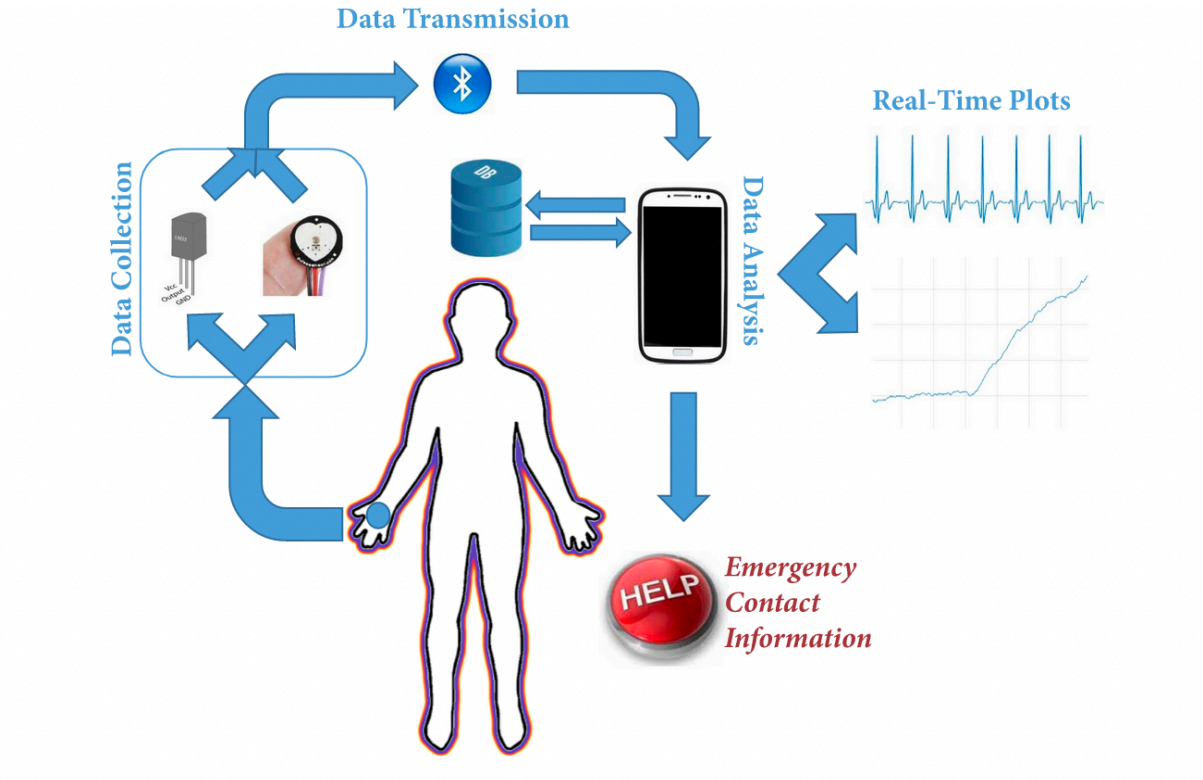
Remote Health Monitoring System

Introduction

The IoT device constantly collects data from the user and sends it to smartphone via a Bluetooth communication module. All the processing and data analysis take place in the application where the user has the option to view user real-time plots. These plots provide the user a basic idea of his/her body's status. The user does not have maintained a record of his/her data to ensure that s/he is in a healthy or unhealthy state since the application's job is to alert the user upon an emergency. Finally, when the algorithm senses an abnormality it immediately alerts the user.

System Architecture

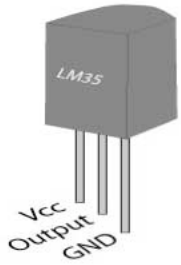


What we have in mind

After soldering all the hardware components on the PCB board, we design the system using Velcro strips to make it wearable.

Hardware

The initial prototype system consists of a low power Bluetooth chip, an Arduino or any other ADC platform chip, a pulse sensor(ECG), and a temperature sensor



Reading analogue signals from the pulse and temperature sensors and create a data packet to convert the signals into digital form. Subsequently, it sends those packets to the phone as a response to the data sending request. It also manages the Bluetooth communication by coordinating with the RN42 Bluetooth chip.

The data read from the sensors is always an analogue value between 0 and 5 volts since that is the operating voltage of this microcontroller. The Arduino then maps those voltage values to digital values ranging from 0 to 1023. Since the y-axis for ECG signals is also a voltage, all we had to do is scale the digital values to back voltage.

Basically, we read the sensor value from the Arduino through analogue pin 0 and then multiply it by 5 and divide it by 1023 to get the correct voltage value. This only applies to the pulse sensor since the expected output from the temperature sensor is in degrees Celsius.

To avoid the inaccuracy in simultaneous reading from multiple analogue pins, we not only need a delay between each reading, but also need to read from the same analogue pin _____times. We read the temperature data from the sensor twice and send the second reading, then do the same for the pulse sensor. We need to send different symbols before the sensor readings to be able to parse the data at the receiving end (android application). Before sending a temperature reading we send a ‘/’ and before sending a pulse reading we send a ‘-’, which makes data parsing simple.

Software

Given that the body temperature does not undergo as many changes as the ECG signal, we increased the ECG's sampling rate by decreasing the temperature's sampling rate. We fixed the sampling rates for the temperature sensor and the ECG signal at __ Hz and ___ Hz

Temperature Data

Novel Analytic Methods Needed for Real-Time Continuous Core Body Temperature Data

Temperature does not need much analysis except for converting the data points to the time main and smoothing the signal for better visual representation. The "noisiness" in temperature signal indicates a need for smoothing

ECG data

Collecting data for various day to day situations like sitting, walking, and running. By using treadmill for approaches other than sitting.
We take ECG and corresponding HR(after applying some transformation technique for accurate reading). Also, the variations that occur as a result of the sensor while doing some activity should be taken into account.

Data Analysis Techniques

- Noise Reduction: Filtering
- Baseline Wander Removal and Removal of High-Frequency Component:

  Baseline wander is a problem that shows ECG signals in a wavy fashion rather than being more of a constant envelope. A high pass filter to the signal improves the "look" of the signal because it removes the low frequency component that manifests itself as a sine-like pattern of the baseline. Removing the baseline wander gives a better signal which can help us process data more accurately.

Time domain operation of a low pass filter for signals is the mathematical operation called the moving average (often addressed to as smoothing).

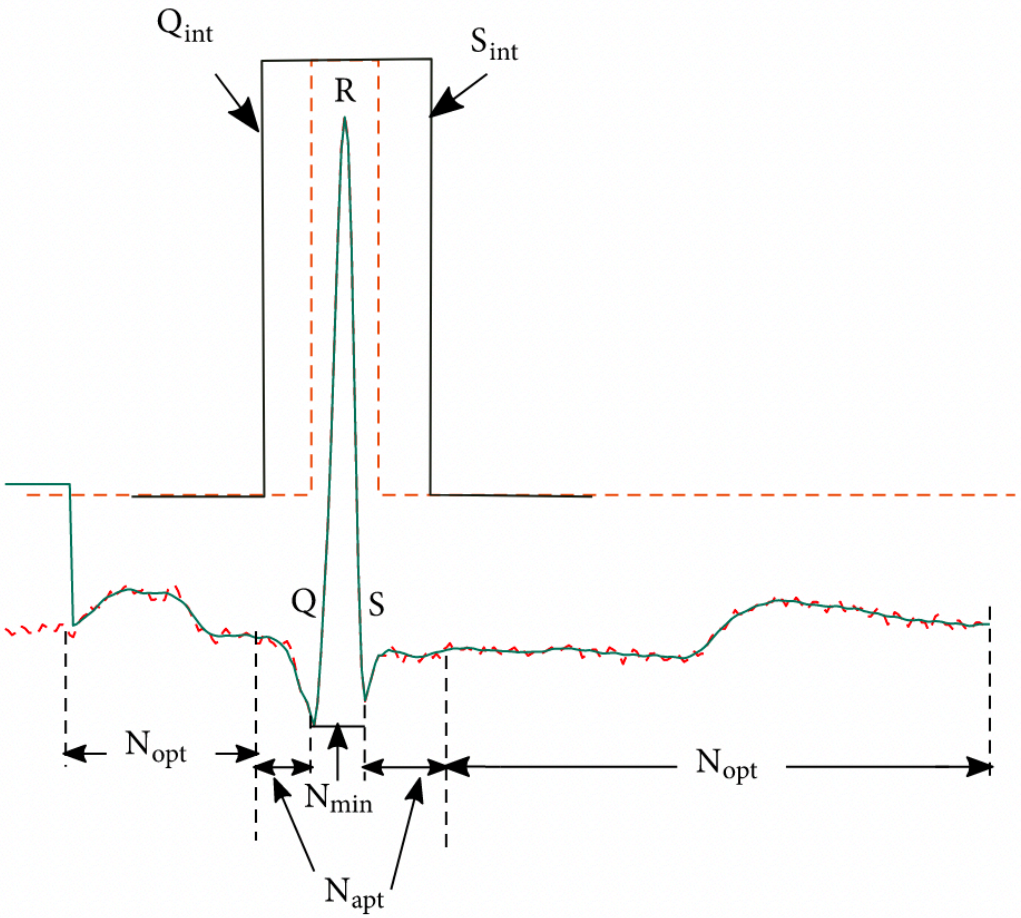
Here is w the cut-off frequency and is N the filter order:

$$|H(\omega)|^2 = \frac{1}{1 + (\omega_c/\omega)^{2N}}$$

Smoothing of ECG signal: (General or p-shift)UFIR smoothing Filtering

$$\widehat{N}_{\text{opt}}(p) = \arg\min_{N} \frac{\partial V(N,p)}{\partial N} + 1.$$

- ECG Signal Denoising on Adaptive Horizons



Next step would be Feature Extraction
Referred:
https://arxiv.org/ftp/arxiv/papers/1005/1005.0957.pdf

1. A feature extraction method using Discrete Wavelet Transform (DWT) to extract the relevant information from the ECG input data in order to perform the classification task

   In the feature extraction module the Wavelet Transform (DWT) is designed to address the problem of non-stationary ECG signals. It was derived from a single generating function called the mother wavelet by translation and dilation operations. Using DWT in feature extraction may lead to an optimal frequency resolution in all frequency ranges as it has a varying window size, broad at lower

frequencies, and narrow at higher frequencies. The DWT characterization will deliver the stable features to the morphology variations of the ECG waveforms.

2. Based on HR:
   We extracted heart rate or Beats per Minutes (BPM) from collected ECG signals. We can calculate BPM using several techniques:
   - taking the number of QRS peaks in a given time
   - using autocorrelation: signal is correlated with a shifted copy of itself as a function of delay or lag. Correlation indicates the similarity between observations as a function of the time lag between them.
     Formula:

$$R(k) = \sum_{n=N1}^{N2-k} x(m) * x(m+k)$$

   - using Fourier transform: The Fourier transform extracts the frequencies and harmonics of the signal. So, we find the location of the maximum harmonic in the frequency plot.

$$F(\omega) = \int_{-\infty}^{\infty} f(t) * e^{-i\omega t} dt$$

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) * e^{i\omega t} d\omega$$

3. R-R intervals:
   The **RR interval**, the time elapsed between two successive R waves of the QRS signal on the **electrocardiogram** (and its reciprocal, the HR), is a function of intrinsic properties of the sinus node as well as autonomic influences. For normal ECG signals, the R-R intervals do not fluctuate or suddenly change in a drastic manner.
4. ST segments:
   The **ST segment** is the flat, isoelectric section of the ECG between the end of the S wave (the J point) and the beginning of the T wave. The **ST Segment** represents the **interval** between ventricular depolarization and repolarization

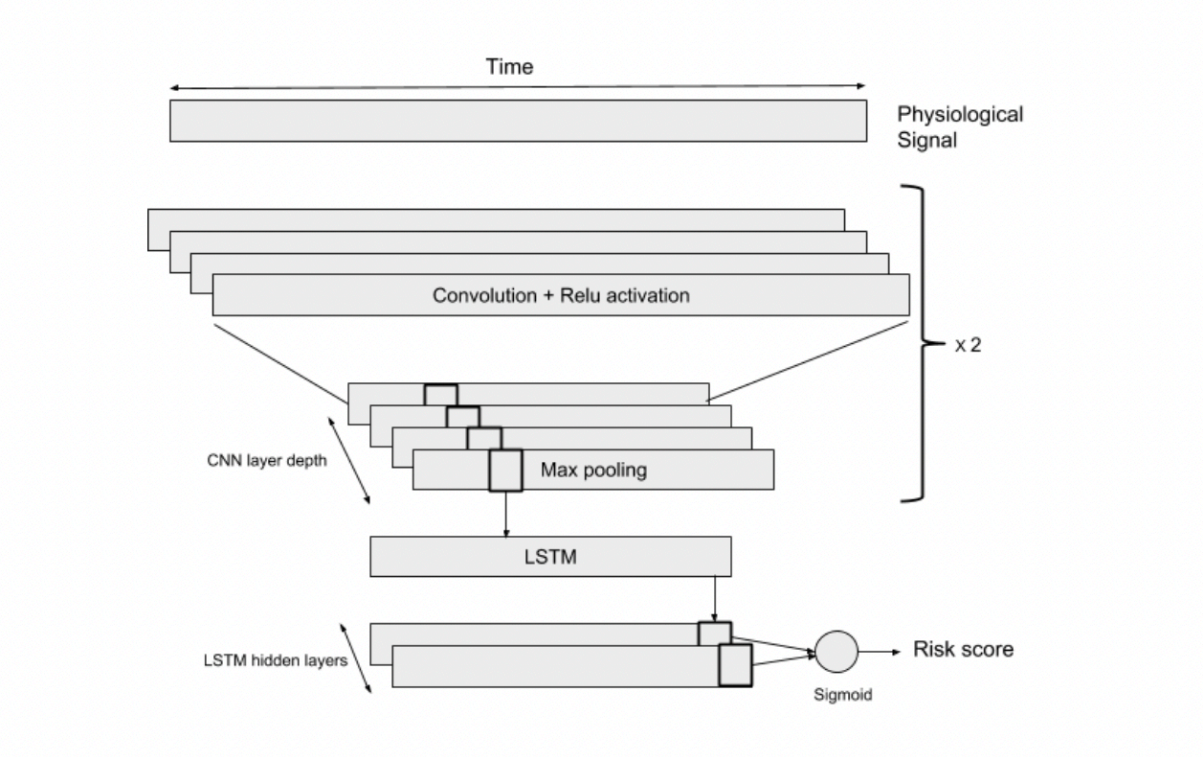
   The most important cause of **ST segment** abnormality (**elevation** or depression) is myocardial ischaemia or infarction. Also, elevated ST segments are one of the biggest indicators of heart attacks

Algorithm:

Many approaches available:

Classification of ECG N second window(can be treated as image classification problem)

(CNN(ReLU activation) dense layers + LSTM)+RNN(Recurrent Neural Network)
LSTM: Used for learning existing trend in the signals
RNN: For temporal data analysis

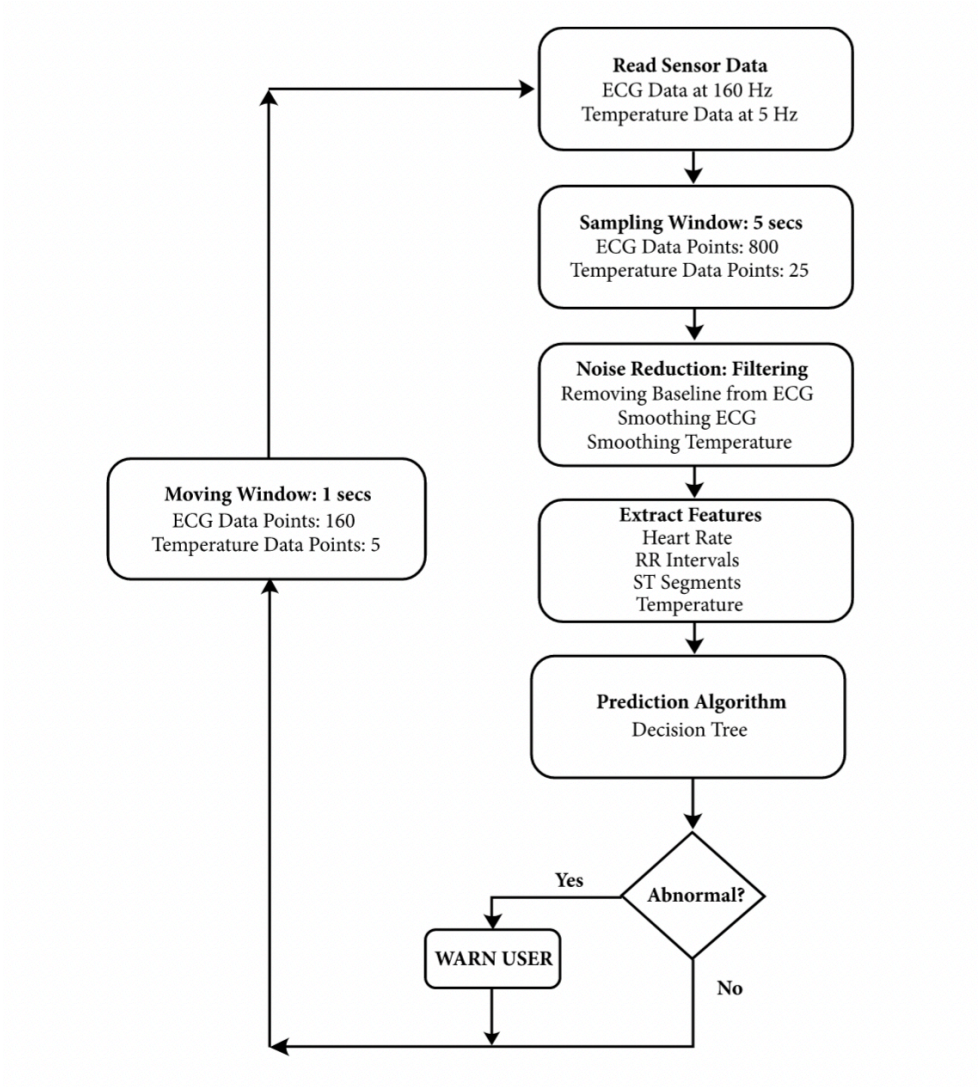


First steps is to read data from the sensors at _ Hz from temperature and __ Hz from ECG data.

We then maintain a sampling window of N seconds on which to perform all computations. After selecting the sample window, we reduce the noise by applying the filtering techniques discussed in Section

After removing all the noise components from the signals, we extract the three features from the ECG and pass on those features along with the temperature data to our prediction algorithm. If the results from the algorithm indicate that the current sample window is normal, the window shifts by n second and takes the next N seconds of data. If the algorithm detects an abnormality, it immediately warns the user. Using a moving window of n second creates the need more computation but it provides faster and more accurate feature extraction and prediction results. This means the next sample window will have n second of new data and N seconds of data from the previous sample window.

For every windows cardiac Arrest Risk Score will be calculated and deviation from previous measurements will be calculated.



For 5secs moving window

The algorithm above uses the decision tree as a prediction algorithm. However, using a Neural network gives a more accurate prediction.

Classification Techniques	Accuracy
Naive Bayes	90.74%
Decision Trees	99.62%
Neural Networks	100%

Reference:
https://d1wqtxts1xzle7.cloudfront.net/32129879/IJERTV1IS8282-with-cover-page.pdf?Expires=1622291700&Signature=LPFANApF7Pd9k5GQwmSBYS8bWtItKsOHsD-uNzvSMQ8f4tVgOLuvHtpkeleFrj-QpasAUlbjp~sw0mEf9YTrf6dEDEwhx9vh9Ib32YdiOatOL8l7VgEmTygpQP8W-hKYA4nv5zAnY-IJrHsSRtQnlkDwoqiVe87nz0cTbinlAMn~ngZFdH-DMBhRmfXfcTiaZdYgfnLq4rgab6B94-pK9HEEB6N1DGL~U6z25KUopLF~H~1Y6hX-d~iOC9ITmMS1zOa1IZ7m8qgbo5Uqxh13T2F2BDb-rOOZ655LjJtuloKNuL65hVwE0zgONqJeXXRt3G6e5C3uIXBwKdSmXMvQbw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

The MorphNet Algorithm

MorphNet iteratively shrinks and expands a network, shrinking via a resource-weighted sparsifying regularizer on activations and expanding via a uniform multiplicative factor on all layers. It is a method to reduce the model size.

Background:

We consider deep feed-forward neural net-works, typically composed of a stack of convolutions, biases, fully-connected layers, and various pooling layers, and in which the output is a vector of scores. In the case of classification, the final vector contains one score per each class. We number the parameterized layers of the DNNL=1,...,M+1. Each layer L corresponds to a convolution or fully-connected layer and has an input width I_L and output width O_L associated with it. In the case of a convolutional layer, I_L, O_L correspond to the number of input and output channels, respectively, and$O_{L-1}=I_L$ for most networks without concatenating residual connections. We consider L=M+ 1 to be the last layer of the neural network. Thus O_{M+1} is the size of the final output vector.

The neural network is trained to minimize a loss: $\min_{\theta} \mathcal{L}(\theta)$, where θ is the collective parameters of the neural network and L is a loss measuring a combination of how well the neural network fits the data and any additional regularization terms (e.g.,L2 regularization on weight matrices).

Method:

We motivate our approach by first presenting a naive solution to

$$O^*_{1:M} = \operatorname*{argmin}_{\mathcal{F}(O_{1:M}) \leq \zeta} \min_{\theta} \mathcal{L}(\theta).$$

the width multiplier.

Let $\omega \cdot O_{1:M} = \{b\omega O_1 c, ..., b\omega O_M c\}$ for ω>0.
Observe that ω<1 results in a shrunk network and ω >1 results in an expanded network. To solve Eq. (2) one may perform the following process:

1. Find the largest ω such that $F(\omega \cdot O_{\circ 1:M}) \leq \zeta$.

2. Return$\omega \cdot O_{\circ 1:M}$.

Algorithm 1 The MorphNet Algorithm

1: Train the network to find
$\theta^* = \underset{\theta}{\operatorname{argmin}}\{\mathcal{L}(\theta) + \lambda \mathcal{G}(\theta)\}$, for suitable λ.
2: Find the new widths $O'_{1:M}$ induced by θ^*.
3: Find the largests ω such that $\mathcal{F}(\omega \cdot O'_{1:M}) \leq \zeta$.
4: Repeat from Step 1 for as many times as desired, setting
$O^{\circ}_{1:M} = \omega \cdot O'_{1:M}$.
5: **return** $\omega \cdot O'_{1:M}$.

Reference:
https://arxiv.org/pdf/1711.06798.pdf

Another way to reduce the model size is to prune the Neural Network as in this reference
https://news.mit.edu/2020/foolproof-way-shrink-deep-learning-models-0430

Pruning a Neural Network refers to the compression of neural networks by removing parameters.

For more info, refer to this
https://proceedings.neurips.cc/paper/2020/file/46a4378f835dc8040c8057beb6a2da52-Paper.pdf