have authority to set parameters of sensor readings and access real-time or historical data readings. Most importantly, the system will remain watchful, promptly notifying users if sensor parameters are ever exceeded through alerts and notifications. Design II for the design team centers around the process of testing, evaluation, improvement, and redesign.

3.4.1 Level 2 Diagram

In this section, a detailed Level 2 diagram outline the low-level components of the final intended design for the SafePaws vest.

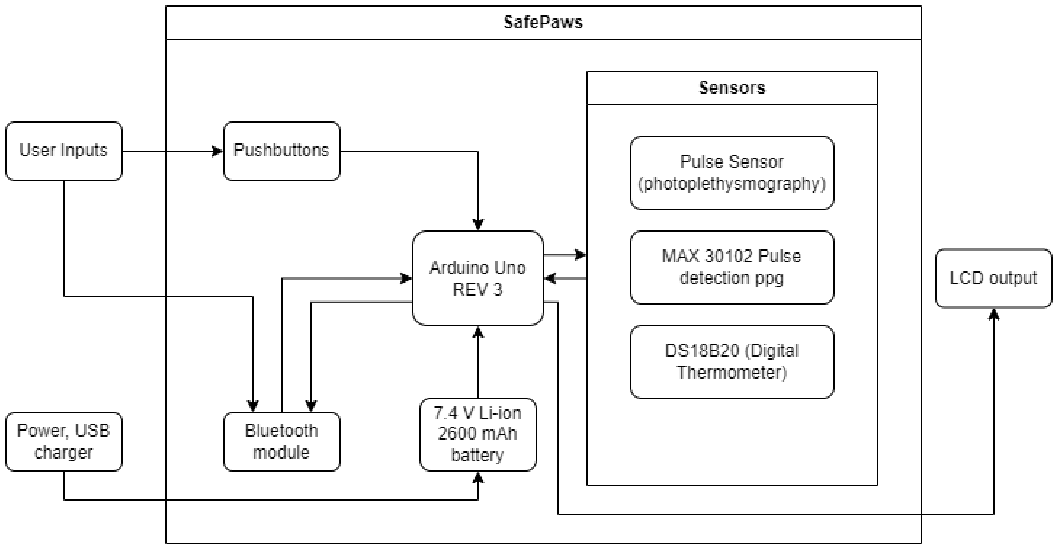
SafePaws
Sensors
User Inputs
Pushbuttons
Pulse Sensor (photoplethysmography)
Arduino Uno REV 3
MAX 30102 Pulse detection ppg
LCD output
DS18B20 (Digital Thermometer)
Power, USB charger
Bluetooth module
7.4 V Li-ion 2600 mAh battery

Fig. 3-4: Diagram for Safe Paws (Level 2)

The Level 2 diagram describes how the software and hardware components connect to become the SafePaws prototype.

The SafePaws vest represents our solution for remote health monitoring of dogs, as the prototype aims to integrate a heart rate sensor and body temperature thermometer in a simple wearable design for on-the-go tracking. With autonomous data collection and wireless control through a mobile app, our design choices prioritize data accuracy, user-friendliness, and pet safety. This section encapsulates our final design considerations, from exploring multiple options to subsystem descriptions. Moving forward, this document records the testing and improving phases of the design process.

3.5. References

[1] The Arduino Team, "Arduino Uno R3 Specification Sheet," Arduino Documentation, https://docs.arduino.cc/hardware/uno-rev3 (accessed Oct. 20, 2023).

[2] S. K, "Basys 3 reference manual," Basys 3 Reference Manual - Digilent Reference, https://digilent.com/reference/programmable-logic/basys-3/reference-manual (accessed Oct. 20, 2023).

[3] S. K, "Basys 3 reference manual," Basys 3 Reference Manual - Digilent Reference, https://digilent.com/reference/programmable-logic/basys-3/reference-manual (accessed Oct. 20, 2023).

[4] Raspberry Pi Team, “Raspberry pi 4 model B specifications,” Raspberry Pi Documentation, https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ (accessed Oct. 20, 2023).

[5] Circuit IO Team, “Arduino vs. Raspberry Pi: Which board is best?,” circuito.io blog, https://www.circuito.io/blog/arduino-vs-raspberry-pi/ (accessed Oct. 20, 2023).

[6] MC33269 - voltage regulator adjustable output, low dropout 800 ma - onsemi, https://www.onsemi.com/pdf/datasheet/mc33269-d.pdf (accessed Oct. 21, 2023).

[7] Last Minute Engineers, “In-Depth: Detect, Measure & Plot Heart Rate using Pulse Sensor & Arduino,” Last Minute Engineers, Jan. 16, 2021. https://lastminuteengineers.com/pulse-sensor-arduino-tutorial/

[8] “General Description.” Available: https://www.analog.com/media/en/technical-documentation/data-sheets/max30102.pdf (accessed Oct. 23,2023).

[9] “AST1041.”Available: https://www.digikey.com/en/products/detail/tinycircuits/AST1041/11594531 (accessed Oct. 23, 2023)

[10] A. Industries, “Waterproof 1-Wire DS18B20 Digital temperature sensor,” *www.adafruit.com*. https://www.adafruit.com/product/381?gclid=CjwKCAjws9ipBhB1EiwAccEi1I_Cpjkz6gxzXgaTveIUXcOG1HRZNsbkSlXBfyg8ZbmgEaWvQ2WrZRoCuzIQAvD_BwE (accessed Oct. 23, 2023).

[11] “DFRobot I2C Temperature & Humidity Sensor (Stainless Steel Shell),” *RobotShop USA*. https://www.robotshop.com/products/dfrobot-i2c-temperature-humidity-sensor-stainless-steel-shell?gclid=CjwKCAjws9ipBhB1EiwAccEi1CuoByaq1hzph8IVKYTeXjcKXWlgK6sr-yW5Cu4VAhkbwjlkPy68TBoCAgoQAvD_BwE (accessed Oct. 23, 2023).

[12] L. Miller, “How to parse JSON data with Arduino (arduinojson tutorial),” Learn Robotics, https://www.learnrobotics.org/blog/parse-json-data-arduino/#:~:text=To%20parse%20JSON%20files%20using%20Arduino%2C%20you%E2%80%99ll%20need,set%20up%20the%20sketch%20as%20you%20would%20normally. (accessed Oct. 23, 2023).