Cardiac Arrest Detector

Kyle Hanks 

*Electrical and Computer Engineering Department*

*George Mason University*

Fairfax, Virginia, 22030

[khanks@gmu.edu](mailto:khanks@gmu.edu)

Kyle Loyd Guthrie 

*Electrical and Computer Engineering Department*

*George Mason University*

Fairfax, Virginia, 22030

[kguthrie@gmu.edu](mailto:kguthrie@gmu.edu)

Shamili Mownika Tetali

*Electrical and Computer Engineering Department*

*George Mason University*

Fairfax, Virginia, 22030

[stetali@gmu.edu](mailto:stetali@gmu.edu)

Sunanda Roy

*Electrical and Computer Engineering Department*

*George Mason University*

Fairfax, Virginia, 22030

[sroy9@gmu.edu](mailto:sroy9@gmu.edu)

*Abstract*— Internet of Things is considered one of rapidly advancing innovative technologies that has influenced several areas, modelling a shift in traditional healthcare systems. Proliferation of IoT devices has enabled several opportunities to improve quality of life and providing methods to reduce mortality rates. Healthcare Domain is one important area requiring high standards of accuracy and precision in determining key metrics. The objective of this project is to design a Proof-of-concept prototype based on research with Connected Healthcare multisensory IoT system that extracts human body vital parameters and derive an early alarm of fatal cardiac arrest. The design goal primarily focuses interfacing various sensors and implementing data collection software methodologies by means of considering cost-effective techniques to potentially bring the technology closer to users and aid larger human population.

Keywords—IoT, Arduino, Bluetooth, pulse, temperature, Android, Firebase

1. INTRODUCTION

Advancements in various fields of modern Computing and Communication technology has embraced new ideas in IoT design implementations. These innovations provided extensive opportunities to improving how traditional systems operated for betterment of developing enhanced lifesaving solutions. The importance of Healthcare and especially Cardiac monitoring systems has come a long way with extensive research put into characterizing cardiac abnormalities and human organ functions and tremendous studies have proved that timely monitoring of various vital parameters will have a lifesaving outcome. With the evolution of IoT it was possible to bring in these technologies together and implement medical solutions for proactive detection of health issues.

The high-level design idea on this project is to develop an embedded system using Arduino Nano and interfacing it to various sensors like pulse and temperature which are basic parameters to detecting anomalies in cardiac function. Essentially, the design aspect would leverage in-built low power Bluetooth communication method to integrating sensor data into android based application which would then establish a secure channel to cloud based solution for data acquisition and analysis.

1. USE CASE DESCRIPTION

Diagram

Description automatically generated

Fig. 1. Block diagram of key components

1. IOT TECHNICAL SOLUTION

We propose building a hardware prototype along the lines of the experimental setup described in [1]. The goal of this prototype IoT device is to capture the instantaneous heart rate and body temperature readings of a human and perform computations which would result in the detection of an imminent danger of cardiac arrest in that person.

1. *System Architecture*

We split the entire architecture of the solution into five essential components:

1. *Sense*
2. *Compute*
3. *Transmit*
4. *Store*
5. *Retrieve*

The above components are described below in greater detail.

1. *Sense*

The following sensors have been used to capture user data:

1. *Tragoods pulse sensor/heart rate monitor*

ARDUINO LIBRARY: PulseSensor Playground

This module [2] can be used to detect our heart rate by clipping it to sensitive areas like an earlobe or fingertip. In our setup, we connect it to a fingertip. The captured signal is a raw voltage which can be plotted on any serial plotter to show the heart rate varying over time.



Fig. 2. Components of pulse sensor package

The front side of the sensor (with the heart logo) makes the contact with the skin. This front side has a small round hole, which is where the LED (reverse mounted at the back side) shines through from the back. Just below this round hole is a little square, which is an ambient light sensor.

The LED shines light into the fingertip or earlobe, or other capillary tissue, and sensor reads the light that bounces back.

1. *FTCBLock TMP36 temperature sensor*

ARDUINO LIBRARY: TMP36

This module [3] is a low voltage and high precision temperature sensor, which is a possible candidate for reading body temperatures.

CHALLENGE: This sensor does not give stable and accurate values obfuscating our future calculations.

A picture containing text, electronics

Description automatically generated

Fig. 3. TMP36 temperature sensor

1. *Compute*

For detecting a heartbeat, we set a threshold value and classify any reading above that as a heartbeat and below as none.

CHALLENGES: Obtaining readings in the expected range required setting the correct threshold value and sampling interval, so the correct number of heartbeats are detected per unit time interval. We solved this problem by observing the signal waveform to detect threshold and surveyed online implementations for setting the sampling interval. As a part of this verification, we added support for computing the Beats Per Minute (BPM) periodically during a user session. Our obtained BPM values were in the expected normal range (~60).

The temperature readings can be obtained using the associated library functions or manually calculated (ensure Arduino Nano 33 IoT operating voltage is factored into calculations).

1. *Transmit*

The computed sensor data are transmitted to the receiver device (a smartphone) using the in-built Bluetooth Low Energy (BLE) module on Arduino Nano 33 IoT device.

1. *Store*

The smartphone device (running Android OS) stores these sensor readings in real-time into a cloud database (Firebase).

1. *Retrieve*

The data stored in the cloud database is queried back by our Android app with the future goal of plotting them into a user-friendly waveform for the convenience of the end user.

1. IMPLEMENTING BLE COMMUNICATION AND FIREBASE STORAGE
2. *Overview of BLE*

BLE is a wireless personal area network technology which provides a low power and low-cost interface for data transmission. Using BLE, data can be transmitted up to a range of 100 m and Android apps can communicate to edge devices having stricter power requirements such as proximity sensors, heart rate monitors, and fitness devices.

1. *Overview of Firebase storage*

Firebase is a cloud platform developed by Google for developing mobile and web applications. It provides end-users a means to store their data on a cloud server in real-time and perform different analytics on user data. Firebase provides numerous platforms for supporting a wide range of user requirements like performing user authentication, storing data into a real-time database, setting up a cloud server, executing cloud functions and implementing machine learning applications.

1. *Implementing BLE*

In this project, BLE has been used to transmit sensor data from an Arduino Nano 33 IoT acting as the peripheral device (client) to a Samsung Galaxy M11 smartphone acting as the central device (server). Arduino’s FreeRTOS libraries have been used to implement multithreaded reading of sensor values.

1. *CLIENT side*

Sensor readings from pulse sensor and temperature sensor are captured continuously and sent in parallel over two respective threads to the server end. These 2 sensor readings aretransmitted as Arduino BLE objects of type *BLEIntCharacteristic* to the Android app listening on server side.

1. *SERVER side*

The server side comprises of an Android app developed in-house using Android Studio software on Windows 10 platform. This app captures the exported Arduino BLE characteristic readings and converts them to Android objects of type *BluetoothGattCharacteristic* for subsequent processing.

1. *Implementing Firebase storage*

In this project, the Cloud Firestore database (server) provided by Firebase has been used to receive real-time data from an Android app running on a smartphone (client). This stored data is also read back by the Android app for possible future applications.

1. *CLIENT side*

Due to the asynchronous nature of the Android API (which reads the incoming sensor data over multiple threads), the data are stored to Cloud Firestore storage asynchronously as well.

1. *SERVER side*

For each continuous execution of the experimental setup, all the incoming data from Android data are stored as a single collection of multiple documents. Here, each document represents an individual sensor reading.

1. *Benefits of this communication interface*

The BLE communication protocol allows for the transmission of sensor data from client to server with extremely low latency, which may not be possible if implemented using other communication protocols like Wi-Fi, Bluetooth (BLE’s ancestor), etc. In addition, the low power consumption of BLE meets the requirement of a robust IoT device driven by power supplies of low range.

1. *Algorithm*
2. *Edge device (Arduino Nano 33 IoT)*

**Input:** Sensor readings: pulse (p\_raw) and temperature (t\_raw).

**Output:** BLE characteristic objects of sensor readings: p\_char and t\_char.

**Procedure:**

1. Initialize a BLE service object (srv).
2. Initialize BLE characteristic objects (p\_char and t\_char).
3. Add p\_char and t\_char to srv and advertise srv.
4. Create separate RTOS threads for reading respective sensor values (p\_thread and t\_thread).
5. Optionally create RTOS thread for monitoring tasks usage.
6. Schedule all RTOS tasks for continuous reading of sensor values at fixed intervals.
7. *Gateway device (Smartphone with Android OS)*

**Input:** Number of samples (set of pulse and temperature readings) to be captured from peripheral device (Arduino) - N.

**Output:** Cloud Firestore collection (user session) of multiple documents (sensor readings).

**Procedure:**

1. Establish connection to the peripheral device.
2. Initialize a counter for tracking the number of samples read.
3. Convert each (Arduino) BLE characteristic object to a (Android) BluetoothGatt characteristic object.
4. Establish connection to the Firebase instance.
5. On successful completion of a read characteristic function, convert to desired format and store into the cloud database.
6. Final collection consists of N sensor values.
7. RESULTS
8. *List of deliverables*
9. Arduino code and serial log
10. Android app and logcat log
11. Demo videos
12. Fritzing diagram

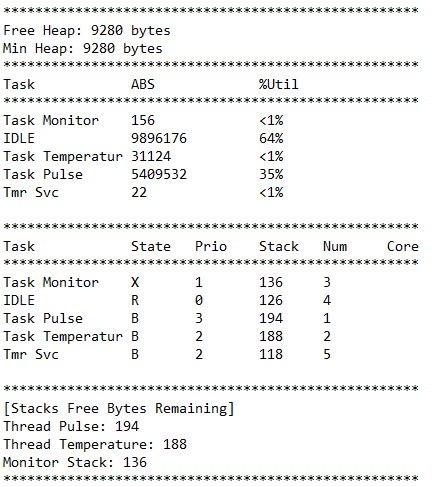


Fig. 4. Example of RTOS task usage

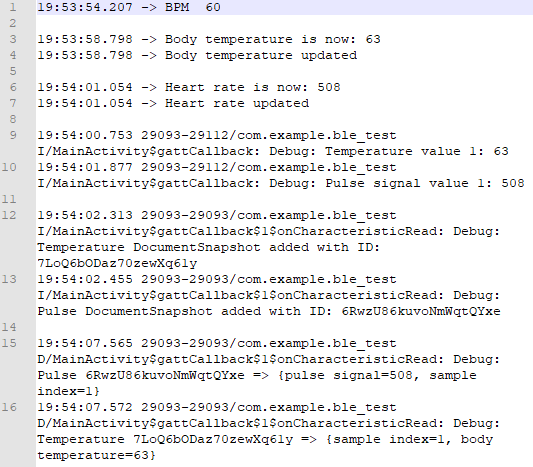


Fig. 5. Data flow of a sample sensor reading

1. *Future scope*

The current Android app can read stored values from the cloud after a round of data collection has been completed. These retrieved data can be plotted into a graph on the app’s user interface itself for subsequent analysis. Nevertheless, stored data from Cloud Firestore can always be plotted into any user-friendly format offline using popular software platforms like Python and Microsoft Excel.

The accuracy of sensor readings can be improved with further market survey of IoT devices, although the base IoT device of Arduino Nano 33 IoT will always be preferred over its higher footprint counterparts.

1. CONCLUSION
2. DISCUSSION

REFERENCES

1. AKM Jahangir Alam Majumder, Yosuf Amr ElSaadany, Roger Young, Donald R. Ucci, "An Energy Efficient Wearable Smart IoT System to Predict Cardiac Arrest", Advances in Human-Computer Interaction, vol. 2019.
2. Pulse Sensor Playground. [<https://github.com/WorldFamousElectronics/PulseSensorPlayground>]
3. TMP36 Temperature Sensor. [<https://learn.adafruit.com/tmp36-temperature-sensor>]
4. The Ultimate Guide to Android Bluetooth Low Energy. [<https://punchthrough.com/android-bleguide/>]
5. Android Developer Guides. [<https://developer.android.com/guide>]
6. Add Firebase to your Android project. [<https://firebase.google.com/docs/android/setup>]
7. Cloud Firestore. [<https://firebase.google.com/docs/firestore>]
8. Things You Should Know About Bluetooth Range [<https://blog.nordicsemi.com/getconnected/things-you-should-know-about-bluetooth-range>]

PAGE TO BE DELETED LATER

RED – In progress

The grad project paper will follow

* IEEE paper format, be at least 8 pages and must contain but not limited to the following sections
* Title and Student names
* Abstract
* Use Case description *-> Intro and diagram done*
* IoT Technical solution
  + Description for all components of the technical solution
  + Data flow diagram *-> Output done*
* Results from your IoT system
* Conclusion
* References

The final project final presentation will be 20 minutes broken into two sections: 15 minutes for presenting the use-case, technical solution and IoT system implementation/results and 5 minutes for Q&A

Each final project is graded based on the depth of covered material. The final project paper and presentation should provide answer to the following key questions:

* Why this use-case is important for IoT devices?
* What are the challenges related to this use-case? *->* *sensor readings done*
* What is the solution offered by literature research papers?
* What is the technical solution for the use-case and components for your IoT system?
* What are the main use-case requirements that determined your IoT component selections and solution?
* What were the challenges with microcontroller programming?
* What is the dataflow of your IoT system? *-> Output done*
* What are your results?