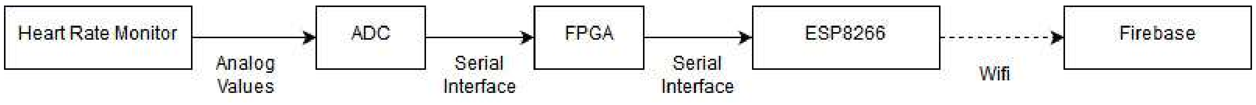
ECE 540 Final Project Proposal

Project Name: "I've Got Heart" Wireless Heartrate Monitor

Engineers: Alex Olson, Ryan Bentz, Ryan Bornhorst, and Andrew Capitana

Project Description

We are going to create a wireless heart rate monitor that reads heart rate readings locally and updates a Google Firebase. An Android app will read the values and display the heart rate readings on the mobile device.



Design Approach

Hardware

- Serial interface for the ADC that reads the Heart Rate monitor analog values
- Serial interface for the ESP8266
- Sound generator for missing heart rate
- AHB-Lite peripherals for ADC and ESP8266 that main application uses to interact with serial interfaces
- LED's for blinking heart rate
- Seven Segment display for showing BPM

Software

- Algorithm that translates incoming data from the ADC into heart rate, frequency, and variability
- Timer interrupt used to measure beats per minute
- ESP8266 Wifi interface written in Arduino that handles the interconnect between the peripheral interface and the Wifi transmission
- Data flow control that handles the reading of raw data from heart monitor and writing final values to the ESP8266 interface

The heart rate monitor is connected to an ADC. The ADC is connected to an AHB-Lite peripheral that reads the data on a periodic interval and stores it in a circular buffer. The peripheral also provides a memory address for the application program to read from the buffer. The mipsFPGA application reads the data in a buffer on an interval that is on the order of 10-20x the sampling rate of the peripheral serial connection to the ADC. For example, the peripheral samples the ADC every 10ms and the application program reads all the samples every 1s and uses the data to do its calculations. Once the application program reads the data, it processes the data to find the BPM and sends the calculated BPM to the database.

There is a second AHB-lite peripheral that acts as the interface for the application program and the NodeMCU. The application program writes to a register in the peripheral and the peripheral takes the data and sends it through a serial protocol (SPI) to the NodeMCU Wi-Fi hardware module. The NodeMCU will then read in this serial data from the FPGA module output and transmit the data back out through its Wi-Fi signal to a server. The server will store the collected heart rate data. The real-time heart rate data can be displayed directly on the 7-segment display as it is read in. The purpose of the Android app is to be able to collect historical data and display that as well.

The role of the NodeMCU is to relay real-time heart rate data calculated in the mipsFPGA (heart rate, frequency, etc.) to a database that may be read by an android app. The Android app will simply act as a read-only GUI for the FPGA that displays the data that is collected.

The heart rate sensor used in this project is an open source pulse sensor called "Pulse Sensor Amped" which comes from www.pulsesensor.com. The sensor circuit uses an LED coupled with an APDS-9008 light sensor which is then amplified with an LCP-6001 linear op amp to produce an analog output signal.

Will you still communicate w/ Firebase if you have to abandon the mobile Android app portion of the project?

Yes. We feel that getting the NodeMCU to connect and transmit data to the Firebase will be the most challenging part of the Wifi communication and we are focusing on that first. The Android app itself can be simplified to simply read the values and show them in a single activity if we run out of time.

How will you communicate w/ Firebase from the embedded CPU?

The embedded CPU writes the calculated data to an AHB-Lite peripheral. The peripheral sends the data to the NodeMCU via serial interface. Preliminary research shows that the NodeMCU software can be written with Arduino and the community has provided Firebase libraries to facilitate the connection with Firebase.

Demonstrating Success

A successful demonstration will show the heart rate calculation on the 7-Segment displays and on the mobile device. We will show the heart rate can change due to physical activity or when disconnected from a person. We can confirm a good heart rate calculation by using a smartphone health app.

If we run out of time we will begin scaling down the project in the following order:

1. Abandon the android app
2. Simplify the heart rate calculation algorithm
3. Simplify the ADC serial interface buffer
4. Abandon the Wi-Fi interface and just display the heart rate readings on the 7 Segment Displays

Milestones

Week of Feb24:	Submit proposal and divide labor according to team strengths.
Week of Mar3:	Determine implementation of hardware and software solutions. Design interfaces and agree on communication protocols. Get hardware peripherals synthesized on the board. Test that hardware is responding to software testing.
Week of Mar10:	Verify valid heart rate monitor readings. Verify communication with Firebase. Get basic software algorithm functioning.
Week of Mar17:	Finish up anything left and add features to the device.