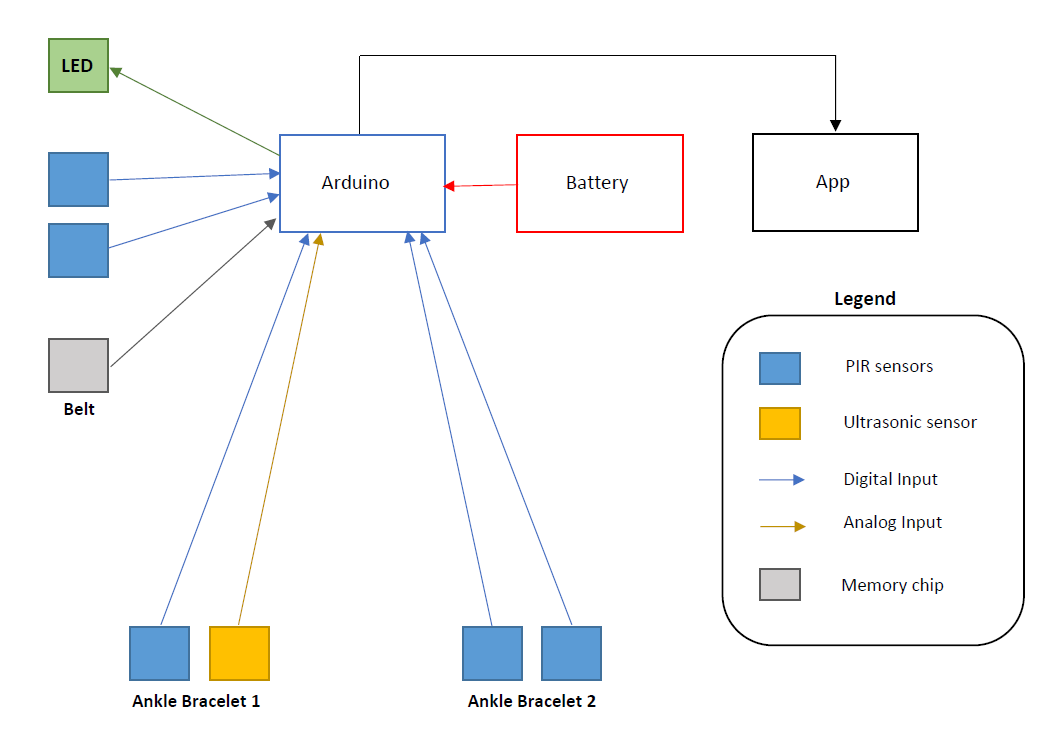
# **2. APPROACH**

GAS aims for an inexpensive, lightweight solution for physical therapists to monitor key aspects of a patient’s gait between therapy sessions. In particular, GAS measures the average width between the user’s feet, the average time taken for each step, the average time and direction of arm swaying, and the position of the arms relative to the feet. To accomplish these measurement goals, GAS utilizes Arduino boards paired with Arduino-based passive infrared (PIR) sensors and an ultrasonic motion sensor, all of which are located on a belt and two ankle bracelets. GAS stores gait-related information on a memory chip located on the belt, and these data can be accessed through an external app launched on an Android mobile device. GAS is easily integrable into the user’s life and accommodates a variety of clothing and safety precautions. Figure 2.1 outlines the design of GAS.



**Figure 2.1** Design Overview

As shown in Figure 2.1, there are a total of five PIR sensors, one ultrasonic sensor, and one LED in the design. On one ankle bracelet, there is one PIR sensor and one ultrasonic sensor. The PIR sensor on this ankle bracelet is placed on the outside of the ankle to determine whether there are nearby objects, such as animals or people. When this sensor detects nearby objects, measurements taken during this time period are discarded, as these objects can disrupt normal measurements. The ultrasonic sensor on this ankle bracelet takes a distance measurement of how far apart the legs are during the patient’s walking session. On the other ankle bracelet, there are two PIR sensors. One PIR sensor is on the outside of the ankle, while the other is on the inside of the ankle. The outer PIR sensor on this ankle bracelet serves the same function (nearby object detection) as the outer PIR sensor on the other ankle bracelet. The inner PIR sensor determines when the opposite leg is in a position parallel to the leg with the inner PIR sensor; this measurement then informs the ultrasonic sensor on the opposite ankle bracelet to take a distance measurement. The belt houses the Arduino, battery, memory chip, two PIR sensors, and LED. The Arduino is the brain of the device, recording and storing all the information taken by the sensors. The battery is connected to the Arduino and powers the device, while the memory chip holds all the stored data. The two PIR sensors, located on the left and right sides of the belt, detect the relative position of the user’s arms while they are swinging as well as the time it takes for the user to swing their arms. The LED serves as a reminder to the user that the device is on by displaying a green light.

## **2.1 Hardware**

GAS includes the following hardware subsystems: the belt, the left ankle bracelet, and the right ankle bracelet. This section details the numerous hardware components required for each hardware subsystem in GAS. Given that there are no pre-existing tools that record gait-related data, these components were chosen based on functionality, price, and quality. For all tables listed in the hardware subsections, cells with ideal traits are highlighted in green, cells with acceptable traits are highlighted in yellow, and cells with undesirable traits are highlighted in red.

**2.1.1 Microcontroller**

The microcontroller for this device needs to include at least five input/output digital pins, at least one input/output analog pin, a 5-V voltage regulator for the usage of the PIR sensors and ultrasonic sensor, a 3.3-V voltage regulator for the usage of the LED, and a relatively inexpensive price tag to make the device affordable. Table 2.1 compares the aspects of the Arduino Uno Rev3, the Arduino Mega 2560, the Arduino Due, and the Arduino Nano microcontrollers to the requirements for GAS’s microcontroller.

**Table 2.1 Comparison of Microcontrollers**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Microcontroller** | **Cost** | **Number of Digital Pins** | **Number of Analog Pins** | **Includes 5-V and 3.3-V Regulators** |
| Requirements | < $40 | At least 5 | At least 1 | Yes |
| Arduino Uno Rev3 [1] | $23.00 | 14 | 6 | Yes |
| Arduino Mega 2560 [2] | $40.30 | 54 | 16 | Yes |
| Arduino Due [3] | $40.30 | 54 | 12 | Yes |
| Arduino Nano [4] | $20.70 | 22 | 8 | No |

As shown in Table 2.1, the Arduino Uno Rev3 (referred to as “Arduino” in this document) is the most viable choice for the device’s microcontroller. Not only is the Arduino Uno Rev3 relatively inexpensive and lightweight, but it is also customizable and easy to use [1]. The online community surrounding the Arduino products is centralized around the Arduino website that teaches users to program their Arduino boards [5], and the products are well documented. With this information in mind, the Arduino Uno Rev3 is the obvious choice for the microcontroller used in GAS.

**2.1.2 Motion Sensors**

The motion sensors used in this device are responsible for a variety of measurements: the time it takes for the user to step, the relative position of the user’s feet, the time it takes for the user to swing their arms, the relative position of the user’s arms, and the detection of other people, animals, and objects around the user. PIR sensors are used for these measurements as opposed to other types of sensors due to their ability to detect infrared radiation (IR), which allows them easily to detect the presence of an individual [6]. Therefore, this sensor provides the most efficient way to detect the presence of the user, where the presence of the user, in this case, refers to the arms and legs. Other options that were considered along with PIR sensors were photodiodes and accelerometers. Photodiodes are useful since they are able to pinpoint when a specific LED light comes into their field of view [7]. However, photodiodes would require extra wiring and LED components to be placed on the leg opposite to the leg where the sensors are located. The incorporation of photodiodes in the device would not only increase the total manufacturing cost of the device, but it would also complicate the wiring and component connections. In addition, photodiodes struggle to work well during the day. On the other hand, accelerometers are even worse for determining the required measurement for the device. Accelerometers determine the magnitude and direction of the acceleration they experience via vibration [8]. To determine position, the acceleration values from the accelerometer must be integrated twice. In doing these integrations over relatively long periods of time, the error of the device drastically increases [9]. The comparisons among these three types of sensors are summarized in Table 2.2.

**Table 2.2 Comparison of Sensors**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type of Sensor** | **Photoelectric (Able to Detect IR)** | **Works Well in Daylight** | **Accumulated Error** | **Sensitivity to Temperature** |
| Requirements | Yes | Yes | Small | Low-High |
| PIR [6] | Yes | Yes | Small | High |
| Photodiode [7] | Yes | No | Small | High |
| Accelerometer [8], [9] | No | Yes | Large | N/A |

Due to the various requirements listed in Table 2.2, the PIR sensor is an excellent candidate for the motion sensor. The main options for the type of PIR sensor used in GAS are the following: the HC-SR501 [10], the SEN-13968 [11], and the AMN34111 [12]. General information on the three PIR sensors is listed in Table 2.3.

**Table 2.3 PIR Sensors**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Product** | **Cost** | **Range** | **Measuring Angle (°)** | **Time for One Measurement** |
| Requirements | < $10 | < 10 m | 90 < x < 180 | < 0.6 sec |
| HC-SR501 [10] | $2.60 | Up to 3 m to 7 m (adjustable) | 110 | 3 sec to 5 min |
| SEN-13968 [11] | $15.95 | Up to 2 m to 5 m (adjustable) | 110 | 0.4 sec to 7.5 sec |
| AMN34111 [12] | $14.68 | Up to 10m | 110 | 30 sec |

As shown in Table 2.3, the PIR sensor choice is narrowed down to the HC-SR501 and the SEN-13968. The measuring angle needs to be an obtuse angle between 90° and 180° so that the outer PIR sensors have a wide field-of-view for detecting objects, animals, or individuals passing by the user during a walking session. Although all the sensors listed in Table 2.3 meet this angle requirement, the minimum time required for taking one measurement for the AMN34111 sensor is too long in comparison to the other two sensors, and it has an unnecessarily long range. This PIR sensor’s long-range capabilities cause it to register people, animals, and objects relatively far away from the user. Furthermore, the lengthy-time delay causes it to miss all the user’s movements within that time period. On the other hand, the SEN-13968 and the HC-SR501 have somewhat similar attributes. The SEN-13968 is expensive when compared to the HC-SR501; however, SEN-13968 takes measurements rapidly, which allows this sensor to record the user’s arm and leg movements successfully without missing any movements. This conclusion is based on the fact that the average human cadence is 110 – 115 steps per minute [13], which is approximately 1.83 to 1.92 steps per second. To measure the average cadence, the PIR sensor needs to record the user’s steps every 0.52 to 0.55 seconds on average. Since only 0.4 seconds are needed for recording one measurement, the SEN-13968 can detect cadences of up to 150 steps per minute, which is approximately 30% faster than the upper bound of the average human cadence. Since users are instructed exclusively to walk with the device, the SEN-13968 has more than enough cushion to record all the user’s leg movements. In addition, the average human arm swing rate closely follows the average human cadence [13]; therefore, this same time requirement can be applied to the PIR sensors on the belt that detect the user’s arm movements. The range of the SEN-13968 is also adjustable. This fact is important because the average width between a user’s legs is 7.7 to 10 centimeters [14]; therefore, since the PIR sensor on the inner-ankle detects the motion of the opposite leg, the selected PIR sensor must be able to take measurements given a restricted range. An adjustable range also is beneficial to the PIR sensors located on the belt, since this range can be modified such that the sensors are only taking measurements close to the user’s body (i.e., measurements involving the arms). Therefore, the SEN-13968 is the PIR sensor of choice.

**2.1.3 Distance Sensor**

The distance between the user’s feet must be measured, as it relays key aspects of the user’s gait to the physical therapist. A variety of sensors can be used to accomplish this task, such as photodiode time-of-flight, LIDAR, and ultrasonic. Photodiode time-of-flight sensors measure distance from a reflected wave pulse; however, they are expensive and are sensitive to different environmental factors [15]. For example, sunlight can poorly affect the performance of these sensors. Since patients may be inclined to complete walking sessions outside, photodiode time-of-flight sensors are not a viable option. The second option is LIDAR sensors. LIDAR sensors are more advanced than infrared sensors and can even measure 3D structures while maintaining a high accuracy [15]. However, as a result of their advanced technology, they are expensive and can cost hundreds of dollars. On the other hand, ultrasonic sensors are less expensive and do not have the same sensitivity drawbacks experienced by the time-of-flight sensors. Ultrasonic sensors emit ultrasonic waves, and the time taken for these waves to be received back at the target determines the distance between the target and another object. Ultrasonic sensors also have a high frequency and high penetrating power, allowing them to recognize objects easily at a high accuracy rate [16]. Based on these facts, an ultrasonic sensor is an excellent candidate for measuring the average width between an individual’s feet. The major options for ultrasonic sensors are HC-SR04 [17], PING))) #28015 [18], and US-100 [19]. Important characteristics considered when selecting an ultrasonic sensor are reflected in Table 2.4.

**Table 2.4 Ultrasonic Sensors**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Product** | **Cost** | **Range** | **Measuring Angle (°)** | **Time for One Measurement(ms)** | **Time between Measurements (ms)** | **Output** |
| Requirements | < $10 | 2 cm – 100 cm | As small as possible | < 400 | < 400 | Analog |
| HC-SR04 [17] | $3.95 | 2 cm – 400 cm | 30 | 0.01 | 60 | Analog |
| PING))) #28015 [18] | $29.99 | 2 cm – 300 cm | Angle not disclosed\* | 0.05 | 0.2 | Analog |
| US-100 [19] | $6.95 | 2 cm – 450 cm | Less than 15 | 0.01 | 60 | Analog |

*\*Stated on product website that PING))) has a narrow acceptance angle but does not give specifics*

Two important aspects that were considered when selecting the ultrasonic sensor were the time between measurements and time for one measurement, as these measurements help ensure that all data is recorded accurately. As shown in Table 2.4, it is required that the time between measurements is less than 400 milliseconds, due to the fact that the PIR sensor has a minimum 400 millisecond delay time. The activation of the PIR sensor drives the ultrasonic sensor’s distance measurement, as the PIR sensor reports when the user’s legs move into a parallel position. Due to this constraint, the ultrasonic sensor cannot have a delay time of more than 400 milliseconds. However, all the reported ultrasonic sensors meet this requirement. In addition, all the sensors record data rapidly, which allows ample time for all measurements to be taken and stored to the memory chip before the next instance of the user’s feet in parallel. Another important aspect that was considered in the selection process was the range of the sensor. Since the main goal of the sensor is to measure the width between the user’s legs, the range of the sensor must be adjusted to measure only this width, which is typically 7.7 to 10 centimeters for adults [14]. Additionally, the measuring angle is important. Depending on how low the ankle bracelets are to the ground, the measuring angle, if too large, can cause reflection off the ground, and the ultrasonic sensor would fail to function properly and record inaccurate data. While HC-SR04 offered the best price and range, the measuring angle was a bit large. However, to counteract this problem, the ankle bracelets can be moved up the user’s legs (near the shins). By adjusting the position of the bracelets, a proper measurement of the user’s step width is maintained, but any possibility of reflection off the ground is eliminated. Therefore, even if some adjustments to the ankle bracelet placement must be made, HC-SR04 remains the best option for the ultrasonic sensor.

**2.1.4 Subsystem Connection**

There are two main options to consider when it comes to the subsystem connection within the device: wired or wireless. A wired approach to GAS simplifies the design process but complicates the user’s interaction with the device. Connecting the ankle bracelets to the belt via wires means that the ankle bracelets would not require an Arduino or batteries to operate. Instead, the Arduino and batteries are housed in the belt, which makes the ankle bracelets lighter and less bulky. Wired connection also allows all the subsystems of GAS to operate synchronously since the delay time in communication with wires is insignificant. Despite these benefits, relatively long wires typically hinder user interaction with a device since the wires can get tangled. Furthermore, users are faced with a trip hazard (i.e., loose wires that cause a person to fall while walking) if worn outside of his or her clothing. On the other hand, a wireless approach to GAS simplifies the user’s interaction with the device but complicates the design process. In addition, connecting the ankle bracelets to the belt via wireless connections means that the user does not have to handle tangled wires. However, using wireless connections complicates the design process and makes the ankle bracelets bulkier than necessary. The complication of the design process comes from working with a wireless connection method such as Bluetooth. A wireless connection also causes the design to have two extra battery sources and two extra Arduinos (one for each ankle bracelet). Along with increased bulkiness, an extra delay in communication exists, which results in the subsections not working synchronously. The primary motivation for whether to include or exclude wires is the typical patient in outpatient therapy. Since the typical patient in outpatient therapy suffering from gait-related problems is 65+ years of age [20], increased bulkiness is an obstacle, as this extra weight may impede the elderly patient from completing his or her walking sessions and result in fatigue. Since GAS is designed to be lightweight, adding extra battery sources and Arduinos runs the risk of exceeding this weight constraint and creating additional problems for the patient to overcome. To combat the potential trip hazard, GAS incorporates a knee brace on each leg to hold the wires in place. The knee brace ensures that the wires stretching from the belt to the ankles are taut and do not serve as a hindrance to the user when walking. The knee brace allows GAS to be worn with a variety of clothing and enables the device to also be worn outside of clothing with no potential trip hazard. In addition, the wired approach keeps all subsystems connected; therefore, it is less likely that the user will misplace one or more components of the device. Between these two options, the wired approach appears to be the more viable subsystem connection method for the device.

**2.1.5 Memory Chip**

GAS stores gait-related data and holds these data until the physical therapist extracts them. There are several measurements that are stored on the device for each walking session: the length of the walking session in minutes, the total number of steps for the walking session, the total number of arm swings during the walking session, the width measurement between the user’s feet in centimeters for every step the user takes, the average width between the user’s feet for the walking session, the time taken for every step, the average time taken for each step in steps per minute, the time taken for every arm swing, the average time taken for each arm swing in swings per minute, the correct position of the arms relative to the feet, the ratio of how often the position of the arms relative to the feet was correct, and a flag that is raised every time an object is detected by the outer PIR ankle bracelet sensors. Separate counters are used for the total number of steps for the walking session, the total number of arm swings during the walking session, and the correct position of the arms relative to the feet. For the correct position of the arms relative to the feet, the device checks that the right foot is forward if the left arm is forward and vice versa. This counter increases by one for every instance that the relative position of the arms and feet is correct. The ratio of how often the position of the arms relative to the feet was correct is calculated by dividing the counter value of the correct position of the arms relative to the feet by the counter value of the total number of steps. Since the walking sessions last up to an hour each, the timer will need to last for at least sixty minutes. The time taken for every step and the time taken for every arm swing are stored as separate arrays, with the total number of indices of both arrays equal to the total number of steps and the total number of arm swings respectively. The average step time and average arm swing time are calculated by adding all the values in their respective arrays and dividing by the total number of steps or arms swings during the walking session. Since the PIR sensors can take a measurement every 0.4 seconds [11], the device handles cadences of up to 150 steps per minute. This measurement constraint means that the device needs to handle at least 9,000 measurements for the time taken for every step and 9,000 measurements for the time taken for every arm swing due to the device handling up to 9,000 measurements per hour. The measurement constraint also means that the counters each need to increment to at least 9,000. The raised flag only needs a single bit, as it serves to override the sensors to temporarily halt the measurements being taken while an object is detected near the user. With that information stated, Table 2.5 details the minimum required number of bits that the memory chip must store.

**Table 2.5 Minimum Required Bits Stored by Memory Chip**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Measurement/ Calculation** | **Minimum Required Decimal Value** | **Minimum Required Binary Value** | **Maximum Number of Bits** | **Total Number of Bits Needed for Arduino (type of variable)** |
| Length of Walking Session (minutes) | 60 minutes = 360,000 hundredths of a second | 1010111111001000000 | 19 | 32 (long) |
| Total Number of Steps | 9,000 | 10001100101000 | 14 | 16 (integer) |
| Total Number of Arm Swings | 9,000 | 10001100101000 | 14 | 16 (integer) |
| Feet Width Measurements | 9,000 values of 10,000 hundredths of a centimeters | 10011100010000 | 14 \* 9,000 = 126,000 | 16 (integer) \* 9,000 = 144,000 |
| Average Feet Width (centimeters) | 10,000 hundredths of a centimeter | 10011100010000 | 14 | 16 (integer) |
| Time Measurements for Every Step (seconds) | 9,000 values of 6,000 hundredths of a second | 1011101110000 | 12 \* 9,000 =108,000 | 16 (integer) \* 9,000 = 144,000 |
| Average Time per Step (seconds) | 6,000 hundredths of a second | 1011101110000 | 12 | 16 (integer) |
| Time Measurements for Every Arm Swing (seconds) | 9,000 values of 6,000 hundredths of a second | 1011101110000 | 12 \* 9,000 =108,000 | 16 (integer) \* 9000= 144,000 |
| Average Time per Arm Swing (seconds) | 6,000 hundredths of a second | 1011101110000 | 12 | 16 (integer) |
| Correct Position of Arms Relative to Feet | 9,000 | 10001100101000 | 14 | 16 (integer) |
| Ratio of How Often Arm and Feet Position was Correct (percentage) | Up to 100 | 1100100 | 7 | 16 (integer) |
| Flag | 1 | 1 | 1 | 1 (bool) |
| **Total Number of Bits Required** |  |  | 255,085 | **432,145** |

It is important to mention that the Arduino software is coded in C++. C++ can store several variable types [21], but only the integer, long, and bool types are used for the device. In Table 2.5, the second column labeled “Minimum Required Decimal Value” provides the lowest required value for each measurement and calculation. The blue-highlighted cells in this column provide their respective values in terms of a hundredth of their particular units. The values are represented in this way to obtain a precision of two decimal places for those values while they are stored as integers or longs in the Arduino code. Every time these values are transferred to the GAS app, each value is divided by 100 in order to maintain this precision. The third column labeled “Minimum Required Binary Value” in Table 2.5 shows the minimum required binary value converted from the decimal values in the second column. The number of digits of the binary values in the third column adds up to the values in the fourth column labeled “Maximum Number of Bits.” These values in the fourth column are used to determine which variable type is required to store the values in the Arduino. The fifth column labeled “Total Number of Bits Needed for Arduino” lists the variable type for each measurement, along with the calculation, and states how many bits in the Arduino that each variable would require. From Table 2.5, the minimum number of required bits needed for the memory chip is 432,145. This value in bytes is approximately 54,019. GAS is designed to store data for at least one walking session; therefore, based on this information, GAS needs roughly 55 kB of storage space in its memory chip to accomplish this goal. The Arduino includes a 1 kB EEPROM and a 32 kB flash memory chip [1], which are not sufficient to store the minimum required amount of data. Table 2.6 outlines three options for memory expansion.

**Table 2.6 Memory Chips**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Memory Chip** | **Cost** | **Storage Space** | **Current Draw** | **Supply Voltage** | **Type of Chip** |
| Requirements | < $20 | ≥ 55 kB | As small as possible | 3.3V or 5 V | Flash Memory |
| CAT25512 [22] | $0.60 | 512 kb = 64 kB | 3 mA | 1.8 V to 5.5 V | EEPROM |
| LE25S81A [23] | $0.59 | 8 Mb = 1000 kB | 5 mA | 1.65 V to 1.95 V | Flash Memory |
| Arduino MKR MEM Shield  [24] | $19.50 | 2 MB = 2000 kB | 4 mA | 2.7 V to 3.6 V | Flash Memory |

As shown in Table 2.6, the LE25S81A’s supply voltage range does not include the required supply voltage; therefore, this memory chip is not a viable option. The two primary candidates are the CAT25512 and the Arduino MKR MEM Shield. While the Arduino MKR MEM Shield is expensive in comparison to the CAT25512, the Arduino MKR MEM Shield is a flash memory chip. On the other hand, the CAT25512 is an EEPROM chip. Flash memory is similar to EEPROM; however, the main difference between the two memory classifications is that flash memory is frequently rewritten while EEPROM is rarely rewritten [25]. Flash memory works best for GAS since the recorded data needs to be rewritten frequently. Therefore, the Arduino MKR MEM Shield is the optimal choice for the memory chip.

**2.1.6 Battery**

The battery or batteries chosen to power GAS supply the various components during a typical walking session. Table 2.7 lists the expected average power consumption for each component in GAS.

**Table 2.7 Power Consumption**

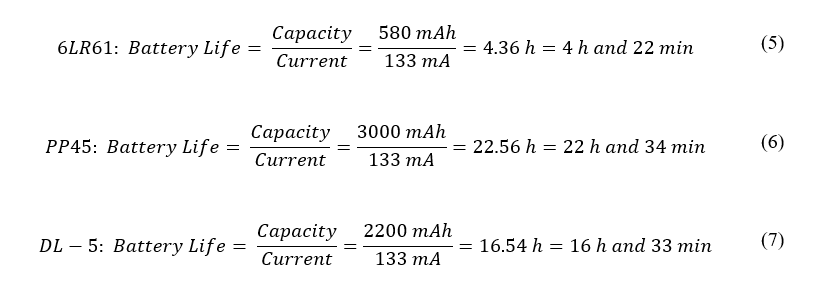
|  |  |  |  |
| --- | --- | --- | --- |
| **Part** | **Current (Active)** | **Voltage** | **Power Consumption** |
| Arduino [1] | 45 mA | 5 V | 0.225 W |
| PIR Sensor  (x5) [11] | 15 mA | 5 V | 0.075 W |
| Ultrasonic Sensor [17] | 50 mA | 5 V | 0.25 W |
| LED [7] | 20 mA | 5 V | 0.1 W |
| Memory Chip [24] | 4 mA | 3.3 V | 0.0132 W |
| **Total** | 134 mA |  | 0.6632 W |

The Arduino powers the sensors, LED, and memory chip at 5 V or 3.3 V. The calculations performed in Table 2.7 consider the average-case power consumption scenario with the duty-cycle equal to 1. The total power value of 0.6632 W shows that the device runs at relatively low power, while the total current value of 134 mA is used to determine the battery capacity needed to run the device. For the device to run a minimum of one hour, a minimum battery capacity of 134 mAh is required. According to the Arduino information website, the recommended input voltage for the battery ranges from 7 to 12 V [1]. It is also necessary that the battery is lightweight, small, and inexpensive. The 6LR61, the PP45, and the DL-5 are the three batteries considered for use in GAS. Table 2.8 details the specifications of each of these battery types.

**Table 2.8 Types of Batteries**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Product** | **Cost** | **Volume (cm3)** | **Weight (g)** | **Output Voltage (V)** | **Capacity (mAh)** | **Chemical Makeup** | **Rechargeable** |
| Requirements | < $10 | < 100 | < 454 | 7 to 12 | > 134 | -- | -- |
| 6LR61 [26] | $3.36 | 21.0 | 46.0 | 9 | 580 | ZnMnO2 | No |
| PP45 [27] | $44.99 | 258.75 | 579 | 12 | 3000 | NiMH | Yes |
| DL-5 [28] | $3.49 | 43.26 | 54.4 | 6.0 | 2200 | ZnMnO2 | No |

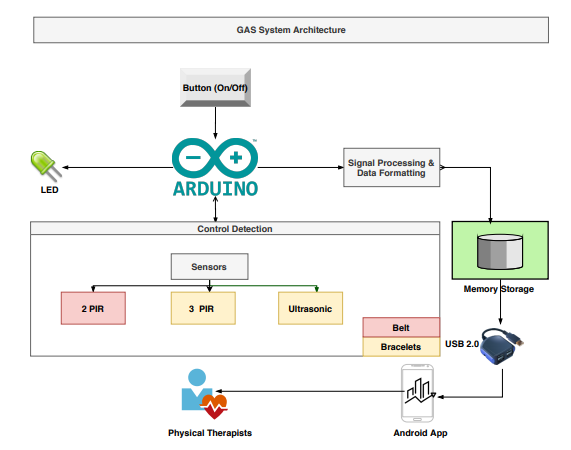
The 6LR61 and DL-5 batteries in Table 2.8 are replaceable and non-rechargeable, while the PP45 is rechargeable. Both rechargeability and replaceability have their merits; however, replaceable batteries are better for GAS since it will simplify the interaction between the therapist or user and the device. Including rechargeable batteries creates additional barriers; for example, rechargeable batteries, such as the PP45, are expensive, and either the user or the physical therapist is tasked with keeping track of a charger. If this charger gets misplaced, the user will not be able to record any data for his or her walking sessions and will have to wait for the physical therapist to order a new one. Furthermore, the PP45 takes two hours to recharge [27]. This recharge time can cause problems if the user plans on using the device during a specific timeframe and forgets to recharge it earlier. Instead, replaceable batteries are inexpensive and are found online or in stores. Because replaceable batteries can be found in a variety of stores, they can instantly be replaced and would not hinder the user from using the device during his or her preferred timeframe. Based on the data from Table 2.8, the following equations provide the battery life of each of the products:



The DL-5 requires that two batteries be placed in series since the output voltage of one DL-5 is 6 V; therefore, Equation 7 assumes that two DL-5 batteries are in series providing an input voltage of 12 V to the device. Equations 5, 6, and 7 state that all three batteries will provide the device a lifetime of greater than one hour. The PP45 provides the longest battery life, but it is too expensive and bulky. Therefore, the best choice for the battery used in GAS is the 6LR61 which is the smallest, lightest, and most inexpensive option.

## **2.2 Software**

The software used in GAS includes the Arduino coding, the data storage, and the smartphone app. The Arduino collects data from the sensors and does any necessary calculations. The Arduino then stores these data until they are transferred to the app. The Arduino is responsible for the data processing necessary for data storage and transfers these data to the smartphone application for presentable calculations to the physical therapists. All collected data is stored in the memory chip. The app is the interface for the physical therapist to view the patient’s data, which is visualized in a clear, concise manner. Figure 2.2 demonstrates how the components of GAS work together to gather, store, and display information.

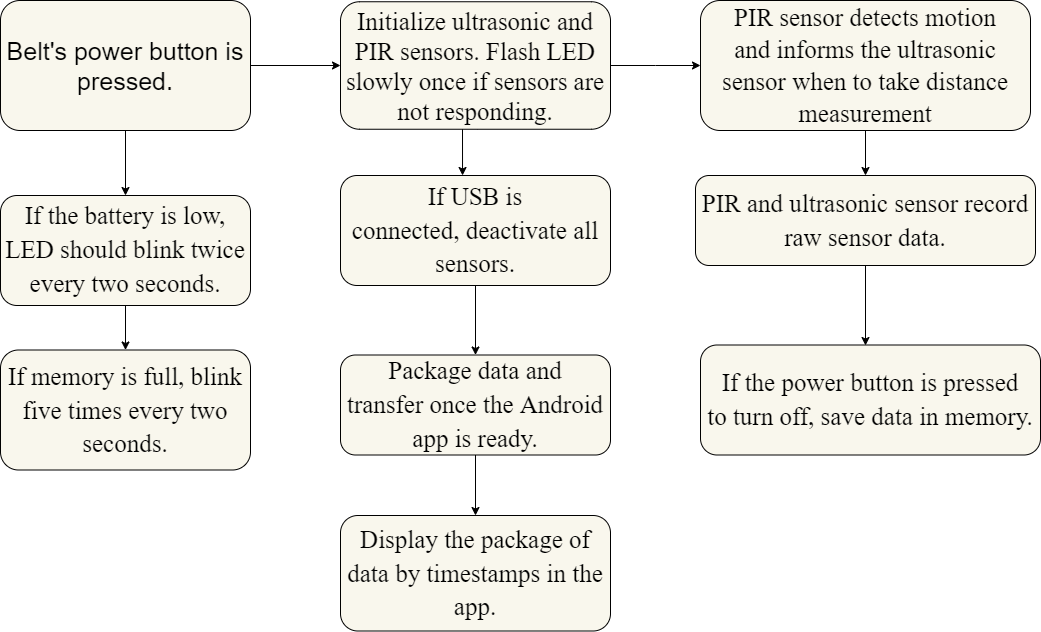


**Figure 2.2 GAS System Architecture**

Figure 2.2 shows the overall GAS software system. The inputs to the system come from the sensors’ measurements, which are stored in the database until they are transferred into the user app.

**2.2.1 Control Detection**

Various control states are required for the proper functioning of GAS. These states will allow GAS to not only record all necessary measurements but also present reminders to the user and physical therapist when the memory is full and battery is low. Figure 2.3 describes the crucial control states needed for the Arduino operating system.



**Figure 2.3 Control Detection Conditions**

As shown by Figure 2.3, when the power button is pressed, initialization of all sensors begins. Simultaneously, operating checks of the ultrasonic and PIR sensors are done to ensure the successful functionality of recording the gait data. These operating checks confirm that the sensors are working properly by triggering each sensor to see if the sensor outputs any data; however, these outputs from the operating checks are not recorded. For the detection of walking and arm-swaying patterns, the sensors adhere to a distinct hierarchy. First, the system ensures the outer-ankle PIR sensors do not detect objects near the user. Next, the inner-ankle ultrasonic sensor measures the parallel distance between the user’s legs after being triggered by the inner-ankle PIR sensor when the opposite leg is detected. After that measurement, the software timer begins when the user’s feet are in parallel to increment the step counter and obtain the step time. Finally, the timer resets once the user’s feet return to the parallel position as signaled by the inner-ankle PIR sensor. The following is the pseudocode for this process:

Power button PRESSED:

INIT PIR sensors

INIT Ultrasonic sensor

INIT LED

Check Memory

Check sensors

IF sensors not working THEN

BLINK LED

IF memory is full THEN

BLINK LED 5 TIMES

TRIGGER PIR SENSORS

WHILE outerPIRSensors detect Nothing

IF innerPIRSensor don’t detect motion THEN

CLEAR FLAG

ELSE IF !FLAG && innerPIRSensor detect motion THEN

GET DISTANCE, TIME, STEPS

IF armPIRSensors detect Motion

GET # of Arm Sways

IF leftAnklePIR sensor && leftWristPIRSensor

INCREMENT Arm-Walk Counter

**2.2.2 Data Processing**

Data processing starts with reading correct pin voltages and taking the raw data from the sensors. The Arduino’s application programming interface converts the ultrasonic sensor’s analog data to digital data. A built-in Arduino function, “pulseIn(),” is used to gather the speed of sound, distance, and time. As for the PIR sensor, it differs in detection, as the software needs to check only if the pin is high. Utilizing both sensors, the arm-sway counter, step time, step counter, and the width between the user’s feet are immediately measured. The PIR sensor data appends to the ultrasonic sensor’s data to form a calculated table of the average width between the user’s feet, the average time taken for each step, the average time and direction of arm swaying, and the position of the arms relative to the feet.

**2.2.3 Memory Storage**

The memory storage is where all of the data is stored from the Arduino. Since the physical therapists need to transfer data into the Android app, the data must be stored in a specific format that is recognizable by the application. A one-hour walking session is approximately 55 kB of data; therefore, to help the user recognize when the memory storage is full, the LED blinks 5 times every 2 seconds. This visual reminder keeps users from continuing a session when the Arduino halts data collection. Finally, the calculated data is stored as 16-bit integers, 32-bit longs, and a 1-bit bool within memory [21].

**2.2.4 Data Transferring**

The data communication between the app and the Arduino is a wired communication through USB 2.0. First, the user needs a USB cable to connect from the device to a smartphone containing the installed application. GAS houses a start button on the belt, which turns the system on or off. Once the user connects GAS to the smartphone via USB and presses the start button on the belt, the Arduino deactivates all sensors and prepares the system for data transfer. The application has a connection button, which is handled by an onClick method. The onClick method searches for all connected devices and connects only with an Arduino that has the same vendor ID that was given to the user. Once the method finds the correct device, it requests permission from the device to connect [29]. The application also contains an upload button, which is handled by another onClick method and can be pressed after the serial connection is established. Whenever the upload button is pressed, its handle function sends commands to the Arduino to start packaging data from the memory. The transferring process is done by the USBSerial library on Android, which is responsible for serial communication over the USB cable [30]. The packaged data of the walking session is then stored on the database of the application along with its timestamp. Finally, the application displays the results synchronously on the user’s interface and is organized based on the dates of the completed walking sessions.

**2.2.5 Operating System**

Table 2.9 outlines the requirements for GAS’s app operating system, along with three potential selections. Cells with ideal traits are highlighted in green, cells with acceptable traits are highlighted in yellow, and cells with undesirable traits are highlighted in red.

**Table 2.9 Operating System Comparisons**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OS** | **USB 2.0 Compatible?** | **Acceptable**  **SDK?** | **Portable**  **(High – Low)** | **Cost to Publish** |
| Requirements | Yes | Yes | High | ≤$25 |
| Android | Yes | Yes | High | $25 (one time) [31] |
| iOS | Yes | Yes | High | $99 (per year) [32] |
| Windows/PC | Yes | Yes | Low | $0 |

Table 2.9 describes key aspects considered when choosing GAS’s operating system. All options contain USB 2.0 compatibility for data transferring and acceptable software development kits (SDK) to develop a complementary application for the Arduino. However, the differences begin with the portability constraint. A Windows/PC, for one, is less portable than an iOS or Android smartphone. Smartphones, on the other hand, allow the physical therapist to analyze and review a patient’s data easily, whether the physical therapist is working in the office or traveling to a home-health appointment. Finally, the cost to publish an application entails releasing the application to a distribution center, such as the Google Play Store, the App Store, or for download on GAS’s website. Android and Windows/PC require lower fees and fewer compliant policies than iOS. As a result, Android is the best option when regarding the constraints of the preference of operating system.

**2.2.6 Smartphone Application**

After the measurement data are stored in the external memory of the system, the user can start transferring data into the application. Since the users of the application are primarily physical therapists, the application provides them with different charts and diagrams as a tool for analyzing their patient’s walking data. The smartphone application also allows the physical therapist to view all the data from the previous walking sessions and organizes these sessions based on their timestamps. Figure 2.4 shows the user interface (UI) design of the application with its functionalities. To upload the data of a new session, the user needs to connect the smartphone with the belt via a USB cable. Then, the user presses the upload button to start the uploading process. Finally, the application updates the list of the sessions and displays the last uploaded walking session.



**Figure 2.4 GAS’s UI Design**

As shown in Figure 2.4, physical therapists can view all the current sessions organized by the date of the sessions. Through the use of charts and graphs, the physical therapist is easily able to see how certain aspects, such as the time measurements for each step, fluctuate over the walking session, allowing the physical therapist to gain a better understanding of how his or her patient is walking. Furthermore, each patient has a unique ID that helps the physical therapist identify their record. There are other functionalities on the application such as clearing sessions, settings, and some instructions. Overall, this app makes it easier and more convenient for physical therapists to monitor improvements in their patients and make any corrections to resolve any afflictions that impede a person from walking correctly.

**References:**

[1] “Arduino Uno Rev3,” *Arduino | store.arduino.cc,* 2020. [Online]. Available: https://store.arduino.cc/usa-uno-rev3 [Accessed 16-Sept-2020].

[2] “Arduino Mega 2560 Rev3,” *Arduino | store.arduino.cc*, 2020. [Online]. Available: https://store.arduino.cc/usa/mega-2560-r3 [Accessed 28-Sept-2020].

[3] “Arduino Due,” *Arduino | store.arduino.cc,* 2020. [Online]. Available: https://store.arduino.cc/usa/due [Accessed 28-Sept-2020].

[4] “Arduino Nano,” *Arduino | store.arduino.cc*, 2020. [Online]. Available: https://store.arduino.cc/usa/arduino-nano [Accessed 13-Oct-2020].

[5] “Getting Started with Arduino UNO,” *Arduino* | *ardunio.cc,* 2018. [Online]. Available: https://www.arduino.cc/en/Guide/ArduinoUno [Accessed 16-Sept-2020].

[6] T. DiCola, “How PIRs Work,” *Adafruit,* 2014. [Online]. Available: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work [Accessed 16-Sept-2020].

[7] F. Mims III, “How to Use LEDs to Detect Light,” *Make: Community,* 2014. [Online]. Available: https://makezine.com/projects/make-36-boards/how-to-use-leds-to-detect-light/#:~:text=Insert%20the%20fiber%20and%20cement,to%20power%20a%20second%20LED [Accessed 16-Sept-2020].

[8] “Accelerometer,” *Omega.* [Online]. Available:ttps://www.omega.co.uk/prodinfo/accelerometers.html#:~:text=An%20accelerometer%20is%20a%20device,the%20force%20exerted%20upon%20it [Accessed 16-Sept-2020].

[9] “Using Accelerometers to Estimate Position and Velocity,” *CHRobotics.* [Online]. Accessed: http://www.chrobotics.com/library/accel-position-velocity [Accessed 16-Sept-2020].

[10] “HC-SR501 Passive Infrared (PIR) Motion Sensor,” [Online]. Available: https://www.epitran.it/ebayDrive/datasheet/44.pdf [Accessed 17-Sept-2020].

[11] “SparkFun OpenPIR,” *Sparkfun Electronics.* [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/Sparkfun%20PDFs/SEN%E2%80%9013968\_Web.pdf [Accessed 17-Sept-2020].

[12] “Passive infrared human detection sensor with built-in amp,” *Panasonic.* [Online]. Available: https://www.alliedelec.com/m/d/9b0187f5be4c4127a102d8868f3222c2.pdf [Accessed 17-Sept-2020].

[13] Blackstock, T., “Step Speed,” September 2020.

[14] Blackstock, T., “Step Width,” September 2020.

[15] “Leveraging the benefits of Time-of-Flight camera technology in 3D machine vision applications,” *Adept Turnkey.* [Online]. Available: http://www.adept.net.au/news/newsletter/201111-nov/article\_tof\_Mesa.shtml [Accessed 28-Sept-2020].

[16] “Ultrasonic Sensor working applications and advantages,” *Microcontrollers tutorials & projects,* 2017. [Online]. Available: https://microcontrollerslab.com/ultrasonic-sensor-working-applications-advantages/#:~:text=presence%20detection%20system.-,Advantages%20of%20Ultrasonic%20sensor,the%20external%20or%20deep%20objects.&text=These%20sensors%20have%20greater%20accuracy,and%20depth%20of%20parallel%20surface [Accessed 16-Sept-2020].

[17] “Ultrasonic Ranging Module HC-SR04,” *ElecFreaks.* [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf [Accessed 16-Sept-2020].

[18] “PING))) Ultrasonic Distance Sensor (#28015),” *Parallax.* [Online]. Available: http://www.farnell.com/datasheets/2622313.pdf [Accessed 16-Sept-2020].

[19] “US-100 Ultrasonic Distance Sensor – 3V or 5V Logic,” *Adafruit.* [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/4019\_Web.pdf [Accessed: 16-Sept-2020].

[20] T. Blackstock, Private Communication, September 2020.

[21] “C++ Variable Types,” *TutorialsPoint*.[Online]. Available: <https://www.tutorialspoint.com/cplusplus/cpp_variable_types.htm> [Accessed: 27-Sept-2020].

[22] "CAT25512" *OnSemi.* [Online]. Available: <https://www.onsemi.com/pub/Collateral/CAT25512-D.PDF> [Accessed: 28-Sept-2020].

[23] "LE25S81A" *OnSemi.* [Online]. Available: <https://www.onsemi.com/pub/Collateral/LE25S81A-D.PDF> [Accessed: 28-Sept-2020].

[24] "Arduino MKR MEM Shield," Arduino | store.arduino.cc, 2020. [Online]. Available: <https://store.arduino.cc/usa/mkr-mem-shield> [Accessed: 28-Sept-2020].

[25] “What is the Difference Between Flash Memory and EEPROM?” *ElectronicsForU.* [Online]. Available: <https://www.electronicsforu.com/resources/learn-electronics/eeprom-difference-flash-memory> [Accessed: 14-Oct-2020].

[26] "6LR61," *Energizer.* [Online]. Available: <https://data.energizer.com/PDFs/Ind-6LR61PL_EU.pdf> [Accessed: 27-Sept-2020].

[27] “12V 3,000 mAh NiMH Battery Pack, PP45,” *Studica.* [Online]. Available: https://www.studica.com/us/en/Studica-Robotics/12v-3-000-mah-nimh-battery-pack-pp45\_1/70018\_1.html?ex\_ref=google\_feed&gclid=EAIaIQobChMIzZDt8dDx6wIVF0qGCh0B3AZ4EAQYAiABEgJHv\_D\_BwE [Accessed: 17-Sept-2020].

[28] “DL-5 Battery,” *Complete Battery Source.* [Online]. Available: https://www.completebatterysource.com/dl-5-battery.php?gclid=EAIaIQobChMI\_YXw7NHx6wIVGEmGCh1lVgdXEAQYAiABEgJYtvD\_BwE [Accessed: 17-Sept-2020].

[29] H. Mathavan, “Communicate with Your Arduino Through Android - Projects,” *All About Circuits*, 2015. [Online]. Available: https://www.allaboutcircuits.com/projects/communicate-with-your-arduino-through-android/. [Accessed: 29-Sep-2020].

[30] felHR85, “felHR85/UsbSerial,” *GitHub*. [Online]. Available: https://github.com/felHR85/UsbSerial/. [Accessed: 18-Sep-2020].

[31] “How to use the Play Console,” *Google*. [Online]. Available: https://support.google.com/googleplay/android-developer/answer/6112435?hl=en. [Accessed: 14-Oct-2020].

[32] KMT, “How much does Apple charge for free apps hosted on the app store?,” *Apple*. [Online]. Available: https://developer.apple.com/forums/thread/76587. [Accessed: 14-Oct-2020].