Sensor Design Project Report

Justas Brazauskas

December 17, 2021

1 Introduction

In the domain of healthcare sensors, the majority of the market is dominated by expensive high-end products. With the need for pulse and temperature sensors further exacerbated by the pandemic, the niche for cheap, open-source non-medical grade products has grown in the last several years. To tackle this, we propose a cheap, DIY-based healthcare sensor prototype for pulse and temperature readings.

2 Background

With the ongoing pandemic, the need to swiftly monitor vital signs in patients is more important than ever. Furthermore, in the developing world, particularly sub-Saharan Africa, there is only 1 doctor per 10'000 inhabitants, compared with 28 in the UK. Hence, there is a dire need not only for the growth of the healthcare sector in general but also for its efficiency by using technology [1]. There exist several options to elevate the burden of the healthcare sector in the developing world, for example through the use of telemedicine or faster in-person appointments. In both of these cases, having faster, more efficient ways of reading and monitoring vital healthcare data would be vastly beneficial. Rather than depending on old, inaccurate and sometimes hazardous medical devices such as mercury thermometers or measuring pulse manually, digital sensors could provide cheaper and safer alternatives.

For this reason, we chose to create a cheap, DIY prototype of a smart healthcare sensor for key vital signs monitoring. According to the Johns Hopkins University[1], the three vital body signs are body temperature, pulse rate and respiration rate. However, due to time and monetary constraints we chose to only focus on temperature and pulse rate.

The shift towards such devices can already be seen worldwide - the electronic temperature sensor market size is projected to grow from $ 6.3 billion in 2020 to $ 8.8 billion by 2027, while the Optical Pulse Sensor market is projected to grow by $ 1.36 billion till 2024 [2].

3 Literature Review

The increased interest in healthcare sensors can not only be seen in the increasing market growth for such products but also in research. The following section looks at similar student-led DIY sensor projects, as well as more advanced research papers on healthcare sensors.

Overall there has been a relatively wide range of similar students DIY-based approaches to healthcare sensors. For example, Yassin et al propose an Arduino-based LM35 temperature and heart rate monitoring system with wireless Bluetooth data transfer capabilities [3]. However, the project does not make use of signal amplifiers which can have an adverse effect on the precision and accuracy of the system. Furthermore, the project has no Internet connectivity or any real-time data visualisation examples that would be beneficial in providing direct feedback to the user.

Similarly, Devara et al propose a wearable system with pulse and temperature sensors, packed together with an integrated OLED display and wireless phone connectivity [4]. Additionally, the system makes use of filter and amplifier circuits that help provide more reliable data.

Lastly, Sollu et al propose a mixed Arduino and Raspberry Pi setup, using temperature and pulse sensors[5]. The system does not make use of operational amplifiers or filtering, hence hindering its

[1]Body vital signs: www.hopkinsmedicine.org/health/conditions-and-diseases/vital-signs-body-temperature-pulse-rate-respiration-rate-blood-pressure

real-world usage. However, the proposed system has an interactive dashboard to display its readings that could make it especially useful for quick data analysis.

In general, the majority of the student-led research projects include some sort of wireless connectivity to their devices to transfer the data from the Arduino to an external device. However, only half of the projects made use of more advanced usage of electronics circuits via operational amplifiers.

On a more professional level, research papers tend to focus on smaller, better integrated wearable devices with more reliable data outputs. Marathe et al present a mobile patient-monitoring system by using four sensors in one unified package [6]. The proposed system collects the data from the connected sensors - pressure sensor, pulse oximeter, temperature sensor and ECG module and then transmits the data over WiFi.

Finally, Wu et al presents a a ring-type wearable sensor with temperature and pulse sensing capabilities with RFID data transmission [7]. The device features a novel design features and an accompanying application that provides user with the data. Overall the more advanced research papers provide more in-depth and thorough solutions in a smaller form factor.

4 Setup

Our healthcare system diagram is comprised of three main components: the sensor unit, Raspberry Pi and the Cloud. These components are illustrated in the diagram below, where the square rectangle represents the sensor unit, also called UNOsense, the triangle represents the Raspberry Pi which acts as a local sensor node for data-preprocessing and the rounded edge rectangle represents the Cloud.

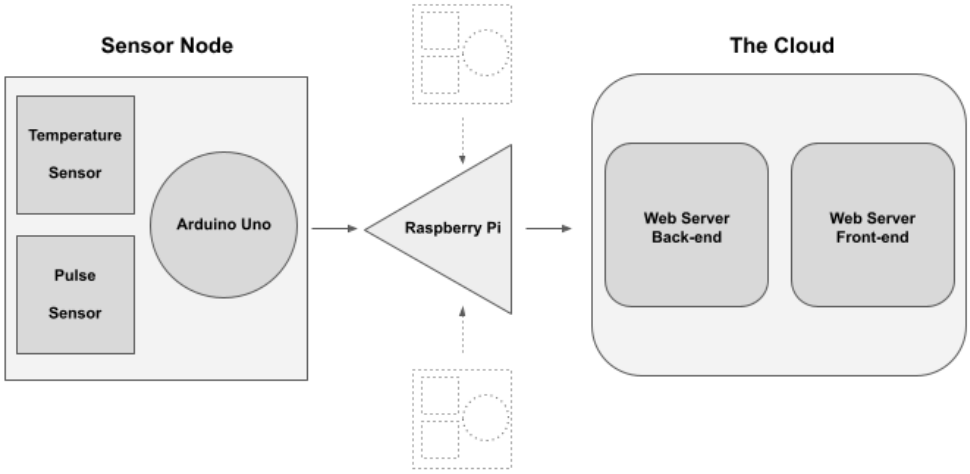

Figure 1: System diagram.

4.1 Sensor Node

The sensor node, or UNOsense, is comprised of two sensing units - a temperature sensor and a pulse sensor (photodiode). The two are connected to the Arduino using I/O pins and continuously collect the data after being powered on. The temperature sensor and pulse sensor signals go through op-amps before sending their data to the Arduino microcontroller. This provides more accurate and less noisy results.

In order to provide some direct feedback to the user, we have also installed an OLED display that shows the sensors readings in real-time. The I2C OLED display provided clear image quality for the use case targeted with adequate contrast and high brightness. Furthermore, it uses lower power consumption than comparable LCD displays.

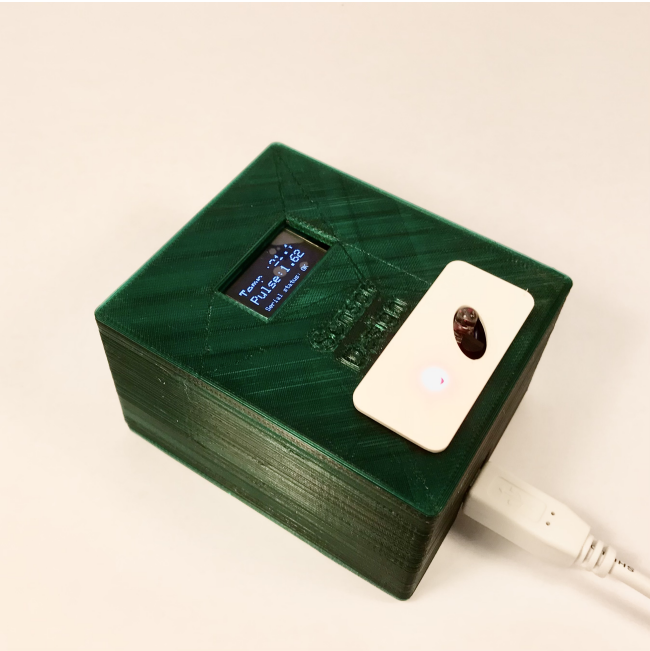

Figure 2: Sensor Node.

4.1.1 Temperature Sensor

The selected temperature sensor was LM35. The sensor features linear +10-mV/ °C scale factor and 0.5°C ensured accuracy (at 25°C). Finally, it provides a rough temperature measuring range of -55 to 150°C, however, the range we were mostly interested in was 35°C to 39°C.

In order to properly make use of the sensor we first had to calibrate it. The Arduino ADC is 10 bits - 1024 steps, while it's voltage range is 0-5V. Since each ADC step is around 5mV per step, or 0.5°C, this did not provide sufficient resolution prior to calibration. The calibration was done by setting two points to determine both offset and slope of the linear relation using the alcohol thermometer as a reference.

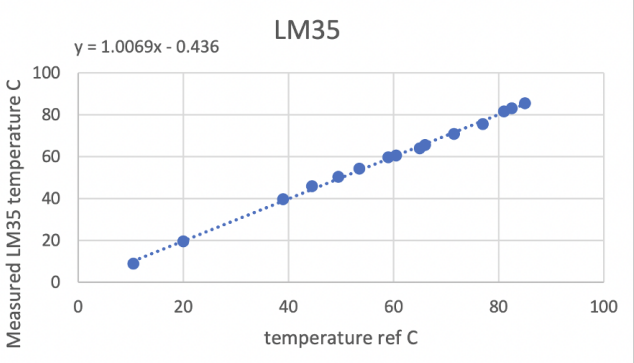

Figure 3: Calibrated LM35 readings.

The non-inverting amplifier gain had to be recalculated from the starting value (11) since the thermometer would not show very high temperatures. In other words, we were not using the full range of possible sensor reading values. After some calculations using the equations presented in the slide, the optimal gain was found to be 4.11. The previously used resistors were replaced to be 47k and 15k ohms. After the calibration, it became apparent that amplified LM35 had less variation in reading values (was more consistent) and generally matched the physical thermometer values better. With the

amplifier the resolution became 0.5/Gain, where Gain was calculated as the following:

$$Gain = 1 + (R2/R1) \tag{1}$$

and R2 and R1 were the resistors in the circuit. The circuit was based on the diagram provided below.

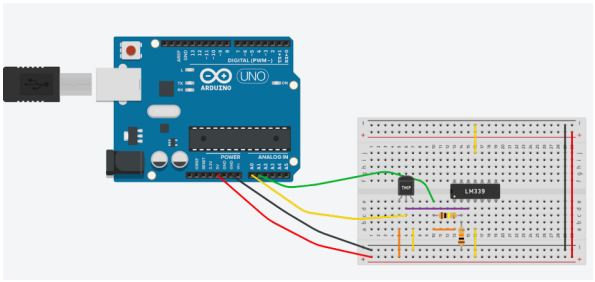

Figure 4: LM35 connected to Arduino Uno, the op amp is MCP6004.

4.1.2 Pulse Sensor

Similarly to body temperature, pulse rate is one of the main vital signs routinely monitored by medical professionals. In our implementation of the pulse sensor, we used a combination of an LED and a photodiode, otherwise known as photoplethysmography. It is a non-invasive pulse monitoring technique when heart rate is calculated using light penetrating the skin.

The LED was used to illuminate the finger, while the photodiode was used to detect the pulsating light that is transmitted through the finger. The low-pass filter with low cut-off frequencies filtered out the pulsating signal and the trans-impedance amplifier then subtracted the filtered from the original signal and converted the small current into a voltage that we finally measured with the Arduino ADC. The circuit is based on the diagram provided below.

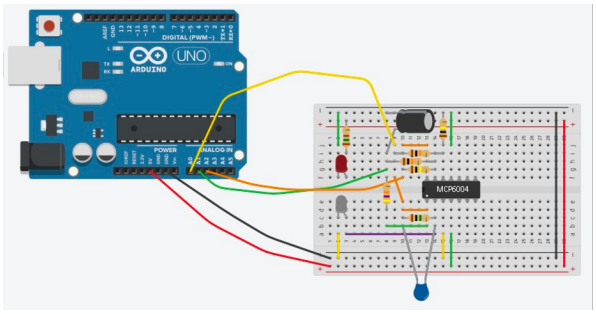

Figure 5: Pulse sensor circuit with a photodiode and LED.

Overall it was found that the positioning of the finger on the photodiode was very important, hence when designing the case this was taken into consideration.

4.2 Raspberry Pi

The Raspberry Pi in our system acts as a central broker between the sensor node and the Cloud. We chose such a sensor node configuration because first of all, the sensor unit lacked any networking capabilities and Raspberry Pi had WiFi, which allowed us to send data to the Cloud. Secondly, having an intermediary broker between the sensor and the Cloud meant we could form a star network configuration (seen in Figure 6) where we could deploy several sensor nodes and have all of them send the data back to the central broker. There, the collected data could be backed up, analysed and further transmitted to the Cloud. This not only allowed us to have cheaper sensor units but also created more flexibility when deploying sensors in the real world. E.g. in a deployment, one ward in a hospital could have a central Raspberry Pi, which would then transmit the data from all of the UNOsense nodes in that room.

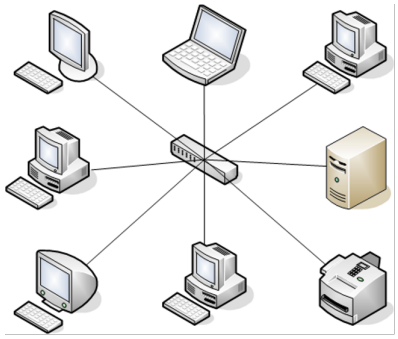

Figure 6: Star network configuration.

4.3 The Cloud

After the Raspberry Pi connects to the Internet, it is used as an intermediary between the Arduino and the Cloud. When referring to the Cloud, we mean both the back-end and the front-end systems.

With the back-end working, we were able to receive HTTP requests from the Pi and keep a historical log of the data. For this, we used a PHP program that accepts HTTP requests and writes the data to the server. Continuing with the low-cost, DIY theme of the project we achieved this by using a free web hosting service.

For the front-end website, we wanted a simple and straightforward way of viewing sensed pulse and temperature data. Most of the user interface was created using HTML and JavaScript. In order to plot the incoming and collected data on real-time interactive graphs made using CanvasJS.

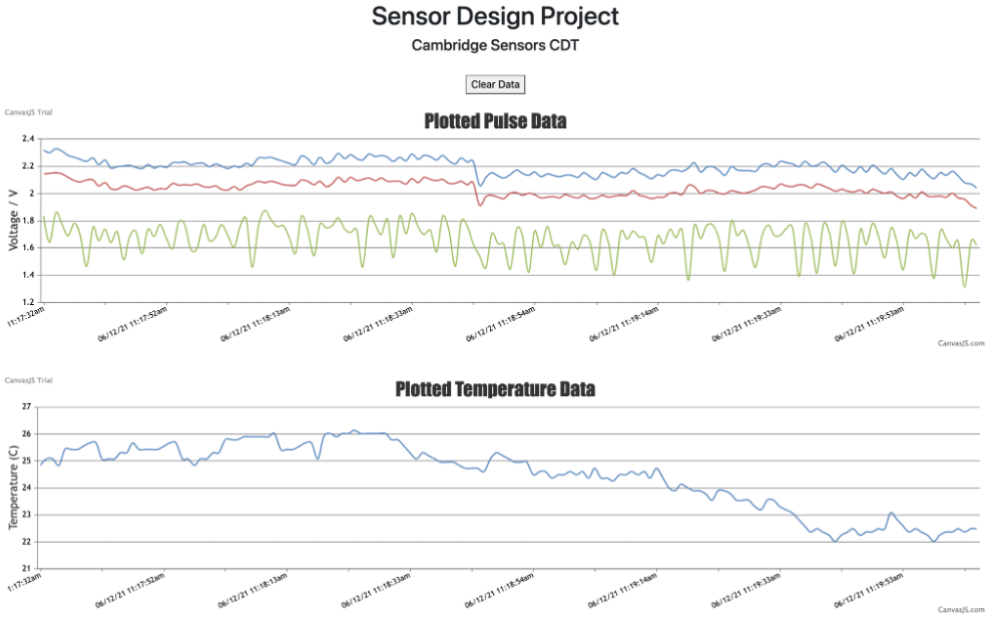

Figure 7: Sample graphs from the website.

While having the sensor connected to the Cloud is an essential feature of the project, the UNOsense sensor node is also capable of functioning just by itself with the Raspberry Pi and the Cloud, with the readings appearing on the installed OLED display.

5 Experimental Data

The pulse analysis was inspired by the pre-existing code on Moodle. After analysing the collected data, we noticed that finding the pulse rate was effectively finding time intervals between the peaks

in the waveform. While there were several different ways to find the pulse rate, i.e. using Fourier transform, we found that the algorithm that worked best for us was peak detection. It was chosen both as combination of hardware limitations and the ease of use that worked best with the data set we had collected.

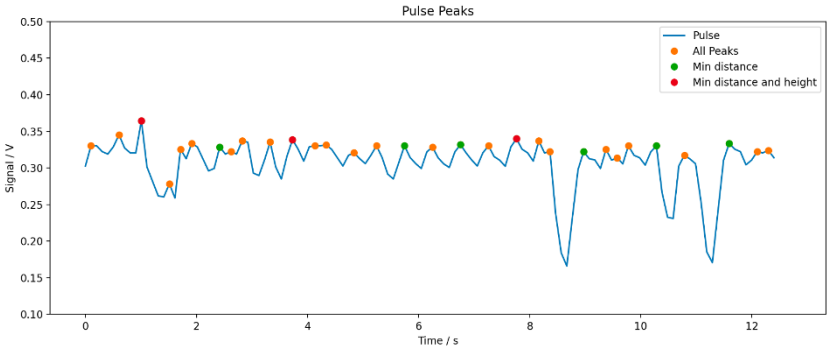

Figure 8: The calculated pulse was around 80BPM which is very acceptable for a resting heart rate.

As seen in Figure 5, we managed to get every peak, which meant that the pulse rate could be calculated effectively, only when excluding some of the outliers. For example, some of the inconsistencies in readings are very clear around 1 - 3 second mark as well as around 9 - 10 second mark. This could either happen due to finger being moved slightly or due to an algorithmic error.

Finally, the outlier-free results were consistently around 80-90BPM, which is well within the acceptable pulse rate range. However, further investigations might be useful to see how the pulse rate varies within different conditions, e.g. after exercising.

6 Discussion

The following sections discuss the performance of the entire system and some of the findings in more detail. This includes both positive and negative aspects of the created prototype.

Figure 9: The entire system in use.

6.1 Temperature

We found that the temperature mostly worked within acceptable limits, however, some future improvements were needed. The aforementioned issues had to do with Arduino I/O limitations, sensor placement and sensitivity.

First of all, due to our decision to install the OLED display on the Arduino board, we ran out of available I/O pins. This meant that we had to choose between showing raw and amplified LM35 readings. Due to the reasons described in the section on the temperature sensor, we decided to exclude the unamplified readings from the project. While we could have opted for a bigger board, we decided that the raw readings were not essential for this part of the project.

Secondly, after installing the temperature sensor on the case, we found that the placement of the temperature sensor next to the pulse sensor did not work as well as expected. We found that the readings took longer to measure since the contact area with the sensor was reduced. Additionally, since the readings were only measuring the fingers temperature it wasn't always realistic that the temperature would fully match that of the body in some conditions, i.e., cold.

One potential workaround would be to change the location of the sensor, for example, place it closer to the palm. Furthermore, the LM35 sensor could be changed with a more expensive alternative, i.e. BME280[2], however, that might come with a risk of undermining the cheap DIY aspect of the entire project.

6.2 Pulse

We found that the pulse sensor worked reasonably well. The separately 3D printed cradle for the finger provided sufficiently good conditions for pulse monitoring and the sensor was able to function properly.

Similarly to the temperature sensor, any future iterations of the project could benefit from increased accuracy by using pulse sensors with integrated electronics[3] to minimise the size of the sensor unit. Additionally, the screen could be updated with a heart animation that would mimic the heartbeat of the user.

Overall, we found that with the given sensor setup, the pulse sensor performed better than the temperature sensor, given its operational setting.

6.3 The Case

The design of the sensor unit was created with functionality in mind first. One of the primary goals was to hide all of the electronics components so that the user would not accidentally destroy the circuit by pulling out or shorting any wires. The rectangular shape of the box mimics the general dimensions of the breadboard and provided ample volume on the inside to house all of the electronics. Rounded edges reduce the discomfort of using the device and the screen is located in the most visible location.

The case is comprised of three parts – the bottom part of the case on which the Arduino is placed, the rectangular box that houses all of the electronics and the finger cradle with holes for the pulse and temperature sensors. The color of the finger cradle was chosen to be different from the rest of the case to subtly guide the user where to place the finger for sensing. The case is closed with four screws on the bottom.

Given that this is just a prototype, the case is rather large and clunky, however, since it is designed to be stationary it doesn't cause any inconvenience to the end-user.

The obvious next step would be to transition from using a breadboard to a printed PCB design and potentially a different form factor. With the decreased size and improved ergonomics, this could even become a wearable device that the user would be much more comfortable with.

6.4 Networking

One of the major challenges for this project, apart from the successful use of both sensors, was online connectivity. Since the Arduino Uno does not have any inherent WiFi functionally and that there

[2]https://www.adafruit.com/product/2652
[3]https://pulsesensor.com/

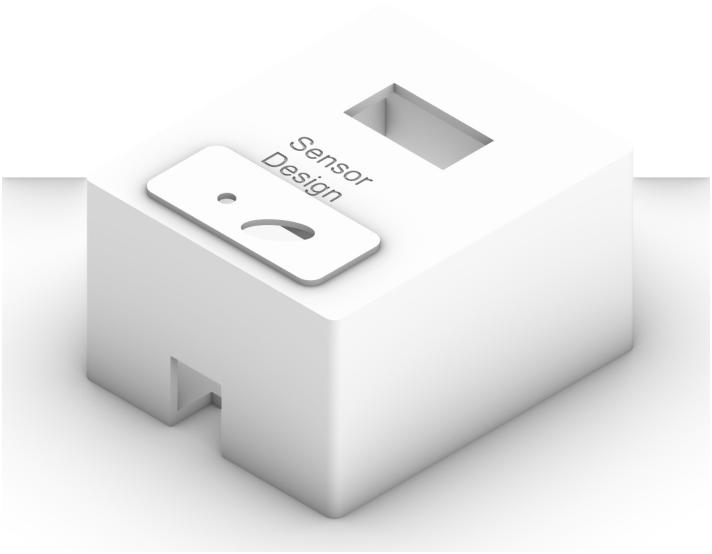

Figure 10: Case render.

were a limited number of I/O pins available, we decided it was best to connect the Arduino to the Raspberry Pi using a cable.

The cable transmits the data using Serial protocol, which is useful if we wanted to move away from using the capable and transition to using Bluetooth. This would be achieved easily since the TX/RX pins on the Arduino are unused and BLE transceivers use Serial from data transmission.

Furthermore, this would simplify the star configuration of the UNOsense sensor as the number of sensor units connected to the Raspberry Pi would not be limited by the number of available USB ports.

6.5 Assembly

With all of the components in place, the breadboard with the electronics components was carefully placed inside the case, which itself was snugly secured with four screws on the bottom. We found that with this configuration we had to be very careful not to accidentally destroy the circuit, hence some of the wires were secured with tape to avoid any damage during the assembly procedure.

In its current form, the assembled UNOsense node is dependent on being plugged into the Raspberry Pi for power and networking features. While it is possible to use the device without the Pi, it would be inconvenient to have it be always plugged in. Therefore, the project could be further improved by adding a BLE transceiver, batteries, and a battery charger inside the case.

7 Conclusion

We have designed a cheap, Arduino-powered sensor node called UNOsense with temperature and pulse sensing capabilities. The UNOsense node can be used by itself or connected to a Raspberry Pi with additional nodes to form a sensor cluster.

The used temperature and photodiode-based pulse sensors had to be calibrated and improved using operational amplifiers. In order to overcome Arduino limitations and provide networking capabilities, UNOsense was connected to Raspberry Pi to transmit the data to the Cloud for visualisation and analysis. Finally, the sensor node was placed in a proprietary 3D printed case with an OLED display attached to provide direct feedback to the user.

Both sensors provided adequate results, however, future iterations of the project could benefit from a smaller form factor and increased temperature sensor accuracy,

References

[1] Chris Campbell Chelsea Bruce-Lockhart. In charts: healthcare technology in low-income countries. *Financial Times*.

[2] BusinessWire. Global temperature sensors market worth $8b by 2027. *BusinessWire*.

[3] FM Yassin, NA Sani, and SN Chin. Analysis of heart rate and body temperature from the wireless monitoring system using arduino. In *Journal of Physics: Conference Series*, volume 1358, page 012041. IOP Publishing, 2019.

[4] Kresna Devara, Savira Ramadhanty, and Tomy Abuzairi. Design of wearable health monitoring device. In *AIP Conference Proceedings*, volume 1933, page 040022. AIP Publishing LLC, 2018.

[5] Tan Suryani Sollu, Muhammad Bachtiar, Benyamin Bontong, et al. Monitoring system heartbeat and body temperature using raspberry pi. In *E3S Web of Conferences*, volume 73, page 12003. EDP Sciences, 2018.

[6] Sachi Marathe, Dilkas Zeeshan, Tanya Thomas, and S Vidhya. A wireless patient monitoring system using integrated ecg module, pulse oximeter, blood pressure and temperature sensor. In *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, pages 1–4. IEEE, 2019.

[7] Yu-Chi Wu, Pei-Fan Chen, Zhi-Huang Hu, Chao-Hsu Chang, Gwo-Chuan Lee, and Wen-Ching Yu. A mobile health monitoring system using rfid ring-type pulse sensor. In *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 317–322. IEEE, 2009.