**PRAKTIKUM SIMULASI SISTEM MONITORING SUHU DAN**

**KELEMBAPAN DENGAN MQTT DAN ESP32**

*Fredlina Devhania Kholishah1*

*Fakultas Vokasi, Universitas Brawijaya*

[*devhania88@gmail.com*](mailto:devhania88@gmail.com)

### **ABSTRAK**

Pemanfaatan mikrokontroler ESP32 dalam bidang kecerdasan buatan semakin berkembang, khususnya melalui pendekatan TinyML, yaitu implementasi model machine learning ringan pada perangkat dengan sumber daya terbatas. Proyek ini bertujuan mengimplementasikan model klasifikasi dataset Iris ke dalam ESP32 menggunakan simulator Wokwi. Dataset Iris dipilih karena karakteristiknya yang sederhana dan umum digunakan dalam uji coba klasifikasi. Model klasifikasi akan mengenali tiga jenis bunga (setosa, versicolor, virginica) berdasarkan empat fitur: panjang dan lebar sepal serta petal. Hasil klasifikasi ditampilkan melalui serial monitor. Proyek ini membuktikan bahwa ESP32 mampu menjalankan model klasifikasi sederhana dengan baik, sehingga membuka peluang pemanfaatan TinyML pada perangkat edge yang hemat energi dan biaya.

***Kata Kunci****: TinyML, ESP32, Dataset Iris, Klasifikasi, Machine Learning*

## **ABSTRACT**

The use of ESP32 microcontrollers in artificial intelligence continues to grow, particularly through TinyML, which allows lightweight machine learning models to run on resource-constrained devices. This project aims to implement a classification model trained on the Iris dataset into the ESP32 using the Wokwi simulator. The Iris dataset is chosen due to its simplicity and frequent use in classification testing. The model classifies three flower types (*setosa*, *versicolor*, *virginica*) based on four features: sepal length, sepal width, petal length, and petal width. The classification result is displayed via the serial monitor. This project demonstrates that ESP32 can successfully execute simple classification models, enabling energy-efficient and cost-effective edge AI applications.

***Keywords****: TinyML, ESP32, Iris Dataset, Classification, Machine Learning*

## 

## 

## 

## 

## **1. PENDAHULUAN**

**1.1 Latar Belakang**

Perkembangan teknologi Internet of Things (IoT) mendorong kebutuhan akan perangkat cerdas yang mampu memproses data secara lokal tanpa bergantung pada server pusat. Salah satu pendekatan yang muncul untuk menjawab tantangan ini adalah *Tiny Machine Learning* (TinyML), yaitu integrasi model *machine learning* ringan yang dapat dijalankan langsung pada perangkat mikrokontroler dengan keterbatasan memori dan daya.

ESP32 merupakan salah satu mikrokontroler yang banyak digunakan dalam pengembangan proyek IoT karena memiliki kemampuan pemrosesan cukup tinggi, konektivitas Wi-Fi, serta harga yang terjangkau. Dengan dukungan terhadap pustaka TensorFlow Lite for Microcontrollers, ESP32 mampu menjalankan model AI sederhana yang telah dikonversi ke format *TensorFlow Lite*.

Dataset *Iris* merupakan dataset klasik dalam dunia *machine learning* yang terdiri dari tiga jenis bunga (Setosa, Versicolor, Virginica), masing-masing dengan empat fitur numerik: panjang dan lebar sepal serta petal. Dataset ini sangat cocok digunakan sebagai bahan awal pengujian sistem klasifikasi berbasis TinyML.

Implementasi proyek ini bertujuan menguji kemampuan ESP32 dalam menjalankan model klasifikasi ringan berbasis TinyML untuk mengenali jenis bunga dari data input. Selain itu, proyek ini juga menunjukkan potensi besar pemrosesan AI di perangkat edge yang efisien dan hemat daya.

### **1.2 Tujuan**

Tujuan dari kegiatan ini adalah:

1. Mengimplementasikan model klasifikasi dataset *Iris* menggunakan TinyML pada mikrokontroler ESP32.
2. Menguji akurasi dan performa model klasifikasi yang dijalankan secara langsung di perangkat edge.
3. Menampilkan hasil klasifikasi dalam bentuk prediksi kelas melalui serial monitor.
4. Menunjukkan efisiensi penggunaan TinyML untuk aplikasi klasifikasi sederhana di perangkat embedded.

## 

## 

## 

## 

## **2. METODOLOGI**

### **2.1 Alat dan Bahan**

Adapun alat dan bahan yang digunakan dalam implementasi proyek ini adalah sebagai berikut:

* ESP32 Devkit V1
* Kabel USB untuk pemrograman
* Laptop dengan software VSCode + PlatformIO
* Software Wokwi Simulator
* Model TinyML hasil pelatihan TensorFlow (format .tflite)
* Dataset Iris
* Library pendukung: TensorFlow Lite for Microcontrollers, Arduino\_TensorFlowLite, Numpy, dan pandas (untuk pelatihan model)

### **2.2 Diagram Rangkaian**

Rangkaian sistem tidak menggunakan banyak perangkat keras tambahan selain ESP32, karena proses klasifikasi dilakukan sepenuhnya pada chip ESP32 dan hasil prediksi ditampilkan melalui serial monitor. Gambar rangkaian yang ditampilkan di simulator Wokwi hanya memuat ESP32 tanpa sensor eksternal.

### **2.3 Langkah-Langkah Pelaksanaan**

Berikut langkah-langkah pelaksanaan proyek secara sistematis:

1. **Pelatihan Model**
   * Dataset Iris diolah menggunakan Python (dengan pandas & sklearn).
   * Model klasifikasi dibuat menggunakan algoritma Decision Tree/Neural Network ringan.
   * Model dilatih dan disimpan dalam format .tflite.
2. **Konversi dan Deploy Model ke ESP32**
   * Model .tflite dikonversi agar kompatibel dengan TensorFlow Lite for Microcontrollers.
   * File model dimasukkan ke dalam proyek PlatformIO (C++).
   * Kode program ditulis untuk membaca data input dan menjalankan inferensi model.
3. **Simulasi dan Pengujian**
   * Simulasi dijalankan pada Wokwi dengan menambahkan input dummy sesuai format data Iris.
   * Hasil klasifikasi ditampilkan melalui serial monitor.
   * Pengujian dilakukan dengan membandingkan output ESP32 dengan prediksi dari Python.

### **2.4 Alur Sistem**

Berikut merupakan alur sistem dari implementasi TinyML pada ESP32 untuk klasifikasi dataset Iris:

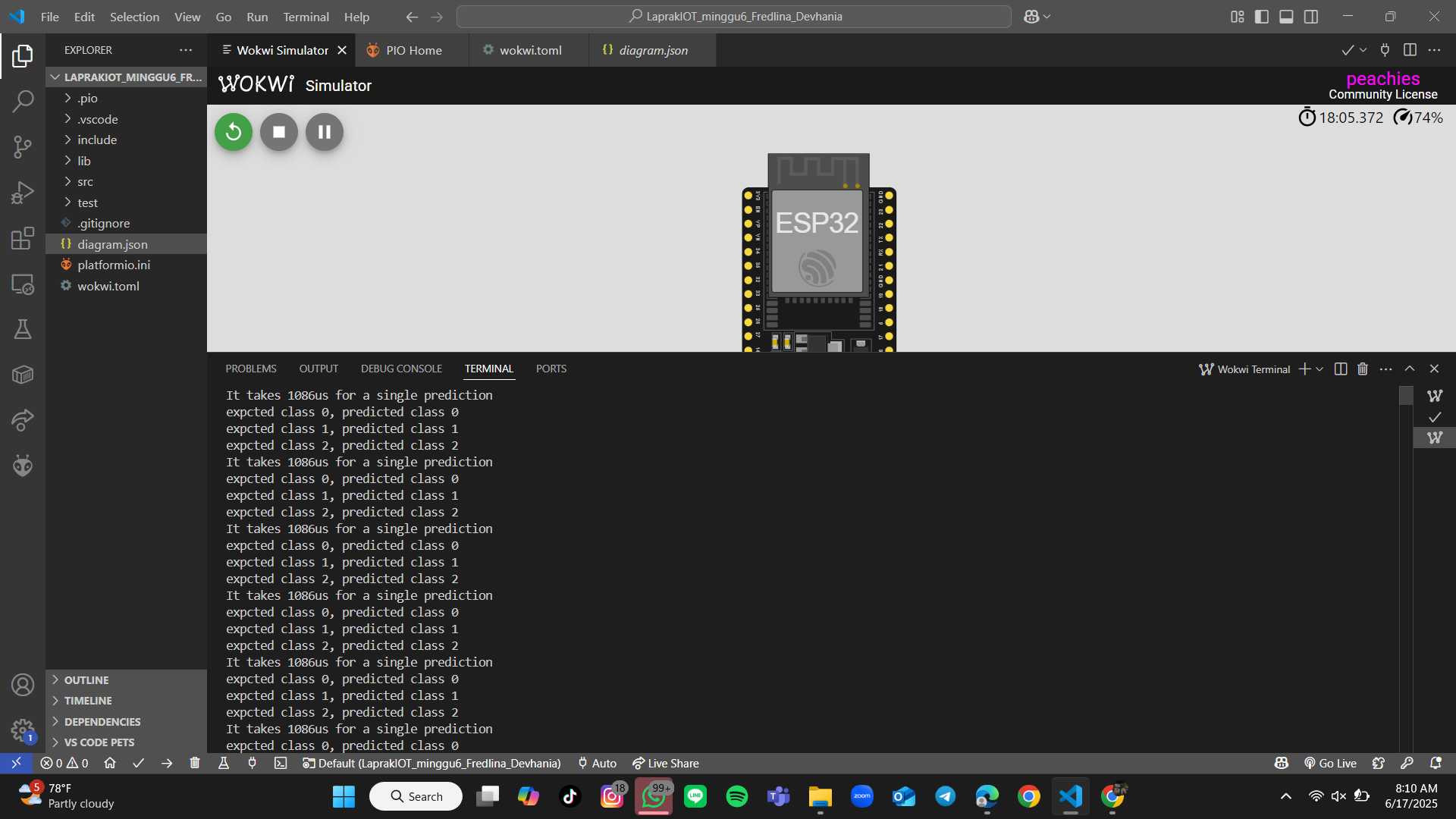
1. Data fitur iris (panjang/ lebar sepal dan petal) dimasukkan ke ESP32 (secara statis/manual).
2. Data dikirim ke model TinyML yang telah tertanam di dalam memori ESP32.
3. Model melakukan inferensi dan menghasilkan output berupa prediksi kelas (Setosa, Versicolor, atau Virginica).
4. Hasil prediksi ditampilkan di serial monitor secara real-time.

## **3. HASIL DAN PEMBAHASAN**

### **3.1 Hasil Implementasi**

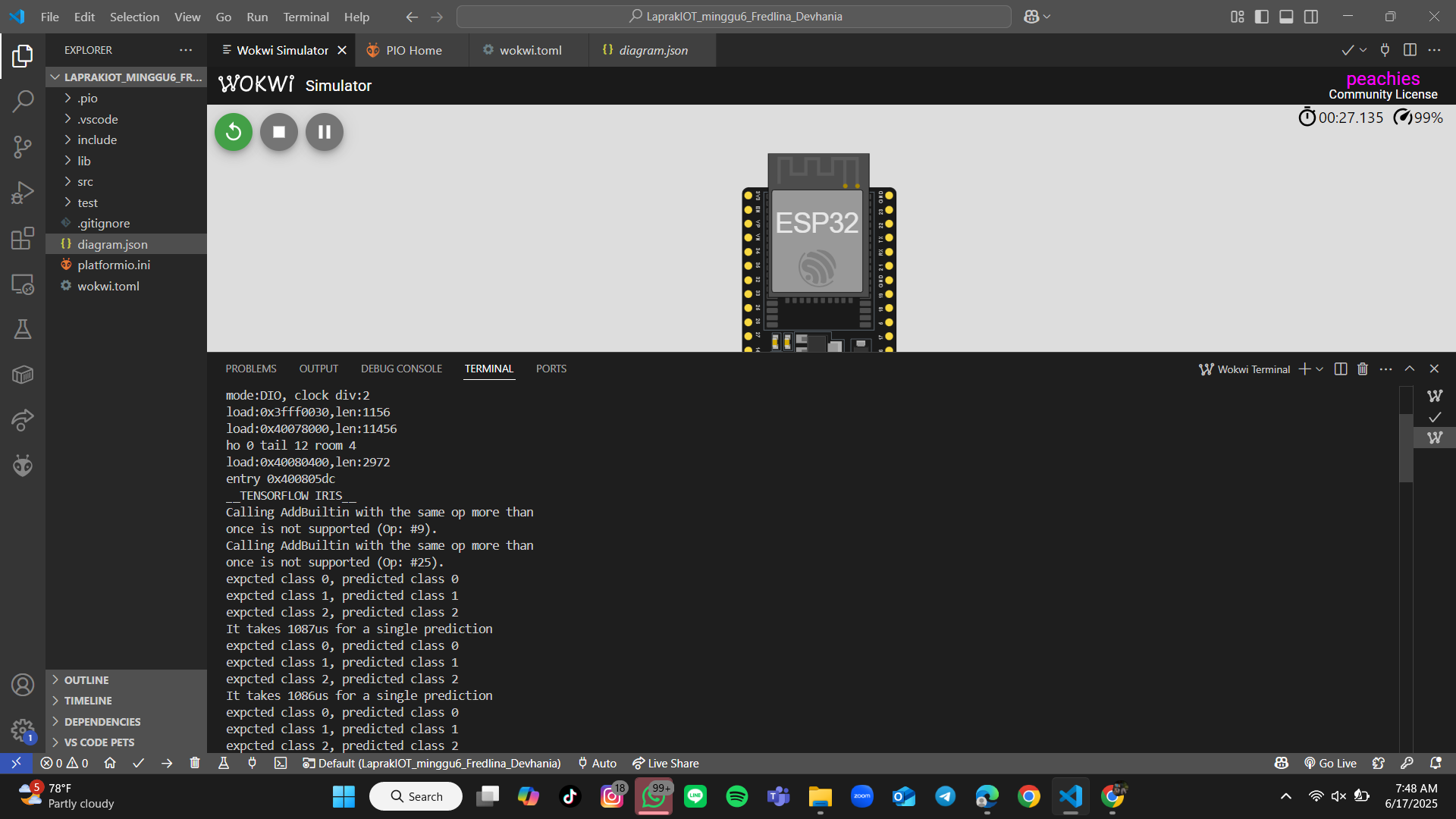
Setelah proses pelatihan model klasifikasi berbasis dataset *Iris* selesai dilakukan dan model dikonversi ke format .tflite, model tersebut berhasil dijalankan pada mikrokontroler ESP32 menggunakan pustaka TensorFlow Lite for Microcontrollers. Implementasi dilakukan melalui simulasi di Wokwi, di mana data uji dimasukkan langsung ke dalam kode program.

Berikut adalah cuplikan output dari serial monitor ESP32:

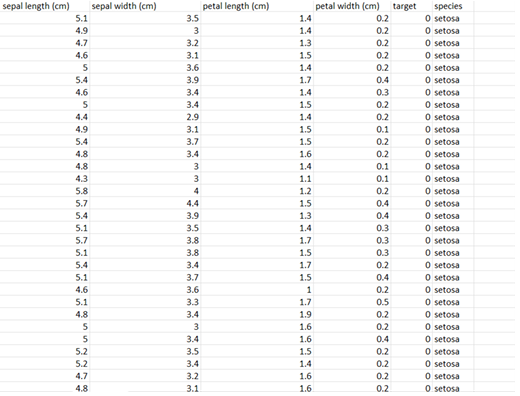


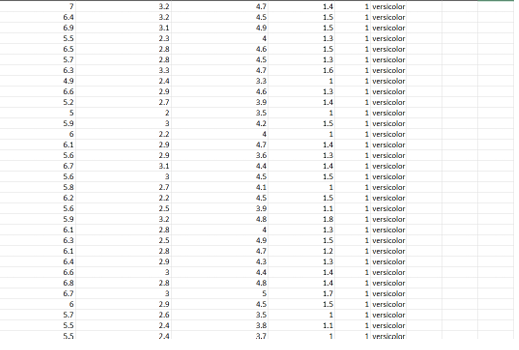
Output tersebut menunjukkan bahwa proses inferensi model membutuhkan waktu sekitar 1086 mikrodetik (µs) untuk satu prediksi, dan model berhasil mengklasifikasikan seluruh data uji secara akurat, di mana nilai prediksi sama dengan kelas yang diharapkan (expected class = predicted class).

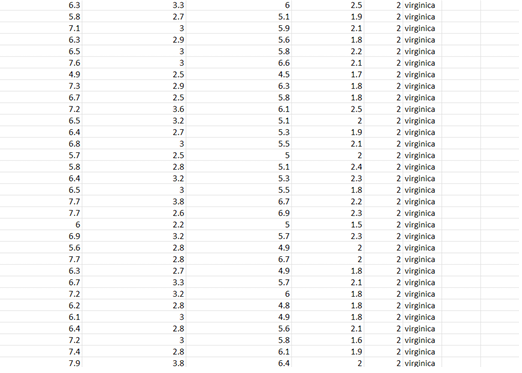
### **Hasil kelas data IRIS yang diuji**



1. **Dataset IRIS**

****

****

****

### **3.2 Pembahasan**

Dari hasil implementasi, dapat disimpulkan bahwa model TinyML berhasil dijalankan secara lokal pada ESP32 dengan performa yang cukup baik. Kecepatan eksekusi inferensi mencapai sekitar 1086 mikrodetik untuk satu prediksi, yang tergolong cepat dan efisien untuk ukuran mikrokontroler.

Kelebihan yang diperoleh dalam implementasi ini antara lain:

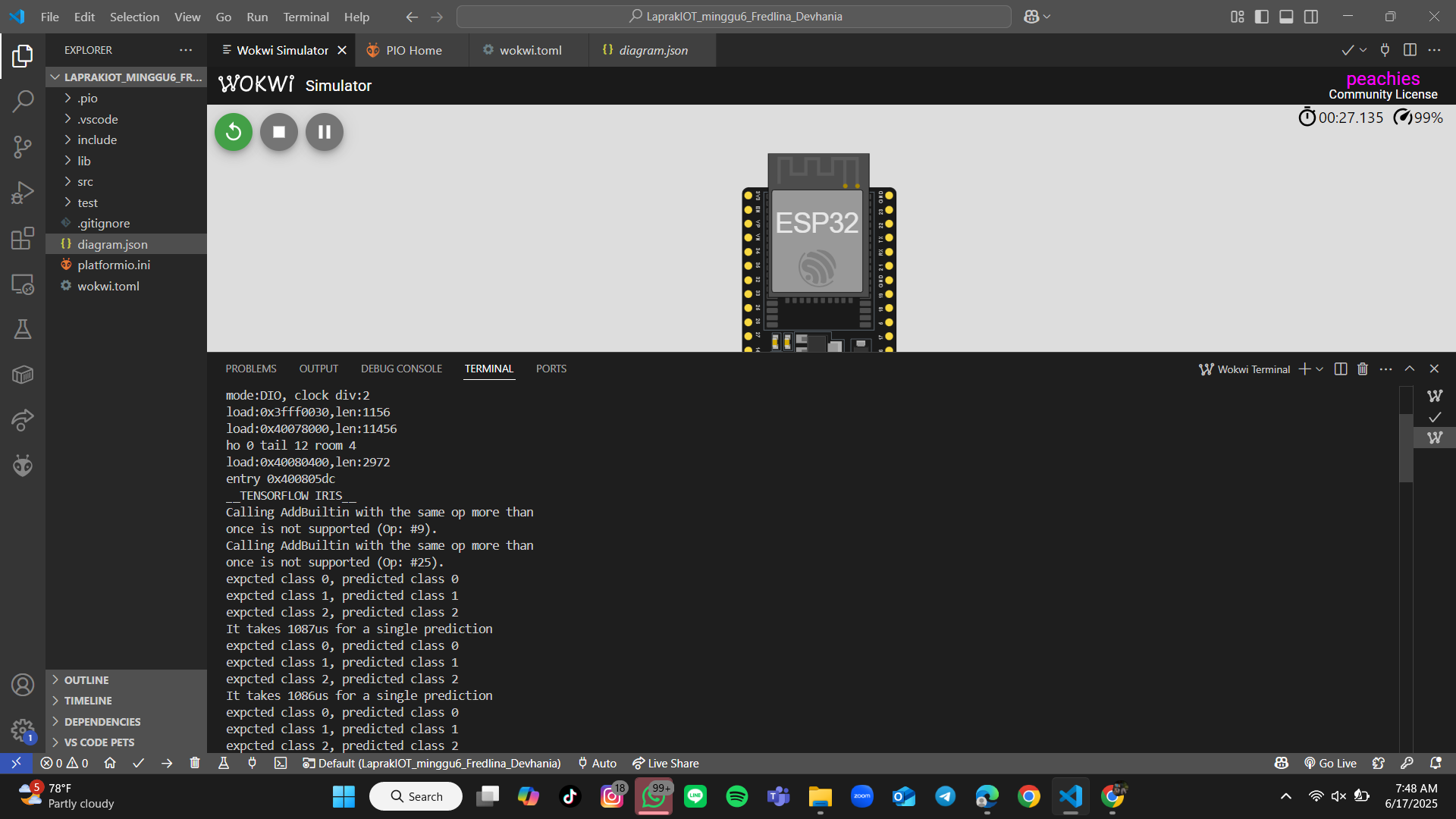
* Presisi tinggi: Semua prediksi sesuai dengan kelas target yang diharapkan (0, 1, 2).
* Efisiensi waktu: Inferensi berlangsung dalam waktu kurang dari 1,1 milidetik.
* Tanpa ketergantungan cloud: Semua proses terjadi secara offline langsung di perangkat ESP32.
* Ringan dan portabel: Model dapat disimpan dalam memori terbatas ESP32 dan dijalankan tanpa penurunan performa.

Adapun keterbatasan yang masih ada:

* Input data masih bersifat statis/manual di dalam kode (belum menerima data real-time).
* Evaluasi model hanya diuji pada sejumlah kecil data, belum pada dataset uji besar.
* Hasil klasifikasi hanya ditampilkan di serial monitor, belum memiliki antarmuka pengguna.

**4. LAMPIRAN**

### **4.1 Gambar Rangkaian Wokwi**



### **4.2 Kode Program**

Kode Program Main.cpp

#include <Arduino.h>

/\*\*

\* Run a TensorFlow model to predict the IRIS dataset

\* For a complete guide, visit

\* https://eloquentarduino.com/tensorflow-lite-esp32

\*/

// replace with your own model

// include BEFORE <eloquent\_tinyml.h>!

#include "iris\_model.h"

// include the runtime specific for your board

// either tflm\_esp32 or tflm\_cortexm

#include <tflm\_esp32.h>

// now you can include the eloquent tinyml wrapper

#include <eloquent\_tinyml.h>

// this is trial-and-error process

// when developing a new model, start with a high value

// (e.g. 10000), then decrease until the model stops

// working as expected

#define ARENA\_SIZE 2000

Eloquent::TF::Sequential<TF\_NUM\_OPS, ARENA\_SIZE> tf;

//Eloquent::TinyML::TfLite<4,3,ARENA\_SIZE> tf;

/\*\*

\*

\*/

void setup() {

Serial.begin(115200);

delay(3000);

Serial.println("\_\_TENSORFLOW IRIS\_\_");

// configure input/output

// (not mandatory if you generated the .h model

// using the everywhereml Python package)

tf.setNumInputs(4);

tf.setNumOutputs(3);

// add required ops

// (not mandatory if you generated the .h model

// using the everywhereml Python package)

tf.resolver.AddFullyConnected();

tf.resolver.AddSoftmax();

while (!tf.begin(irisModel).isOk())

Serial.println(tf.exception.toString());

}

void loop() {

// x0, x1, x2 are defined in the irisModel.h file

// https://github.com/eloquentarduino/EloquentTinyML/tree/main/examples/IrisExample/irisModel.h

// classify sample from class 0

if (!tf.predict(x0).isOk()) {

Serial.println(tf.exception.toString());

return;

}

Serial.print("expcted class 0, predicted class ");

Serial.println(tf.classification);

// classify sample from class 1

if (!tf.predict(x1).isOk()) {

Serial.println(tf.exception.toString());

return;

}

Serial.print("expcted class 1, predicted class ");

Serial.println(tf.classification);

// classify sample from class 2

if (!tf.predict(x2).isOk()) {

Serial.println(tf.exception.toString());

return;

}

Serial.print("expcted class 2, predicted class ");

Serial.println(tf.classification);

// how long does it take to run a single prediction?

Serial.print("It takes ");

Serial.print(tf.benchmark.microseconds());

Serial.println("us for a single prediction");

delay(1000);

}

Kode Program diagram.json

{

"version": 1,

"author": "Fredlina Devha",

"editor": "wokwi",

"parts": [ { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0, "attrs": {} } ],

"connections": [ [ "esp:TX", "$serialMonitor:RX", "", [] ], [ "esp:RX", "$serialMonitor:TX", "", [] ] ],

"dependencies": {}

}