KLASIFIKASI BUNGA IRIS MENGGUNAKAN MODEL TENSORFLOW LITE PADA MIKROKONTROLLER ESP 32

Feby Kurnia Putri

Teknologi Informasi, Fakultas Vokasi, Universitas Brawijaya Jl. Veteran No.10-11, Ketawanggede, Kec. Lowokwaru, Kota Malang, Jawa Timur 65145

febystudentub@gmail.com

Abstrack -- This practicum aims to implement the Iris dataset classification model locally on the ESP32 microcontroller using Tiny Machine Learning (TinyML) technology with the help of the TensorFlow Lite for Microcontrollers (TFLM) library. The model used is a classification model with three classes (Setosa, Versicolor, Virginica) based on four feature parameters (sepal and petal length and width). The model was pre-trained and converted into a TensorFlow Lite format that can be run on microcontroller devices. In the inference process, the model received four numerical inputs and predicted the class of Iris flowers locally without an internet connection. The prediction results were displayed via a serial monitor, along with the time taken to run each prediction. Tests were conducted on the Wokwi simulation platform, which showed that the system was able to perform inference quickly and efficiently. This project proves that the ESP32 has the ability to run simple machine learning models locally, which opens up vast opportunities for the development of AI applications on edge devices.

Keyword : ***TinyML, ESP32, TensorFlow Lite, Iris, Machine Learning, Embedded AI, Inferensi Lokal***

Abstrak **--** Praktikum ini bertujuan untuk mengimplementasikan model klasifikasi dataset *Iris* secara lokal pada mikrokontroler ESP32 menggunakan teknologi *Tiny Machine Learning* (TinyML) dengan bantuan *library TensorFlow Lite for Microcontrollers (TFLM).* Model yang digunakan adalah model klasifikasi dengan tiga kelas *(Setosa, Versicolor, Virginica)* berdasarkan empat parameter fitur (panjang dan lebar sepal serta petal). Model telah dilatih sebelumnya dan dikonversi menjadi format TensorFlow Lite yang dapat dijalankan pada perangkat mikrokontroler. Dalam proses inferensi, model menerima empat input numerik dan memprediksi kelas dari bunga Iris secara lokal tanpa koneksi internet. Hasil prediksi ditampilkan melalui serial monitor, bersamaan dengan waktu yang dibutuhkan untuk menjalankan setiap prediksi. Pengujian dilakukan pada *platform* simulasi Wokwi, yang menunjukkan bahwa sistem mampu menjalankan inferensi dengan cepat dan efisien. Proyek ini membuktikan bahwa *ESP32* memiliki kemampuan untuk menjalankan model pembelajaran mesin sederhana secara lokal, yang membuka peluang luas bagi pengembangan aplikasi *AI* di perangkat *edge.*

Kata Kunci : ***TinyML, ESP32, TensorFlow Lite, Iris, Machine Learning, Embedded AI, Inferensi Lokal***

I. PENDAHULUAN

Perkembangan teknologi machine learning telah merambah hingga ke perangkat mikrokontroler melalui konsep Tiny Machine Learning (TinyML), yang memungkinkan model-model AI dijalankan secara lokal di perangkat kecil dan hemat daya. Salah satu platform yang mendukung TinyML adalah TensorFlow Lite for Microcontrollers (TFLM), yang dirancang untuk menjalankan model inferensi tanpa perlu sistem operasi atau koneksi internet.

ESP32 merupakan salah satu mikrokontroler yang cukup kuat untuk digunakan dalam implementasi TinyML karena memiliki prosesor dual-core dan RAM yang relatif besar dibandingkan dengan mikrokontroler lainnya. Dalam proyek ini, ESP32 digunakan untuk menjalankan model klasifikasi dataset Iris, sebuah dataset terkenal dalam pembelajaran mesin yang berisi informasi morfologi dari tiga jenis bunga iris: Setosa, Versicolor, dan Virginica. Model ini menerima empat input numerik berupa panjang dan lebar dari sepal dan petal, dan memberikan prediksi kelas dari bunga tersebut. Dengan menjalankan inferensi secara langsung di ESP32.

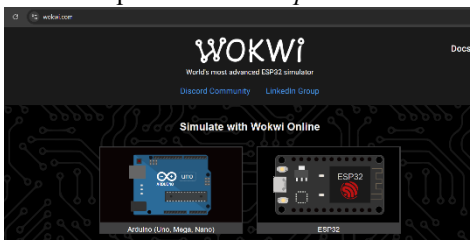
II. METODOLOGI

A. ALAT & BAHAN

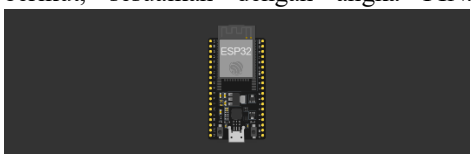
Mikrokontroler *(ESP32),* Library eloquent_tinyml, Library tflm_esp32, Dataset *Iris* yang telah dilatih dan dikonversi ke format .tflite *Visual Studio Code* (C++*), PlatformIO IDE. Koneksi Wi-Fi Wokwi-GUEST), Komputer dengan software Arduino IDE.*

B. LANGKAH IMPLEMENTASI

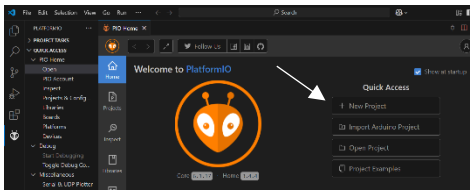
1. Buka https://wokwi.com/ pilih ESP32, kemudian pilih *Starter Templates.*



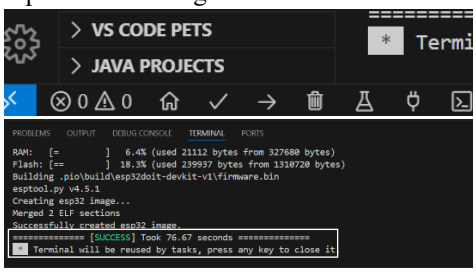
2. Atur rangkain diagram seperti gambar berikut, sesuaikan dengan angka PIN.



3. Buka VScode, kemudian *install PlatformIO IDE.* Buat proyek baru seperti gambar berikut:



4. *Build Wokwi* dengan cara klik centang pada bagian bawah dari *Visual Studio Code,* seperti gambar berikut.



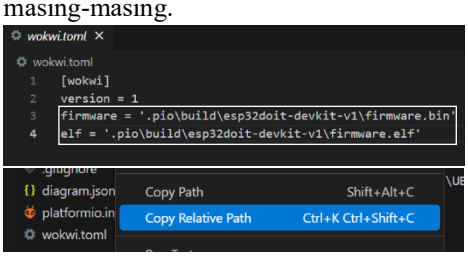
5. Buat *file* baru pada *Visual Studio Code*, kemudian buka *wokwi.toml* pada *google*, kemudian *copy* ke *file* baru yang diberi nama *wokwi.toml* di *VScode.*

wokwi.toml

A basic `wokwi.toml` file looks like this:

```
[wokwi]
version = 1
firmware = 'path-to-your-firmware.hex'
elf = 'path-to-your-firmware.elf'
```

6. Setelah itu ganti menjadi *firmware bin*, dan *esp32 firmware elf* pada *Visual Studio Code* masing-masing.



7. Buat *file* baru bernama *diagram.json* pada *Visual Studio Code.* Kemudian pergi ke *wokwi* untuk meng *copy code diagram.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": {}
}
```

8. *Paste* ke *Visual Studio Code,* pada *file diagram.json. Jika diagram* tidak berfungsi, maka lakukan edit pada *diagram.json*, kemudian *paste code* yang sudah di *copy* pada *wokwi.*

9. Kemudian *wokwi* akan menampilkan hasil dari *diagram.json* pada simulasi *Temperature dan Humidity.*

10. Buat code C++ pada main

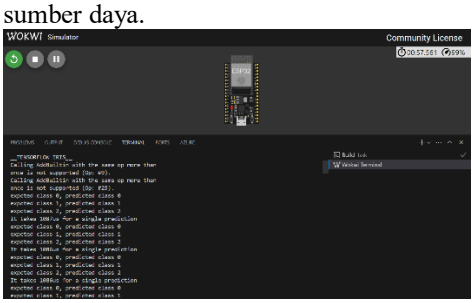
```
#include <Arduino.h>
#include <iris_model.h>
#include <tflm_esp32.h>
#include <eloquent_tinyml.h>

#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);
}
```

III. HASIL DAN PEMBAHASAN

1. Integrasi model *TensorFlow Lite* pada mikrokontroler *ESP32* menggunakan simulator Wokwi berjalan dengan lancar. Waktu inferensi yang konsisten di kisaran 1086 mikrodetik membuktikan bahwa *ESP32* mampu menangani komputasi machine learning sederhana secara real-time. Ini membuka peluang besar untuk implementasi kecerdasan buatan langsung di perangkat *IoT* tanpa perlu bergantung pada server eksternal, yang berarti sistem dapat lebih hemat daya, cepat, dan mandiri dalam pengambilan keputusan.

2. *TensorFlow Lite* yang dijalankan pada b*oard ESP32* berhasil melakukan proses klasifikasi terhadap data uji *IRIS* dengan baik. Meskipun muncul peringatan terkait penggunaan operasi AddBuiltin lebih dari satu kali, hal ini tidak mengganggu jalannya inferensi. Hasil prediksi menunjukkan bahwa model mampu memetakan data uji ke kelas yang sesuai, misalnya “expected class 0, predicted class 0”, dan seterusnya, dengan waktu rata-rata inferensi sekitar 1086 mikrodetik per prediksi. Hal ini menunjukkan bahwa model dapat berjalan cukup cepat dan efisien pada perangkat *edge* seperti *ESP32,* sehingga cocok untuk aplikasi embedded *AI* dengan keterbatasan sumber daya.



IV. KESIMPULAN

Setelah melakukan perancangan dan pengujian terhadap alat secara keseluruhan. Maka dapat diambil kesimpulan :

1. Perangkat yang digunakan oleh penulis dapat bekerja dengan baik sesuai dengan yang diharapkan.
2. Mikrokontroler *ESP32* yang digunakan sebagai pengendali utama dapat bekerja dalam menjalankan perintah yang diberikan.

V. DAFTAR PUSTAKA

Loquent Arduino. (2022). *TinyML for Arduino and ESP32 with TensorFlow Lite Models*. https://eloquentarduino.github.io/

Espressif Systems. (2023). *ESP32 Technical Reference Manual*. Retrieved from https://www.espressif.com/en/products/socs/esp32/resources

Fisher, R. A. (1936). *The use of multiple measurements in taxonomic problems.*