**Implementasi Model Machine Learning untuk Klasifikasi Dataset Iris pada Mikrokontroler ESP32**

*Muhammad Fau Zan Sabani1*

*Fakultas Vokasi, Universitas Brawijaya*

*Email:* [*mfauzan18@student.ub.ac.id*](mailto:mfauzan18@student.ub.ac.id)

# Abstrak:

Laporan praktikum ini membahas tentang implementasi *machine learning* pada perangkat *embedded* dengan memanfaatkan TensorFlow Lite pada mikrokontroler ESP32. Proyek ini bertujuan untuk menerapkan model klasifikasi yang mampu mengenali jenis bunga Iris secara langsung di perangkat keras (*on-device*), tanpa bergantung pada koneksi internet atau server eksternal.

Eksperimen dimulai dengan membangun dan melatih model klasifikasi di lingkungan Google Colab, yang kemudian dikonversi ke format TensorFlow Lite (.tflite). Model ini selanjutnya diubah menjadi sebuah *array* C dan diintegrasikan ke dalam proyek PlatformIO sebagai *file header*. Kode program kemudian dibuat untuk memuat model tersebut di ESP32, melakukan inferensi atau prediksi pada data input, dan menampilkan hasil klasifikasinya melalui Serial Monitor.

Hasil utama dari praktikum ini menunjukkan bahwa model *machine learning* dapat diimplementasikan dan berjalan secara efisien pada mikrokontroler ESP32, dengan tingkat akurasi yang tinggi dan waktu prediksi yang cepat. Implementasi ini berhasil membuktikan bahwa perangkat *embedded* dengan sumber daya terbatas mampu menjalankan tugas-tugas inferensi AI secara mandiri, membuka jalan untuk pengembangan aplikasi *TinyML* atau AI di perangkat IoT yang lebih cerdas dan responsif.

Kata Kunci *Kecerdasan Buatan, IoT, Machine Learning, ESP32, MQTT, Laporan Praktikum*

# Abstract:

This lab report discusses the implementation of machine learning on embedded devices by utilizing TensorFlow Lite on the ESP32 microcontroller. This project aims to implement a classification model capable of recognizing Iris flower types directly on-device, without relying on an internet connection or external server.

Experiments began by building and training the classification model in the Google Colab environment, which was then converted to TensorFlow Lite (.tflite) format. The model was then converted into a C array and integrated into the PlatformIO project as a header file. Program code was then created to load the model on the ESP32, perform inference or prediction on the input data, and display the classification results via Serial Monitor.

The main results of this practicum show that machine learning models can be implemented and run efficiently on the ESP32 microcontroller, with a high level of accuracy and fast prediction time. This implementation successfully proves that embedded devices with limited resources are capable of performing AI inference tasks independently, paving the way for the development of more intelligent and responsive TinyML or AI applications in IoT devices.

Keywords *Artificial Intelligence, IoT, Machine Learning, ESP32, MQTT, Practical Report*

# 1. Pendahuluan

* 1. **Latar Belakang**

Sistem *Internet of Things* (IoT) konvensional seringkali bergantung pada pengiriman seluruh data mentah ke *cloud* untuk dianalisis. Pendekatan ini menimbulkan beberapa tantangan, seperti latensi yang tinggi karena proses kirim-terima data, ketergantungan pada konektivitas internet yang stabil, serta risiko privasi karena data sensitif harus dikirim keluar dari perangkat lokal.

Dengan kemajuan pesat di bidang *Artificial Intelligence* (AI), muncullah sebuah solusi yang dikenal sebagai Tiny Machine Learning (TinyML). Teknologi ini memungkinkan model *machine learning* yang telah dioptimalkan untuk dijalankan secara langsung di perangkat mikrokontroler dengan sumber daya terbatas. Dengan pendekatan ini, perangkat IoT tidak hanya mampu mengumpulkan data, tetapi juga dapat memproses dan membuat keputusan secara mandiri di tingkat lokal (*on-device*), sehingga sistem menjadi lebih cepat, aman, dan efisien.

Oleh karena itu, praktikum implementasi model AI pada mikrokontroler menjadi penting untuk memahami potensi TinyML. Dalam proyek ini, dataset bunga Iris digunakan sebagai studi kasus untuk melatih model klasifikasi. Model yang sudah jadi kemudian dikonversi ke format TensorFlow Lite (.tflite) dan diintegrasikan langsung ke dalam mikrokontroler ESP32 sebagai *file header*. Melalui implementasi ini, dibuktikan bagaimana sebuah perangkat keras sederhana mampu menjalankan inferensi AI untuk melakukan klasifikasi data secara *real-time* dan mandiri.

* 1. **Tujuan Eksperimen**

Tujuan dari eksperimen simulasi pemantauan IoT menggunakan MQTT dan Wokwi ini adalah sebagai berikut:

* Merancang dan mengimplementasikan sistem pemantauan suhu dan kelembapan secara *end-to-end*, mulai dari akuisisi data oleh sensor DHT22 hingga visualisasi *real-time* pada antarmuka website.
* Menguji dan memvalidasi fungsionalitas pengiriman data dari ESP32 ke *broker* MQTT menggunakan platform simulasi Wokwi sebelum diterapkan pada perangkat keras.
* Memahami arsitektur sistem IoT secara menyeluruh, mencakup mekanisme komunikasi *publish-subscribe* pada protokol MQTT dan cara mengintegrasikannya dengan antarmuka pengguna berbasis web.
* Mengembangkan kemampuan teknis dalam pemrograman mikrokontroler ESP32 serta dasar-dasar pengembangan web untuk visualisasi data yang diterima dari *broker* MQTT.
* Menyediakan dasar untuk membangun aplikasi IoT yang lebih lengkap dan ramah pengguna (*user-friendly*), di mana data tidak hanya dikumpulkan tetapi juga disajikan dalam bentuk informasi yang mudah dipahami.

# 2. Metodologi

* 1. **Alat dan Bahan**

Untuk melakukan praktikum simulasi, alat dan bahan yang digunakan

* Mikrokontroler ESP32
* Platform wokwi, digunakan untuk membuat simulasi
* CNN/Google Colab
* Arduino IDE/PlatformIO jika menggunakan Visual Studio Code
  1. **Langkah Implementasi**

Berikut langkah-langkah implementasi simulasi lampu lalu lintas dengan menggunakan Mikrokontroler ESP32 di Wokwi/Arduino:

* Install Arduino IDE atau jika menggunakan VsCode install Extension PlatfromIO di bagian ekstensinya. Buka juga website wokwi untuk tampilan simulasi nya secara virtual (jangan lupa untuk install ekstensi Wokwi juga di VsCode).



* Buat proyek baru di Wokwi dengan memilih ESP32 sebagai mikrokontroler.
* Untuk pemogramannya menggunakan bahasa Arduino (C/C++) di Arduino IDE/PlatformIO (bisa juga langsung dimasukkan ke file ‘main.c’ di Wokwi).

#include <Arduino.h>

#include <iris\_model.h>

#include <tflm\_esp32.h>

#include <eloquent\_tinyml.h>

#define ARENA\_SIZE 2000

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

void setup() {

    Serial.begin(115200);

    delay(3000);

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

    tf.setNumInputs(4);

    tf.setNumOutputs(3);

    tf.resolver.AddFullyConnected();

    tf.resolver.AddSoftmax();

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

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

}

void loop() {

    if (!tf.predict(x0).isOk()) {

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

        return;

    }

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

    Serial.println(tf.classification);

    if (!tf.predict(x1).isOk()) {

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

        return;

    }

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

    Serial.println(tf.classification);

    if (!tf.predict(x2).isOk()) {

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

        return;

    }

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

    Serial.println(tf.classification);

    Serial.print("It takes ");

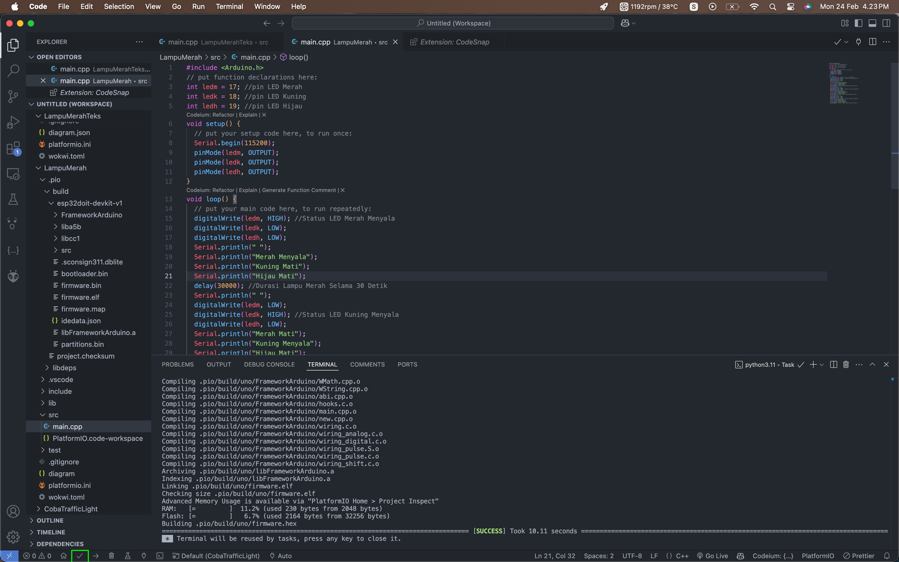
    Serial.print(tf.benchmark.microseconds());

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

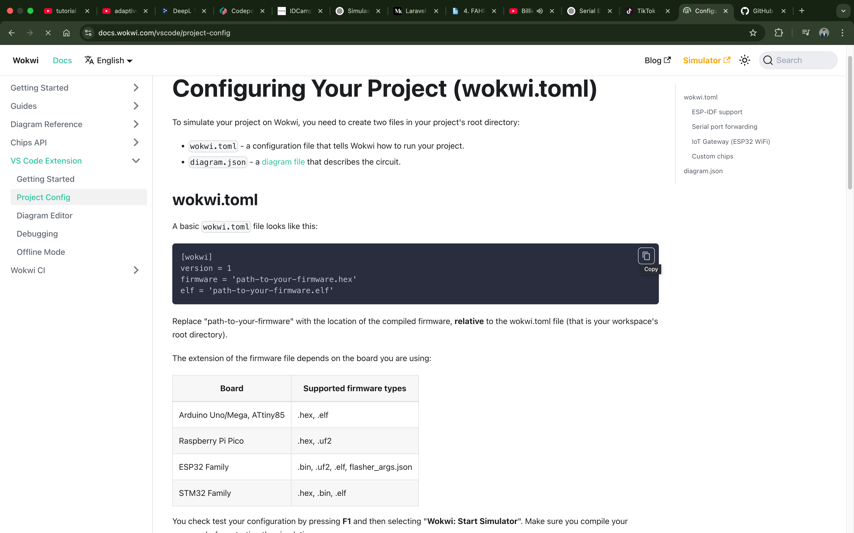
    delay(1000);

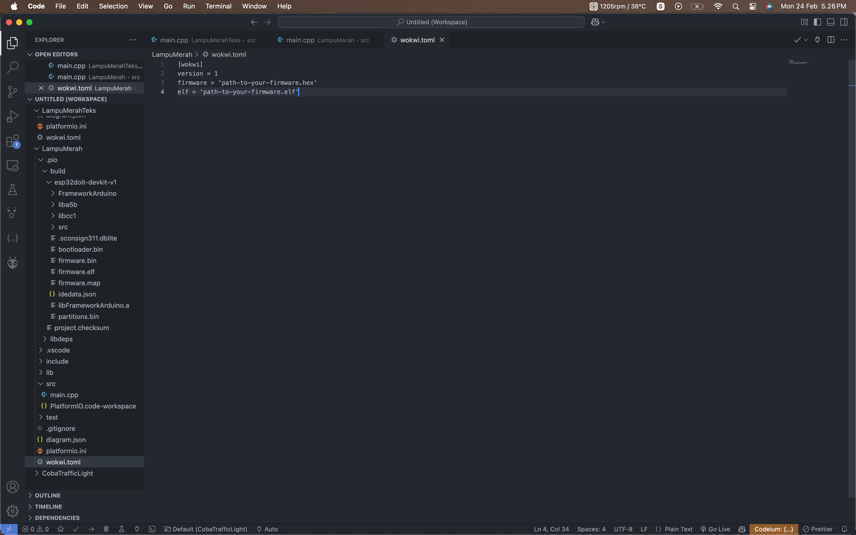
}

* Jika di platform Wokwi file ‘main.c’ tidak bisa ter-compile/dijalankan maka kita bisa mengetes sukses atau tidak kode tersebut di PlatformIO dengan menggunakan tombol centang dibawah.

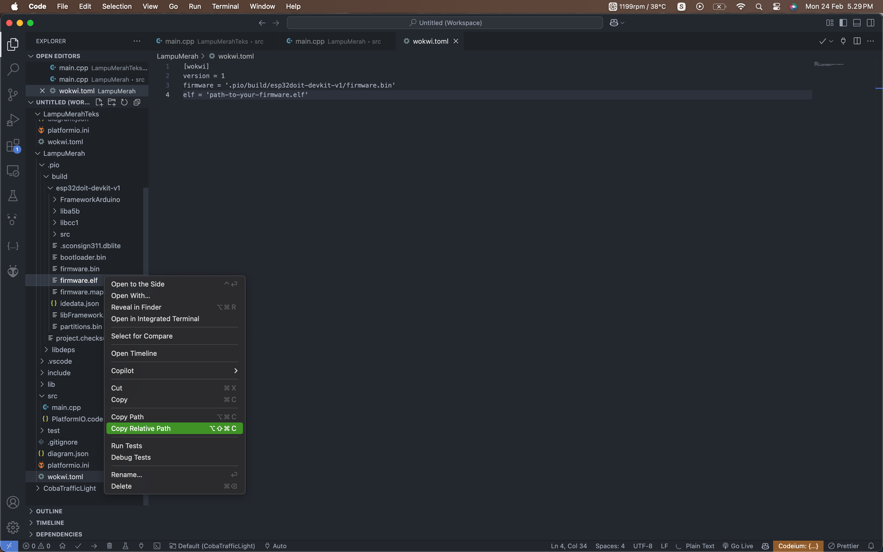


* Jika kode di file ‘main.cpp’ sukses di *Build* maka langkah selanjutnya untuk dapat melakukan virtualisasi simulasi lampu merah adalah membuat file bernama ‘diagram’ yang nanti diisi oleh kode dari file ‘diagram.json’ di projek Wokwi sebelumnya. Lalu kalau sudah, ubah nama file ‘diagram’ sbelumnya menjadi ‘diagram.json’ seperti di platform Wokwi sebelumnya.
* Melatih model klasifikasi menggunakan dataset *Iris* di Google Colab dengan library Keras.
* Mengonversi model ke format .tflite menggunakan TFLiteConverter.
* Mengubah file .tflite menjadi array C (iris\_model.h) dengan fungsi hex\_to\_c\_array().
* Memasukkan file iris\_model.h ke dalam folder proyek PlatformIO.
* Menulis kode utama dalam file main.cpp menggunakan library eloquent\_tinyml dan tflm\_esp32.
* Mengatur jumlah input dan output model.
* Melakukan pemanggilan model untuk memprediksi tiga contoh data (x0, x1, x2).
* Menampilkan hasil klasifikasi dan waktu eksekusi prediksi pada Serial Monitor.
* Untuk dapat menjalankan simulasi dari file ‘diagram.json’ pada platform.io tersebut maka kita harus menambahkan file yang kita namakan ‘wokwi.toml’. Langkah berikutnya cari di web browser dengan keyword ‘Wokwi.toml’ dan salin kode konfigurasinya yang berasal dari halaman web dokumentasi wokwi tersebut ke file ‘wokwi.toml’ yang sudah kita buat.





* Langkah terakhir untuk dapat mengkonfigurasi file ‘wokwi.toml’ tersebut agar dapat membantu menjalankan file diagramnya maka kita perlu untuk mengganti path ‘firmware’ dan ‘elf’ path file yang berada di dalam rute folder ‘pio/build/esp32doit-devkit-v1’. Salin tiap file dengan format file yang bertuliskan ‘.hex’ dan ‘.elf’ dengan menggunakan klik kanan dan klik ‘Copy relative path’ lalu tempel ke path yang harus diisi di file ‘wokwi.toml’.

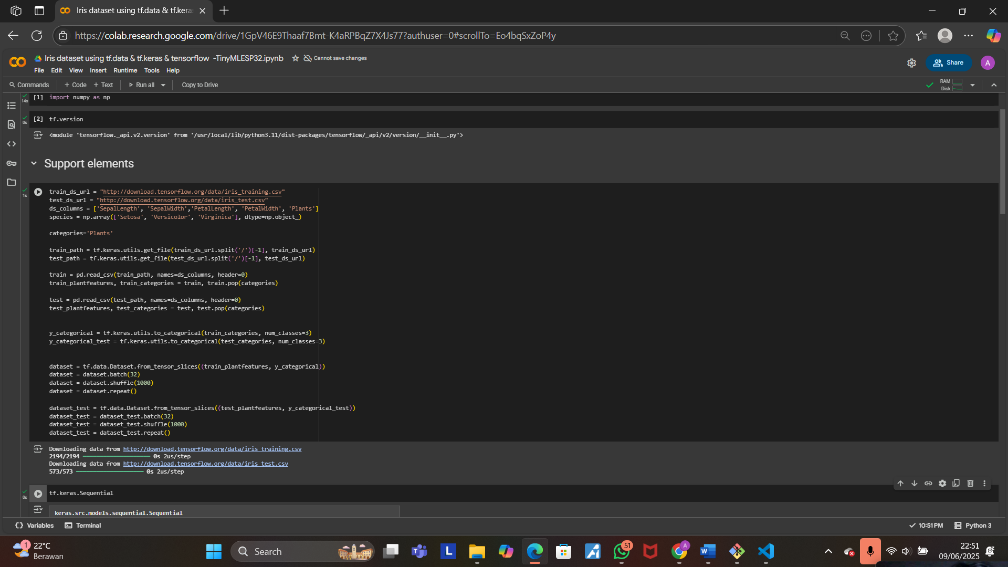


# 3. Hasil dan Pembahasan

1. **Hasil Eksperimen**

Dalam praktikum kali ini hasil yang didapat sebagai berikut :

* Melatih Model pada Google Colab dan Mengubah metode
* Link URL: <https://colab.research.google.com/drive/1GpV46E9Thaaf7Bmt-K4aRPBqZ7X4Js77?usp=sharing>



A screenshot of a computer

AI-generated content may be incorrect.

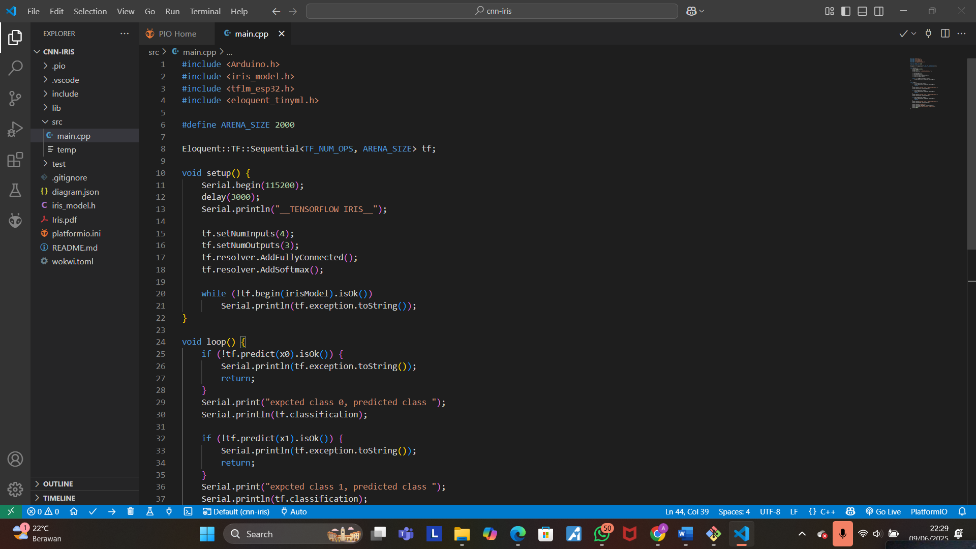
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Pada VSCode



A computer screen shot of a computer program

AI-generated content may be incorrect.

* Output yang didapat

A computer screen shot of a computer

AI-generated content may be incorrect.

# 4. Lampiran

* Kode Program .json:

{

  "version": 1,

  "author": "subairi",

  "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": {}

}

* Kode Iris.h:

#pragma once

#ifdef \_\_has\_attribute

#define HAVE\_ATTRIBUTE(x) \_\_has\_attribute(x)

#else

#define HAVE\_ATTRIBUTE(x) 0

#endif

#if HAVE\_ATTRIBUTE(aligned) || (defined(\_\_GNUC\_\_) && !defined(\_\_clang\_\_))

#define DATA\_ALIGN\_ATTRIBUTE \_\_attribute\_\_((aligned(4)))

#else

#define DATA\_ALIGN\_ATTRIBUTE

#endif

// automatically configure network

#define TF\_NUM\_INPUTS 4

#define TF\_NUM\_OUTPUTS 3

#define TF\_NUM\_OPS 2

#define TF\_OP\_SOFTMAX

#define TF\_OP\_FULLYCONNECTED

// sample data

float x0[4] = {0.22222222222f, 0.62500000000f, 0.06779661017f, 0.04166666667f};

float x1[4] = {0.75000000000f, 0.50000000000f, 0.62711864407f, 0.54166666667f};

float x2[4] = {0.55555555556f, 0.54166666667f, 0.84745762712f, 1.00000000000f};