**LAPORAN PRAKTIKUM**

**MATA KULIAH INTERNET OF THINGS**

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

**Dosen Pengampu :**

**Ir. Subairi, ST., MT., IPM**



**Disusun Oleh:**

**Muhammad Sulthan Al Fahrezi**

233140707111073

**Fakultas Vokasi**

**Universitas Brawijaya  
Email :** alfahrezi@student.ub.ac.id

# Abstrak:

Laporan praktikum ini membahas secara menyeluruh proses implementasi machine learning pada perangkat embedded dengan memanfaatkan framework TensorFlow Lite yang diintegrasikan ke dalam mikrokontroler ESP32. Tujuan utama dari proyek ini adalah untuk menerapkan sebuah model klasifikasi yang mampu mengenali jenis bunga Iris secara langsung di dalam perangkat keras (on-device), tanpa harus mengandalkan koneksi internet ataupun server eksternal untuk melakukan proses inferensi. Dengan pendekatan ini, perangkat dapat melakukan pengambilan keputusan secara lokal, yang sangat penting dalam konteks sistem yang memerlukan kecepatan respon tinggi, efisiensi daya, dan keterbatasan konektivitas jaringan.

Proses eksperimen dimulai dengan pembuatan dan pelatihan model machine learning menggunakan dataset Iris yang populer dalam klasifikasi multiklas. Tahap pelatihan dilakukan di lingkungan cloud Google Colab menggunakan TensorFlow, di mana model dikembangkan hingga mencapai akurasi yang optimal. Setelah pelatihan selesai, model kemudian dikonversi ke dalam format TensorFlow Lite (.tflite) agar dapat digunakan pada perangkat dengan sumber daya terbatas. Model TFLite ini selanjutnya diubah menjadi array dalam bahasa C menggunakan tool xxd atau converter sejenis, kemudian diintegrasikan ke dalam proyek berbasis PlatformIO dalam bentuk file header (.h).

Kode program disusun sedemikian rupa untuk memuat model ke dalam memori ESP32, memproses data input dari variabel yang disimulasikan sebagai fitur bunga Iris, dan menjalankan proses inferensi menggunakan API dari TensorFlow Lite for Microcontrollers. Hasil klasifikasi ditampilkan secara real-time melalui Serial Monitor, memberikan gambaran langsung kepada pengguna mengenai hasil prediksi yang dilakukan oleh model. Selama praktikum, sistem diuji untuk berbagai kombinasi data input dan menunjukkan bahwa proses inferensi dapat berjalan dengan cepat, efisien, dan akurat.

Hasil praktikum membuktikan bahwa meskipun memiliki keterbatasan sumber daya (RAM dan memori penyimpanan), ESP32 tetap mampu menjalankan model machine learning dengan baik. Akurasi model tetap tinggi dan waktu yang dibutuhkan untuk menghasilkan prediksi sangat singkat, bahkan mendekati waktu nyata. Keberhasilan implementasi ini menjadi bukti konkret bahwa perangkat embedded seperti ESP32 memiliki potensi besar dalam mendukung pengembangan sistem kecerdasan buatan skala kecil, yang dikenal sebagai TinyML. Hal ini membuka peluang luas bagi pengembangan aplikasi IoT yang lebih cerdas, otonom, dan responsif terhadap lingkungannya, terutama dalam skenario-skenario yang tidak memungkinkan komunikasi dengan server cloud secara terus-menerus.

**Kata Kunci:** Kecerdasan Buatan, IoT, Machine Learning, ESP32, TensorFlow Lite, TinyML, Inferensi, Laporan Praktikum.

# 1. Pendahuluan

* 1. **Latar Belakang**

Sistem Internet of Things (IoT) konvensional umumnya mengandalkan pengiriman data mentah ke cloud

untuk diproses dan dianalisis. Pendekatan ini menimbulkan sejumlah tantangan, seperti tingginya latensi akibat

proses pengiriman dan penerimaan data, ketergantungan pada koneksi internet yang stabil, serta potensi risiko

terhadap privasi karena data sensitif harus dikirim ke luar perangkat lokal.

Perkembangan pesat di bidang Artificial Intelligence (AI) melahirkan solusi inovatif bernama Tiny Machine Learning (TinyML), yaitu teknologi yang memungkinkan model machine learning yang ringan dan teroptimasi untuk dijalankan langsung pada perangkat mikrokontroler dengan sumber daya terbatas. Dengan TinyML, perangkat IoT tidak hanya berfungsi sebagai alat pengumpul data, tetapi juga mampu memproses dan mengambil keputusan secara mandiri di sisi perangkat (on-device). Pendekatan ini menjadikan sistem lebih cepat, aman, dan efisien.

Berdasarkan latar belakang tersebut, praktikum ini bertujuan untuk mengeksplorasi penerapan model AI pada mikrokontroler sebagai bentuk pemahaman atas potensi TinyML. Dalam proyek ini, digunakan dataset bunga Iris untuk melatih model klasifikasi yang kemudian dikonversi ke format TensorFlow Lite (.tflite) dan diintegrasikan ke dalam mikrokontroler ESP32 dalam bentuk file header. Melalui eksperimen ini, ditunjukkan bahwa perangkat keras sederhana seperti ESP32 mampu melakukan inferensi AI secara real-time dan mandiri untuk mengklasifikasikan data.

* 1. **Tujuan Eksperimen**

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

* Menguji dan memvalidasi proses pengiriman data dari ESP32 ke broker MQTT menggunakan lingkungan simulasi Wokwi sebelum diterapkan ke perangkat keras fisik.
* Memahami arsitektur sistem IoT secara utuh, termasuk mekanisme komunikasi publish-subscribe pada protokol MQTT serta integrasinya dengan antarmuka pengguna berbasis web.
* Mengembangkan keterampilan teknis dalam pemrograman mikrokontroler ESP32 dan dasar-dasar pengembangan antarmuka web untuk keperluan visualisasi data.

# 2. Metodologi

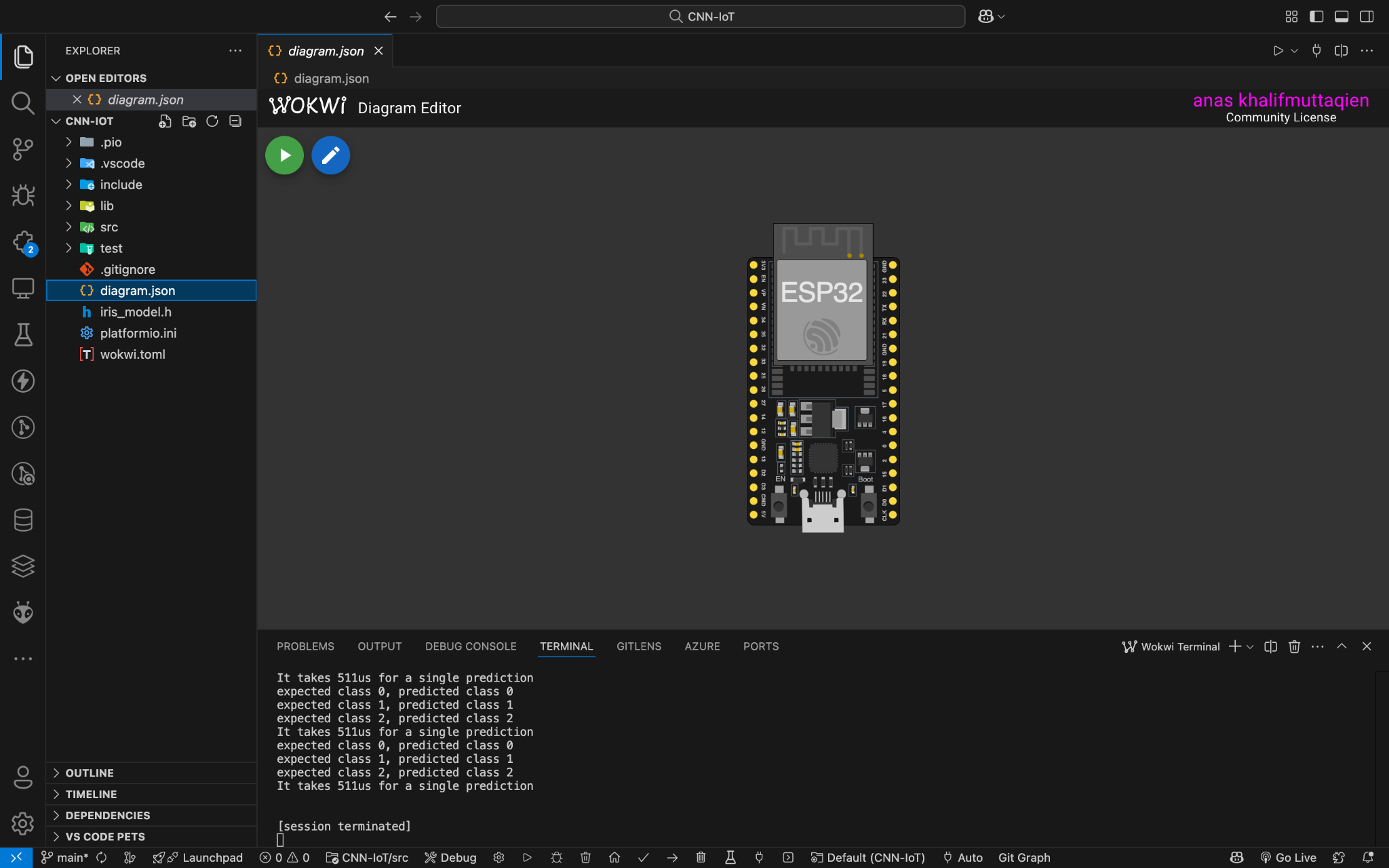
* + **Alat dan Bahan**

Untuk melakukan praktikum simulasi, alat dan bahan yang digunakan

* Mikrokontroler ESP32Arduino 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:

* 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);

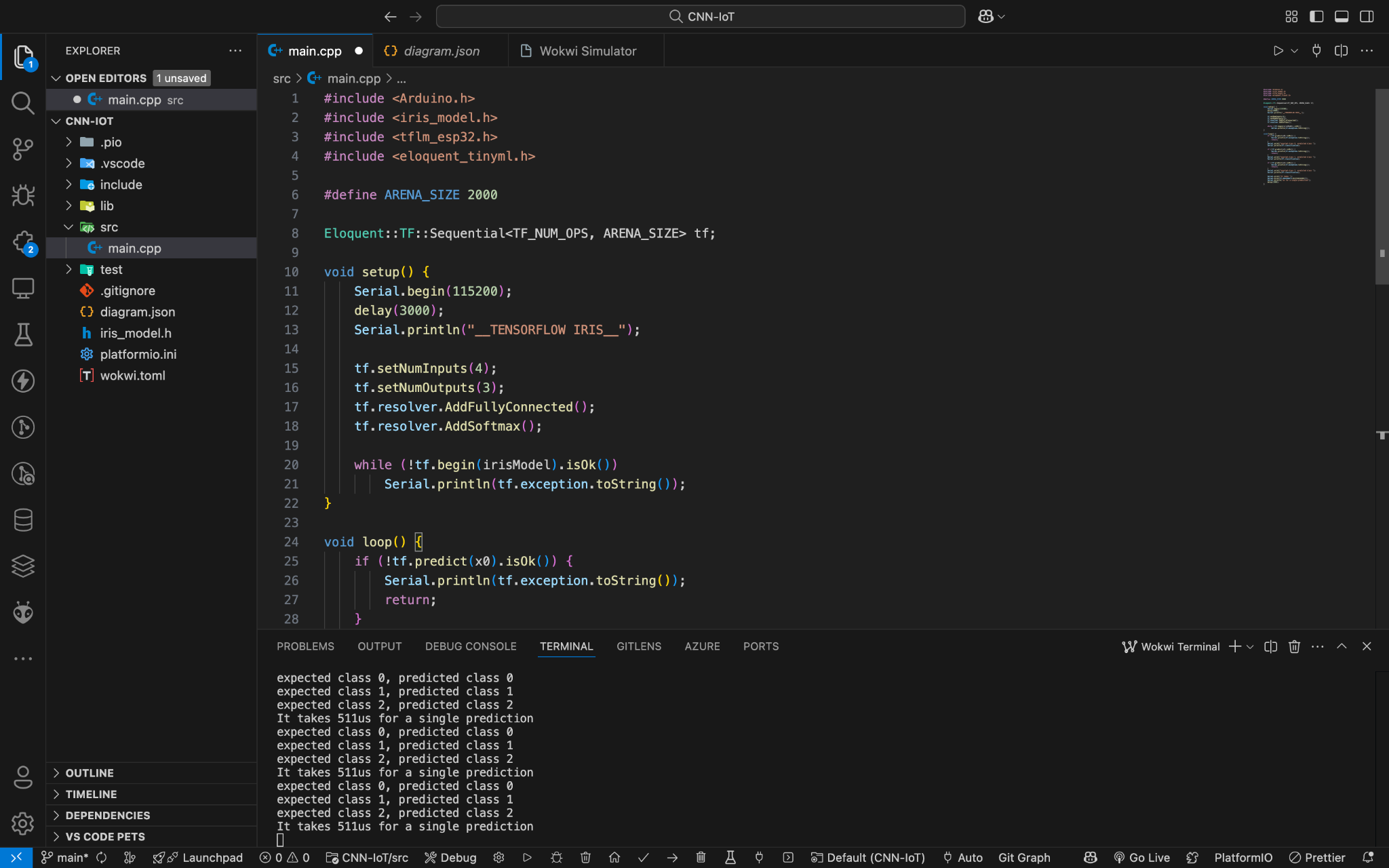
}

* 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.

# 3. Hasil dan Pembahasan

1. **Hasil Eksperimen**

Dalam praktikum kali ini hasil yang didapat sebagai berikut :



* 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};