Deployment of a PPG Based Heart Rate Estimation Model to an Embedded System

JACOB SINDORF, Arizona State University, USA

Photoplethysmography (PPG) based heart rate (HR) estimation has become a staple in health monitoring and wearable health sensing. However, due to the noninvasive optical signal, the data is highly susceptible to noise sources such as motion artifact (MA). Noise corrupts the signal causing inaccurate interpretation, limiting the use of PPG based HR estimation to low motion and more clinical settings. A basis for MA removal and low complexity computation have brought rise to learning based techniques when estimating HR. Larger more diverse PPG activity datasets have been created and deep learning based estimation has shown the ability to interpret accurate HR in realistic MA affected scenarios. This work expands upon previous work and designs towards a wearable embedded HR device capable of continous HR estimation. Through a deep neural network (DNN), HR can be estimated within 10 beats per minute (BPM) at an accuracy of 81%. This model can then be deployed to an embedded system to make HR predictions in real time. The HR estimation embedded device depicted creates the foundation for future projects. With some direction in accuracy increase and integration with more sensor systems, this device can be used to estimate HR during day to day life.

Additional Key Words and Phrases: datasets, neural networks, embedded systems, photoplethysmography (PPG)

ACM Reference Format:
Jacob Sindorf. 2022. Deployment of a PPG Based Heart Rate Estimation Model to an Embedded System . 1, 1 (April 2022), 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Heart Rate (HR) estimation is a staple in almost all medical practice and medical evaluation. Understanding HR and the implications of it remains an integral part of health care and with the rise of wearable devices, HR estimation has become intertwined in many people's daily lives. One example of HR estimation comes from the photoplethysmography (PPG) signal, an non-invasive optical based bio-sensor that can measure changes in blood volume [3]. PPG signals are commonly interpreted for measuring both HR and blood oxygen saturation (SPO2) and are identified by their two distinct peaks. Shown in fig.1 is a simulated, idealized PPG signal where it is possible to see a periodic wave like pattern. The higher peak can be labeled as the systolic peak, and the lower peak as the diastolic peak. Using LEDs and a photodiode, the light received by the diode can be interpreted to create the PPG waveforms described. Two main modes of PPG are used are called transmitive and reflective [2]. Both modes commonly have a red and infrared LED and a photodiode to collect the emitted light. Transmitive PPG sensors measure light through the body by placing the LEDs and photodiode on different sides, whereas reflective measures reflected light, with the LED and photodiode on the same side. An advantage to reflective sensors would be that it can be placed in many different areas on the skin to read PPG, however it is heavily affected by noise.

Author's address: Jacob Sindorf, Arizona State University, 7001 E Williams Field Rd, Mesa, Arizona, USA, jsindorf@asu.edu.

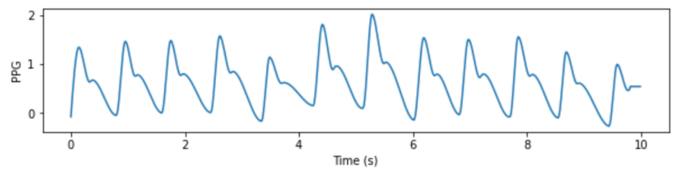


Fig. 1. Simulated idealized PPG signal using neurokit2 [4] with a HR of 75 and minimal added noise.

Creating noise in a PPG signal alters and corrupts it, making noise a major problem in HR estimation and overall signal interpretation. Sources of noise can arise from motion artifact (MA), high frequency burst, power-line interference, drift and more, all of which obscure the obtained signal. Wearing PPG sensors invite these noise sources in almost all environments except when the user holds perfectly still limiting the overall use cases of a PPG based wearable HR estimation system. This problem makes it impractical to continuously monitor heart rate during day to day activities. The use of denoising in interpreting the signals would be required. Actively denoising PPG signals however can be computationally expensive especially on an embedded system or memory constrained device. Larger devices capable of running complex algorithms would also be burdensome to the user. These bring up the motivation to accurately estimate HR in realistic day to day activities which include motion and noise corrupted PPG signals on a memory constrained wearable system.

2 RELATED WORK

Wrist worn PPG sensors are some of the most common types of HR estimation devices both on the market and in research. A summary of HR estimation techniques with wrist worn PPG devices is described in detail in [1]. Overall the effectiveness of specific algorithms on motion artifact removal, low complexity systems, and even disease detection are discussed, but it is important to emphasize a few main points. The first is that methods pointing towards MA removal have brought significant advances in cleaning incoming PPG signals. Reducing the noise assists in bringing out cleaner more interpretable signals, however most techniques still struggle with high motion activities. The next main point are algorithms focused on low complexity computation of real time signals. Extracting HR from a PPG signal in real time requires fast computations on a short window of data, meaning low complexity computations would be ideal for continuous HR monitoring. Methods involving fast fourier transform (FFT) and the inclusion of an inertial measurement unit (IMU) have found some success in estimation. The last main point would be learning based algorithms. Datasets on PPG and IMU values have been made publicly available and new machine learning based HR estimation techniques have been proposed with promising accuracy and real time application. However, a main deficiency with learning based algorithms is that a large and diverse dataset is required to encompass all day to day activities.

Considering these main points, it would be possible to combine the promising results of each into a potential real time system. This would include MA removal and low complexity computation for signal preprocessing, and a learning based algorithm for accurate interpretation. The work done by [5] expands upon this idea by first creating a new publicly available dataset on day to day activities, then applying machine learning to accurately estimate HR.

2.1 PPG-DaLiA Dataset

Expanding publicly available datasets was one of the main contributions from [5] as current PPG based datasets lacked in both size and real time situations. The created dataset, PPG-DaLiA, focuses on day to day activities over an extended

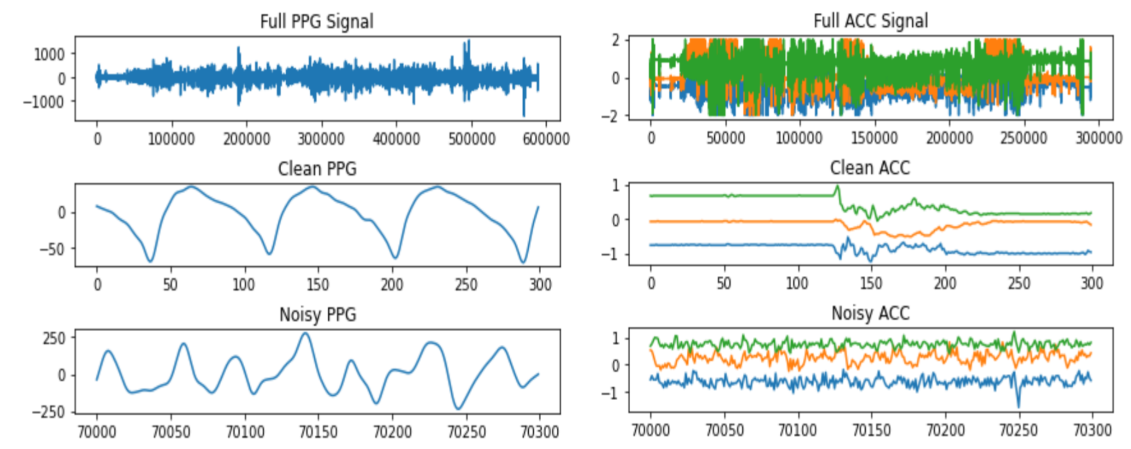


Fig. 2. Plotted data from S1 from the PPG-DaLiA dataset [5]. The left depicts PPG data, with the full 2.5 hour signal, a clean signal from sitting, and a noisy signal from walking. The right shows tri-axial accelerometer from the same time points. Blue represents x acceleration, orange is y, and green is z.

period of time of 2.5 hours for each of the 15 subjects. The activities reflect realistic tasks such as sitting, walking, biking, or even in general working. Each subject has data recorded from a wrist worn device that includes PPG and tri-axial accelerometer, and a chest worn device for electrocardiogram (ECG). ECG signals are used to interpret ground truth heart rate data provided as labels alongside activities in the dataset. An example of collected data for PPG and accelerometer for subject 1 is shown in fig.2, showing the full collected signal, an instance with minimal noise (when sitting), and an instance with noise (when walking). Designing the dataset for day to day activities allows it to be used in continuous HR monitoring problems as it captures a range of data including MA. This makes it very applicable to machine learning problems capable of interpreting noisy and clean sensor input.

2.2 Deep Learning and Embedded Application

Creating the diverse dataset was the first step in the work done in [5], but an accurate HR estimation problem with ML was also addressed. This was done to convey the datasets ability to work in a realistic settings for continuous HR monitoring, even with noise. Overall a convolutional neural network (CNN) was created and compared to be the most accurate model out of other deep learning techniques. The PPG-DaLiA dataset was also compared to other publicly available PPG based datasets and was found to perform with the lowest mean absolute error (MAE). However the proposed CNN model, although highly accurate, remains computationally expensive. To solve this a reduced CNN model intended for embedded application was proposed. A small loss in accuracy was found however it significantly saves memory. This work begins by replicating the proposed reduced CNN model, which is pictured in fig.3. Through python, each sensor signal from the dataset can be preprocessed the same as described [5]. This starts with an 8/2 second window and shift per each of the 4 sensors, a FFT on each of the time series segment, trimming down to only the 0-4Hz range, then finally a z-score to get a 0 mean, 1 standard deviation per segment. The 4 segments can then be stacked into a 257x4 array and fed into the CNN model with the corresponding ground truth HR value (in beats per minute (BPM)). Replicated results of this model are also pictured in fig.3, showing the loss curve (MAE), and a scatter plot of predicted vs actual HR. An r^2 metric was used to evaluate the model alongside MAE, with the model predicting

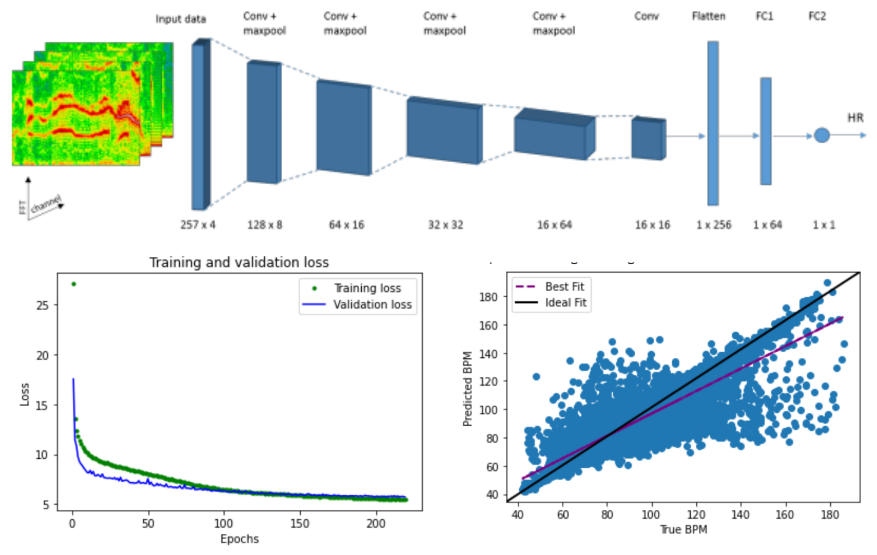


Fig. 3. Replicated CNN results based on the proposed embedded CNN in [5]. The top displays the CNN architecture from [5]. Replicated loss curve results with MAE (bottom left) and a comparison between predicted vs actual HR (bottom right) is also shown.

around 85% of HR within 10 BPM. Replicating the model, it does contain a significant memory savings and would be appropriately sized for an embedded application. However, a few other issues arise when deploying models for specific embedded systems.

Embedded systems would allow for a wearable and uncumbersome device capable of being worn throughout day to day activities. Many factors must be considered in the design and use of such a device. Besides the mentioned memory constraints, it is also imperative to design with specific hardware in mind. This would include a low cost and low power microcontroller capable of running trained models, training and mapping the available hardware's sensors to the trained dataset, and using available software given its limitations. The work done in [5] proposes an important foundation for an embedded application for HR estimation. Using that framework, this work expands upon that and designs for an embedded application intended for continuous HR monitoring.

3 MODEL DESIGN

From the embedded framework presented in [5], a new embedded system can be designed and created for continuous HR monitoring. In this case, the hardware revolves around an Arduino Nano 33 BLE Sense. The following sections describe how the data is preprocessed, how a model can be trained and deployed, and how new sensors can be integrated to make HR predictions on an embedded wrist worn device.

3.1 Dataset Preprocessing

Starting off the design for an embedded system, some changes to the dataset preprocessing must be performed to better fit an Arduino. A summary of the preprocessing steps are shown in fig.4, where the fundamental steps as mentioned previously are mostly kept the same. The biggest difference would be using only real numbers and changing the final

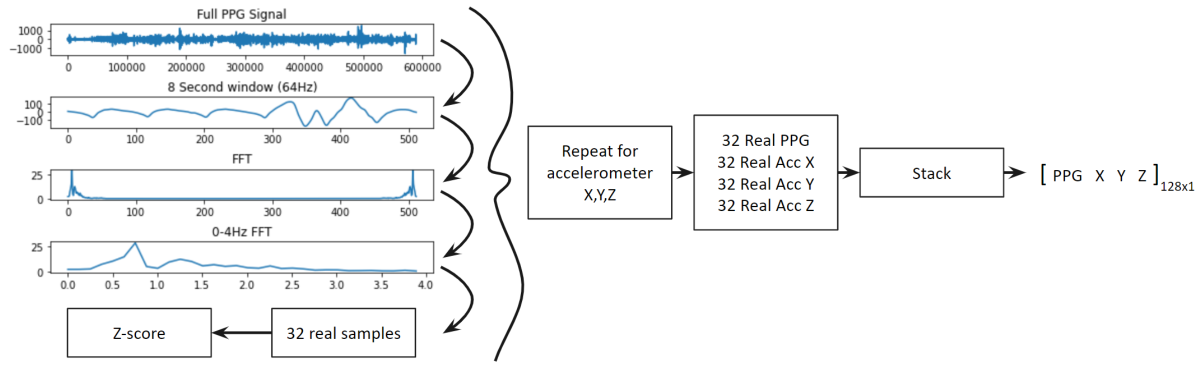


Fig. 4. Dataset preprocessing steps to better fit Arduino limitations. Includes graphical representations of the signals from S1, and the final array size to feed into a deep learning application.

input array size. Real numbers are obtained through the magnitude of the FFT and are needed as the Arduino software struggles to interpret the imaginary part. Changing the input size is done for two reasons, being that Arduino also has difficulty with multidimensional arrays, and a smaller input array saves memory. Overall the final array is changed from 257x4 to 128x1. The choice of 128x1 includes 32 real data points per sensor as this corresponds to the 0-4Hz range from an FFT given an input of 512 samples (64Hz for 8 seconds). This is performed on all 15 subjects, and the data can be stacked into a final (64687,128,1) array, with 64687 corresponding ground truth HR labels. This data is further split into 60% training data (38811 arrays), 20% validation data (12938 arrays), and 20% testing data (12938 arrays).

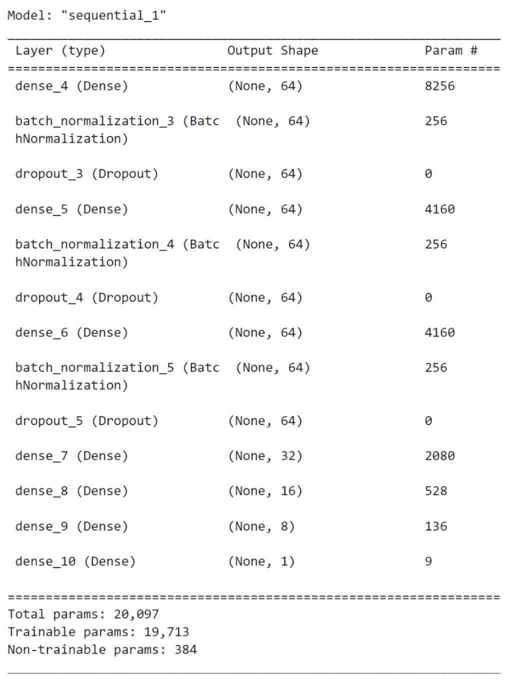
3.2 Deep Neural Network Design

After preprocessing the input data and splitting it into a train, validate, test format, a deep neural network (DNN) can be designed to predict HR. A change from CNN is performed mainly due to software issues between Arduino and Python, as Conv1D currently has dependency issues with newer versions of python. Conv1D would be used based on the (None,128,1) input array, but can be expanded to a 2D in future work by adding another dimension (None, 128,1,1). This work presents a deep neural network using tensorflow and keras which is summarized in fig5. It is trained and validated with the training and validation datasets respectively.

With a trained DNN the next step would be to condense and format the model for tensorflow lite (tflite) a program that allows trained models to be deployed to a microcontroller for embedded applications. Running tflite on the DNN outputs an Arduino compatible hex library where a condensed example of the tflite output is shown in fig6. Overall it creates a long list of hex values and includes the byte size of the trained model at the end denoted with len.

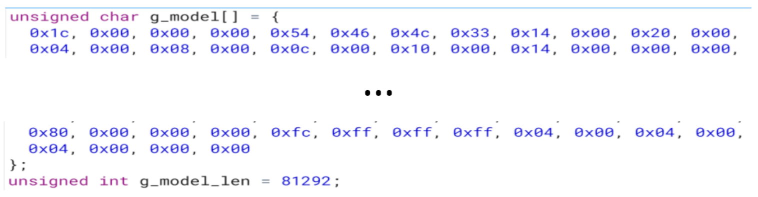
3.3 Hardware

Including the Arduino Nano 33 BLE Sense, there also needs to be a way to measure accelerometer and PPG signals to input into the trained tflite model. Fig.7 includes a circuit diagram of the hardware components used in this system. In order to measure acceleration, the built in LSM9DS1 IMU on the Arduino can be used. To obtain PPG signals, a SparkFun Pulse Oximeter and Heart Rate Sensor (POHR) - MAX30101 MAX32664 (Qwiic) is used. The main PPG signals come from the raw IR LED collected by the photodiode and processed by the MAX30101 portion of the sensor.



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 64)                8256

 batch_normalization_3 (Batc  (None, 64)               256
 hNormalization)

 dropout_3 (Dropout)         (None, 64)                0

 dense_5 (Dense)             (None, 64)                4160

 batch_normalization_4 (Batc  (None, 64)               256
 hNormalization)

 dropout_4 (Dropout)         (None, 64)                0

 dense_6 (Dense)             (None, 64)                4160

 batch_normalization_5 (Batc  (None, 64)               256
 hNormalization)

 dropout_5 (Dropout)         (None, 64)                0

 dense_7 (Dense)             (None, 32)                2080

 dense_8 (Dense)             (None, 16)                528

 dense_9 (Dense)             (None, 8)                 136

 dense_10 (Dense)            (None, 1)                 9

=================================================================
Total params: 20,097
Trainable params: 19,713
Non-trainable params: 384
_________________________________________________________________
```

Fig. 5. Model summary of the proposed DNN using python.



```
unsigned char g_model[] = {
  0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x14, 0x00, 0x20, 0x00,
  0x04, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x10, 0x00, 0x14, 0x00, 0x00, 0x00,

                                  ...

  0x80, 0x00, 0x00, 0x00, 0xfc, 0xff, 0xff, 0xff, 0x04, 0x00, 0x04, 0x00,
  0x04, 0x00, 0x00, 0x00
};
unsigned int g_model_len = 81292;
```

Fig. 6. Condensed tflite output example of a trained DNN model. This output only depicts the first and last few lines as the file contains a large number of values.

To visualize predictions a UCTRONICS 0.96 Inch OLED Module 12864 128x64 Yellow Blue SSD1306 Driver I2C Serial Self-Luminous Display Board, or OLED screen is used. Overall all the components can be driven by a 3.7 V battery making it a relatively low power system.

3.4 Sensor Preprocessing

Preprocessing the dataset was straightforward, but perhaps one of the greatest challenges would be to map the new sensor values to the dataset values. This requires extensive preprocessing and could end up being computationally expensive. This process is limited to both the available Arduino libraries, and memory. For this project, the PPG-DaLiA dataset was preprocessed in a way that fit the capabilities of Arduino, meaning the core preprocessing steps remain the same. The biggest difference would be to add a few steps prior to the 8 second windowing to fit the new senor

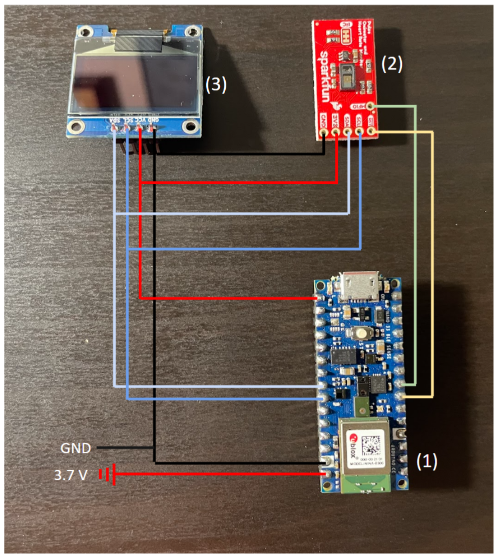


Fig. 7. Circuit diagram of the included hardware components required for embedded application. (1) Arduino Nano 33 BLE Sense, (2) Sparkfun Pulse Oximeter and Heart Rate Sensor, (3) UCTRONICS 0.96 Inch OLED Module.

data to the dataset. The dataset samples at 64Hz giving 512 samples every 8 seconds. Here the Arduino accelerometer samples at 104 Hz, and the SparkFun POHR samples at 100 Hz. To match the dataset sampling rate, a 1/64 delay is added between each collected sample and a 512 sample cap is created. This allows for 512 input samples to be collected in approximately 8 seconds. The PPG sensor required further preprocessing as its raw values come in with noise even without motion artifact. To solve this a moving average filter can be applied to the raw PPG data. Then to fix the amplitude difference, a DC bias removal is performed. Once the sensors data is preprocessed, it can be input to the trained tflite model, allowing for a HR prediction to be made. An overall sensor preprocessing summary is shown in fig.8 for reference.

4 RESULTS

4.1 DNN results

The DNN model discussed performed similar to the replicated CNN model. Fig. 9 shows the MAE loss curve finishing with around 6.5. A final r^2 value of .75 on the training data, and .67 on the test data was found and a final accuracy of 81% within 10 BPM was achieved. Note this is a 4% decrease from the CNN replica model created earlier. The model was converted to tflite as described previously, and shown in fig.6, with a final byte size of 81292. After deploying the model to Arduino, only 51% of the dynamic memory was used. This includes all libraries, arrays, the tflite model, and data storage needed to run predictions. With only about half of the memory being used, this leaves room for larger more computationally complex models to be explored in the future.

4.2 Wearable Hardware and Verification

Testing the embedded system required a wearable wrist worn device capable of making HR predictions. System hardware is displayed in fig.10 and is based on the hardware and circuit diagram as mentioned. The system comes equipped with

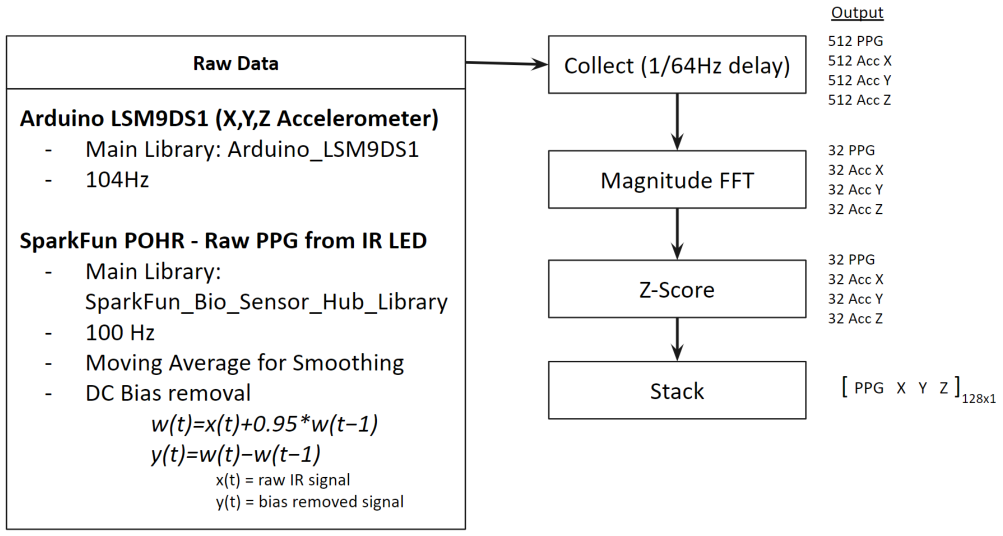


Fig. 8. Preprocessing flow chart for the sensors used in this embedded system.

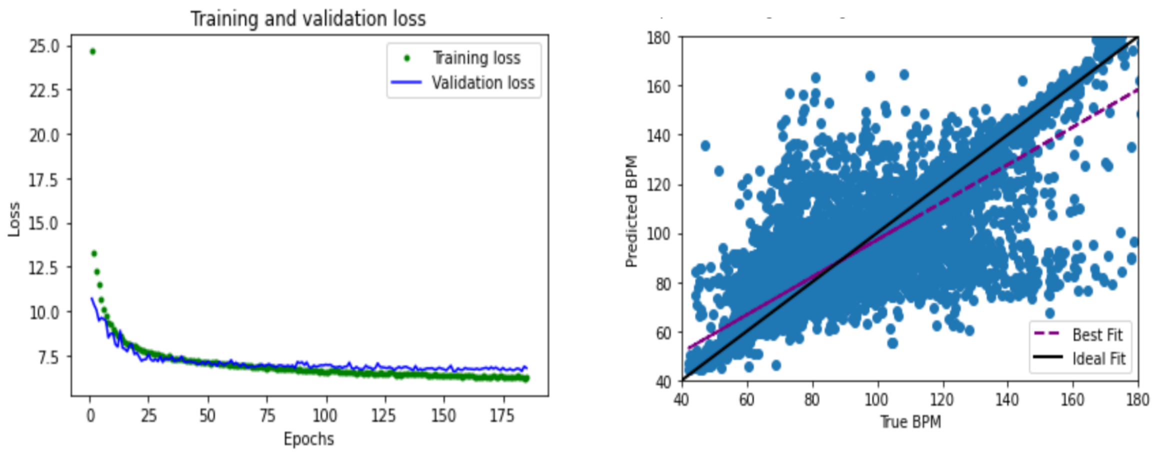


Fig. 9. DNN results, with MAE loss curve on the right and predicted vs real HR scatter plot on the right.

a 3.7V battery and a switch, requiring only to be powered on to make predictions. Due to the nature of the DC bias removal on the PPG sensor, it is best to place the PPG prior to turning on the device, then give it around 2 cycles (16 seconds) before continuous HR predictions begin.

For ground truth HR, the Sparkfun POHR sensor has a built in algorithm to interpret PPG into HR through its MAX32664 chip. It also displays its confidence in each HR estimation and can be used in the same set up as the proposed embedded system, an example verification system is shown in fig11. However, this built in algorithm is extremely

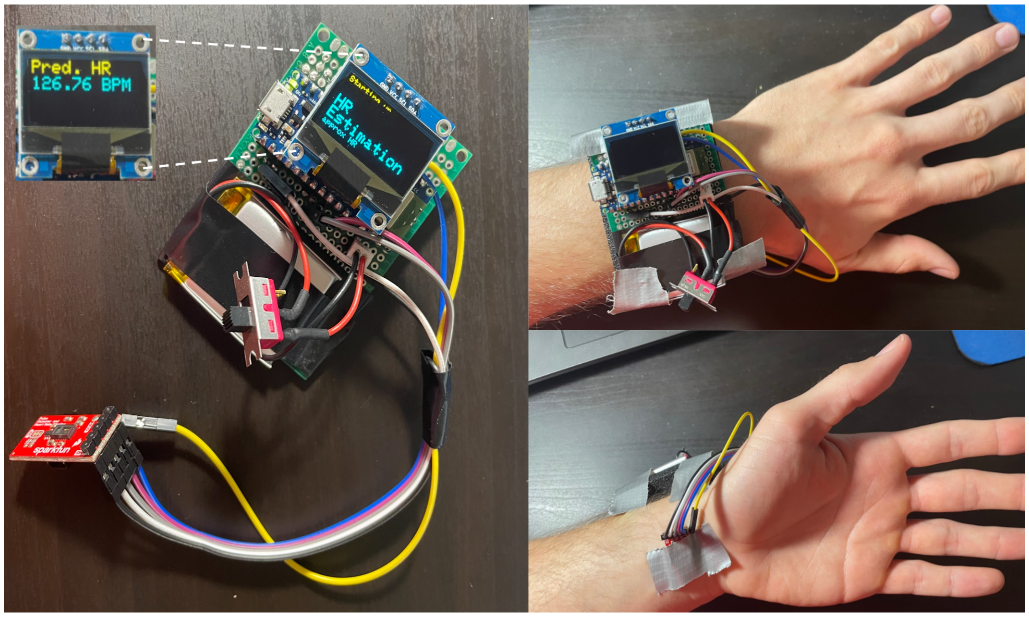


Fig. 10. Wearable wrist worn continuous HR estimation system.

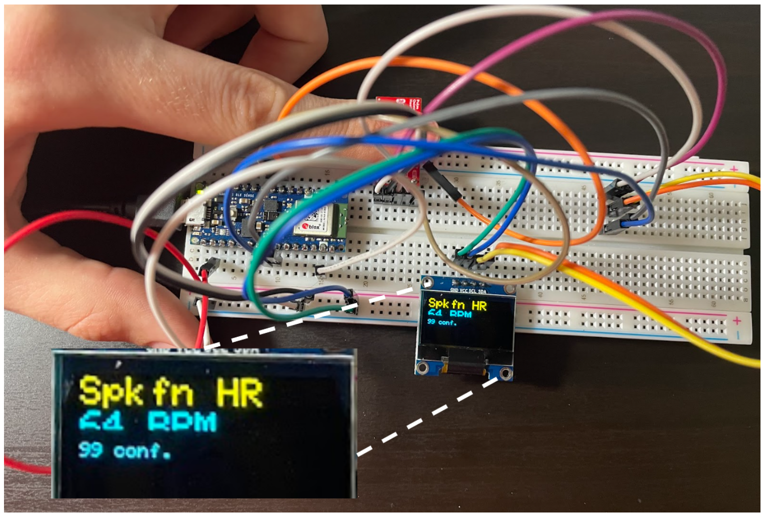


Fig. 11. Verification setup to read HR estimation derived from the Sparkfun POHR sensor.

sensitive to noise and requires a finger to be perfectly placed and unmoving on the LED. Any movement causes the system to make no prediction further motivating the need for a HR estimation system given noisy sensor data.

Overall the wearable continuous HR system faced some issues regarding accuracy in real time. Currently the system over estimates HR and requires future work to improve.

5 FUTURE WORK

This project proposes the foundation for a continuous HR estimation system given noisy sensor data. This would allow for accurate HR estimation in day to day tasks on an embedded wearable device. Some future steps however are required to finalize the project and its promising start.

First would be to finalize the real time sensor input, as this is the main issue in the high HR estimates being received from the model. More work to closely match the sensor output to the dataset sensor output would be necessary to improve model predictions. As well as sensor improvement, model optimization would be necessary. Given the large amount of remaining memory on the device, it is possible to use a more computationally expensive model. This would include re exploring a CNN model designed to fit the Arduino IDE. Once these areas are explored, a real validation procedure would be required to prove the effectiveness of the system. That procedure would include multiple subjects and multiple scenarios testing the system versus a ground truth HR value. This would most likely be derived from ECG as the SparkFun sensor only emphasizes the main problem of poor predictions in noisy environments. Verification of battery life would also be an important measurement as continuous HR estimation is expected to be worn for long periods of time while simultaneously providing HR values.

The most promising direction the project should take would be to start cross platform transfer learning. This potentially could solve the systems issue with the new sensor system and could also help expand it to any new sensor. This would make the device have a larger number of use cases and make it more accessible to other research projects.

6 CONCLUSION

Real time HR estimation of day to day activities struggles due to the addition of MA and noise. Larger datasets and accurate estimation techniques through machine learning have been previously proposed with accurate results. This work expanded upon previous HR estimation design and works towards an embedded device deployment. This work designed towards an embedded application and created the foundation of a real time wearable continuous HR estimation device. Methods to preprocess and design for a specific embedded system are described and can be expanded to streamline future device designs. Given the current system, some future work would be required to solidify and verify the design. Working towards higher accuracy and making the system adaptable to any sensor would allow this foundation to be applied to a wider range of projects making it applicable not only to continuous HR estimation, but activity recognition and disease detection as well.

REFERENCES

[1] Dwaipayan Biswas, Neide Simões-Capela, Chris Van Hoof, and Nick Van Helleputte. 2019. Heart Rate Estimation From Wrist-Worn Photoplethysmography: A Review. *IEEE Sensors Journal* 19, 16 (2019), 6560–6570. https://doi.org/10.1109/JSEN.2019.2914166

[2] Mohamed Elgendi. 2020. *PPG Signal Analysis: An Introduction Using MATLAB.* CRC Press. 27–52 pages. https://doi.org/10.1201/9780429449581

[3] Panicos A Kyriacou and Subhasri Chatterjee. 2022. 2 - The origin of photoplethysmography. In *Photoplethysmography*, John Allen and Panicos Kyriacou (Eds.). Academic Press, 17–43. https://doi.org/10.1016/B978-0-12-823374-0.00004-9

[4] Dominique Makowski, Tam Pham, Zen J. Lau, Jan C. Brammer, François Lespinasse, Hung Pham, Christopher Schölzel, and S. H. Annabel Chen. 2021. NeuroKit2: A Python toolbox for neurophysiological signal processing. *Behavior Research Methods* 53, 4 (feb 2021), 1689–1696. https://doi.org/10.3758/s13428-020-01516-y

[5] Attila Reiss, Ina Indlekofer, Philip Schmidt, and Kristof Van Laerhoven. 2019. Deep PPG: Large-Scale Heart Rate Estimation with Convolutional Neural Networks. *Sensors* 19, 14 (2019). https://doi.org/10.3390/s19143079