UNIVERSITY OF SOUTHAMPTON
Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

Supervisor: Prof Neil white
Secondary Examiner: Prof Hywel Morgan

Personal At-home Monitoring (PAM) device for the diagnosis and treatment of patients

by John Hawk

11th May 2020

Word Count: 8,807

Abstract

The current diagnosis/treatment process revolves around occasional meetings between a doctor and their patient, or possibly measurements taken by patients. This leads to long periods of time where data, that could be helpful to the doctor, is unavailable or inaccurate. To solve this, a proposed device will be developed to act as a data collection and storage device. It will be worn constantly by the patient and will log data from sensors autonomously, and will be an open source platform to allow any sensors to connect, and store data, to the device. The interface between the device and the sensors will be based off the I2C protocol with a custom data package structure. This proposed system will periodically communicate to all sensors connected to the network to gather their current values, and will then process and store the information on an SD card as a CSV for easy access of data. Patients would be able to view and potentially edit their data through a web-based application communicating with the device via the web Bluetooth protocol.

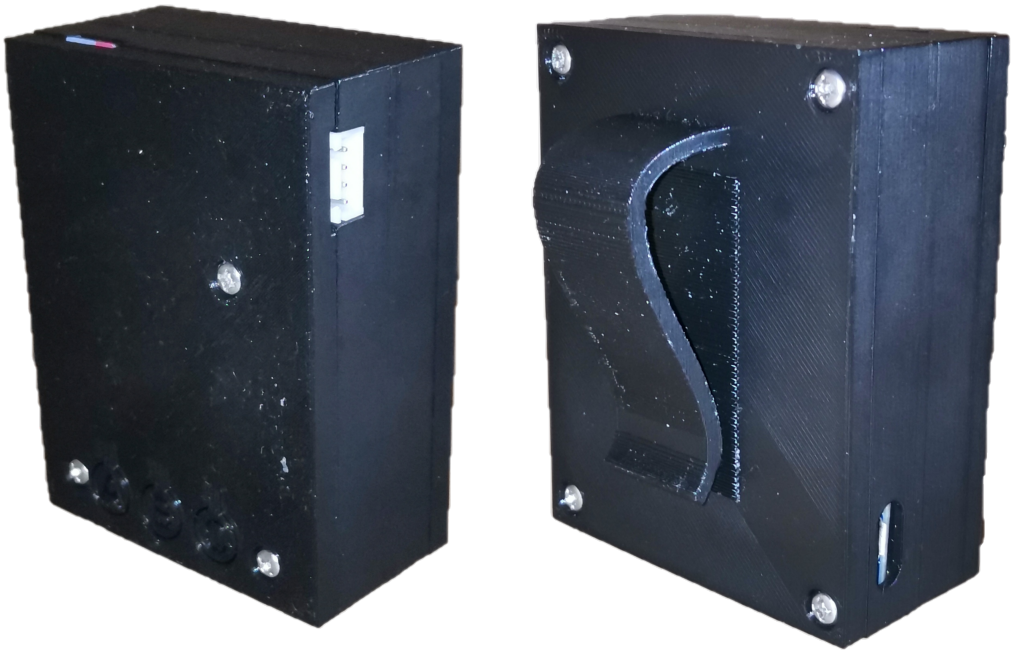


FIGURE 1: The PAM device

Contents

List of Figures

1 Project Problem/Goals

When doctors need to diagnose or treat a patient, their only option is to rely on regular check-ups and potentially inaccurate recording from patients. For the patients, they must remember to keep a record of their health between check-ups. This leads to an unsatisfactory situation where patients may not be getting diagnosed as quickly or treated as well as they would otherwise be if the doctors had access to more accurate information about their day to day lives. To solve this, the PAM device has been created. The aim of the project was to create a prototype, proof of concept, platform for medical data collection, which can easily read data from third party external sensors as well as allowing anyone to develop a data collection device to work with the platform, or an application to access the data. To prove the functionality of the platform, a prototype core PAM device has been made, as well as an application to view the data, and an external sensor.

1.1 Design Philosophy

This product is designed to collect whatever data is required, to make diagnosis and treatment an easier task. It is to be a device prescribed by health care workers, and used by patients during their daily lives, before being returned to doctors who can use the data collected. Access to accurate information about various factors of a patient's life (such as their heart rate, light levels around them, or air quality in their vicinity) will allow doctors and analysts to find the cause of problems, and monitor recovery, far more easily; I.E. being able to see in what scenarios a patient may have a migraine, or whether a stroke patient's symptoms are healing. There is currently no device which provides medical grade information about a patient's health during their daily life. This device would be the first product designed to be small, simple, and comfortable enough to be used for prolonged periods of time without supervision from health care workers, allowing for easier and more accurate data to be collected. This data should assist in decreasing how long it takes from first symptoms to a treatment in certain situations. While an excess of data would not help in every situation, for certain cases it would be extremely helpful. The aim of this project is not to fully develop a deployable ecosystem of sensors and data collection devices, but rather to explore the ways in which such an ecosystem could be created by developing the first steps and defining the standard, culminating in a prototype platform to showcase the potential for such a device.

1.2 Analysis and Specification of the Solution

By designing the monitoring device as a platform instead of a standalone product, it allows for further expansion without the need for first-party sensors. This reduces the future cost of development of first party sensors for the system, or even for expensive development of sensors entirely as sensors currently on the market can be easily modified to function with this platform. This also allows the main device itself to be improved and replaced if the standards defined are upheld, meaning anyone can make a device as a drop-in replacement for the PAM device developed here.

To communicate between the sensors and the control box, multiple protocols were considered, namely: I2C (Inter-Integrated Circuit) [1], SPI (Serial Peripheral Interface) [2], UART (Universal Asynchronous Receiver/ Transmitter) [3], and OWI (One Wire Interface) [4]. To make the choice between the protocols, five aspects were compared: number of wires to connect devices, number of possible sensor devices, additional hardware requirements, complexity of implementation, and maximum data speed.

	I2C	SPI	UART	OWI
No. of wires	4 (VCC, GND, SCL, SDA), including power	3 + a wire per device (SCK, MISO, MOSI, SS), excluding power	4 (VCC, GND, TX, RX)	3 (VCC, GND, DATA), including power
No. of devices	127	Limited by the number of spare pins	1	2^{64}
Additional hardware	N/A	N/A	N/A	Maxim I2C to OWI conversion chip
Complexity	Medium	Medium	Low	High
Max data rates	400 kbps.	No specified limit	Device dependant	16.3 kbps

From this, I2C was chosen as it allows for a large number of connected devices with a small number of wires, and without additional hardware and the added complexity that brings, while still being fast enough for this usage.

By using an SD card to store the data, the information collected and saved by the device can be easily read by and computer with an SD card or a USB port. The CSV file type used to save the data can be accessed and edited by any text editor/spreadsheet manager. This means the device's data can be easily read, edited, and saved by both the patient and doctor.

The choice to use Bluetooth 4.0 with a web-based application as opposed to a dedicated mobile application, was due to the simplicity of using a website to access the data on a range of devices. A web-based application is also simpler to programme than dedicated applications, as one website can be accessed on a multitude of devices, rather than needing to rewrite the application per operating system. Due to the use of a web-based application, Bluetooth 4.0 had to be used (as opposed to older, simpler standards such as Bluetooth 2.0) as the web Bluetooth standard the application relies on, requires Bluetooth 4.0 or higher to function[5].

A separate power board will be used to provide the 5V required by the main board, from the battery, as current experimentation has shown power management to be far more complex than initially anticipated. Therefore, for this prototype, a power board developed and tested by a major manufacturer will be used to reduce the workload and ensure a stable power supply. It will also ensure a safe charging and discharging of the lithium-ion battery that will be used to power the system.

The IMU on the main board has been added to allow basic step tracking without additional sensors, as basic exercise tracking may be very useful information to have for any treatment/-diagnosis. The ATMega328p-au was chosen as a processor that works natively with the Arduino ecosystem, allowing for easy programming, while also being low powered, having a small PCB footprint, and with the necessary serial hardware/pins.

2 Report on Literature Search

There have been a few systems similar to this proposed platform, however none of them function exactly as required for this problem. Three have a similar aim, to collect and store data in real time from patients, however, each of these systems have flaws which prevent them from being used in this precise situation. Each solution examined requires a smartphone to collect the data which limits the ability to control the hardware provided and providing a phone to each patient would be prohibitively expensive.

The first of the three solutions [6], connects wireless sensors to a smartphone using BLE and relays the collected data to a machine-learning algorithm. This system works well for automated diagnosis, however for long periods of time individually powered sensors would be annoying to recharge separately. This system proves that BLE is sufficiently efficient that it can be used to get data from the proposed device, without quickly draining the battery.

The second of the three solutions [7] uses a similar structure. A smartphone connects to a series of sensors through a standard connection interface, and relays that data back to a server to detect issues and changes in the patient's vital signs. The fact that the solution requires the manual

recording of data will lead to issues with patients forgetting to input and therefore having gaps in the data.

The third solution examined [8] is a system used to track the vitals of multiple people at disaster scenes using cheap sensors connected to a central tablet PC. Each sensor is powered with AA batteries, and requires a network connection to function. These make sense for the environment it is designed for, but for the proposed usage the heavy replaceable batteries and network connection would not function well.

There has been a selection of systems that have sections that may be beneficial to this project. A wearable sensor network developed by Righetti and Thalmann [9] used I2C to connect an unknown number of nodes together and allowed for simple communication between them. This network proves that I2C can be used as the interface between the main control box and the various sensors that will be used. The paper does bring up issues with I2Cs address conflict issues, but this can be mitigated with a single master multi-slave set-up.

Another proposed system [10] creates a network of bedside monitors connected to a central system to collate the data. The bus used has a maximum data transfer speed of 100kB/s with each bedside taking 5kB/s bandwidth, resulting in a maximum of 16 bedsides (allowing for overhead),and the researchers used an artificial patient to test their system. With I2Cs maximum data rate, assuming the external sensors use less bandwidth than a hospital bedside monitor, at least 10 external sensors may be connected. The method of simulating patient data will be useful when testing the proposed device.

The final system explored is a universal interface between sensors and microcontrollers [11]. The researchers developed a hardware interface layer to help a range of sensor types (analogue, digital, I2C, etc.) and allow a microcontroller to communicate to them through either an SPI or a UART interface. A similar solution will be used to convert standard sensors to work with the proposed platform, a small microcontroller (possibly an avr-based ATTiny) will be used to read the sensors value and output the data in accordance to the protocol.

To view the data collected from the proposed platform, there must be an application on the user's device, I.E. a smartphone or a laptop. This can either be a dedicated application or a web application. However, as explored by David Forunato and Jorge Bernardino [12], a web based application provide many benefits over tradition applications. The authors point out faster development, ability to run on any device, and centralised implementation allowing constant updating without requiring the user to update an application, as advantages of developing a web based system. They do also note that dedicated applications tend to have superior performance, tend to be more stable, and are installable from a trusted app store. For this project, the benefits of a web application outweigh the drawbacks, as the application development needs to be fast and to work on a wide range of devices.

When discussing the development of web applications, the authors of that paper proposed the use of four frameworks: React, Vue, Angular, and Ionic. All four of these are based on javascript (or typescript, a more strictly typed version of javascript) and aim to provide very similar functionality. The authors argue that the use of a framework speeds up development and allows for a more native feel than with traditional web development. For this project, a framework will be used for these reasons, however, the Boostrap framework was chosen over the ones suggested in the paper. To read the paper, Google Translate for documents [13] was used as the paper was originally published in Portuguese, so the translation may not be entirely accurate, but the general ideas of the paper should have been accurately represented.

3 Detailed Design

The PAM device consists of a central data collection box worn on the waistband of the user, which connect to external sensors such as a heart rate sensor or an air-quality monitor. These sensors communicate with the central box using a custom defined data structure, sent over I2C to minimise the number of cables used. By creating a standard data format and transmission protocol, anyone can create sensors to be used with this system.

As the PAM device collects data, it is stored on an SD card with the time of recording as a CSV file; to sort the data, a new file is created for each day the PAM device is used. A website has been created which connects to the PAM device via BlueTooth and display the data as easy to understand graphs. The transmission of data between the device and the website is in a simple defined structure, such that (as with the external sensors) any third party can develop a data visualisation platform and easily integrate it with the device.

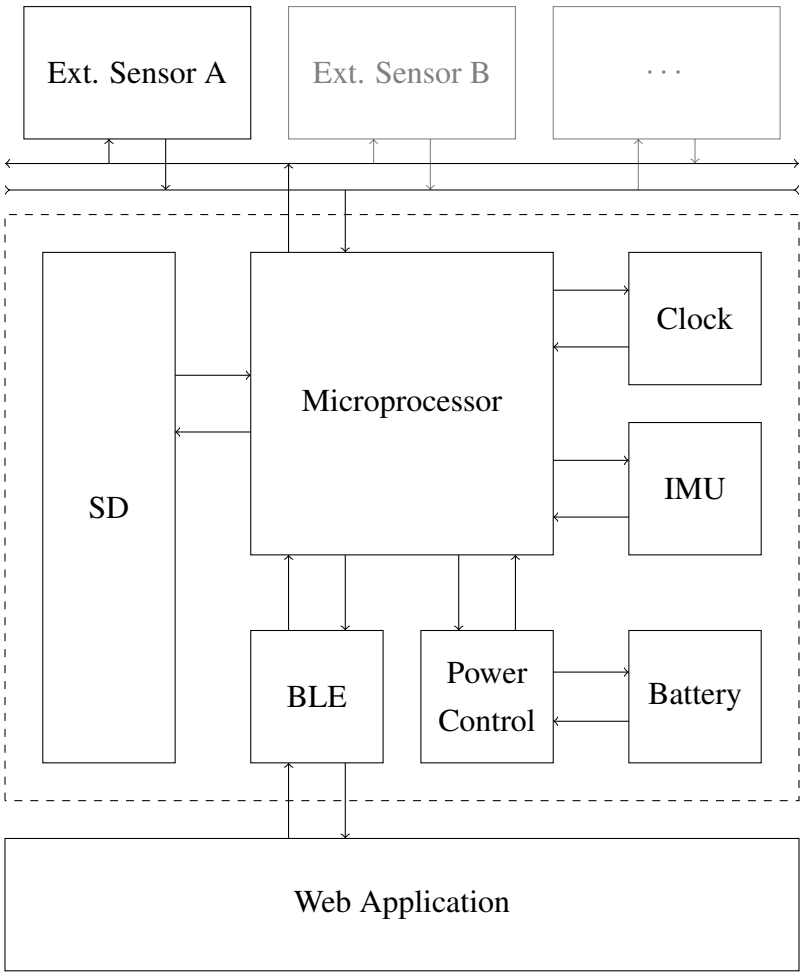


FIGURE 2: Block Diagram of the PAM Device

On the device, there are three buttons designated for user input; these are used for specific inputs that a sensor cannot be made for, such as when the user suffered a migraine, or when they ate. The number of times the buttons are pushed in any given interval is recorded onto the SD card alongside the data from the sensors. In addition to the buttons and the external sensors, the PAM device has an inertial measurement unit on board which is used to count the steps of the user to provide extra information into the user's daily habits.

The device is powered from a 2000mAh lithium polymer battery which is charged and boosted using a separate PCB to provide the stable power required to the main custom PCB. The main board, charging circuit, and battery are all housed in a 3D-printed enclosure with an integrated belt clip to allow the device to be worn easily.

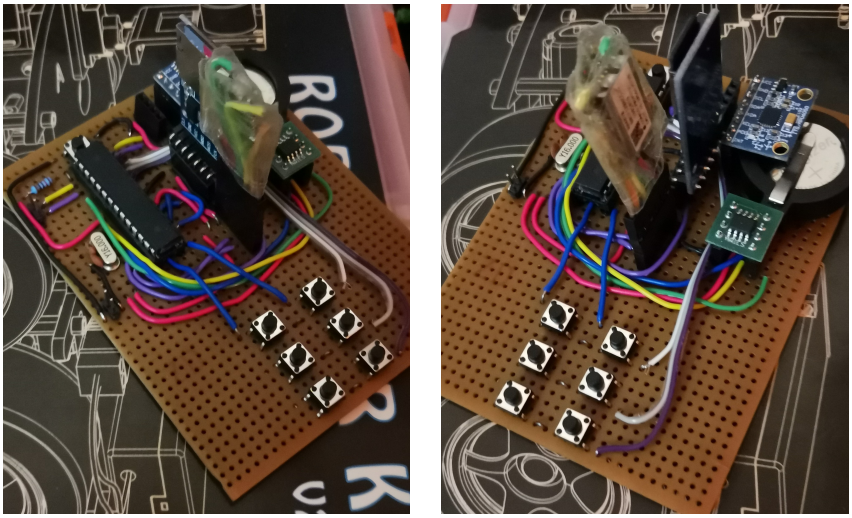


FIGURE 3: Prototype PAM Device

To ensure the designed system was capable before developing the final product, a prototype design using the same components in simpler forms (i.e. breakout boards where available) was built on protoboard. The prototype used sockets to connect the main components to the protoboard, using the Dual In-line Package (DIP) version of the ATMega328p to make rapid production simpler. Through hole versions of the other components, such as the $16MHz$ oscillator, were used as opposed to the surface mount versions used in the final versions to allow this prototype to be developed quickly. As each breakout board has their own voltage regulation if required, there was no need to include a $3.3V$ line on this prototype.

After basic testing, to ensure the prototype had the required basic functionality, the final design was developed.

4 Implementation

4.1 Hardware

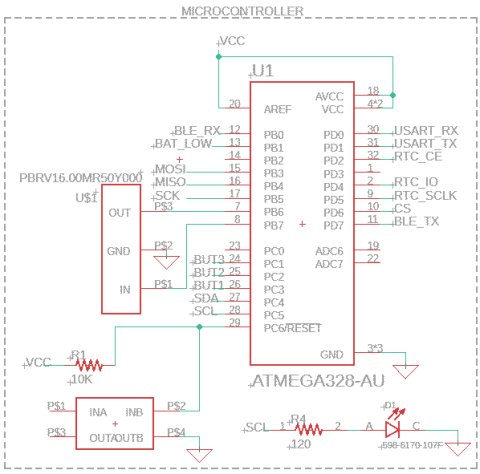


FIGURE 4: Microprocessor segment schematic

As mentioned, the PAM device is powered by the ATMega328p-au AVR microcontroller [14], chosen for its ease of compatibility with the Arduino programming environment. The Arduino environment was selected to make prototyping quicker and easier with a large choice of libraries for each section of this project. The processor operates at 16MHz, provided by an external oscillator, which is fast enough to provide all the functionality required from the PAM device. In addition, it has hardware UART, SPI, and I2C ports allowing simple communication with all areas of the device. Initially an LED was added to the system to add some visual feedback, however as this caused issues, it was not added to the final board.

At the core of the device is a custom designed PCB. This was created using Autodesk Eagle [15] for the design and manufactured by JLCPCB [16], before having the components soldered onto it by hand. This choice of assembly meant the components chosen had to be larger than otherwise necessary. Predominantly choosing the 0805 size (0.08 x 0.05, approximately 2mm x 1.5mm) as this is small enough to be compact, but large enough to be soldered by hand easily. The components also needed to be spread further apart than would be required with a pick and place machine, to allow for easy access and reduce the likelihood of heat damage to surrounding components. By using surface mount components instead of through hole components, the dimension of the PCB can be reduced, which on this iteration are approximately 5cm by 7cm, allowing it to be easily incorporated into a device worn on a user's belt.

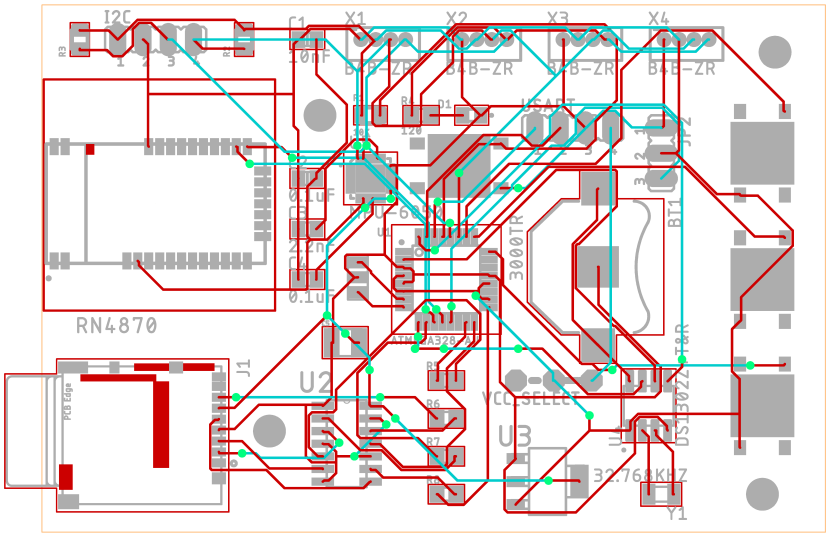


FIGURE 5: Design for the PCB

The time keeping functionality is provided by the DS1302 RTC (Real Time Clock) IC from Maxim [17], this chip stores the current time as epoch time with an arbitrary start time. This RTC has an additional 3V backup coin cell battery to ensure that accurate time is maintained if the main power to the board fails. Communication to the RTC is achieved via a software implementation of a three-wire interface, allowing the device to easily request the time when it is required. The DS1302 also has a small amount of on-board memory, however this was not required.

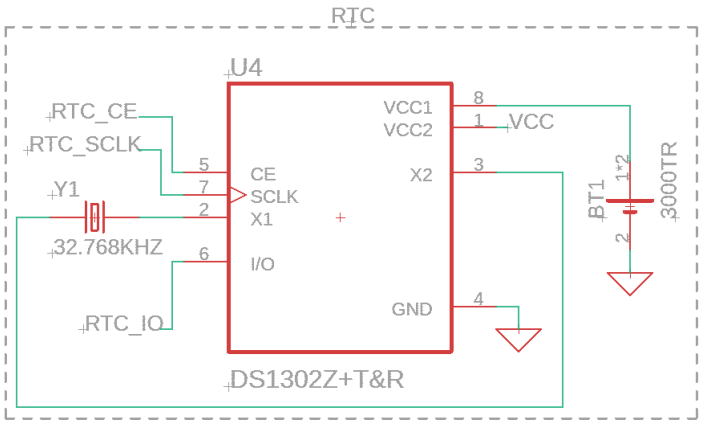


FIGURE 6: RTC segment schematic

The original plan for the SD card was to implement it directly onto the main board, using an SN74LVC125A buffer gate as a 3.3V level shifter because the SD standard [18] does not support the 5V logic from the processor. However, due to an unsolved issue causing shorts between the 5V rail and ground, the entire SD card section of the board has been replaced

with an SD card breakout board designed to work with the chosen processor. This breakout board has been adhered to the main board, with jumper wires soldered to the necessary pads to ensure correct communication and power to and from the SD card. To communicate with the processor, the SD specification supports SPI, allowing simple connections between the card and the processor.

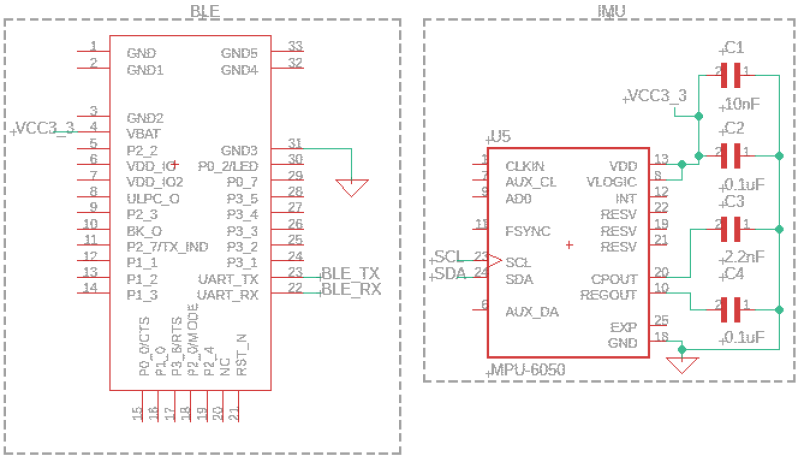


FIGURE 7: BLE and IMU segment schematic

For step counting, the MPU6050 IMU (inertial measurement unit) from Invernesses [19] was chosen due to its simple I2C interface and prevalence, ensuring that there are many easy to use libraries which work with this chip. As discovered after designing the main board, the MPU6050 has a pin used to set the I2C address of the chip, since this was not tied low, the IC defaulted to an address of 0x69 (Decimal 105 as opposed to the standard 0x68 (Decimal 104) used by the majority of libraries. Therefore, to function correctly, the library used had to be modified to use address 0x69. As the step counting only relies on the accelerometer inside the IMU, a simpler accelerometer only chip could have been used. However, to allow for a more advanced pedometer function in the future, the MPU6050 is still a good choice.

As the PAM device needs to be accessible via a web interface, the device has to support Bluetooth 4.0 (BLE), so the RN4870 BLE chip from Microchip [20] was selected as it provides access to all features of BLE, whilst remaining simple to integrate and use. The RN4870 was connected to the microprocessor using a UART interface set to 9600 baud. As the only hardware serial port on the processor was being used for the programming header, the RN4870 had to communicate via a software serial port. This is also why a slower baud rate was chosen, as it was not necessary for instant communication, but it was important that the data was sent accurately. To set the device name and enabling transparent UART, the microprocessor was used to pass all data between the hardware and software serial ports, allowing a terminal to access the RN4870. Once set up, the transparent UART mode on the RN4870 allowed the microprocessor to easily communicate to any Bluetooth device as though it were connected via UART.

To provide stable power to the board, and to reduce development time, the Adafruit PowerBoost 500C [21] was used. This provides a stable 5V from a standard LiPo (Lithium-Polymer) battery (which usually output 3.7V) at 500mA, whilst allowing the battery to be charged via a USB Micro-B port included on the PowerBoost board. The 500C variant was chosen as the device draws less than the rated 500mA, so the larger PowerBoost 1000C was unnecessary.

The battery chosen was a 2000mAh lithium polymer cell as it will comfortably power the device for over a day, whilst remaining physically small enough to easily fit into a case. With the final design of the case, the battery was too thin, so strips of foam were added to either side to ensure the battery remained in place and was not damaged during use. The PowerBoost board does provide a low battery indicator pin, which has been connected to the processor to allow for low battery detection, however this feature has not been implemented in software. As the SD card and RN4870 chip require 3.3V, an AMS1117 LDO voltage regulator [22] has been included to provide a separate 3.3V rail from the 5V supplied to the main board.

For connecting to external sensors, the PAM device has a JST 4 pin connector to carry the two signals for I2C communication (SCL and SDA) in addition to the 5V and ground to power the external devices. This connector was chosen for its ubiquity and ease of use, whilst also being robust and able to withstand daily use. To ensure clean communication via the I2C bus, 3.3k pull-up resistors were added to the SCL and SDA lines. For the user input on the device, three surface mount buttons were added to the main board, each connected directly to a pin on the microprocessor. Due to an error in the PCB design, 10k pull-down resistors were added afterwards to ensure correct readings from the buttons.

FIGURE 8: The PAM device internals

To prove the functionality of external sensors, the PulseSensor analogue heart rate sensor [23] has been used in conjunction with an ATTiny402 [24] to create a sensor that works with the

system. The ATTiny processor waits for a request from the main board, reads the value from the sensor, and returns it to the main processor using the custom data format. As this sensor is being used to prove the concept rather than as a prototype itself, there is no casing, instead the sensor and the processor are held in place using Velcro straps. This heart rate sensor must be worn on the finger to get accurate results, so is not suitable for daily use.

4.2 Casing

The casing for the PAM device was developed as three separate segments which are assembles with the hardware for the device stored in the central segment and separated into two sections. On one half of this central segment, the main PCB is mounted using M3 screws through the three mounting holes. The battery and the PowerBoost 500C were mounted on the other side with foam tape and M3 screws respectively. Initially, the design had no connector allowing for the main PCB and PowerBoost 500C to be separated, so the hole between the two halves of the main segment was designed to be large enough for the PowerBoost 500C to fit through. The two boards are attached on raised mounting points, the threads of which are added by heat pressing M3 threaded knurled brass inserts into the mounting points. Cut-outs for the SD card, the JST connection for the external sensor, and the micro USB for charging, have been incorporated into the design for the central section.

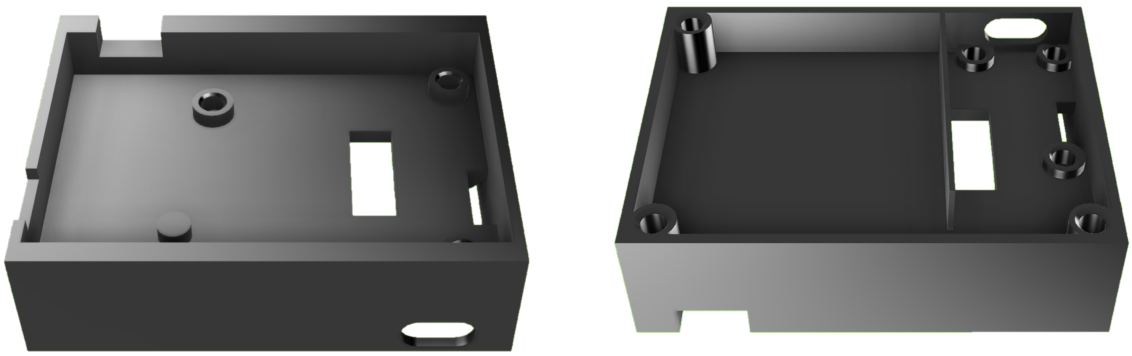


FIGURE 9: Centre section of the casing

To attach the rear cover, three additional threads have been added to the corners of the rear section. For the fourth corner, the mounting hole for the PowerBoost 500C was reused, with a longer screw to reach to the lower thread. The rear cover has the belt clip as part of the design which was modelled by sweeping a profile over a spline to create the required shape. The heads of the screws are recessed into the surface to prevent them getting caught during use.

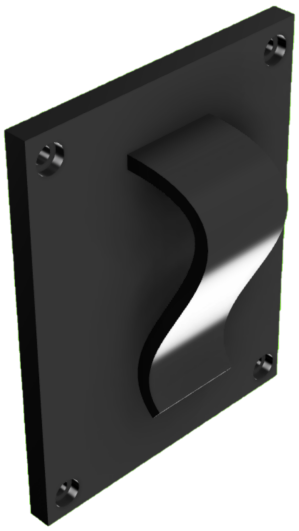


FIGURE 10: Rear panel of the casing

There was no room to add mounting holes in the corners of the front section, without increasing the size of the casing, so the mounting holes for the main PCB were re-used. The front panel has longer supports to ensure the PCB does not move around, and has additional cut-outs for the JST connector and SD card to ensure there is enough clearance for good fit. The buttons are added with a thinner area of the panel, to allow for a slight flex in the material, with pillars resting on the surface of the button, such that when the front panel is pressed, the button below is pressed as well. These thinner areas have letters recessed onto their surface to help distinguish the buttons from one another.

FIGURE 11: Front panel of the casing

This design was chosen as it allowed each section to be developed independently, speeding up development time, and allowing any issues to be solved faster. To produce the casing, 3D printing was chosen as it is perfect for rapid prototyping and is able to create the complex shapes required.

4.3 Software

To help the programme run smoother on the processor, the processing occurs in two stages which run every other millisecond. In the first stage, the device checks if any data has been requested from the website. As the Bluetooth connection is handled via the RN4870, the processor is not required to handle connecting/disconnecting to/from the client's device. If there has been data requested by the web application, the processor reads the data for that day from the SD card, processes it to extract the raw data and data types from the CSV file, and sends it to the website. The data is sent line by line with a slight delay to make receiving and processing the data easier on the website, however this increases how long the information takes to send.

In the second stage, the device records and stores the data. To record data from the board, the device polls both the buttons and IMU every loop to check if there have been any user inputs or steps. The device also checks if a set time has elapsed since the last set of data was saved. If this time has passed, the PAM device polls: the RTC, for the current time of saving; the pedometer function, to see how many steps have been taken since the last data saving; the buttons, to see if any have been pressed since the last data saving, and how many times; and every I2C address, to try and get data from the external sensors. This data is then added to the bottom of the CSV file for the day in the defined data structure, with a new file being created at the start of each day with the date as the title in the format YY-MM-DD, which complies with the 8.3 naming standard set out by Microsoft [25], to ensure compatibility with as many systems as possible.

To make data identification easier, each source of information (time, step count, heart rate, etc.) is provided with an 8-bit identification number (0x00 for time, 0x01 for step count, etc.). When the data is stored and transmitted, it is preceded by the associated identification number. By doing this, the data can be easily sorted by the PAM device, the website, or by a person reading the raw data from the SD card. Data on the device is processed as 32-bit unsigned integers, however, when communicating with the external sensors, for the proof of concept, data is being transmitted as 8-bit integers for simplicity. When requesting data from the external sensors, the PAM device waits for a 16-bit response, where the upper 8-bits correspond to the sensor data, and the lower 8-bits correspond to the sensor type. This limitation was imposed to ensure accurate data transmission; however, the system is capable of handling larger data packages.

To handle I2C/Software serial communication, and reading/writing from the SD card, the libraries provided as part of the Arduino environment were used [26]. These libraries made interfacing with the required sections of the microprocessor/PAM device simple, although they are slower and take up more memory than writing specific functions from scratch, as they have extra functionality that is not required. For connecting to the IMU and RTC, as the Arduino ecosystem does not have integrated libraries, the MPU6050 library by Tockn [27], and the RTC and ThreeWire libraries by makuna [28] were used, respectively. To ensure that the file saved had the correct time-stamp, a call back function was added to modify the file's data and time whenever it was saved.

The ATMega328p has only 2048 bytes of memory for variables. When combined with the required buffers for data recording, as well as the 512 byte buffer for reading/storing from/to the SD card, there was very little memory left for local variables, with around 72% of the dynamic memory reserved for global variables. To reduce the global variable limit, all static strings to be printed to the terminal, were stored in flash-memory using the F() wrapper. While this did allow the programme to run, due to the minimal space left there were many stability issues which caused the programme to crash. To compensate for this, the AVR ATMega series has a watchdog timer built in which allows the microprocessor to reset if a set idle time has elapsed. While this is not an ideal solution for a final product, it functioned well enough to help solve the problem of the processor crashing.

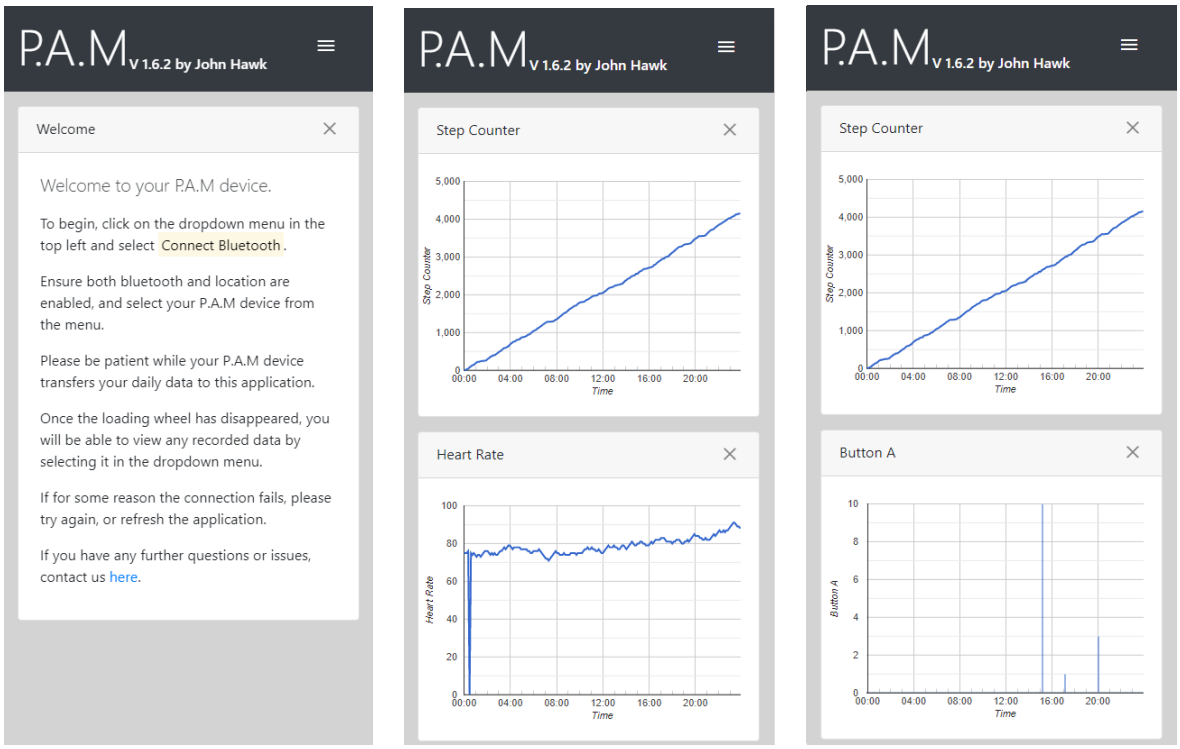


FIGURE 12: Example screenshots of the PAM web application

To view the data, the website pam.johnhawk.tech was developed. The graphical design for the website is built using bootstrap 4.4.1 from twitter [29], which was included by linking to the MaxCDN content delivery network for bootstrap. By using a content delivery network, less work needs to be done to ensure the most recent version of bootstrap is being used, and it can help speed up loading times as users may already have the CDN cached. Bootstrap was chosen for the website as it is designed to create mobile-first responsive interfaces, making it simple to create an easy to use application on both mobile and desktop, while focussing on mobile as most users will be accessing the site from their phones.

The google charts API [30] has been chosen to display the data from the device after it has been received and processed. This API was selected as it is simple to work with while creating easily readable graphs with useful features, such as being able to read precise values by clicking on the graphs. As with Bootstrap, this API has been included using a content delivery network from googleapis.com. In addition, the google icons API [31] has been used to provide a few symbols such as the close symbol.

To handle the Bluetooth communication, the web Bluetooth JavaScript functions have been used. The Web Bluetooth standard is defined by the Web Bluetooth Community Group [5] and are widely accepted by web browser developers; however, some browsers, namely Apple's safari, do not support this feature. This is one of the issues with relying on new features on browsers, however websites are easier to access when compared to downloading applications, so this is an acceptable limitation.

As the RN4870 on the PAM device is set to transparent UART mode, communicating to the chip involves reading and writing to specific characteristics as specified in the standard. To access these characteristics, the website must specify the name of the device it wishes to connect to, extract the data from the primary service (where the UUID is specified by the standard), where the characteristics can be found. At this point the website can set up an event listener to update when new data is received and can write data to the RX characteristic.

FIGURE 13: Scan for the PAM web application

To ensure that the website can be accessed from any device and that the Bluetooth functionality works (which requires an HTTPS connection to ensure security), the website is hosted using GitHub's GitPages [32]. This service allows for easy hosting and editing when combined with

GitHub Desktop [33], however it would not be an ideal host for many concurrent users, where a dedicated host would be better such as Amazon Web Services [34], or Microsoft Azure [35].

5 Testing

5.1 Strategy

Due to the Covid-19 pandemic and the subsequent lockdown, the testing methodology for the PAM device had to be changed. Initially the device was to be given to university students to be used for a day, with a questionnaire to gauge the user experience. However, as interacting with new individuals was forbidden at the time of testing, this technique was impossible. To compensate for the lack of testing participants, and the change to daily routines, the testing for the PAM device was broken into different tests for each section, with a final test to assess the functionality of the whole product.

Before writing the programme for the PAM device, each component of the main board was tested using simple test scripts from their respective libraries where available. However, some components, such as the microprocessor and the buttons, needed to have custom testing scripts written. In addition, a script to count steps from the IMU data was written and run separately to test the ability for the PAM hardware to act as a pedometer. These small programmes predominantly consist of writing values to the serial terminal in response to data from each component. In addition, to ensure the RTC maintained the time in the event the main battery died, the power to the main board was disconnected for a random time interval between ten seconds and a minute, before re-connecting and checking the time saved was still correct.

Once each section was verified, the project was broken into three larger segments: storing sensor data, transmitting sensor data, and viewing sensor data. To test storing the sensor data, the polling time for information was set at one second, far faster than it would be for the final device, to ensure that an adequate amount of data was collected. This data was sent to both the serial terminal for instant comparison and to the SD card. The data saving functionality was tested by comparing the data stored on the SD card with the information on the terminal, the time stamp of the SD card file with the time of day, and the file name with the date. To check the accuracy of the recorded data, the information printed to the terminal was compared to the events which occurred in the time frame. These tests were repeated multiple times to ensure that the recording and saving of data was accurate. While this methodology is not ideal as it does not test long term stability, it does show the ability of the device to gather the required data, how reliably it can store the data, and how accurate the stored data is.

For the transmission of data, a custom data file was created to ensure a known set of expected information at both ends; this way, any issues in transmission could be easily identified. The custom file contains both lines of different lengths and data larger than 8 bytes, to ensure that the transmission system could handle all the potential expected complexities of the data storage structure. The latter of these may have posed an issue as the software serial write functionality requires 8-bit values, whether integers or characters, however this was handled well in software. To test the accuracy of the transmitted data, the three stages of stored, read, and sent were compared. The 'Serial Bluetooth Terminal' android app was used to read the sent data, the serial terminal was used to see the data read from the SD card by the processor, and the custom file was read in a standard spreadsheet processor (in this case, Excel). To ensure a fair test, this was repeated multiple times while varying the custom data file.

As the functionality of the website relies on a source of data from a Bluetooth device, it was determined that the best way to test it would be to use the functioning transmission of the PAM device. Using a custom data file as before, the receiving functionality of the website was checked to ensure the correct data was being read and used. To confirm the functionality of the user interface, each element of the interface was tested while trying to find cases in which it would break. This is not a reliable testing technique, however when bug finding for interfaces, the only suitable technique is repeated use which, given the current situation, must be simulated.

To guarantee an acceptable performance for multiple users, and to ensure the website can be accessed from anywhere, various devices were tested including android/IOS mobiles, laptops, and a desktop PC. While the user experience is impossible to test without participants, the transmission of data and display of the received data could be empirically tested through repetition and comparison of the displayed data and the custom data file the data is being sent from, similar to the testing methodology for the Bluetooth transmission.

Once each segment had been proven to function on their own, the two programmes on the PAM device were combined and used to send data to the website. To simulate daily use in a short space of time, the PAM device was carried around the house with the buttons being pressed occasionally and the heart rate sensor connected, before accessing the device via the website, and viewing the data. The PAM device was set to record data every second to allow for a large collection of data in a short space of time. The resulting graphs on the website were compared to the activities prior to ensure everything functioned as expected given the results of the previous tests.

5.2 Results

Using the testing methods described above, the individual segments worked flawlessly. The hardware of the device worked as expected, being able to read data from the SD card, the IMU, the RTC, and all external devices when required. However, the pedometer function is very basic, so there is not a precise count of how many steps were taken, but it does show the rough values, making it acceptable to use to see trends over the course of the day. In addition, the heart rate sensor was found to be inaccurate most likely due to the low-cost and low power of the sensor. This is acceptable as it still proves the capability of the developed system to read and record the data from external sensors, rather than trying to prove the reliability of a low-cost consumer sensor.

The tests of the three main segments proved that the device is capable of reading and storing data reliably, as well as transmitting data from the SD card via Bluetooth. The website was then able to receive and process that data to produce user-readable graphs. While these segments worked as expected throughout testing individually, when the storing and reading programmes were combined into the final programme, stability issues arose. The introduction of the watchdog timer helped to alleviate these issues so that the tests could be completed, which proved the core idea and functionality of the device worked. However, even with the watchdog timer, the instability of the system meant the user experience of the device was poor as it would often take multiple attempts to connect to the device, and data would be lost as the programme crashed before saving.

There were no specific tests or results for the 3D-printing case, as the only reliable testing method is long-term use by various people which was impossible given the situation. However, during general testing, there were no major issues with the casing.

6 Critical Evaluation

As stated previously, while the complete programme ran on the PAM device, it is unusable due to its instability, caused by the limited SRAM of the ATMegax8 series of microprocessors. As mentioned earlier, the library for the SD card requires at least 512 bytes of memory to act as a buffer when reading and writing which, when combined with the other required variables for that library, uses around 32% of the 2048 bytes of available memory for the one section. As this is a required functionality, there is no way to reduce this value using the current design. However, if the entire programme were written in a lower level language, such as C, the memory could have been allocated in a more usable way for this programme. The easiest solution to this problem, and possibly the only solution, is to upgrade the microprocessor to one with more

RAM, such as the ATMega644p-au (with 4kB of RAM) [36], or the ATMega1280 (with 8kB of RAM) [37].

When transmitting the data from the device to the website, due to the delays included to make processing simpler, the transmission takes longer than ideal. While this is not an issue for smaller files (with fewer than 200 data points), for larger files the delay can result is the transmission appearing to fail to the user. This does not impact functionality, however, it does diminish the user experience. By using a faster baud rate to communicate to the RN4870 chip and reducing/eliminating the delay between lines of data (which would require better processing when receiving the data), this transmission time could be reduced to an acceptable value. In addition, the web Bluetooth standard is acceptable for most users, except for those using iOS devices as mentioned previously as the standard is not supported by Apple. The only way to mitigate this would be to develop a dedicated application for Apple devices, however this was beyond the scope of this project.

For collecting the data, the defined data structure allows for easy identification, storage, and transmission of a variety of different data sources. Due to the 8-bit identification number and the onboard sources of data, there are only 251 available values for external sensors, which must be known to the device. This allows a wide range of devices to be made by third parties, if they are of a type that has been specified. While data with an unknown identification can be stored, the website is unable to display the information from them. This could limit the rate of development of new sensors for the system, however there are enough unique IDs for the foreseeable future, while the addition of unknown IDs is very simple.

The wires used to connect the external sensors to the main device are not comfortable to use, and often get in the way of daily life. Due to the length of the cable required, there is limited way to hide the excess besides under clothing, which is uncomfortable and makes the sensors harder to remove when needed. While inadequate for a final product, the cable allowed the prototype to be tested with external sensor, and shows it was capable of reading data from external sensors. While the chosen sensor was inaccurate, the device did read the data as expected.

Due to the construction of the casing and the ports used, the PAM device is not water-resistant, and so is not recommended for use where contact with water might occur. The use of seals and waterproof ports incorporated into the casing would increase the water-resistance but would have increased cost and development time.

The use of separate boards to handle the power and processing, as well as the size of the chosen battery, resulted in a bulky design which, while usable, is not comfortable to wear. In addition, due to the use of the SD breakout board, one of the mounting holes on the main PCB has been covered. As these mounting points are used to attach the front cover, this results in the front

panel not being flush with the casing. This could have been mitigated with separate mounting points for the front cover, however this would have increased the size of the device; without the separate breakout board, this would not have been an issue. The buttons on the casing can be difficult to clearly see, especially for those with sight issues, even with the indented lettering.

Overall, besides the main processor, the chips selected for each section were ideal, or more than required. The components selected for the SD section on board were the same as those used on the SD breakout board, so the PCB design was the problem for that section, rather than the components. The Bluetooth and IMU chips provide extra functionality which has not been implemented and so they could potentially be replaced with lower cost and lower powered alternatives to save space on the PCB and to save cost. While there are many sections that would not be used in a final product (such as the JST connection for external sensors, or the Powerboost 500C board) for use in the prototype to prove the concept, each component was a good choice.

Although the watchdog does work as a fail-safe to reduce the impact of the instability of the processor, it should not be required for the product to function without failing, further emphasising the point that the processor was an inadequate choice.

7 Conclusion

From the tests, this prototype has been shown to be a success. While the resulting device has problems that prevent it from being acceptable for production in its current form, the core idea and way of handling the data works as expected. Using ID numbers for each type of data and storing the information in a flexible structure allowing for many different data sources to be polled and stored, while dynamically adapting to the number of sensors connected. By using widely accepted communication standards, and by defining a custom data structure which is simple to adhere to, any sensor can be made to work with the platform as desired.

The use of a web-based application for accessing the data is as intuitive to use as a native application, while being easier to access as there are no necessary downloads. This relies on the user having an internet connection, which is not always available, but can be assumed to be the case for most of the time. Using a web-based application allows easy access from a plethora of devices if the browser supports the web Bluetooth standard, and the device accessing the website has support for Bluetooth 4.0 (BLE) or higher.

The hardware elements chosen for this project was perfect for their uses, besides the heart rate sensor and the microprocessor. For these components, while they proved the concept worked as expected they had limitations which meant the final product did not function to the extent of

its capabilities. The heart rate sensor was not accurate or reliable enough to be used to measure a user's heart rate, but the use of the ATTiny to adapt it to the platform worked perfectly. The microprocessor was able to power many of the required functions, however due the limited RAM, there are many instability issues which would need to be sorted before use in a final product.

Overall, this project has achieved its goal. While the final product produced may not be ideal for practical use, the platform developed achieves all the goals, with a clear path for future development. It is worth noting, there has been no inclusion of privacy or security as it would have delayed the development of the core aspects of the PAM device platform.

8 Future Work

To carry the development of this project forward, the main PCB would need to be redesigned. This would allow for the incorporation of a more powerful microprocessor (testing would need to be done to find the ideal alternative as both performance and cost would need to be taken into account), for the inclusion of the pull-down resistors for the buttons, which were not implemented on this iteration and for the SD card segment to be fixed, such that it works directly on the PCB without requiring the breakout board.

With more research, the external power board could also be incorporated into the main PCB, which would reduce the required space inside the PAM device's casing, and allow greater control of the voltages from the battery (removing the need to convert from 3.7V, to 5V, to 3.3V for some of the chips). Furthermore, when producing the final designs, the overall dimensions of the PCB could be reduced by using pick and place machines to assemble the boards, as the components would not need to be soldered by hand and so would not require as much clearance.

To help improve the user experience, the wireless transmission would need to be sped up as explained previously. Using the current hardware, the theoretical maximum data rate would be approximately 1Mbps on average (according to the Bluetooth 4.0 BLE specification [38]), which would be fast enough to transfer multiple days of information in less than one tenth of a second, as the average size of one day of data is between 5KB and 10KB (taking between 0.038s and 0.076s respectively). While this theoretical maximum is unlikely to be reached, the increase in speed would allow for more data to be collected without impacting the user experience. The faster data rates would require more complicated data processing when receiving, as the web Bluetooth standard receives the information as an ArrayBuffer with a maximum length of 155 bytes. Currently each line of data is received as a separate ArrayBuffer, however with faster transfer speeds multiple lines of data would be receive in one ArrayBuffer and would need processing to separate them.

In addition, removing the wires from the external sensors would further improve the user experience. While this could be done using the same Bluetooth chip on the main board, it may be worth including a separate smaller processor, such as the ATTiny used on the external sensor, in addition to a separate dedicated wireless chip, for connecting to wireless sensors. This would allow the connections to be handled separately to the data processing, in a similar way to the RN4870 handling the connection to the website, while the main processor would access the data in the same way it currently does, requesting it when required. For the wireless communication, either the Bluetooth 4.0 or 2.4GHz standard could be used. Bluetooth would require less power to run, but 2.4GHz would provide a stronger connection, and allow easy connection to multiple sensors.

For visualisation of the data, while the web-app is ideal for easy development and accessibility on many devices, a dedicated application should be developed for iOS/iPadOS/macOS users, as they are unlikely to be able to use the web Bluetooth functionality. By further developing the web-app into a progressive web app (PWA), the website could be improved to be equal to, if not better than, a dedicated application for the rest of the users. To do this, service workers will need to be added to the website to cache the necessary information and allow the website to be used offline, and even added to homepage making the site feel like a downloaded application. When the data is being analysed by Doctors, a dedicated desktop application should be developed to read the data directly from the SD card, rather than requiring wireless transmission. This would allow for more controls over the data shown, such as seeing more than one day's worth of information.

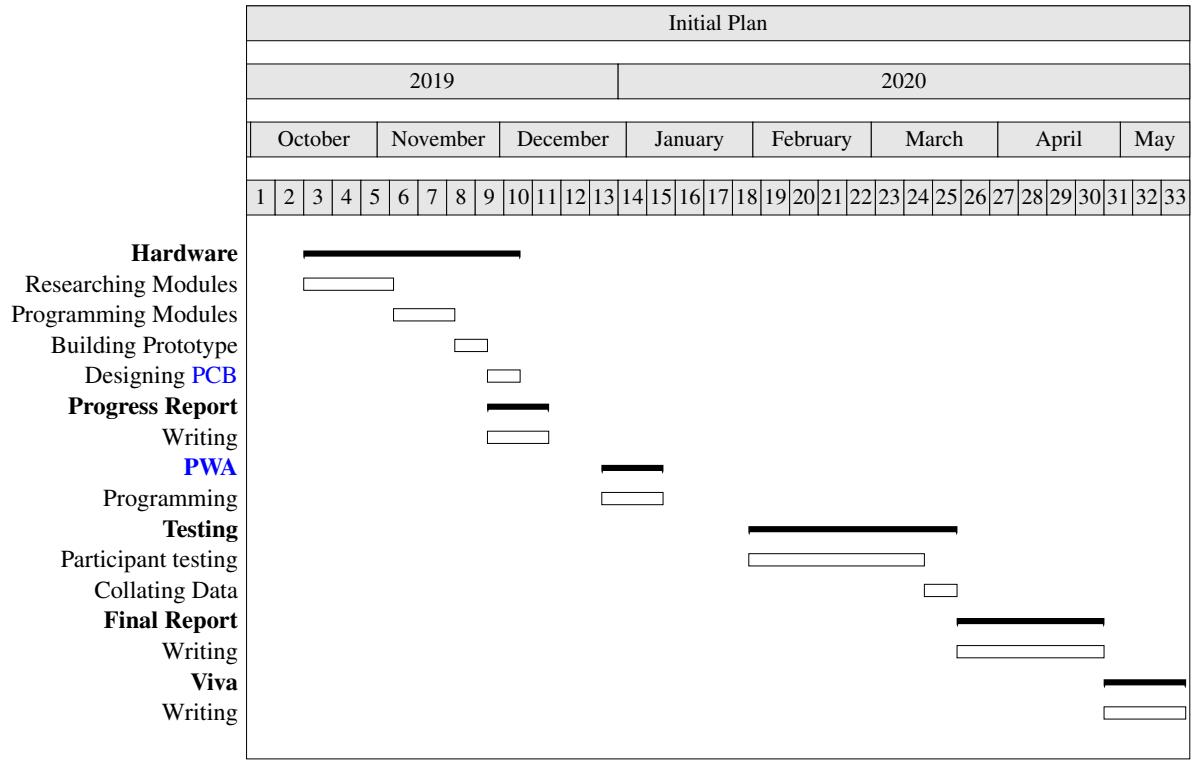
As mentioned previously, the PAM device currently has no security systems put in place, on the device or on the website. If the product were to be used by the general populous, additional protections must be put in place to ensure the data cannot be accessed by unauthorised persons. These may include encrypting the data during storage and transmission, requiring a log in to access the data on the website, and even a pairing button to ensure only the owner of the device can connect to it. In addition, the RTC is not currently updated once the PAM device has been programmed, this functionality was forgone to reduce development time. The original plan included a time update every time the PAM device was connected to an application to read the data, this should be implemented to ensure the device does not lose time for long periods of use.

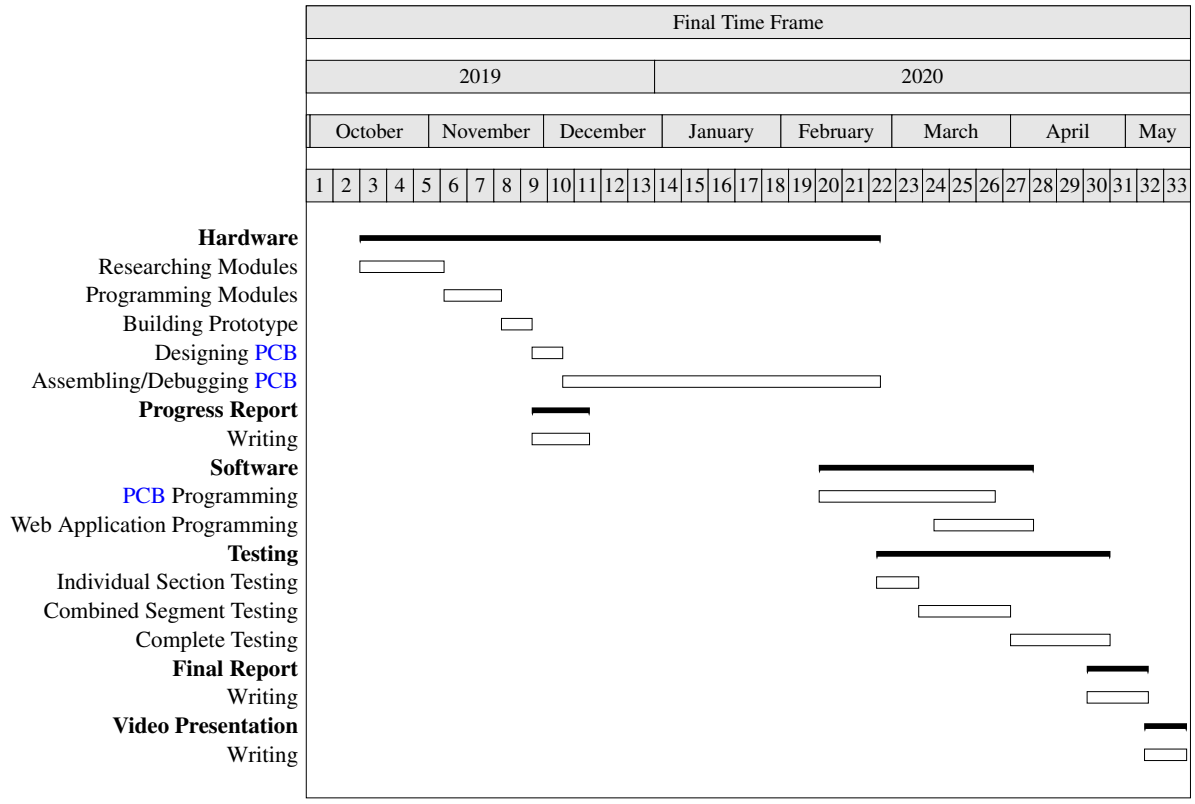
While the current casing is functional, and works well for the prototype, there are many improvements to be made. The size of the case will reduce with the reduction in size of the PCB, which will help increase comfort while wearing. In addition, filleted edges would remove the sharp corners and remove a source of discomfort. To make the product more usable day to day, the casing should be made water-resistant, to ensure it can be used outside in all conditions.

Currently, the buttons on the front panel are difficult to see and use, for future iterations this would need to be improved. This could be done with the use of a separate colour for the lettering, rather than indentation; by increasing the size of the press-able area, which would also help with those who struggle with smaller controls; or by raising the buttons off the surface, rather than keeping them flush. There should be further research done into alternatives to 3D printing to manufacture the casing as it is not currently suitable for final products.

Appendix A

Planning



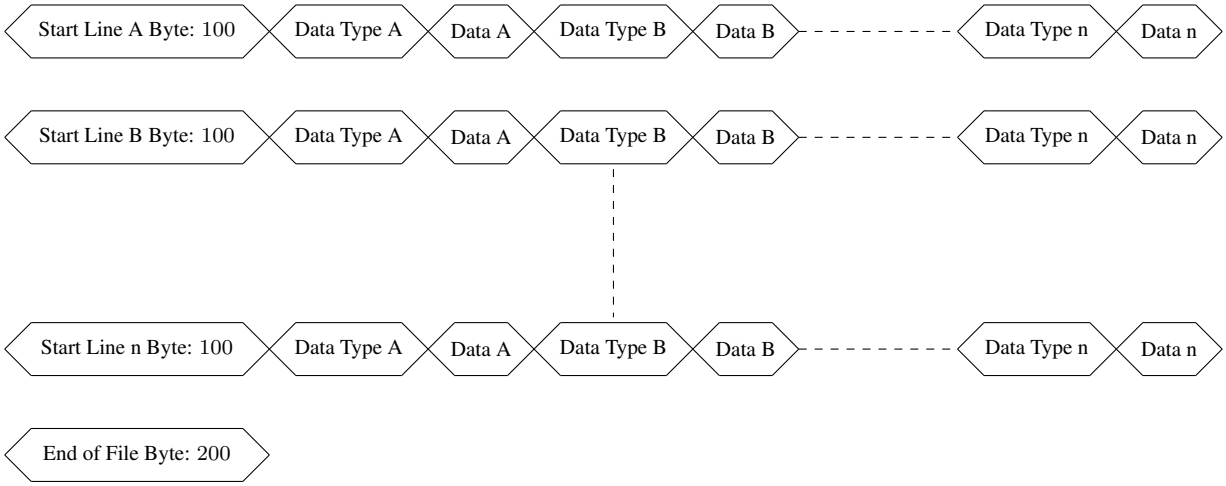


Appendix B

Data Storage Structure

Note: Data Type A < Data Type B < ... < Data Type n

BLE Data Transmission Structure



Appendix C

Glossary

API Application Programming Interface.

BLE Bluetooth Low Energy.

CDN Content Delivery Network.

CSV Comma-Separated Values.

I2C Inter-Integrated Circuit.

IC Integrated Circuit.

IMU Inertial Measurement Unit.

LDO Low-Dropout.

OWI One Wire Interface.

PAM Personal at-Home Monitoring.

PCB Printed Circuit Board.

PWA Progressive Web Application.

RAM Random Access Memory.

RTC Real Time Clock.

SCL Serial Clock.

SD Secure Digital.

SDA Serial Data.

SPI Serial Peripheral Interface.

UART Universal Asynchronous Receiver/Transmitter.

USB Universal Serial Bus.

Appendix D

Original Brief

Personal At-home Monitoring (PAM) to assist with patient diagnosis and treatment

John Hawk
jh16g17@soton.ac.uk
Project Supervisor: Dr Neil White

Problem

When doctors want to diagnose or treat a patient, they need to rely on regular check-ups and inaccurate recording from patients. For the patients, they have to remember to keep a record of their health between check-ups. This leads to an unsatisfactory situation where patients may not be getting diagnosed as quickly or treated as well as they would otherwise be if the doctors had access to more accurate information about their day to day lives.

Project Goals

This project will aim to develop a standardised platform to allow for more detailed and accurate monitoring of patients. It should provide a way to regularly collect unbiased data throughout a patients day to day life, and present the data in a clear format for the doctor to use. The data collected should be saved with the time at which it was collected, so that the doctor and patient can easily track the information throughout the use of the device. The device should also have a simple user interface to allow the patient to easily track their information, and potentially input extra data as required.

This project should serve as an open-source platform to allow third-party sensors to be easily integrated through a standard interface and data format and to allow easy modification and improvements of the design.

Scope of the Project

I will design a data-logging device that will serve as a standard platform for third party sensors to connect to, and which will present the collected data in a simple to understand format.

Main Scope

To achieve this goal, I will develop a self contained device, designed to communicate via a standard protocol (I.E. SPI or I2C) to external sensors, ideally these sensors would be modified open source products so as to reduce development times and costs, and to prove the platform can be made to work with anything. This device would then communicate to a basic progressive web app via BlueTooth 4.0, to allow for easy cross-platform integration, so that the patient or the doctor can easily look at, and keep track of, the collected data in a way that is easy to understand and access.

The device itself will contain a microprocessor, most likely an AVR-based processor for easy development within the Arduino ecosystem, collecting data from the external sensors and saving it to an internal SD card for permanent storage of data. Inside the device, there would also be a power management system capable of recharging the battery and supplying power to all components of the device. In addition, the device will have a built-in RTC (real-time clock) to keep track of the exact time, even if the microprocessor loses power, and an IMU (inertial measurement unit) to allow for simple movement tracking such as step counting.

To prove that the device is capable of working with existing sensors, such as the Gravity heart rate monitor, a sensor currently available to buy will be modified such that it can work seamlessly with the platform developed.

Extended Scope

As a potential further target, this project may also include working with the NHS FIFE database system to allow the device to communicate directly with doctors, without requiring an appointment.

In addition, to assist doctors with analysing the data, a dedicated local programme could be written to allow the doctors to more easily manipulate the recorded values without relying on an internet connection.

Bibliography

[1] NXP Semiconductors. *UM10204 I2C-bus specification and user manual Rev. 6-4 April 2014 User manual Document information Info Content*. Tech. rep. URL: https://www.nxp.com/docs/en/user-guide/UM10204.pdf (visited on 09/12/2019).

[2] *KeyStone Architecture Serial Peripheral Interface (SPI) User Guide Contents SPRUGP2A-March 2012*. Tech. rep. 2010. URL: https://www.ti.com/lit/ug/sprugp2a/sprugp2a.pdf?ts=1588277415454 (visited on 09/12/2019).

[3] *KeyStone Architecture Universal Asynchronous Receiver/Transmitter (UART) User Guide*. Tech. rep. 2010. URL: https://www.ti.com/lit/ug/sprugp1/sprugp1.pdf?ts=1588277452715 (visited on 09/12/2019).

[4] *1-Wire Enumeration Application Report*. Tech. rep. 2013. URL: http://www.ti.com/lit/an/spma057c/spma057c.pdf (visited on 09/12/2019).

[5] Web Bluetooth Community Group. *Web Bluetooth*. 2017. URL: https://webbluetoothcg.github.io/web-bluetooth/ (visited on 29/04/2020).

[6] Ganjar Alfian et al. "A Personalized Healthcare Monitoring System for Diabetic Patients by Utilizing BLE-Based Sensors and Real-Time Data Processing". In: *Sensors* 18.7 (2018), p. 2183. ISSN: 1424-8220. DOI: 10.3390/s18072183. URL: http://www.mdpi.com/1424-8220/18/7/2183.

[7] Jorge Gómez, Byron Oviedo and Emilio Zhuma. "Patient Monitoring System Based on Internet of Things". In: *Procedia Computer Science*. Vol. 83. Elsevier B.V., 2016, pp. 90–97. DOI: 10.1016/j.procs.2016.04.103.

[8] Tia Gao et al. "Vital signs monitoring and patient tracking over a wireless network". In: *Johns Hopkins APL Technical Digest (Applied Physics Laboratory)* 27.1 (2006), pp. 66–73. ISSN: 02705214. DOI: 10.1109/iembs.2005.1616352.

[9] Xavier Righetti and Daniel Thalmann. "Proposition of a modular I2C-based wearable architecture". In: *Proceedings of the Mediterranean Electrotechnical Conference - MELECON*. 2010, pp. 802–805. ISBN: 9781424457953. DOI: 10.1109/MELCON.2010.5475965.

[10] Péter Várady, Zoltán Benyó and Balázs Benyó. “An open architecture patient monitoring system using standard technologies”. In: *IEEE Transactions on Information Technology in Biomedicine* 6.1 (2002), pp. 95–98. ISSN: 10897771. DOI: 10.1109/4233.992168.

[11] Md Sajjad Rahaman et al. “VSIB: A sensor bus architecture for smart-sensor network”. In: *2009 WRI World Congress on Computer Science and Information Engineering, CSIE 2009*. Vol. 3. 2009, pp. 436–439. ISBN: 9780769535074. DOI: 10.1109/CSIE.2009.1014.

[12] D. Fortunato and J. Bernardino. “Progressive web apps: An alternative to the native mobile Apps”. In: *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*. 2018, pp. 1–6.

[13] Google. *Google Translate*. URL: https://translate.google.com/?tr=f&hl=en#view=home&op=docs&sl=auto&tl=en (visited on 07/05/2020).

[14] *ATmega328P 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash DATASHEET*. Tech. rep. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (visited on 29/04/2020).

[15] Autodesk. *Autodesk Eagle*. URL: http://www.autodesk.co.uk/products/eagle/overview (visited on 29/04/2020).

[16] *JLCPCB manufacturer*. URL: http://www.jlcpcb.com (visited on 29/04/2020).

[17] Maxim. *DS1302 RTC*. 2020. URL: https://datasheets.maximintegrated.com/en/ds/DS1302.pdf.

[18] *SD Standard Overview - SD Association*. URL: https://www.sdcard.org/developers/overview/ (visited on 29/04/2020).

[19] *MPU-6000 and MPU-6050 Product Specification Revision 3.4 MPU-6000/MPU-6050 Product Specification*. Tech. rep. 2013. URL: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf (visited on 09/12/2019).

[20] Microchip. *RN4870 Datasheet*. Tech. rep. 2017. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/50002466B.pdf.

[21] *PowerBoost 500 Charger - Rechargeable 5V Lipo USB Boost @ 500mA+ ID: 1944 - $14.95 : Adafruit Industries, Unique & fun DIY electronics and kits*. URL: https://www.adafruit.com/product/1944{\#}technical-details (visited on 29/04/2020).

[22] Advanced Monolithic Systems. *AMS1117 LDO Datasheet*. 2020. URL: http://www.advanced-monolithic.com/pdf/ds1117.pdf (visited on 29/04/2020).

[23] *Heartbeats in Your Project, Lickety-Split World Famous Electronics llc*. URL: https://pulsesensor.com/ (visited on 29/04/2020).

[24] *ATtiny202/402 AVR® Microcontroller with Core Independent Peripherals and picoPower® Technology*. Tech. rep. 2018. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/ATtiny202-402-AVR-MCU-with-Core-Independent-Peripherals{_}and-picoPower-40001969A.pdf.

[25] Microsoft. *Naming Files, Paths, and Namespaces - Win32 apps — Microsoft Docs*. URL: https://docs.microsoft.com/en-us/windows/win32/fileio/naming-a-file?redirectedfrom=MSDN (visited on 29/04/2020).

[26] *Arduino - Libraries*. URL: https://www.arduino.cc/en/reference/libraries (visited on 29/04/2020).

[27] *GitHub - tockn/MPU6050_tockn: Arduino library for easy communicating with MPU6050*. URL: https://github.com/tockn/MPU6050{_}tockn (visited on 05/12/2019).

[28] *GitHub - Makuna/Rtc: Arduino Library for RTC, Ds1302, Ds1307, Ds3231, & Ds3234 with deep support*. URL: https://github.com/Makuna/Rtc (visited on 05/12/2019).

[29] *Bootstrap · The most popular HTML, CSS, and JS library in the world*. URL: https://getbootstrap.com/ (visited on 29/04/2020).

[30] Google. *Charts — Google Developers*. URL: https://developers.google.com/chart (visited on 30/04/2020).

[31] Google. *Material Design Icons*. URL: https://material.io/resources/icons/?style=baseline (visited on 30/04/2020).

[32] *GitHub Pages — Websites for you and your projects, hosted directly from your GitHub repository. Just edit, push, and your changes are live*. URL: https://pages.github.com/ (visited on 30/04/2020).

[33] *GitHub Desktop — Simple collaboration from your desktop*. URL: https://desktop.github.com/ (visited on 30/04/2020).

[34] *Amazon Web Services (AWS) - Cloud Computing Services*. URL: https://aws.amazon.com/ (visited on 30/04/2020).

[35] *Inside Azure Web Hosting Plans*. URL: https://azure.microsoft.com/en-gb/resources/videos/inside-azure-web-hosting-plans/ (visited on 30/04/2020).

[36] Microchip. *ATMega644p Datasheet*. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega164P-324P-644P-Data-Sheet-40002071A.pdf (visited on 30/04/2020).

[37] Microchip. *ATMega1280 Datasheet*. URL: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561{_}datasheet.pdf (visited on 30/04/2020).

[38] *Core Specifications — Bluetooth® Technology Website*. URL: https://www.bluetooth.com/specifications/bluetooth-core-specification/ (visited on 30/04/2020).