NUANSA INFORMATIKA
Volume 20 Nomor 2, Juli 2026

p-ISSN :1858-3911 **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

TinyML-Based Stress Detection Using Time-Domain HRV Features and a Lightweight DNN on ESP32

Sarmayanta Sembiring[1], Kemahyanto Exaudi[2], Abdurahman[3*], Jorena[4], Hadir Kaban[5], M. Buffon Prima[6], Rahmat Fadli Isnanto[7]
[3]Departemen Sistem Komputer Universitas Sriwijaya, Indonesia
[4,5]Departemen MIPA Fisika Universitas Sriwijaya, Indonesia
[1,2,5,7]Program Studi Teknik Komputer Departemen Sistem Komputer Universitas Sriwijaya, Indonesia

E-mail: [1]yanta@unsri.ac.id, [2]kemahyanto@ilkom.unsri.ac.id, [3*]abdurahman@unsri.ac.id,
[4]jorena@mipa.unsri.ac.id, [5]hadir_kaban@mipa.unsri.ac.id,
[6]bufonprima8@gmail.com,[7]rahmatfadliisnanto@ilkom.unsri.ac.id

Abstract

Stress is a psychophysiological condition that requires continuous and objective monitoring. However, existing wearable stress detection systems often rely on cloud-based processing or computationally intensive algorithms, limiting their applicability for real-time inference on resource-constrained embedded devices. This study presents a TinyML-based framework for real-time stress detection using the MAX30102 sensor and an ESP32 microcontroller. The proposed framework integrates four time-domain Heart Rate Variability (HRV) features (BPM, SDNN, RMSSD, and pNN50), a lightweight Deep Neural Network (DNN), full INT8 TensorFlow Lite quantization, and on-device inference to enable efficient edge-based stress classification. The DNN model was trained and evaluated using the WESAD dataset. Experimental results showed that a decision threshold of 0.70 yielded the best classification performance, achieving an accuracy of 82% and an F1-score of 0.63 for the stress class. The quantized TensorFlow Lite INT8 model preserved 100% prediction compatibility between the Python and ESP32 implementations. Furthermore, the MAX30102 sensor achieved a BPM measurement accuracy of 98.74%, while the HRV feature extraction implemented on the ESP32 produced results consistent with the reference calculations. These findings demonstrate that the proposed end-to-end TinyML framework enables accurate and computationally efficient HRV-based stress detection on resource-constrained microcontrollers, providing a practical foundation for real-time wearable edge-health monitoring.

Keywords : *TinyML, Stress Detection, Heart Rate Variability, MAX30102, ESP32*

Submission: June 19, 2026 *Accepted: July 06, 2026* *Published: July 15, 2026*
DOI Article: ***https://doi.org/10.25134/ilkom.v20i2.624***

1. INTRODUCTION

Stress is a psychophysiological condition associated with an increased risk of physical and mental health disorders, including depression, anxiety, diabetes, and hypertension [1], [2], [3]. According to the World Health Organization (WHO), stress-related conditions remain a significant global health burden that adversely affects quality of life and productivity [1].

Stress assessment is commonly performed using subjective approaches, such as psychological questionnaires, clinical interviews, and self-reports [3], [4]. However, these methods are susceptible to bias, reflect only an individual's condition at a specific time, and are unsuitable for continuous monitoring of psychophysiological changes [1], [3]. Continuous monitoring enables earlier stress detection and timely intervention without repeated manual assessments, making it more appropriate for wearable health monitoring systems [6], [7]. Moreover, the limited availability and high cost of psychological services further hinder effective mental health monitoring [4]. These limitations have encouraged the development of objective stress detection systems based on physiological signals and wearable technologies [3], [4].

Wearable technology enables continuous and objective acquisition of physiological signals, including heart rate, skin temperature, and galvanic skin response (GSR), to support real-time stress monitoring and digital healthcare applications [4], [5]. Nevertheless, many wearable systems still depend on cloud-based processing, which increases

p-ISSN :1858-3911 **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

communication latency and limits real-time performance. TinyML addresses this issue by performing machine learning inference directly on resource-constrained microcontrollers, reducing both latency and cloud dependency [14].

Heart Rate Variability (HRV) is commonly characterized using time-domain, frequency-domain, and nonlinear features [15], [16]. This study employs time-domain features because they are recommended by HRV analysis standards [15], [16]. To accommodate the memory and computational limitations of TinyML on resource-constrained microcontrollers [14], a compact set of selected time-domain HRV features is used to develop a lightweight model for on-device inference.

HRV measurements can be performed using Electrocardiogram (ECG) or Photoplethysmography (PPG), which are widely used in wearable devices to continuously obtain heart rate information and HRV parameters [7]. The MAX30102 PPG sensor is widely used in stress monitoring because it can measure heart rate, SpO_2, and HRV, and supports real-time physiological monitoring on wearable devices [5].

Various time domain HRV features such as Standard Deviation of NN intervals (SDNN), Root Mean Square of Successive Differences (RMSSD), and Percentage of successive NN intervals differing by more than 50 ms (pNN50) are widely used in the analysis of stress and relaxation states because they are related to autonomic nervous system activity [8], [9], [10], [11]. Additionally, HRV features in the time, frequency, and nonlinear domains have been extensively utilized as inputs for artificial intelligence models to predict stress [10]. These findings indicate that HRV is an important biomarker in the development of wearable based stress recognition systems and digital health applications [10], [11].

Research on stress detection based on physiological signals continues to evolve alongside advancements in wearable technology and artificial intelligence. Dalmeida and Masala [8] demonstrated that HRV features are good indicators for stress detection, with Average of NN intervals (AVNN), SDNN, and RMSSD being key features in the stress detection process. Additionally, Alnufaily et al. [9] reported that PPG-based HRV features, such as SDNN, RMSSD, and pNN50, exhibit significant differences between normal and stressed conditions. The study also indicated that PPG signals can serve as an alternative to ECG for wearable physiological monitoring and non-invasive HRV analysis. To facilitate comparison among representative studies, Table 1 summarizes the datasets, sensing modalities, machine learning models, deployment platforms, and limitations of previous HRV-based stress detection approaches.

Table 1. Comparison of Previous HRV-Based Stress Detection Studies

Ref	Dataset	Model	Limitation
[8]	Driver ECG dataset	KNN, SVM, MLP, RF, GB	No TinyML or embedded deployment
[9]	WESAD + PTT-PPG datasets	Statistical HRV analysis	No TinyML implement-ation
[12]	SWELL + experimental PPG data	Decision Tree	No micro-controller deployment
[13]	SWELL-KW	1D CNN	No TinyML implement-ation
[2]	Hospital nurse wearable dataset	XGBoost	Does not utilize PPG-derived HRV features

As presented in Table 1, previous studies have investigated either HRV-based stress detection [8], [9], [12], [13] or TinyML deployment on wearable devices [2]. However, the reviewed literature indicates that no prior study has integrated PPG-derived time-domain HRV features (BPM, SDNN, RMSSD, and pNN50), a DNN classifier, and TinyML deployment on an ESP32 microcontroller for real-time stress detection [2], [8], [9], [12], [13].

Based on this description, this study develops a TinyML based stress detection system that integrates the MAX30102 sensor, BPM, time domain HRV features (SDNN, RMSSD, and pNN50), and a DNN model trained using the Wearable Stress and Affect Detection (WESAD) dataset. The model is then implemented on the ESP32 to perform real-time classification of stress and relaxation states. The main contribution of this research lies in the use of BPM and a small number of time-domain HRV features to produce a lightweight, efficient model suitable for implementation on wearable devices with limited resources.

2. METHODS

This research was conducted in several stages, including a literature review, data processing, model development,

p-ISSN :1858-3911 **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

implementation, and evaluation of a TinyML-based stress detection system on the ESP32. These stages were designed to produce a model capable of performing real-time inference on devices with limited resources. The research framework used in this study is shown in Fig 1.

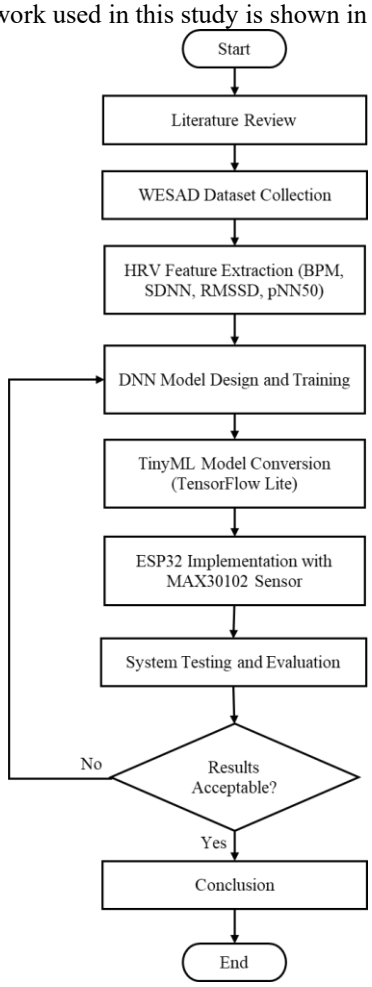

Figure 1. Research Implementation Framework

Based on the research framework shown in Figure 1, the research process began with a literature review to establish a theoretical foundation regarding the concepts and technologies supporting the study, including stress, HRV, the MAX30102 sensor, DNN, and TinyML. The next stage is data collection using the WESAD dataset, which serves as the data source for developing the stress detection model. The dataset then undergoes a feature extraction process to obtain the BPM, SDNN, RMSSD, and pNN50 parameters, which act as input features for the DNN model. The designed model subsequently undergoes training and evaluation to achieve optimal classification performance. Afterward, the model is converted to the TensorFlow Lite format so it can run on devices with limited computational resources. The converted model is then implemented on an ESP32 microcontroller integrated with the MAX30102 sensor to build a real-time stress detection system. The subsequent stage involves testing and evaluating the system to assess the model's ability to classify stress and relaxation conditions. The evaluation results are used as a basis for refining both the model and the system. The optimization and testing processes are repeated until the established criteria are met, before the research concludes with the formulation of conclusions based on the obtained results.

2.1. Literature Review

The research began with a literature review to establish a conceptual and theoretical foundation supporting the development of a stress detection system. The literature review covered various relevant topics, including stress detection, HRV, the MAX30102 sensor, DNN, and TinyML. The review process drew upon various scientific sources, such as journal articles, conference proceedings, and reference books related to the research topic. The analysis focused on previous studies regarding the use of HRV parameters in stress classification, the application of machine learning and deep learning methods for recognizing psychological conditions, and the implementation of TinyML on computing devices with limited resources. The results of the literature review were subsequently used as a basis for determining the HRV features to be used namely BPM, SDNN, RMSSD, and pNN50 designing the DNN model architecture, and implementing the TinyML model on an ESP32 microcontroller integrated with a MAX30102 sensor.

2.2. WESAD Dataset Collection

This study uses the WESAD dataset introduced by Schmidt et al. [21] for stress detection using wearable physiological signals. Only the wrist-worn Blood Volume Pulse (BVP) signal acquired from the Empatica E4 device was used. The original labels were resampled from 700 Hz to match the BVP sampling frequency of 64 Hz. The BVP signal was filtered using a fourth-order Butterworth band-pass filter (0.5–5 Hz), normalized using Z-score normalization, and segmented into 60 s windows with 25% overlap. Windows with a label purity of at least 70% were retained. Four time-domain HRV features, namely BPM, SDNN, RMSSD, and

p-ISSN :1858-3911 **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

pNN50, were extracted as input features for the DNN classifier.

For binary classification, the original WESAD labels were regrouped into Relax (baseline, amusement, and meditation) and Stress classes. The final dataset contained 1,167 samples, comprising 910 Relax samples (77.98%) and 257 Stress samples (22.02%), indicating an imbalanced class distribution. To preserve the original data distribution, no oversampling or undersampling was applied. Instead, class imbalance was addressed during model training using class weights computed from the training data. A summary of the processed dataset is presented in Table 2.

Table 2. Summary of the Dataset Used in This Study

Parameter	Value
Dataset	WESAD
Subjects	15
Signal used	BVP
Sampling frequency	64 Hz
Window length	60 s
Overlap	25%
Features	BPM, SDNN, RMSSD, pNN50
Total samples	1,167
Relax samples	910 (77.98%)
Stress samples	257 (22.02%)
Data balancing	Class weight
Classifier	DNN

2.3. HRV Feature Extraction

Heart Rate Variability (HRV) feature extraction was performed to obtain physiological parameters representing autonomic nervous system activity from the Blood Volume Pulse (BVP) signal. This study employed four time-domain HRV features, namely Beats Per Minute (BPM), Standard Deviation of NN intervals (SDNN), Root Mean Square of Successive Differences (RMSSD), and Percentage of Successive NN Intervals Differing by More than 50 ms (pNN50). These features were selected because they are recommended by HRV measurement standards and are widely used for stress assessment [15], [16].

Prior to HRV feature extraction, the wrist-worn BVP signal acquired from the Empatica E4 device was preprocessed to improve signal quality. The original BVP signal sampled at 64 Hz was filtered using a fourth-order Butterworth band-pass filter with cutoff frequencies of 0.5 Hz and 5 Hz to suppress baseline drift and high-frequency noise while preserving the physiological frequency components associated with cardiac activity. Subsequently, the filtered signal was normalized using Z-score normalization to reduce inter-subject variability before heartbeat detection.

Heartbeat detection, peak correction, and beat-to-beat (RR) interval estimation were performed using the HeartPy library [22]. The extracted RR intervals were subsequently used to compute the HRV features, namely BPM, SDNN, RMSSD, and pNN50, using the mathematical formulations presented in [17]. The overall HRV feature extraction procedure is illustrated in Fig. 2.

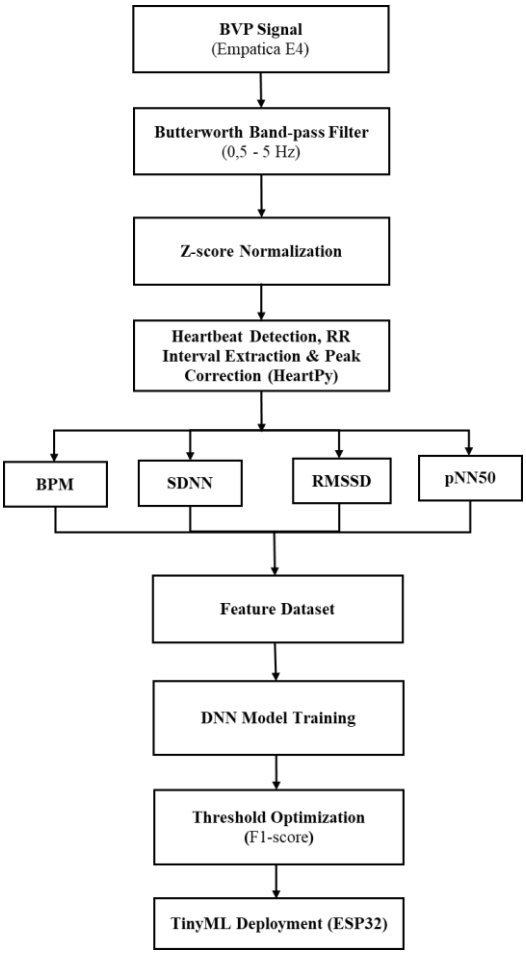

Figure 2. Workflow of the proposed HRV feature extraction, DNN model training, and TinyML deployment.

$$BPM = \frac{60}{RR} \quad (1)$$

Where:
RR = time interval between two consecutive pulse peaks (s).

$$SDNN = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(RR_i - \overline{RR})^2} \quad (2)$$

p-ISSN :1858-3911 **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

$$RMSSD = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}(RR_{i+1} - RR_i)^2} \quad (3)$$

$$pNN50 = \frac{1}{N-1}\sum_{i=1}^{N-1}\delta_i \; x \; 100\% \quad (4)$$

$$\delta_i = \begin{cases} 1, & |RR_{i+1} - RR_i| > 50\ ms \\ 0, & Other \end{cases} \quad (5)$$

2.4. Design and Training of the DNN Model

A DNN is a deep learning method consisting of multiple layers of neurons designed to learn complex data patterns and representations. Through its multilayer structure, a DNN is capable of performing feature learning and classification simultaneously, making it widely used in various pattern recognition and classification problems [18], [19].

In this study, the DNN was employed to classify stress and relaxation states using four time-domain HRV features (BPM, SDNN, RMSSD, and pNN50). Prior to training, the features were normalized using StandardScaler to improve learning stability. A subject-wise split was adopted, where subjects S2–S11 were used for training and subjects S13–S17 for testing to reduce data leakage and improve model generalization to unseen subjects.

To identify an appropriate architecture for TinyML deployment, four candidate DNN architectures (4-8-4-1, 4-16-8-1, 4-16-8-4-1, and 4-32-16-8-1) were experimentally evaluated under identical training settings. Only the network architecture was varied, while the optimizer, learning rate, batch size, regularization methods, and stopping criteria were kept constant to ensure a fair comparison. The models were evaluated using Accuracy, Macro Precision, Macro Recall, Macro F1-score, the number of trainable parameters, and the INT8 model size. Table 3 presents the classification performance and computational complexity of the evaluated candidate architectures.

Table 3. Comparison of Candidate DNN Architectures

Arch	Acc	Macro F1	Params	INT8 (KB)
4-8-4-1	0.78	0.73	129	3.37
4-16-8-1	0.80	0.75	321	3.83
4-16-8-4-1	0.82	0.76	353	4.50
4-32-16-8-1	0.85	0.78	1057	6.06

As shown in Table 3, the 4-32-16-8-1 architecture achieved the highest classification performance but required approximately three times more trainable parameters than the 4-16-8-4-1 architecture. Considering the memory and computational constraints of ESP32-based TinyML deployment, the 4-16-8-4-1 architecture was selected because it provides the best trade-off between classification performance and computational complexity. Therefore, the selected architecture was determined through an experimental comparison of candidate architectures, rather than through hyperparameter optimization or direct adoption from previous studies.

The final DNN architecture consists of an input layer with four neurons representing the HRV features (BPM, SDNN, RMSSD, and pNN50), followed by three hidden layers containing 16, 8, and 4 neurons with ReLU activation. A single output neuron with sigmoid activation performs binary classification of stress and relaxation. To improve model generalization, Gaussian Noise, Batch Normalization, and Dropout were incorporated into the network. The model was trained using the Adam optimizer with the binary cross-entropy loss function. The final DNN architecture is illustrated in Fig. 3.

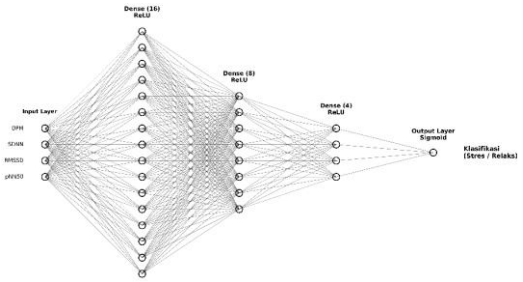

Figure 3. HRV-Feature-Based DNN Architecture for Stress and Relaxation Classification.

2.5. Converting the Model to TinyML

TensorFlow Lite for Microcontrollers (TFLM) enables machine learning inference on resource-constrained embedded devices, requiring only a small memory footprint, making it suitable for TinyML applications [20]. After training, the DNN model was converted from Keras (.h5) to TensorFlow Lite (.tflite) format using the TensorFlow Lite Converter. Full integer (INT8) quantization was then applied using a representative training dataset to convert both weights and activations from 32-bit

p-ISSN :1858-3911 **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

floating-point values to 8-bit integers. This process enables integer-only inference, reduces model size and computational complexity, and preserves satisfactory classification performance for deployment on the ESP32. The overall conversion process is illustrated in Fig. 4

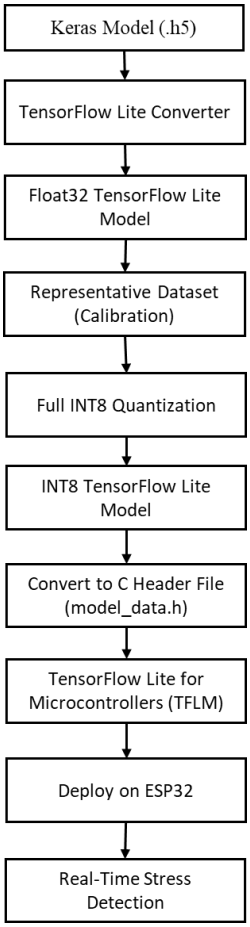

Fig. 4. TinyML model conversion and deployment process for ESP32 implementation.

The characteristics of the final TinyML model are summarized in Table 4. The selected 4-16-8-4-1 DNN architecture produced an INT8 TensorFlow Lite model of 4.50 KB, while TFLM required a 20 KB Tensor Arena as the runtime memory buffer during inference. The compact model size and memory footprint satisfy the memory constraints of the ESP32, demonstrating the feasibility of real-time stress classification using locally extracted HRV features.

Table 4. Summary of the Final TinyML Model for ESP32 Deployment

Parameter	Value
Model Architecture	4–16–8–4–1
Input Features	Four HRV Features
Output Classes	Two (Relax, Stress)
Model Format	TensorFlow Lite (INT8)
Quantization Method	Full Integer Quantization (INT8)
Final Model Size	4.50 KB
Runtime Memory (Tensor Arena)	20 KB
Target Platform	ESP32

2.6. Implementation on the ESP32 and the MAX30102 Sensor

To support implementation on the ESP32 microcontroller, the pre-trained TinyML model, feature normalization parameters, system configuration, and output class information are converted into header files that can be directly integrated into the microcontroller program. These files provide all the necessary information so that the ESP32 is capable of performing data preprocessing, running model inference, and interpreting classification results independently without relying on external computing devices.

After the model was fully trained, evaluation was conducted using several threshold values: 0.50, 0.55, 0.60, 0.65, 0.70, and 0.75. Each threshold was evaluated using the accuracy, precision, recall, and F1-score metrics. The threshold with the best F1-score was selected for use in the system implementation phase.

2.7. Hardware Implementation

During the implementation phase, the trained TinyML model along with feature normalization parameters, system configurations, and output class information is converted into a header file format that can be integrated into and directly accessed by programs running on the ESP32 microcontroller. The integration of these files allows the ESP32 to perform all processing stages independently, from input data preprocessing and model inference to classification result interpretation, without requiring external computing support.

The system's power source comes from a battery connected to the TP4056 module, which serves as a charging and battery protection circuit. The battery's output voltage is then boosted using the MT3608 module before being used to power the ESP32 and other components. The overall system circuit is shown in Fig 5

p-ISSN :1858-3911 **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

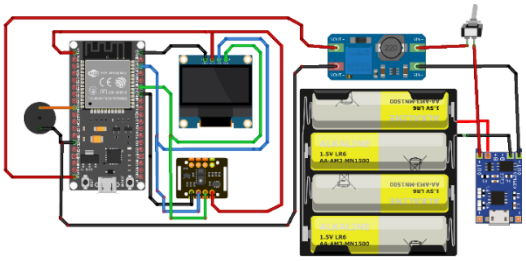

Figure 5. Implementation diagram of a TinyML-based stress detection system

As shown in Figure 5, communication between the MAX30102 sensor and the SSD1306 OLED display with the ESP32 uses the I²C interface. The SDA pins of both devices are connected to GPIO 21, and the SCL pins are connected to GPIO 22. The MAX30102 sensor receives a 3.3 V power supply from the ESP32, while the buzzer is connected to GPIO 25 as an audio indicator.

2.8. Software Implementation

During the software implementation phase, the trained model which has been converted to the TensorFlow Lite format and quantized to INT8 is integrated into the ESP32 program as a header file. In addition to the model, feature normalization parameters, system configurations, and output class information are also stored in the header file to support the inference process on embedded devices.

The program begins with the initialization of the MAX30102 sensor, the SSD1306 OLED, the buzzer, and the TinyML model. After successful initialization, the system waits until the user's finger is detected by the sensor. When a finger is detected, PPG signal data is collected over a specific time interval to obtain the RR interval and calculate the features BPM, SDNN, RMSSD, and pNN50. The obtained features are then normalized using training parameters before being used as input for the TinyML model. The inference results are subsequently used to determine stress or relaxation conditions and displayed on the OLED. Additionally, the buzzer will provide a notification once the classification process is complete. The program workflow implemented on the ESP32 is shown in Fig 6.

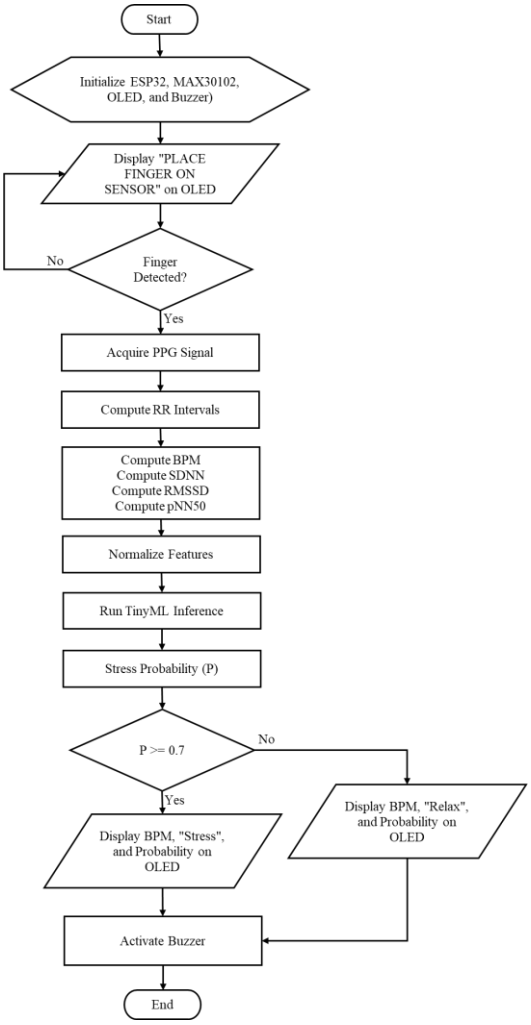

Figure 6. Flowchart of the program implementation on the ESP32

3. RESULT AND DISCUSSIONS

This chapter presents the results of the implementation and testing of the TinyML-based stress detection system designed in the previous chapter. Testing was conducted in stages, beginning with the evaluation of the classification model to determine the optimal threshold value, followed by testing the model implementation on the ESP32 microcontroller, testing the measurement accuracy of the MAX30102 sensor, and finally, functional testing of the entire system. Each testing phase aims to verify the system's performance and ensure that the obtained results align with the research objectives. The results and analysis of each test are presented in the following subsections.

p-ISSN :1858-3911 **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

3.1. Training Results and Threshold Selection

The stress classification model was developed using four time-domain HRV features (BPM, SDNN, RMSSD, and pNN50). The proposed DNN consists of four input neurons, three hidden layers with 16, 8, and 4 neurons, and one output neuron for binary classification of relax and stress conditions. The selected architecture contains 353 trainable parameters, providing a compact model suitable for TinyML deployment on resource-constrained embedded devices. The rationale for selecting this architecture is presented in Section 2.4.

After training, the model outputs were evaluated using six decision thresholds (0.50, 0.55, 0.60, 0.65, 0.70, and 0.75). Model performance was assessed using accuracy, precision, recall, and F1-score to determine the optimal operating threshold. Figure 7 compares the performance obtained at different threshold values.

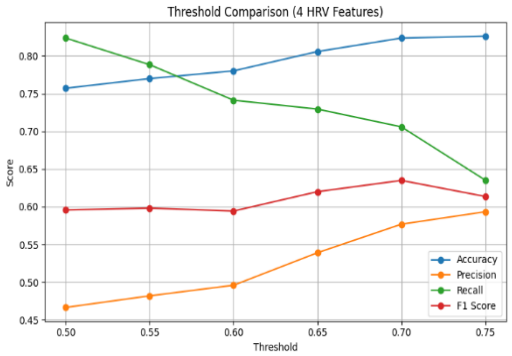

Figure 7. Comparison of accuracy, precision, recall, and F1-score values at various thresholds

As shown in Fig. 7, increasing the threshold generally improves precision while reducing recall, indicating a more conservative prediction of the stress class. From a machine learning perspective, the decision threshold controls the precision–recall trade-off by determining the confidence required to assign the stress label. A threshold of **0.70** achieved the highest F1-score, indicating the best balance between false positive and false negative predictions. Since the F1-score jointly considers precision and recall, this threshold provides the most appropriate operating point for stress classification and was therefore selected for TinyML deployment instead of relying solely on the highest accuracy. The classification performance obtained using the selected threshold of 0.70 is summarized in Table 5.

Table 5. Classification Report Results at Threshold 0.70

	precision	recall	f1-Score	support
Relax	0.91	0.86	0.88	306
Stress	0.58	0.71	0.63	85
Accuracy			0.82	391
Macro avg	0.74	0.78	0.76	391
Weighted avg	0.84	0.82	0.83	391

The proposed model achieved an overall accuracy of 0.82, with a macro-average F1-score of 0.76 and a weighted-average F1-score of 0.83. The stress class obtained a precision of 0.58, recall of 0.71, and F1-score of 0.63, whereas the relax class achieved higher classification performance. The relatively lower precision of the stress class can be attributed to three factors. First, the extracted HRV dataset is imbalanced (910 relax vs. 257 stress samples), causing the classifier to be biased toward the majority class and increasing false positive predictions for the stress class. Second, the four selected HRV features (BPM, SDNN, RMSSD, and pNN50) provide limited discriminative capability because physiological responses to stress partially overlap with normal conditions across individuals. Third, the subject-wise split evaluation introduces substantial inter-subject physiological variability by testing the model on subjects unseen during training, making the classification task more challenging. Despite these challenges, the proposed four-feature HRV-based TinyML model demonstrates satisfactory performance while maintaining a lightweight architecture suitable for real-time deployment on ESP32.

3.2. Testing the Model Implementation on the ESP32

The TinyML implementation on the ESP32 was evaluated to verify that the quantized TensorFlow Lite INT8 model produced the same classification results as the reference implementation in Python. Twenty test samples from Subjects S13–S17 of the WESAD dataset were used, with each sample consisting of four HRV features (BPM, SDNN, RMSSD, and pNN50). To validate only the TinyML implementation, the input features were directly provided to the ESP32 without involving signal acquisition or feature extraction. The prediction results obtained from Python and the ESP32 are compared in Table 6.

p-ISSN :1858-3911 **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

Table 6. Comparison of Inference Results from Python and ESP32 on Test Data for Subjects S13–S17

No	Label Dataset	Probability		Prediction
		Python	ESP32	
1	Stress	0.4687	0.4687	Relax
2	Stress	0.2265	0.2265	Relax
3	Stress	0.9179	0.9179	Stress
4	Stress	0.8632	0.8515	Stress
5	Stress	0.5507	0.5507	Relax
6	Stress	0.9648	0.9648	Stress
7	Stress	0.8945	0.8945	Stress
8	Stress	0.9023	0.9023	Stress
9	Stress	0.7460	0.7460	Stress
10	Stress	0.8515	0.8515	Stress
11	Relax	0.1289	0.1289	Relax
12	Relax	0.0195	0.0195	Relax
13	Relax	0.0898	0.0898	Relax
14	Relax	0.0156	0.0156	Relax
15	Relax	0.4296	0.4296	Relax
16	Relax	0.1250	0.1250	Relax
17	Relax	0.0429	0.0429	Relax
18	Relax	0.2109	0.2109	Relax
19	Relax	0.0078	0.0078	Relax
20	Relax	0.2382	0.2382	Relax

The ESP32 produced prediction results identical to those obtained in Python, with only a minor probability difference (0.0117) observed in one sample due to INT8 quantization, which did not affect the final classification. Consequently, the TinyML implementation achieved 100% prediction compatibility with the Python implementation and 85% classification accuracy on the evaluated test samples, confirming the correctness of feature normalization, quantization, inference, and dequantization on the ESP32.

To further evaluate the feasibility of deployment, the TinyML implementation was assessed using model size, Flash memory usage, runtime memory consumption, inference latency, and execution time. The deployment metrics, averaged over 20 repeated measurements after signal stabilization, are summarized in Table 7.

Table 7. Deployment Performance of the Proposed TinyML Model on ESP32

Parameter	Value
Model Architecture	4–16–8–4–1
INT8 Model Size	4.50 KB
Tensor Arena	20 KB
Flash Memory Usage	386.48 KB (30%)
Free Heap Memory	295,536 Bytes
Minimum Free Heap	289,848 Bytes
Average Inference Latency	0.194 ms*
Average Execution Time	12.077 ms*
Prediction Compatibility (Python vs ESP32)	100%

*Average of 20 repeated measurements after signal stabilization.

As shown in Table 7, the proposed TinyML model requires only 4.50 KB of INT8 model storage and a 20 KB Tensor Arena, while the complete application occupies 386.48 KB (30%) of the available Flash memory. The ESP32 also maintains 295,536 bytes of free heap memory during execution. Furthermore, the average inference latency and execution time are 0.194 ms and 12.077 ms, respectively, demonstrating that the proposed TinyML model satisfies the memory and computational constraints of the ESP32 while maintaining 100% prediction compatibility with the Python implementation.

3.3. MAX30102 Sensor Evaluation

This section evaluates the performance of the MAX30102 sensor as the physiological signal acquisition device used in the proposed TinyML-based stress detection system. The evaluation includes BPM measurement accuracy using a reference pulse oximeter and validation of the HRV features extracted on the ESP32. These experiments verify that the sensor and the implemented feature extraction process provide reliable inputs for the TinyML classifier. In this study, the MAX30102 sensor serves to acquire PPG signals, which are then processed to obtain the parameters BPM, SDNN, RMSSD, and pNN50. These four parameters are used as input features in a TinyML-based stress classification model

3.3.1. BPM Testing of a Pulse Oximeter

To evaluate the heart rate measurement accuracy of the MAX30102 sensor, its BPM readings were compared with those obtained from an Omicron Fingertip Pulse Oximeter (Model LK88), which served as the reference device. The reference pulse oximeter measures heart rate within a range of 25–250 BPM and peripheral oxygen saturation (SpO_2: 70–100%) using photoplethysmography (PPG). During testing, the MAX30102 sensor and the pulse oximeter were placed on different fingers of the same participant and operated simultaneously

p-ISSN :1858-3911* , *e-ISSN : 2614-5405
https://journal.fkom.uniku.ac.id/ilkom

under identical conditions. The participant remained seated and at rest, while measurements were recorded only after stable BPM readings were displayed by both devices.

Ten paired measurements were collected by simultaneously recording the BPM values from the MAX30102 sensor and the reference pulse oximeter. The percentage error between the two devices was then calculated to evaluate the heart rate measurement accuracy of the MAX30102 and to assess its suitability as the physiological signal acquisition device for the proposed TinyML-based stress detection system. The comparison results are presented in Table 8.

Table 8. MAX30102 Sensor Accuracy Test Results

No	BPM		Error (%)
	Pulse Oximeter	MAX30102	
1	87	86	1.15
2	85	85	0
3	82	81	1.22
4	91	93	2.20
5	106	104	1.89
6	97	97	0
7	74	75	1.35
8	83	82	1.20
9	83	81	2.41
10	85	86	1.18

The BPM measurements in Table 8 were used to determine the mean percentage error and the corresponding measurement accuracy using (6) and (7), respectively.

$$Average\ Error = \frac{\sum_{i=1}^{n} Error_i}{n}\ x\ 100\% \qquad (6)$$

$$Accuracy\ =\ 100\% - Average\ Error \qquad (7)$$

The calculated results indicate a mean percentage error of 1.26% and an average accuracy of 98.7%, demonstrating that the MAX30102 provides reliable heart rate measurements for the proposed stress detection system.

3.3.2. Verification of HRV Calculations

HRV calculation verification was performed to ensure that the feature extraction algorithm implemented on the ESP32 microcontroller produced outputs consistent with the results from the reference device. The verification process involved comparing the HRV feature values generated by the ESP32 with the results calculated using a Python program that implemented identical algorithms and mathematical equations. The input data on both platforms consists of RR intervals obtained from the heart rate detection process using the MAX30102 sensor.

In the Python program, the BPM value is calculated based on RR interval data using the same method as the implementation on the ESP32, namely through the calculation of instantaneous BPM, which is then averaged using a buffer that holds a maximum of the last 15 data points. Subsequently, the SDNN, RMSSD, and pNN50 features are calculated directly from the RR interval data based on equations (1) through (5).

Verification was performed for ten experiments by comparing the BPM, SDNN, RMSSD, and pNN50 values generated by both platforms. The results of this comparison were used to evaluate the suitability of the HRV feature extraction algorithm implementation on the ESP32 and are presented in Table 9

Table 9. Comparison of Python and ESP32 Calculation Results

No	BPM		SDNN		RMSSD		pNN50	
	Py	Esp	Py	Esp	Py	Esp	Py	Esp
1	86	86	0.07	0.07	0.122	0.122	57.58	57,58
2	85	85	0.11	0.11	0.159	0.159	71.43	71,43
3	81	81	0.14	0.14	0.228	0.228	100.0	100,0
4	93	93	0.04	0.04	0.071	0.071	44.44	44,44
5	104	104	0.06	0.06	0.102	0.102	55.56	55,56
6	97	97	0.14	0.14	0.221	0.221	66.67	66,67
7	75	75	0.08	0.08	0.111	0.111	44.44	44,44
8	82	82	0.14	0.14	0.229	0.229	77.78	77,78
9	81	81	0.09	0.09	0.157	0.157	63.16	63,16
10	86	86	0.22	0.22	0.234	0.234	66.67	66,67

As shown in Table 9, all HRV features calculated using Python yielded values identical to those produced by the ESP32. The only differences observed were in the last decimal place, due to the rounding of floating-point numbers

3.4. System Functional Testing

Functional testing was conducted to verify the correct operation of the complete TinyML-based stress detection system, including finger detection, PPG signal acquisition, HRV feature extraction, TinyML inference, buzzer notification, and OLED visualization. This evaluation focused on verifying the correctness of the ESP32 implementation under controlled resting conditions rather than assessing stress detection performance under different user activities or physiological conditions.

Figure 8 illustrates the operational sequence and output of the proposed TinyML-based stress detection system during functional testing. After sufficient RR intervals are collected, the

p-ISSN :1858-3911 **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

extracted HRV features are processed by the TinyML model on the ESP32. The classification result, predicted probability, and BPM value are then displayed on the OLED, while a double buzzer notification indicates that the inference process has been completed.

Figure 8. System Functional Test Results

The functionality of each system component was subsequently verified through a series of functional tests. The results for all evaluated functions are summarized in Table 10.

Table 10. System Functional Test Results

No	Test Scenario	Expected Result	Test Result	Status
1	The system is activated without touching the sensor	The OLED displays instructions for placing your finger on the sensor	The OLED displays the message "PLACE FINGER ON SENSOR"	Successful
2	Place your finger on the MAX30102 sensor	The system detects the presence of a finger, and the buzzer beeps twice to signal the start of the measurement The	The finger was successfully detected and the buzzer beeped twice	Successful
3	The sensor collects the necessary heart rate data	System performs HRV feature extraction	The HRV feature has been successfully calculated	Successful
4	The HRV data is suitable for inference	The TinyML model classifies user conditions	The inference process is working properly	Successful
5	The classification process is complete	The buzzer beeps twice to indicate that the measurement process is complete	The buzzer beeps twice	Successful
6	The inference has been completed	The OLED displays the classification results, probabilities, and BPM	The OLED displays the STRES/RELAX label, the probability value, and the BPM value	Successful

As summarized in Table 10, all functional tests were successfully completed, confirming the correct operation of the complete processing pipeline, from finger detection and PPG signal acquisition to HRV feature extraction, TinyML inference, buzzer notification, and OLED visualization. These results demonstrate that the proposed ESP32 implementation correctly reproduces the intended system functionality in real time.

It should be noted that the functional evaluation was conducted under controlled resting conditions because the primary objective of this study was to verify the correctness of the TinyML implementation on the ESP32. Therefore, the evaluation focused on implementation correctness rather than robustness under different user activities, such as walking, exercise, or daily movements. Assessing system robustness under dynamic conditions is beyond the scope of this study and will be considered in future work.

4. CONCLUSION

This study proposed a TinyML-based stress detection framework integrating the MAX30102 sensor, four time-domain HRV features (BPM, SDNN, RMSSD, and pNN50), a lightweight DNN classifier, and ESP32-based deployment for real-time stress classification. The primary contribution of this work is the development of an end-to-end TinyML framework that performs physiological signal acquisition, HRV feature extraction, model quantization, and on-device inference within the limited memory and computational resources of an ESP32 microcontroller.

Experimental results demonstrated that the proposed DNN achieved an accuracy of 82%, while the TensorFlow Lite INT8 model maintained 100% prediction compatibility with the Python implementation on the evaluated test samples. Furthermore, the quantized model required only 4.50 KB of storage and a 20 KB Tensor Arena, confirming its feasibility for deployment on resource-constrained embedded platforms. The MAX30102 sensor also achieved a BPM measurement accuracy of 98.74%, and the HRV feature extraction implemented on the ESP32 produced results consistent with the reference calculations, demonstrating the

p-ISSN :1858-3911 **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

reliability of the proposed embedded implementation.

Future work will investigate the integration of additional HRV features from the frequency and nonlinear domains to improve classification performance. The proposed framework will also be evaluated using larger and more diverse datasets to enhance model generalizability. In addition, further optimization of memory utilization and energy consumption will be explored to support long-term wearable stress monitoring applications on low-power edge devices.

5. ACKNOWLEDGMENT

The authors would also like to acknowledge the support of the SPCIES (Strategic Pervasive Computing and Intelligent Embedded System) research group

REFERNCES

[1] Y. Segono, E. Y. M. Hutagalung, H. G. Simbolon, N. I. Us, and A. Ridwan, "IoT-Based Stress Monitoring Using CNN for HRV-GSR Analysis," *Sinkron*, vol. 10, no. 1, pp. 621–637, 2026, doi: 10.33395/sinkron.v10i1.15671.

[2] A. Abu-Samah *et al.*, "Deployment of TinyML-Based Stress Classification Using Computational Constrained Health Wearable," *Electron.*, vol. 14, no. 4, p. 687, Feb. 2025, doi: 10.3390/electronics14040687.

[3] P. Ganesan, Y. R. Thota, H. Shehata, and T. Nikoubin, *TinyML Based Stress Detection utilizing PPG Signals: A Lightweight Approach for Smart Wearable Devices*, vol. 1, no. 1. Association for Computing Machinery, 2025. doi: 10.1145/3716368.3735274.

[4] A. Calvo, J. Martin, and C. Martin, "Early Detection of Chronic Stress Using Wearable Devices: A Machine Learning Approach with the WESAD Database," in *International Conference on Information and Communication Technologies for Ageing Well and e-Health, ICT4AWE - Proceedings*, 2025, pp. 189–196. doi: 10.5220/0013209700003938.

[5] G. Tyulepberdinova, M. Kunelbayev, G. Amirkhanova, M. Tokhtassyn, and A. Amirkhanov, "Development of a Stress Monitoring System Architecture for Heart Rate Measurement," *Eng. Technol. Appl. Sci. Res.*, vol. 15, no. 6, pp. 29551–29565, 2025, doi: 10.48084/etasr.13150.

[6] R. Al Abdi, S. AlKaabi, S. Elsifi, and J. Yousaf, "Mental Stress Detection Using Physiological Sensors and Artificial Intelligence: A Review," *Sensors*, vol. 26, no. 5, pp. 1–31, 2026, doi: 10.3390/s26051616.

[7] A. Pinge, V. Gad, D. Jaisighani, S. Ghosh, and S. Sen, "Detection and monitoring of stress using wearables: a systematic review," *Front. Comput. Sci.*, vol. 6, no. c, Dec. 2024, doi: 10.3389/fcomp.2024.1478851.

[8] K. M. Dalmeida and G. L. Masala, "HRV Features as Viable Physiological Markers for Stress Detection Using Wearable Devices," *Sensors*, vol. 21, no. 8, p. 2873, Apr. 2021, doi: 10.3390/s21082873.

[9] Wadeea Alnufaily, Ramasamy Srinivasagan, Purushothaman R, and Mohammed Alnaeem, "Stress Detection from Photoplethysmography Signals Using Multi-Domain Heart Rate Variability Analysis," *J. Comput. Biomed. Informatics*, vol. 10, no. 02, Mar. 2026, doi: 10.56979/1002/2026/1234.

[10] Y. Haque *et al.*, "State-of-the-Art of Stress Prediction from Heart Rate Variability Using Artificial Intelligence," *Cognit. Comput.*, vol. 16, no. 2, pp. 455–481, Mar. 2024, doi: 10.1007/s12559-023-10200-0.

[11] M. Bahameish, T. Stockman, and J. Requena Carrión, "Strategies for Reliable Stress Recognition: A Machine Learning Approach Using Heart Rate Variability Features," *Sensors*, vol. 24, no. 10, pp. 1–21, 2024, doi: 10.3390/s24103210.

[12] Y. Y. Tsai, Y. J. Chen, Y. F. Lin, F. C. Hsiao, C. H. Hsu, and L. De Liao, "Photoplethysmography-based HRV analysis and machine learning for real-time stress quantification in mental health applications," *APL Bioeng.*, vol.

p-ISSN :1858-3911 **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

9, no. 2, pp. 1–22, 2025, doi: 10.1063/5.0256590.

[13] J. A. Mortensen, M. E. Mollov, A. Chatterjee, D. Ghose, and F. Y. Li, "Multi-Class Stress Detection Through Heart Rate Variability: A Deep Neural Network Based Study," *IEEE Access*, vol. 11, no. May, pp. 57470–57480, 2023, doi: 10.1109/ACCESS.2023.3274478.

[14] S. Heydari and Q. H. Mahmoud, "Tiny Machine Learning and On-Device Inference: A Survey of Applications, Challenges, and Future Directions," *Sensors*, vol. 25, no. 10, 2025, doi: 10.3390/s25103191.

[15] T. F. of the E. S. of C. the N. A. S. of P. Electrophysiology, "Heart rate variability: standards of measurement, physiological interpretation, and clinical use," *Circulation*, vol. 93, no. 5, pp. 1043–1065, 1996.

[16] F. Shaffer and J. P. Ginsberg, "An Overview of Heart Rate Variability Metrics and Norms," *Front. Public Heal.*, vol. 5, no. September, pp. 1–17, 2017, doi: 10.3389/fpubh.2017.00258.

[17] B. Kołakowski, P. Noga, P. Mańczak, and M. Nowak, "Comparative analysis of methods for calculating HRV values on heart rate monitoring devices," *TASK Q.*, vol. 29, no. 2, 2025.

[18] M. M. Taye, "Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions," *Computers*, vol. 12, no. 5, p. 91, Apr. 2023, doi: 10.3390/computers12050091.

[19] I. H. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," *SN Comput. Sci.*, vol. 2, no. 6, pp. 1–20, 2021, doi: 10.1007/s42979-021-00815-1.

[20] R. Immonen and T. Hämäläinen, "Tiny Machine Learning for Resource-Constrained Microcontrollers," *J. Sensors*, vol. 2022, pp. 1–11, Nov. 2022, doi: 10.1155/2022/7437023.

[21] Schmidt, P., Reiss, A., Duerichen, R., Marberger, C., & Van Laerhoven, K. (2018, October). Introducing wesad, a multimodal dataset for wearable stress and affect detection. In Proceedings of the 20th ACM international conference on multimodal interaction (pp. 400-408).

[22] Van Gent, P., Farah, H., Van Nes, N., & Van Arem, B. (2019). HeartPy: A novel heart rate algorithm for the analysis of noisy signals. Transportation research part F: traffic psychology and behaviour, 66, 368-378.