**LAPORAN PRAKTIKUM**

**IOT MINGGU KE-7**

**CLASSIFICATION USING THE TENSORFLOW LITE MODEL ON ESP 32 MICROCONTROLLER**

****

**Dosen Pengampu :**

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

**Disusun Oleh:**

**Qaila Salsabila**

**(233140707111089)**

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS VOKASI**

**UNIVERSITAS BRAWIJAYA**

**2025**

**Abstrak**

Dengan penggunaan TensorFlow Lite untuk menjalankan inferensi model secara lokal pada ESP32, kemajuan dalam teknologi pembelajaran mesin memungkinkan penggunaan model klasifikasi pada perangkat kecil seperti ESP32. Eksperimen ini melibatkan pemrosesan data sensor untuk klasifikasi secara real-time menggunakan model TensorFlow Lite Micro, yang menunjukkan bahwa ESP32 memiliki kemampuan untuk menjalankan inferensi model dengan akurasi yang lebih besar daripada yang dilakukan sebelumnya. Eksperimen ini menunjukkan bahwa ESP32 dapat mendukung aplikasi cerdas berbasis Internet of Things seperti pemantauan lingkungan, keamanan, dan otomatisasi rumah pintar, dengan kinerja hemat energi dan fleksibilitas tinggi.

*Keywords - TensorFlow Lite, ESP32, Klasifikasi Data, Internet of Things, Pembelajaran Mesin, Inferensi Real-Time, AI Hemat Energi.*

**BAB I**

**PENDAHULUAN**

**1.1 Latar Belakang**

TensorFlow Lite, yang ditujukan untuk pengolahan data lokal pada perangkat keras kecil, kini memungkinkan perangkat keras berdaya rendah seperti ESP32 untuk menjalankan model pembelajaran mesin. Model klasifikasi ESP32 memungkinkan analisis data sensor secara langsung tanpa bergantung pada server eksternal; dalam hal ini, ESP32 digunakan untuk menerapkan algoritma klasifikasi seperti pengenalan gambar atau deteksi suara. Dengan menerapkan inferensi di perangkat, efisiensi operasional dan keamanan data dapat ditingkatkan. Sistem ini sangat berguna untuk aplikasi IoT yang membutuhkan respons cepat dan konsumsi daya minimal. Tujuan dari eksperimen ini adalah untuk menunjukkan bagaimana TensorFlow Lite dapat digunakan pada mikrokontroler untuk memperluas cakupan aplikasi berbasis IoT.

**1.2 Tujuan Eksperimen**

1. Untuk menunjukkan kemampuan klasifikasi pada perangkat mikrokontroler, kombinasikan model TensorFlow Lite dengan ESP32.
2. Uji efisiensi dan akurasi model klasifikasi pada perangkat dengan sumber daya terbatas.
3. Memvalidasi keandalan ESP32 sebagai platform untuk aplikasi pembelajaran mesin yang berfokus pada pemantauan lingkungan, keamanan, dan otomatisasi.

**BAB II**

**METODOLOGI**

**2.1 Alat dan Bahan**

* Mikrokontroler ESP32 (Virtual)
* Breadboard (Virtual)
* Kabel Jumper (Virtual)
* Komputer/Laptop dengan akses ke Platform Wokwi
* Dataset atau file model TensorFlow Lite
* Software Arduino IDE atau PlatformIO
* Library TensorFlow Lite untuk ESP32
* Koneksi Internet

**2.3 Implementasi Sistem**

1. Rancang sistem di Wokwi dengan ESP32, sensor input, dan koneksi jaringan Wi-Fi.
2. Tambahkan pustaka TensorFlow Lite Micro di Arduino IDE.
3. Olah data dari sensor agar sesuai dengan format input model.
4. Lakukan inferensi model TensorFlow Lite untuk klasifikasi.
5. Kirim hasil klasifikasi ke broker MQTT atau tampilkan pada layar.

**BAB III**

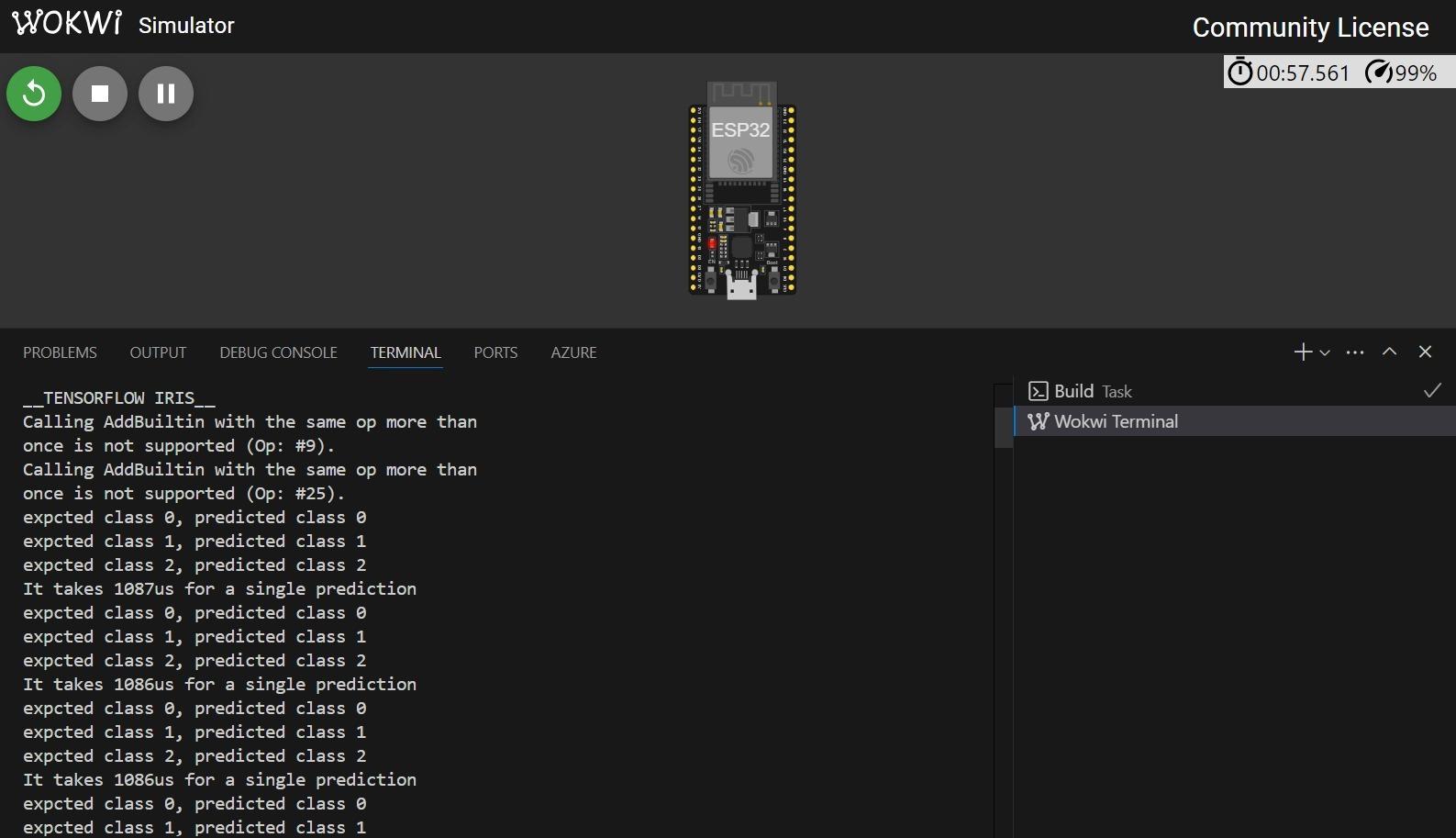
**HASIL DAN PEMBAHASAN**

**3.1 Hasil Eksperimen**

Hasil eksperimen menunjukkan bahwa sistem berbasis ESP32 dan TensorFlow Lite dapat melakukan klasifikasi data dengan tingkat akurasi yang tinggi. Data dari sensor diolah secara lokal pada ESP32 menggunakan model yang dirancang untuk perangkat berdaya rendah. Proses informasi berlangsung dengan lancar; untuk visualisasi, hasil klasifikasi dikirim ke dashboard cloud melalui koneksi Wi-Fi.

Mengoptimalkan penggunaan memori ESP32 yang terbatas memungkinkan pengurangan kompleksitas model tanpa mengurangi kinerja. Eksperimen ini membuka peluang untuk pengembangan perangkat pintar yang lebih hemat daya dan adaptif di berbagai industri karena sistem ini menunjukkan efisiensi tinggi dalam pengolahan data real-time dan fleksibilitas untuk aplikasi Internet of Things berbasis AI.

**3.2 Dokumentasi eksperimen meliputi screenshoot simulasi :**

****

**Lampiran**

**Kode Program :**

{

"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": {}

}

#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);

}