AIoT-Driven Human Activity Recognition for Versatile Framework on Multipurpose Applications

Nanik Triwahyuni[1], Eni Dwi Wardihani[2], Aminuddin Rizal[3], Samuel Beta K.[4], Ricky Sambora[5], Rindang Ayu Oktaviani[6]

[1,2,5,6]Program Studi Teknik Telekomunikasi,
Jurusan Teknik Elektro,
Politeknik Negeri Semarang, Semarang
[1]naniktriw@gmail.com, [2]edwardihani@polines.ac.id, [5]ricky.43120121@mhs.polines.ac.id,
[6]rindang.43120022@mhs.polines.ac.id

[3,4]Program Studi Teknik Elektronika,
Jurusan Teknik Elektro,
Politeknik Negeri Semarang, Semarang
[3]aminuddin.rizal@polines.ac.id, [4]sambetak2@polines.ac.id

Abstract

This paper presents the implementation of an Artificial Intelligence-Internet of Things (AIoT) based Human Activity Recognition (HAR) framework designed for multipurpose applications. The framework integrates sensor data from wearable IoT devices, which is then processed by AI algorithms to classify and predict human activities in real-time. It utilizes machine learning models, particularly machine learning techniques, to analyze complex activity. Our framework uses all open-source development environment, which is able to replicate and modify. For this paper, we give examples of how to use our framework as gamification of a workout practice. The Idea is to recognize user activity including (bicep curl, shoulder press, and front rise) using accelerometer data, and then send recognized activity to our developed online game (pop-balloon). M5StickC Plus used as the hardware which already equipped with inertial measurement unit (IMU) sensor and power management. Furthermore, it has small form factor fit for wearable application. Before real-time performance was taken, we evaluated 5 different machine learning model and choose which one is more optimized. The five models include Naïve Bayes, Support Vector Machine (SVM), AdaBoost, ZeroR, and Random Forest. Accuracy given for offline analysis were 97.68%, 98.97%, 41.62%, 25%, 100% respectively to the previous model. In the final, for real-time performance we choose SVM model which most optimized even though the accuracy reduced to 89.67% for this task.

Keywords: artificial intelligence internet of things, embedded device, human activity recognition, signal processing

1. Introduction

The merging of Artificial Intelligence (AI) with the Internet of Things (IoT) has given rise to a transformative paradigm known as AIoT (Artificial Intelligence of Things) [1],[2]. This integration merges the pervasive connectivity of IoT devices with the intelligent processing capabilities of AI, resulting in highly autonomous, efficient, and responsive

Received: 20 Agustus 2024 Revised: 23 September 2024 Accepted: 23 September 2024

systems. One of the most promising applications of AIoT is Human Activity Recognition (HAR), a technology that leverages data from various sensors to identify and interpret human activities in real-time.

Human Activity Recognition has gained significant attention due to its wide array of applications across different domains such as healthcare, smart homes, security, sports, and industrial environments [3]-[7]. By deploying AIoT-driven HAR systems, it is possible to create versatile frameworks capable of addressing multiple use cases, thus offering multipurpose solutions that can adapt to varying requirements and contexts [8]. Human Activity Recognition is not a new concept; its roots can be traced back to the early development of pattern recognition and machine learning techniques. Traditional HAR systems often rely on rule-based approaches and statistical methods to interpret data from sensors. However, these methods had limitations in handling complex, non-linear, and context-dependent activities [9].

With the advent of machine learning and deep learning, HAR has experienced significant advancements. Modern HAR systems utilize a wide range of data sources, including wearable sensors, cameras, and ambient sensors, and apply sophisticated algorithms to accurately recognize and classify human activities [10]. The integration of AI into IoT has further enhanced HAR by enabling real-time processing, reducing latency, and improving the scalability of these systems. The fusion of AI and IoT in HAR systems brings about several key benefits including real-time processing and decision-making [11], scalability and flexibility [12], contextual awareness [13], and energy efficiency [14].

More focused on HAR using an accelerometer sensor as a data source offers significant advantages over camera-based methods, particularly in terms of privacy, cost-effectiveness, and energy efficiency [15]. Accelerometers can be attached to wearable devices, allowing for continuous monitoring without compromising user privacy, as they do not capture visual data. They are less dependent on environmental factors such as lighting or weather, and their lower power consumption makes them ideal for long-term use. Moreover, accelerometer data is less complex, enabling real-time processing and greater scalability across multiple users. In contrast, camera-based HAR systems are more intrusive, require more computational resources, and are often less practical for mobile or subtle activity monitoring. These factors make accelerometer-based HAR a more versatile and practical choice for many applications.

Despite the potential benefits about using wearable accelerometer and AIoT algorithm for HAR, several challenges must be addressed including issues with data quality due to noise and variability across users, as well as the need for efficient power management to preserve battery life [16]. Real-time processing is crucial but challenging on resource-constrained devices, especially when balancing latency and computational demands. Privacy and security concerns arise due to the sensitive nature of the data collected, and compliance with regulations adds complexity. The complexity of AI algorithms, particularly in adapting to changes in user behavior, can strain the limited processing power of wearable devices [17]. Accurate activity recognition often requires context awareness, which is difficult with accelerometer data alone, and integrating data from multiple sensors increases system complexity. Deployment at scale poses challenges in managing large data volumes, ensuring interoperability, and maintaining performance across diverse devices. Additionally, user acceptance depends on the comfort and unobtrusiveness of wearable devices, as well as trust in the system's reliability and data handling practices.

By knowing the importance of human activity recognition (HAR) in many area and how AIoT enhance capabilities of the HAR performance. To deal with the challenge of AIoT algorithm development, we propose an end-to-end framework to develop AIoT for multipurpose HAR applications. The framework includes data collection, model training, model evaluation, and real-time inference. Tools and IDEs used on this framework are open source so for anyone who wants to replicate is possible.

As an example of the usage of our framework we perform gamification for a workout practice to engage the subject do activity they are asked to. There are 3 activities that need to be classified including bicep curl, shoulder press, and front rise. Stick5C Plus 2 used as the hardware, while Weka and processing IDE mainly used as the software. We conduct experiment to collect and evaluate our framework for this application. In the final, it can be used for playing a game from our developed online game.

The future of our framework can be applied to many applications such as agriculture area or even industrial area. The framework is easy to use and configure. It also scalable for low into high complexity of application.

2. Method

Our proposed framework workflow is shown in Figure 1. There are 4 steps in total namely hardware design, data collection, training data, and real-time inference, the detail of each step as follows:

1. Hardware design: our framework does not see the hardware as specific; we allow the concept of virtualization. Wherever and whenever the hardware can send the data needed for the application everything is fine. However, it is better to develop the hardware related to application need. We suggest to use Easyeda [18] tool which handy and open source simulate electronic circuits simulation and create printed circuit board (PCB) layouts. And for the firmware we recommend to use Arduino IDE [19] which is an open-source software platform used for writing, compiling, and uploading code to the hardware. And for the hardware functionalities itself mainly to read the sensor, do pre-processing if needed and send it to the host PC which running the data collection program. Communication protocol may be various such as Wi-Fi (UDP/TCP), Bluetooth, Zigbee, LoRa, NB-IoT or many more.
2. Data collection: we wrote a program for this task by using Processing IDE [20] which is an open-source integrated development environment designed for the user interface. This program collect data from the user hardware and handling it relevant to the communication protocol and format (data separation). And then, one most important function is to calculate the feature to feed to the machine learning program. There are several features can be use, from time-domain type or frequency domain-type [21]. After calculation, the program will store and formatting the feature bank (dataset) in *.arff* format.
3. Training data: in this stage we use Weka machine learning tool [22], the process is streamlined, making Weka a user-friendly tool for both beginners and experienced machine learning practitioners. Building a model in Weka involves a few straightforward steps. First, we need to load our dataset into Weka, which *.arff* file we built before. Once the data is loaded, we can explore and preprocess it by applying filters to clean, normalize, or discretize the data. After preprocessing,

navigate to the *"Classify"* tab, where we can choose a machine learning algorithm to build our model. Weka provides a wide range of algorithms, including decision trees, Naive Bayes, and support vector machines. Select the desired algorithm and configure any specific parameters. Next, we can perform a cross-validation or use a separate test set to evaluate our model's performance. Finally, once the model is trained, Weka allows us to visualize the results, such as confusion matrices or ROC curves, and save the model (the format is .model) for real-time inference relevant to the application we build.

4. Inference: we wrote another program also by using Processing to build our usage application. Any application can build depending on the need and purpose, such as in medical area (fall detection, medical rehabilitation), sport (to know correct movement on specific sport), smart home (to know the effectiveness of energy based on sensor), in industry (to know the effectiveness of the machinery), and many more. This program also handling the data from user hardware, and then calculate the same feature in data collection stage, and finally do the inference. And result from inference can be used for further function.

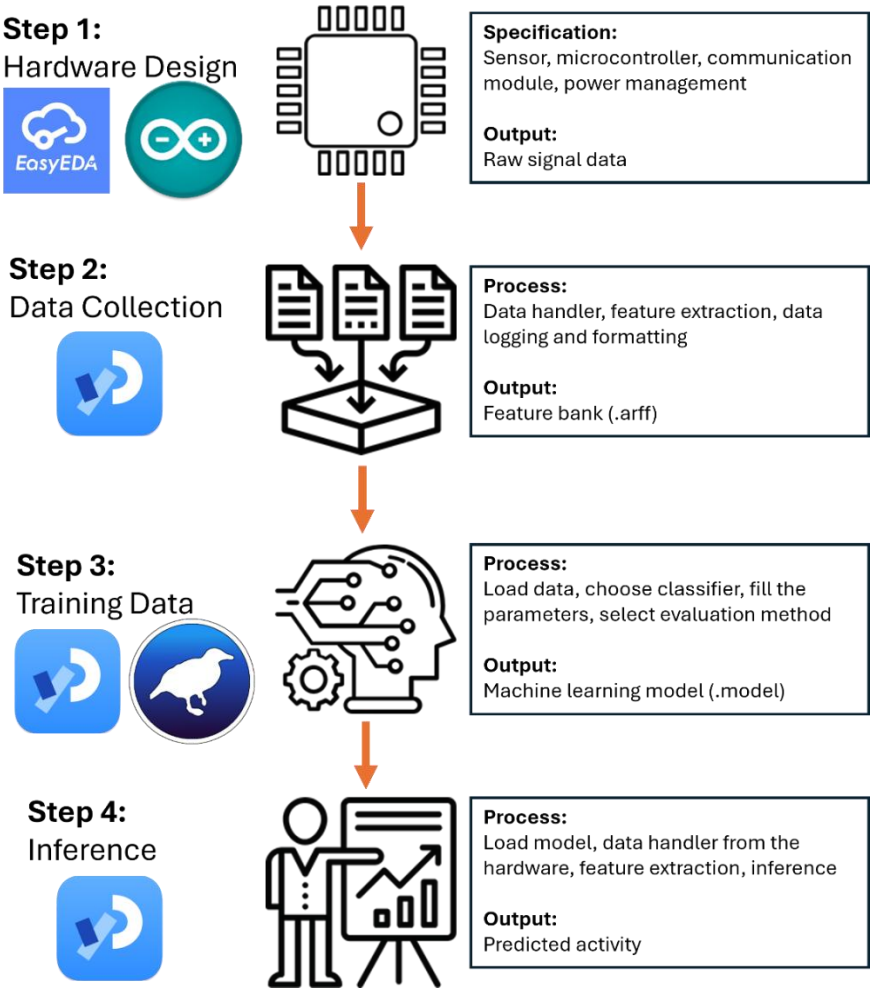


Figure 1. Framework concept

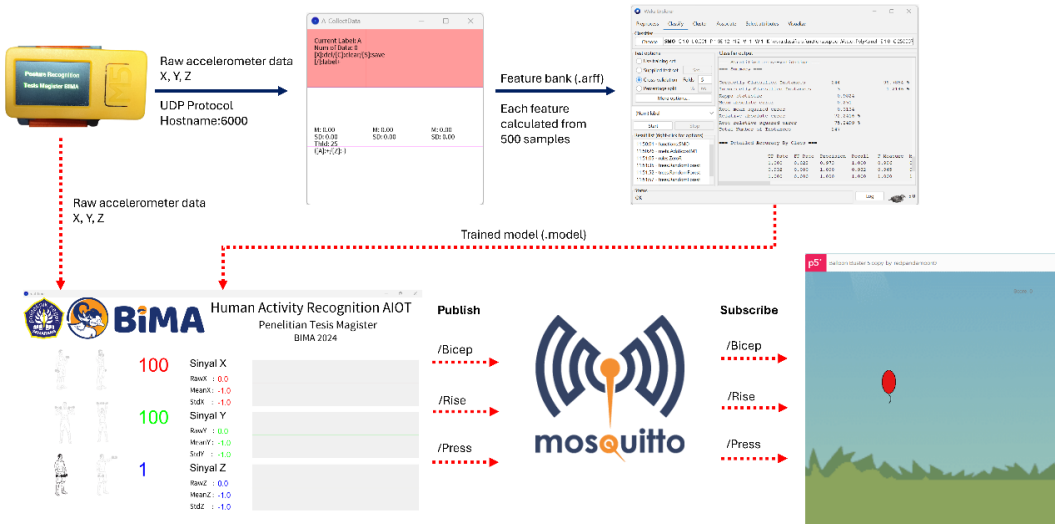


Figure 2. Overall system of workout practice gamification

The overall usage example of our framework is shown in Figure 2. The application recognizes 3 activities to pop the ballon from our developed online game to engage the user do the activity. Not only that, but we also show how many each activity done by the user in our Processing program. The development of this application follows all of our framework steps. From Figure 2, the blue arrow means mode creation, while the red arrow means real-time inference system flow. The details of our system are described below.

2.1. The Hardware

For this application, we opt to use a development IoT product which is M5StickC Plus 2. The board itself is a compact and portable development board, ideal for wearable tech, portable gadgets, and IoT devices. It features a 1.14-inch color TFT display, offering better readability. Powered by an ESP32 microcontroller, it provides robust processing, Wi-Fi, and Bluetooth connectivity. With a built-in 120mAh LiPo battery and USB-C charging, it supports both portability and ease of use. The device includes multiple sensors, such as an IMU [23]. The board we use is shown in Figure 3.

Figure 3. Wearable device M5StickC Plus 2

Even though so many features offered by the development board, we focus on IMU sensors, power management, microcontrollers, and connectivity (Wi-Fi). What our hardware does is connect to the access point as the client using Wi-Fi, and then read accelerometer sensor data for all 3 axes (x,y,z) in 50 Hz sampling frequency, format the data before sending it to the host PC with format *("data_x, data_y, data_z")*, connect to host PC by using UDP protocol with address of the host PC and selected port 6000.

2.2. Data Collection

To collect the data, we use and modify the framework from Weka4P [24] especially for the data handle and feature calculation. 3 main functions of this program as,

1. Data handling: the program receives raw accelerometer data from the hardware with the same UDP protocol. Since the format is combination of the three axes, it needs to be parse into individual axis which are *data_x*, *data_y*, and *data_z*. Streamed data buffered into 500 samples for each axis.
2. Feature calculation: the buffered data then processed in 20 samples window to calculate a line of feature. There is total 6 features extracted from the signal of accelerometer data, those are mean axis x, standard deviation axis x, mean axis y, standard deviation axis y, mean axis z, standard deviation axis z. the calculation follows Equation (1) for the mean and Equation (2) for the standard deviation. Where axis is the axis of the accelerometer (x, y, z), sample [n] is sample in number n of 20 samples window.

$$mean_{axis} = \frac{\sum_{n=0}^{19} sample[n]}{20} \tag{1}$$

$$std_{axis} = \sqrt{\frac{\sum_{n=0}^{19} sample[n] - mean_{axis}}{20}} \tag{2}$$

3. Data formatting: after calculation of the features, then it formatted do can be easily use in machine learning tool. The data format simply follow: $mean_x$, std_x, $mean_y$, std_y, $mean_z$, std_z, *label*. Label is the activity want to predict with label A is represent idle bicep, B is bicep activity, C is idle rise, D is rise activity, E is idle press, F is press activity.

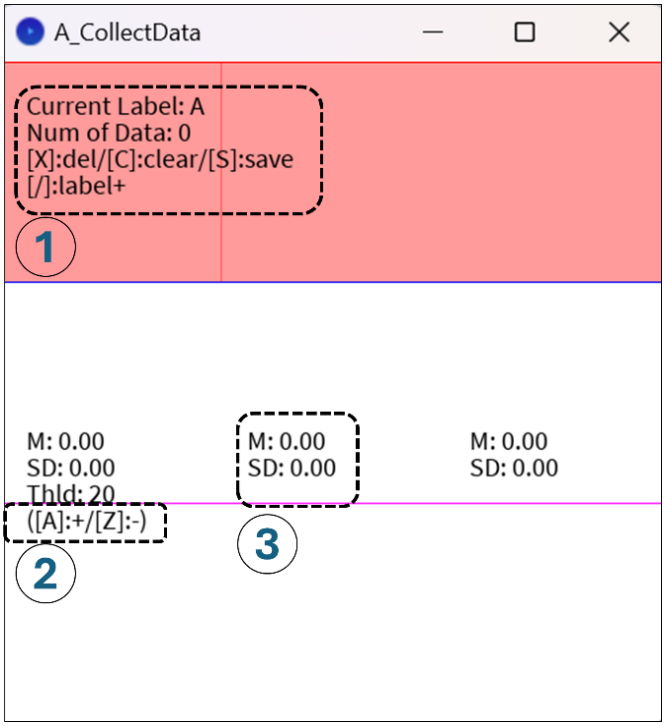
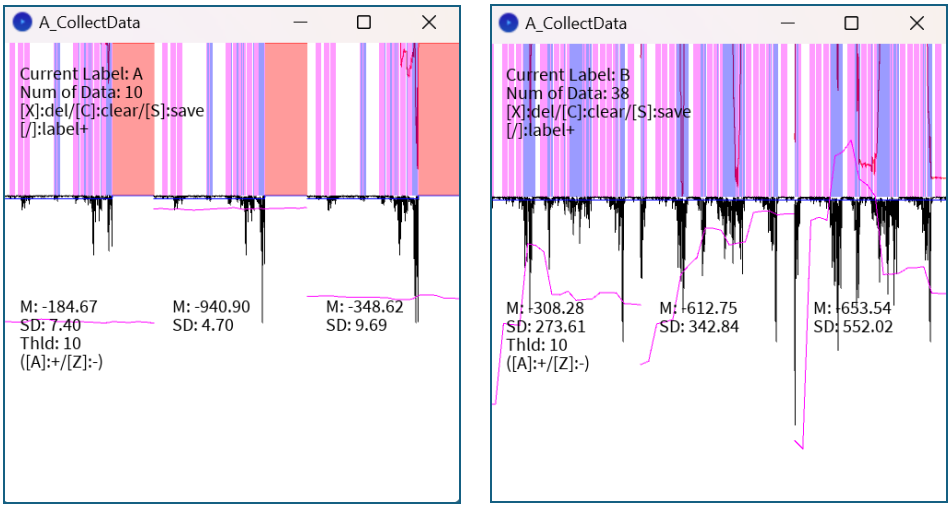


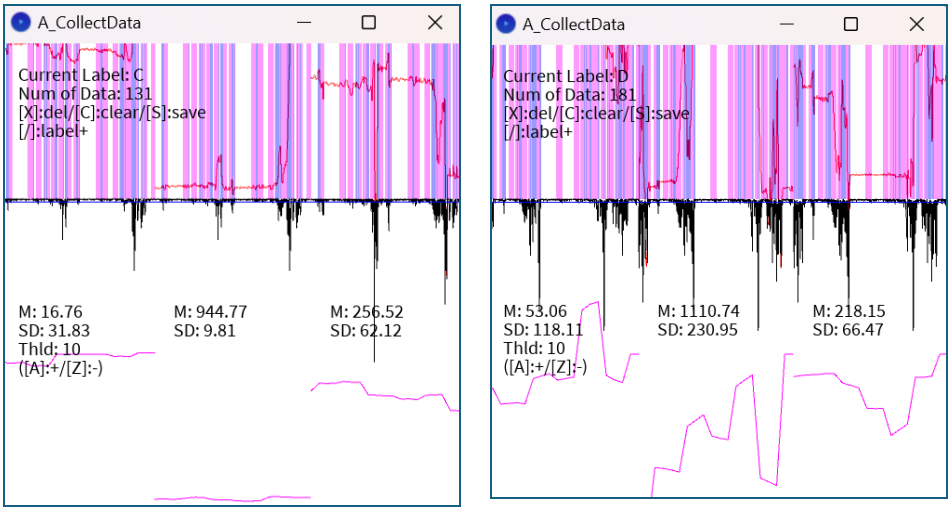
Figure 4. Data collection program

The program interface is shown on Figure 4, there are several components to show informative data. In the figure red area used to show the 500 sampled signal while the white area is used to show 20 samples of data. The dashed rectangle 1 are is to show the current label we collect relevant to the activity data (A, B, C, D, E, or F), number of feature line also on the same area as well as control command to save, delete, clear the data and change the current label. Dashed rectangle area 2 is showing the automatic threshold to save the data, if the signal has value above of the threshold the features data automatically added to the file. Dashed rectangle area 3 is to display current feature (mean and standard deviation of a time window). The example collection of data related to the activity shown in Figure 5, Figure 6, and Figure 7 for Bicep activity (label A, B), Rise activity (label C, D) and Rise activity (label E, F) respectively. The output of this program is a dataset for machine learning tool to build and evaluate the model, the file has extension *.arff*.



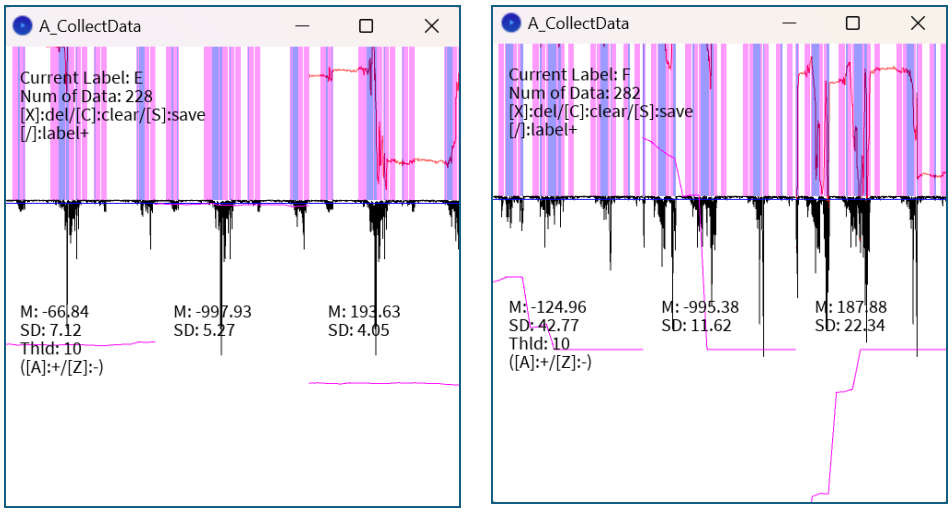
(a) (b)

Figure 5. (a) Signal of label A (idle bicep); (b) Signal of label B (bicep activity)



(a) (b)

Figure 6 (a) Signal of label C (idle rise); (b) Signal of label D (rise activity)



(a) (b)

Figure 7 (a) Signal of label E (idle press); (b) Signal of label F (press activity)

2.3. Model Training

After the dataset was collected, it underwent a comprehensive pre-processing phase to ensure its suitability for model training in Weka. The pre-processing involved several critical steps, including the handling of missing values through techniques such as imputation or exclusion, depending on the nature and extent of the missing data. Additionally, data normalization was performed to standardize the range of independent variables, which is crucial for algorithms sensitive to the scale of data, such as support vector machines. Feature selection was also carried out to identify the most relevant attributes, thereby reducing dimensionality and enhancing model performance. This process was guided by domain knowledge and statistical methods, ensuring that the retained features were both meaningful and influential in predicting the target variable. However, on this research all 6 features are used to feed to the model.

The Weka user interface to train the model shown in Figure 8. Dashed rectangle 1 is where we can choose the model we want to build, and also what parameters we want to configure that are suitable to our application. While the dashed rectangle 2 is for evaluation purposes, there are different options of the evaluation method, in our case we use K-fold cross validation because to mitigate the risk of overfitting and to ensure that the model's performance is not dependent on a particular subset of the data. The area on dashed rectangle 3 give us information about model's accuracy. Related to the accuracy, we can understand our model performance while predicting the activity more detail by looking at dashed rectangle 4. Last but not least important information is shown in dashed rectangle 5 which give us information how long time needed to train the model.

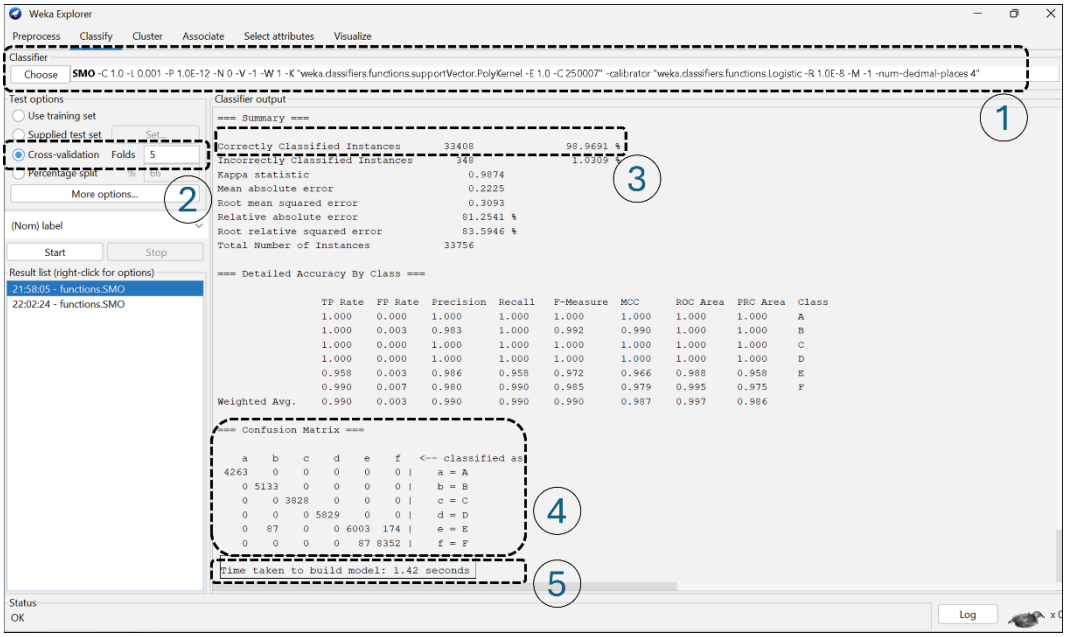


Figure 8. Training and evaluating the model

2.4 Real-time Inference and Application

After machine learning model trained, we can use the model for any several applications. As an example in our application, we built a workout practice with three different movement namely Bicep activity, Rise activity, and Rise activity. Application user interface was built also by using Processing IDE. The user interface can be seen in Figure 9. There is several information displayed on the user interface. the program communicates with the subject's hardware through the same protocol when we collect the data. Raw x, y, z signal from accelerometer data also buffered in the same 500 samples length as same as the feature calculation formed by 20 samples window. The value of raw signal and features are shown in dashed rectangles 3 on the Figure 9. After feature was feed to the model, the inference process starts to predict. The prediction result will be shown as the vector image related to the activity, for example in Figure 9 (dashed rectangle 1) vector image of label E (idle rise) is more contrast compared to the others it means the model predict label E. The unpredicted model has more transparency to differs with predicted one. Dashed rectangle 2 give the information how many time the subject do the activity relevant what they did and predicted by the model.

To engage the subject to do more workout activity we also develop online game modified from [25]. The online game communicates with our application program interface through MQTT Broker which we choose Mosquito. The application act as publisher and the game act as the subscriber. There are three topics made related to the activity, they are "/*Bicep*", "/*Rise*", and "/*Press*". When the application program predicts the activity, it will send "1" to the related topic. And this one value will pop the balloon following the rule in the Table 1. If the subject successfully pops the balloon, it will add to the score. The game screenshot shown in Figure 10, with dashed rectangle 1 is the score while dashed rectangle 2 is the example of blue balloon.

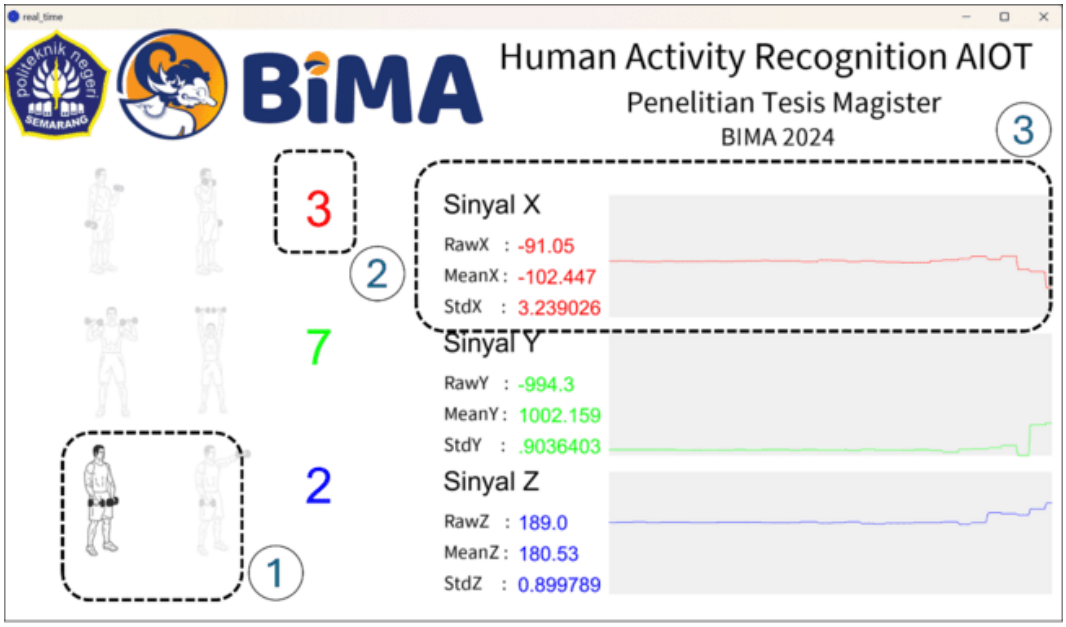


Figure 9. User interface workout practice

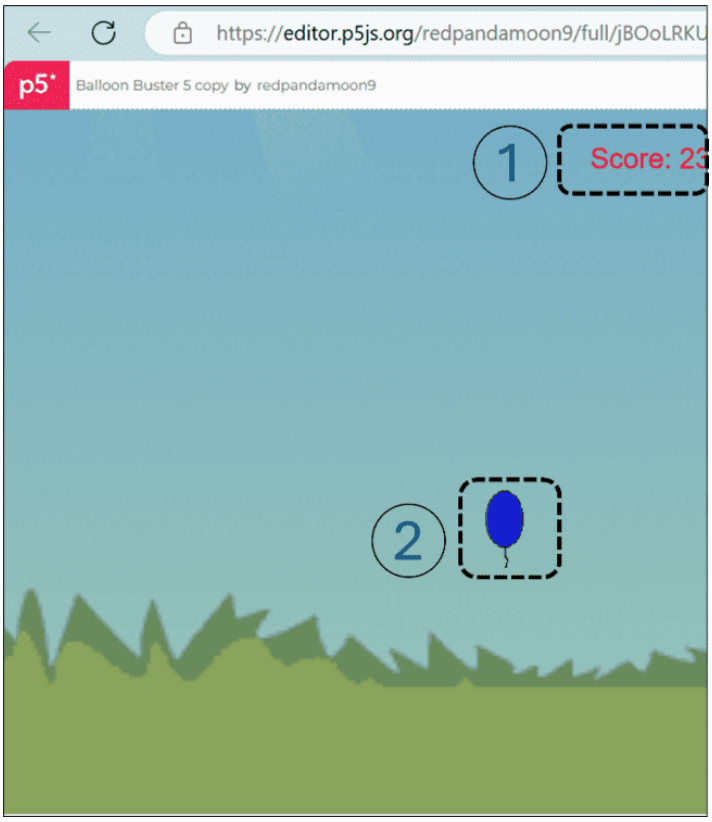


Figure 10. Online game pop the balloon

Table 1. Game rules

Label	Activity	Balloon Color
B	Bicep	Red
D	Rise	Green
F	Press	Blue

3. Experiment

We conducted an experiment to collect the data for the dataset and evaluate our model in both offline and real-time performance. A total 20 participants were involved in this experiment, with different genders (14 male and 6 female) and ages around 16 – 18 years old. All the participants were already well-informed about the experiment and research itself, and willingly agreed to join the experiment. The consent form was filled in by the participant as the proof of participant's understanding and following the ethics of the research.

Each participant did all the activities including idle Bicep (A), Bicep activity (B), idle Rise (C), Rise activity (D), idle Press (E), Press (F), and also playing the game as shown in Figure 11. Each activity is recorded as 5.5 s, so that if all activity is done 1650 samples (sampling rate is 50 Hz) are collected. Since there are 20 participants around 33,000 samples were collected for this experiment.

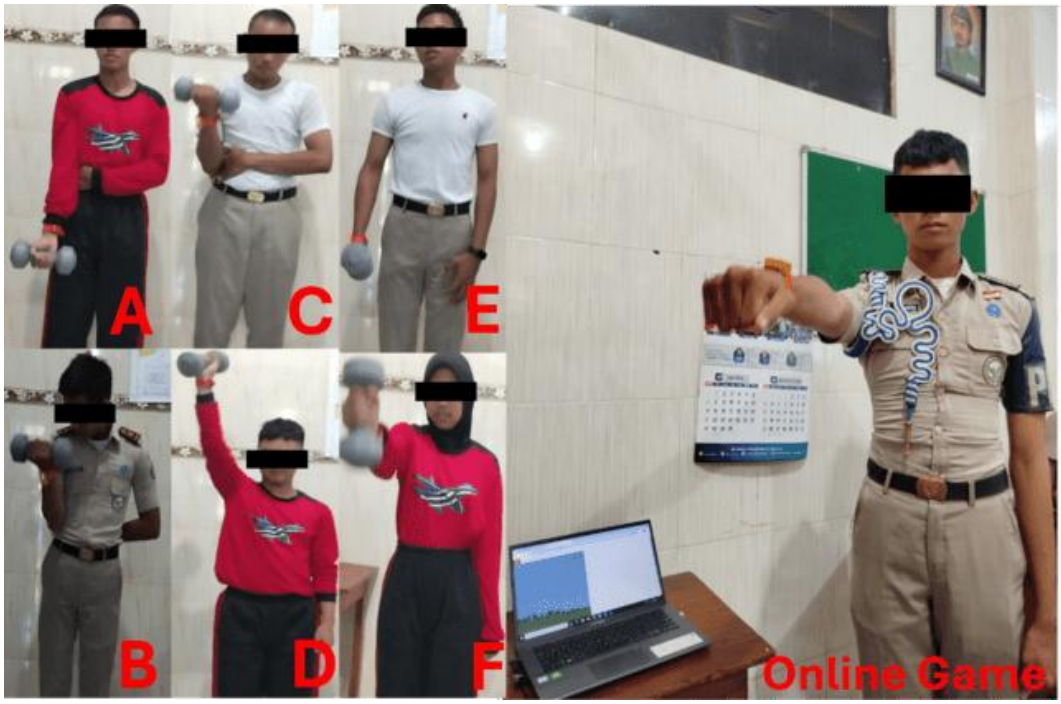


Figure 11. Conducted experiment

4. Results and Evaluation

After we get the data from the experiment then we can train and evaluate our model in Weka. Evaluation done in both offline and real-time performance (playing games). For offline performance we test it with 5-fold cross validation. Performance metrics we use is simple accuracy since it rule-of-thumb the performance of the model and confusion matrix. The formula of accuracy calculations follows Equation (3).

$$accuracy = \frac{Correct\ Prediction}{All\ Prediction} \tag{3}$$

ZeroR Model

Actual / Out	A	B	C	D	E	F	SUM
A	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	4263 12.63%	4263 0.00% 100.00%
B	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	5133 15.21%	5133 0.00% 100.00%
C	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	3828 11.34%	3828 0.00% 100.00%
D	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	5829 17.27%	5829 0.00% 100.00%
E	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	6264 18.56%	6264 0.00% 100.00%
F	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	8439 25.00%	8439 100.00% 0.00%
SUM	0 NaN% NaN%	0 NaN% NaN%	0 NaN% NaN%	0 NaN% NaN%	0 NaN% NaN%	33756 25.00% 75.00%	8439 / 33756 25.00% 75.00%

(a)

AdaBoost Model

Actual / Out	A	B	C	D	E	F	SUM
A	0 0.00%	0 0.00%	0 0.00%	4263 12.63%	0 0.00%	0 0.00%	4263 0.00% 100.00%
B	0 0.00%	0 0.00%	0 0.00%	4611 13.66%	0 0.00%	522 1.55%	5133 0.00% 100.00%
C	0 0.00%	0 0.00%	0 0.00%	3828 11.34%	0 0.00%	0 0.00%	3828 0.00% 100.00%
D	0 0.00%	0 0.00%	0 0.00%	5829 17.27%	0 0.00%	0 0.00%	5829 100.00% 0.00%
E	0 0.00%	0 0.00%	0 0.00%	174 0.52%	0 0.00%	6090 18.04%	6264 0.00% 100.00%
F	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	8439 25.00%	8439 100.00% 0.00%
SUM	0 NaN% NaN%	0 NaN% NaN%	0 NaN% NaN%	18705 31.16% 68.84%	0 NaN% NaN%	15051 56.07% 43.93%	14268 / 33756 42.27% 57.73%

(b)

Naive Bayes Model

Actual / Out	A	B	C	D	E	F	SUM
A	4263 12.63%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	4263 100.00% 0.00%
B	0 0.00%	4959 14.69%	0 0.00%	0 0.00%	174 0.52%	0 0.00%	5133 96.61% 3.39%
C	0 0.00%	0 0.00%	3741 11.08%	87 0.26%	0 0.00%	0 0.00%	3828 97.73% 2.27%
D	0 0.00%	0 0.00%	0 0.00%	5829 17.27%	0 0.00%	0 0.00%	5829 100.00% 0.00%
E	0 0.00%	87 0.26%	0 0.00%	0 0.00%	6003 17.78%	174 0.52%	6264 95.83% 4.17%
F	0 0.00%	0 0.00%	0 0.00%	0 0.00%	261 0.77%	8178 24.23%	8439 96.91% 3.09%
SUM	4263 100.00% 0.00%	5046 98.28% 1.72%	3741 100.00% 0.00%	5916 98.53% 1.47%	6438 93.24% 6.76%	8352 97.92% 2.08%	32973 / 33756 97.68% 2.32%

(c)

SVM Model

Actual / Out	A	B	C	D	E	F	SUM
A	4263 12.63%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	4263 100.00% 0.00%
B	0 0.00%	5133 15.21%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	5133 100.00% 0.00%
C	0 0.00%	0 0.00%	3828 11.34%	0 0.00%	0 0.00%	0 0.00%	3828 100.00% 0.00%
D	0 0.00%	0 0.00%	0 0.00%	5829 17.27%	0 0.00%	0 0.00%	5829 100.00% 0.00%
E	0 0.00%	87 0.26%	0 0.00%	0 0.00%	6003 17.78%	174 0.52%	6264 95.83% 4.17%
F	0 0.00%	0 0.00%	0 0.00%	0 0.00%	87 0.26%	8352 24.74%	8439 98.97% 1.03%
SUM	4263 100.00% 0.00%	5220 98.33% 1.67%	3828 100.00% 0.00%	5829 100.00% 0.00%	6090 98.57% 1.43%	8526 97.96% 2.04%	33408 / 33756 98.97% 1.03%

(d)

Random Forest Model

Actual / Out	A	B	C	D	E	F	SUM
A	4263 12.63%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	4263 100.00% 0.00%
B	0 0.00%	5133 15.21%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	5133 100.00% 0.00%
C	0 0.00%	0 0.00%	3828 11.34%	0 0.00%	0 0.00%	0 0.00%	3828 100.00% 0.00%
D	0 0.00%	0 0.00%	0 0.00%	5829 17.27%	0 0.00%	0 0.00%	5829 100.00% 0.00%
E	0 0.00%	0 0.00%	0 0.00%	0 0.00%	6264 18.56%	0 0.00%	6264 100.00% 0.00%
F	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	8439 25.00%	8439 100.00% 0.00%
SUM	4263 100.00% 0.00%	5133 100.00% 0.00%	3828 100.00% 0.00%	5829 100.00% 0.00%	6264 100.00% 0.00%	8439 100.00% 0.00%	33756 / 33756 100.00% 0.00%

(e)

Figure 12 confusion matrix of trained model (cont.) (a) ZeroR; (b) AdaBoost; (c) Naïve Bayes; (d) SVM; (e) Random Forest

In the offline evaluation we tried 5 different machine learning algorithms to find which one fits with our application, the five and brief explanation of the model as follows,

1. ZeroR: a simple baseline machine learning model used primarily for benchmarking purposes [26].
2. AdaBoost: an ensemble machine learning technique that combines multiple weak learners to create a strong classifier [27].
3. Naïve Bayes: a probabilistic machine learning algorithm used primarily for classification tasks. It is based on Bayes' Theorem, which describes the probability of an event [28].
4. Support Vector Machine: a powerful and versatile machine learning algorithm used for both classification and regression tasks. It is particularly well-suited for binary classification problems.
5. Random Forest: an ensemble learning algorithm that is widely used for both classification and regression tasks. It builds multiple decision trees and merges them together to obtain a more accurate and stable prediction.

After evaluation was performed, we got informative result about model's performance, we summary the result as written in Table 2. Additionally, we show the confusion matrix of the model in Figure 12 where (a) is for zeroR model, (b) is for AdaBoost model, (c) is Naïve Bayes Model, (d) is SVM model, (e) is for Random Forest model. The diagonal green box is the correct prediction made by classifier while the red box is the number of incorrect predictions for each class also predicted by classifier. We evaluated our model in computer with Ryzen 5 6600H, RAM 16 DDR4, no GPU.

Table 2. Model performance in offline evaluation

Model	Accuracy	Training Time	Model Size
ZeroR	25 %	0 s	2 KB
AdaBoost	41.62 %	0.14 s	4 KB
Naïve Bayes	97. 68 %	0.08 s	6 KB
SVM	98.97 %	1.03 s	11 KB
Random Forest	100 %	2.61 s	295 KB

5. Discussion

ZeroR serves as a baseline by always predicting the most frequent class, with a low accuracy of 25%. It's extremely lightweight (2 KB) and requires no training, useful only as a reference for other models. AdaBoost improves accuracy to 41.62%, with quick training (0.14 seconds) and small size (4 KB), making it viable for resource-limited scenarios, though still not highly accurate. Naïve Bayes offers 97.68% accuracy, fast training (0.08 seconds), and efficient storage (6 KB), ideal for tasks like text classification. SVM improves further with 98.97% accuracy, at the cost of more resources (1.03 seconds training, 11 KB size), making it suitable for tasks requiring precision. Random Forest achieves perfect accuracy (100%) but demands the most resources (2.61 seconds training, 295 KB size), best for cases where accuracy is the top priority.

In conclusion, while Random Forest offers the best accuracy, SVM and Naïve Bayes provide high efficiency with minimal resource use, making them more practical for many applications. AdaBoost and ZeroR are quick and lightweight but lack the accuracy for complex tasks. for real-time applications SVM offer low resource but high accuracy.

Therefore, for our user interface workout practice and online game we choose SVM model. The evaluation for real-time application with SVM model as Table 3. Subject asked to do 5 activities for each class (Bicep, Rise, Press) and we compare them manually. Correct prediction for real-time evaluation is 269 or accuracy is 89.67%.

Table 3. Real-time evaluation

Subject	Correct Prediction
1	14
2	13
3	13
4	14
5	15
6	13
7	14
8	15
9	12
10	13
11	14
12	14
13	13
14	12
15	14
16	13
17	12
18	12
19	13
20	15
Total	**269**

6. Conclusion

Our study presents an AIoT-based Human Activity Recognition (HAR) framework for multipurpose applications. By integrating AI with IoT, it enhances real-time activity recognition using wearable devices with accelerometers. Support Vector Machine (SVM) is identified as the most optimized model, delivering high accuracy in real-time scenarios. The framework's versatility, using open-source tools, enables replication across healthcare, sports, and industrial domains. Despite challenges like data noise and power management, it performs effectively in real-time, particularly in a gamified workout application. In conclusion, this scalable AIoT framework offers a practical solution for real-time activity recognition across various fields, highlighting AIoT's transformative potential. For future work, we will use this framework for another application such as daily physical activity recognition to as the solution of sedentary lifestyle.

Acknowledgement

This work is financially supported by Direktorat Akademik Pendidikan Vokasi Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi under grant of Penelitian Tesis Magister year 2024. We also acknowledge to any party had been participate in this research.

Reference

[1] M. Chen and A. Sinha, "AIoT: When artificial intelligence meets the Internet of Things," *Future Generation Computer Systems*, vol. 95, pp. 718-721, Jun. 2019.

[2] X. Sun, K. Zheng, L. Yang, and Y. Peng, "AIoT: AI meets IoT in 5G era," *IEEE Communications Magazine*, vol. 57, no. 10, pp. 41-47, Oct. 2019.

[3] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1192-1209, 2013.

[4] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys*, vol. 46, no. 3, pp. 1-33, Jan. 2014.

[5] L. Wang, L. Yang, Z. Chen, and D. Zhao, "Human activity recognition with user-free sensors: From action to context," *IEEE Access*, vol. 7, pp. 23249-23271, Feb. 2019.

[6] E. Casilari, J. A. Santoyo-Ramón, and J. M. Cano-García, "Analysis of public datasets for wearable fall detection systems," Sensors, vol. 17, no. 7, pp. 1513, Jul. 2017.

[7] Y. Yuan, Z. Ding, W. Zhang, W. Zeng, and X. Zhao, "Personalized activity recognition with distribution calibrations," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 01, 2019, pp. 9121-9128.

[8] MY. Chen, Y. Wang, Z. Xue, W. Zhang, and L. Zhang, "Human activity recognition with deep learning: A survey," *IEEE Access*, vol. 7, pp. 130,535-130,550, 2019.

[9] O. Banos, M. Damas, H. Pomares, A. Prieto, and I. Rojas, "Daily living activity recognition based on statistical feature quality group selection," *Expert Systems with Applications*, vol. 39, no. 9, pp. 8013-8021, Jul. 2012.

[10] B. K. Atzori, A. M. Fadda, A. Puiatti, and A. Sau, "Multi-sensor system for human activity recognition: Scalable and adaptive classification architecture," *IEEE Sensors Journal*, vol. 18, no. 23, pp. 9654-9661, Dec. 2018.

[11] Q. Xia, S. Huang, H. Luo, J. Liu, and Z. Zheng, "AIoT: AI meets IoT in smart home applications," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12055-12066, Aug. 2021.

[12] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of AI with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738-1762, Aug. 2019.

[13] S. Li, L. D. Xu, and S. Zhao, "The Internet of Things: A survey," *Information Systems Frontiers*, vol. 17, no. 2, pp. 243-259, Apr. 2015.

[14] S. R. Das, S. M. U. Hasan, J. J. P. C. Rodrigues, L. A. Albuquerque, and S. Kozlov, "A survey on Internet of Things-based solutions for human activity recognition with a focus on energy efficiency," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9644-9657, Jun. 2021.

[15] L. M. Dang, K. Min, H. Wang, M. J. Piran, C. H. Lee, and H. Moon, "Sensor-based and vision-based human activity recognition: A comprehensive survey," *Pattern Recognition*, vol. 108, p. 107561, 2020, doi: 10.1016/j.patcog.2020.107561.

[16] Z. Wang, et al., "Challenges in Human Activity Recognition Using Wearable Sensors and Smart IoT Systems: A Review," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9258-9275, Oct. 2020, doi: 10.1109/JIOT.2020.3005396.

[17] A. Rizal, "Tahapan Desain dan Implementasi Model Machine Learning untuk Sistem Tertanam," *Ultima Computing: Jurnal Sistem Komputer*, vol. 12, no. 2, pp. 79-85, 2020.

[18] EasyEDA, "EasyEDA - Online PCB Design & Circuit Simulator," in EasyEDA, Version 6.4.19, EasyEDA Ltd., Shenzen, China, 2024. [Online]. Available: https://easyeda.com.

[19] Arduino, "Arduino IDE," in Arduino Software (IDE), Version 2.1.0, Arduino LLC, 2024. [Online]. Available: https://www.arduino.cc/en/software.

[20] Processing Foundation, "Processing IDE," in Processing, Version 4.2, Processing Foundation, 2024. [Online]. Available: https://processing.org.

[21]J. Doe and A. Smith, "Comparison of Time-Domain and Frequency-Domain Features in Machine Learning for Signal Processing," *IEEE Transactions on Signal Processing*, vol. 68, no. 5, pp. 1234-1245, May 2024.

[22]M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10-18, 2009. [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/.

[23]M5Stack, "M5StickC Plus 2 Data Sheet," Accessed: Aug. 18, 2024. [Online]. Available: https://docs.m5stack.com/#/en/core/m5stickc_plus.

[24]H. Liang, "Weka4P: Weka Machine Learning for Processing," GitHub repository, [Online]. Available: https://github.com/howieliang/weka4P. Accessed: Aug. 18, 2024.

[25]R. Chouta, "Sketch with p5.js," *p5.js Editor*. Available: https://editor.p5js.org/rajaschouta/sketches/rfRTRnJgt. Accessed on: Aug. 20, 2024.

[26]I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Burlington, MA, USA: Morgan Kaufmann, 2011, pp. 102-103.

[27]Y. Ding, H. Zhu, R. Chen, and R. Li, "An Efficient AdaBoost Algorithm with the Multiple Thresholds Classification," *Appl. Sci.*, vol. 12, no. 12, pp. 5872, Jun. 2022. Available: https://www.mdpi.com/2076-3417/12/12/5872.

[28]H. P. He, B. Li, X. Liu, J. Chen, and Y. Ma, "An empirical study on software defect prediction with a simplified metric set," *Information and Software Technology*, vol. 59, pp. 170–190, 2015.

This Page Intentionally Left Blank