Cardiac Arrest Detector

Kyle Hanks
Electrical and Computer Engineering Department
George Mason University
Fairfax, Virginia, 22030
khanks@gmu.edu

Shamili Mownika Tetali
Electrical and Computer Engineering Department
George Mason University
Fairfax, Virginia, 22030
stetali@gmu.edu

Kyle Loyd Guthrie
Electrical and Computer Engineering Department
George Mason University
Fairfax, Virginia, 22030
kguthrie@gmu.edu

Sunanda Roy
Electrical and Computer Engineering Department
George Mason University
Fairfax, Virginia, 22030
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

I. 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.

II. USE CASE DESCRIPTION

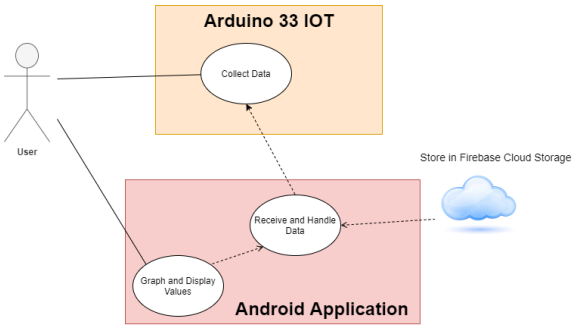

Fig. 1. Block diagram of key components

III. 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.

A. 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:

a) *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.

b) *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.

Fig. 3. TMP36 temperature sensor

2) 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).

3) 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.

4) Store

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

5) 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.

IV. IMPLEMENTING BLE COMMUNICATION AND FIREBASE STORAGE

A. 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.

B. 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.

C. 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 are transmitted as Arduino BLE objects of type *BLEIntCharacteristic* to the Android app listening on server side.

2) 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.

D. 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.

2) 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.

E. 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.

F. Algorithm

1) 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.

2) 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.

V. RESULTS

A. List of deliverables

1) Arduino code and serial log
2) Android app and logcat log
3) Demo videos
4) Fritzing diagram

```
*************************************************
Free Heap: 9280 bytes
Min Heap: 9280 bytes
*************************************************
Task            ABS             %Util
*************************************************
Task Monitor    156             <1%
IDLE            9896176         64%
Task Temperatur 31124           <1%
Task Pulse      5409532         35%
Tmr Svc         22              <1%

*************************************************
Task            State   Prio    Stack   Num     Core
*************************************************
Task Monitor    X       1       136     3
IDLE            R       0       126     4
Task Pulse      B       3       194     1
Task Temperatur B       2       188     2
Tmr Svc         B       2       118     5

*************************************************
[Stacks Free Bytes Remaining]
Thread Pulse: 194
Thread Temperature: 188
Monitor Stack: 136
*************************************************
```

Fig. 4. Example of RTOS task usage

```
19:53:54.207 -> BPM  60

19:53:58.798 -> Body temperature is now: 63
19:53:58.798 -> Body temperature updated

19:54:01.054 -> Heart rate is now: 508
19:54:01.054 -> Heart rate updated

19:54:00.753 29093-29112/com.example.ble_test I/MainActivity$gattCallback: Debug: Temperature value 1: 63
19:54:01.877 29093-29112/com.example.ble_test I/MainActivity$gattCallback: Debug: Pulse signal value 1: 508

19:54:02.313 29093-29093/com.example.ble_test I/MainActivity$gattCallback$1$onCharacteristicRead: Debug: Temperature DocumentSnapshot added with ID: 7LoQ6bODaz70zewXq6ly
19:54:02.455 29093-29093/com.example.ble_test I/MainActivity$gattCallback$1$onCharacteristicRead: Debug: Pulse DocumentSnapshot added with ID: 6RwzU86kuvoNmWqtQYxe

19:54:07.565 29093-29093/com.example.ble_test D/MainActivity$gattCallback$1$onCharacteristicRead: Debug: Pulse 6RwzU86kuvoNmWqtQYxe => {pulse signal=508, sample index=1}
19:54:07.572 29093-29093/com.example.ble_test D/MainActivity$gattCallback$1$onCharacteristicRead: Debug: Temperature 7LoQ6bODaz70zewXq6ly => {sample index=1, body temperature=63}
```

Fig. 5. Data flow of a sample sensor reading

B. 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.

VI. CONCLUSION

VII. 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?