Day 10: Design 3

Sunday, July 30, 2023 3:21 PM

Goals:

- Continue working on software design
  - Pseudocode for app and RTC init still need to be written
  - Revising DAQ pseudocode based on preliminary testing
- Begin developing some of the software for the DAQ and App
- Aid with other parts of the design, time permitting

RTC initialization:
Arduino Code responsible to set up the RTC, runs once.

External Libraries
    Arduino
    uRTCLib
    UnixTime
    WiFiNINA

Globals:
    ssid: char[]
        The WiFi SSID
    pass: char[]
        The WiFi password
    rtc: uRTCLib::uRTCLib
        uRTCLib library object
    stamp: UnixTime::UnixTime
        Unix timestamp converter
    status: int
        WiFi radio status

Arduino Functions:

setup: () => void

    Begin RTC module

    Set status to WiFiNINA::WL_IDLE_STATUS

    While the status is not WiFiNINA::WL_CONNECTED:
        Attempt to connect (Begin WiFi, status is updated)
        Wait for 10 seconds

    Set RTC epoch using Wifi.getTime()

loop: () => void
    EMPTY

App Code:
Framework: Ionic/React
Using "Tabs" template as a baseline

External Libraries:

@ionic/react
@capacitor-community/bluetooth-le
@capacitor/core
@capacitor/filesystem
@capacitor/preferences
@capacitor/local-notifications
ionic-simple-lockscreen
ionicons
chart.js
react-chartjs-2

Hooks:

Globals:

HRV_SERVICE: const string
Bluetooth Low Energy HRV Metric Service UUID
HRV_CHARACTERISTIC: const string
Bluetooth Low Energy Characteristic UUID (receive HRV records)
ERROR_CHARACTERISTIC: const string
Bluetooth Low Energy Characteristic UUID (receive error codes from wearable).
REQUEST_CHARACTERISTIC: const string
Bluetooth Low Energy Characteristic UUID (send data requests to wearable).

data_hook: async (callbacks: array) => void

Try:
Initialize the bluetooth-le BleClient
Let deviceId: string be the output of connect(callbacks)
If there is an error:
Log the error in console

connect: async (callbacks) => string

Let connected:boolean be false

While not connected:
If there is no saved id in preferences:
Search for a device with the HRV_SERVICE as its advertised service

Save selected device id to preferences
Else:
Use saved id from preferences

Attempt to connect to the device wearable 5 times with a timeout of 5 seconds
(On Disconnect parameter is connect(callbacks) function).

If no connection:
Clear preference for id

Let lastWeekDate: string be the UTC ISO8601 representation of 1 week ago

Write lastWeekDate to REQUEST_CHARACTERISTIC.

Start Notifications for HRV and Error characteristics
On Disconnect Parameters:

HRV_CHARACTERISTIC => callbacks[0]
ERROR_CHARACTERISTIC => callbacks[1]

fetchRecords: async (timePeriod:number) => array
Let rawData: array be an empty array

Let endingTimestamp: number be the current unix timestamp

Let currentDatetime: Date be a UTC Date object initialized to endingTimestamp - timePeriod

Let startTimestamp be currentDatetime's unix timestamp

While the unix time of currentDatetime is before endingTimestamp:
Try to open the corresponding HRV data file using currentDatetime (req…)

If the file doesn't exist
Increment currentDatetime by a day
Continue with the loop

Process data file:
Split each record (separated by a newline)
Split the numbers within each record (separated by a space).
Add each element from resulting array from split to rawData

Filter rawData by removing records with timestamps lower than startingTimestamp and greater than endingTimestamp

Return rawData

Pages:
Home Screen:
HomeScreen:React.FC: ({userState}) => JSX
Return JSX object implementing the design of ():
If there is a passcode set, make a lockscreen appear first.
If userState is 0:
Display Normal in status box
If userState is -1:
Display Fatigued in status box
If userState is 1:
Display Stressed in status box

Graph Tab:
GraphTab:React.FC: ({userState, userSettings}) => JSX

Create states for graphData (any) and timeframeSelection (number)

getChartData: async () => void

Let timePeriod:number store the time period to look at in seconds
If timeframeSelection is 0:
timePeriod is set to 1 hour
Else If timeframeSelection is 1:
timePeriod is set to 1 day

Else if timeframeSelection is 2:
timePeriod is set to 1 week
Else:
Log an error to console
End method

Let records:array be the output of fetchRecords(timePeriod)

Aggregate data:
If timeframe is 1:
Separate timestamps by nearest even hour
Average the corresponding HRV values and timestamps and save in records
If timeframe is 2:
Separate timestamps into their corresponding day
Average the corresponding HRV values and timestamps and save in records

Let labels:array be the local time representation of each timestamp. This is rounded to:
The nearest 5th minute when timeframe is 0,
The nearest Even hour when timeframe is 1,
The corresponding Day when timeframe is 2.

Let values:array be the corresponding HRV values of each record.

Let colors:array be the output of colorRecords(records).

Set chartData to a chartjs data object
labels should be labels
For the dataset:
data should be values
backgroundColor should be colors
borderColor should be colors

colorRecords: async (records) => array
Let colors:array be an empty array to store data point colors

Let baselineRecords:array be the output of fetchRecords(3 days (in seconds)).
Let baselineHRV:number be the mean of the HRV values stored in baselineRecords

For each record in records:
If the record is fatigued or stressed (the record's HRV value is greater than 107 or greater than 115% or greater than userUpperThreshold of baselineHRV or less than 16 or less than 85% of baselineHRV or less than userLowerThreshold):
Add the rgb value for a red color to the colors array
Else if the record is greater than 108% of baselineHRV or less than 92% of baselineHRV:
Add the rgb value for a yellow color to the colors array
Else:
Add the rgb value for a green color to the colors array

Return colors.

Return JSX object implementing the design of ()…

Advice Tab:
AdviceTab:React.FC: ( ) => JSX

Return JSX object implementing the design of () …

Settings Tab:
SettingsTab:React.FC: ({userSettings}) => JSX
Return JSX object implementing the design of () …:
Each text field edits its corresponding userSetting

Routing:
App:React.FC: () => JSX
Let userSettings: object be the a collection of useState<number> arrays for user upper and lower thresholds and passcode (userUpperThreshold, userLowerThreshold, passcode)

Let userState: object be the useState<number> array for the user state.

Let readjustError: number be the number of readjust errors from the last hour

hrvCallback: async (rawRecord: DataView) => void

storeRecord(rawRecord).
determineUserState().

determineUserState: async () => void
Let baselineRecords:array be the output of fetchRecords(3 days (in seconds)).
Let sampleRecords:array be the output of fetchRecords(3 minutes (in seconds)).

Let baselineHRV:number be the average of the baselineRecords' HRV values.
Let sampleHRV:number be the average of the sampleRecords' HRV values.

If the sample is stressed (the sampleHRV is greater than 107 or greater than 115% of baselineHRV or greater than userUpperThreshold):
Set userState to +1
Else if the sample is fatigued (the sampleHRV is or less than 16 or less than 85% of baselineHRV or less than userLowerThreshold):
Set userState to -1
Else:
Set userState to 0

storeRecord: async (rawRecord: DataView) => void
Parse rawRecord into timestamp:number and hrvMetric:number

Let currentDate: Date be the UTC date associated with timestamp

Open the corresponding HRV data file using currentDate (req…) for writing/appending

Let record:string be the formatted form of the record (req…)
If record is not already in the currentDate file:
Append the record to the file (end with newline).

errorCallback: async (rawError: DataView) => void
Parse rawError into errorCode:number

If errorCode is 1 (A readjust error):
Add one to readjustError

If readjustError is greater than 5:
Send a local notification to alert user to readjust their sensor

If readjustError is 1:
Set a timeout for an hour to set readjustError to 0.

Log errorCode to the console

Using useEffect (run once):
data_hook([hrvCallback, errorCallback])

Return JSX object with Routing to each page according to (), userState and userSettings should be passed as props to each component that requires it.

BLE Setup:
The device wearable acts as the BLE peripheral device
The device application acts as the BLE central device

HRV Service
UUID: x180F
HRV Characteristic
UUID: x2A19
Permissions: Read | Notify
Data Format: UUUUHHHH
U: Unix timestamp: uint32
H: HRV metric: float32
Error Characteristic
UUID: x2A1A
Permissions: Read | Notify
Data Format: Z
Z: Error Code: uint8
Request Characteristic
UUID: x2A1B
Permissions: Read | Write
Data Format: YYMD
Y: Year: uint16
M: Month: uint8
D: Day: uint8

TODO: make flowchart for between the devices, between each section of the app code, Arduino software priming directions, and app routing

Day 11: Design 4

Monday, July 31, 2023 6:24 PM

Goals:

- Consolidate all pseudocode into the Design Doc
    - Separate out function headers and variables
- Make software flowcharts
    - Between the devices
    - Between each section of the app code
    - Arduino software priming directions
    - app routing
- Time permitting, help with other sections
- Begin writing code for app, rtc init, finish DAQ code.