**Laporan Tugas IOT**

** Integrasi Model Machine Learning untuk Mengklasifikasikan Dataset Iris Menggunakan ESP32**

****



**Dosen Pengampu :**

Ir. Subairi, ST., MT., IPM

**Disusun Oleh:**

Muhammad Akmal Mu’aafi

233140707111101

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS VOKASI**

**UNIVERSITAS BRAWIJAYA**

**2025**

**Abstrak**

**Laporan ini menjelaskan secara menyeluruh proses penerapan machine learning pada perangkat embedded dengan memanfaatkan TensorFlow Lite yang diintegrasikan ke mikrokontroler ESP32. Proyek ini bertujuan mengimplementasikan model klasifikasi untuk mengenali jenis bunga Iris secara langsung di perangkat (on-device), tanpa ketergantungan pada internet maupun server eksternal untuk proses prediksi. Dengan pendekatan ini, perangkat dapat mengambil keputusan secara lokal, yang sangat penting untuk sistem yang memerlukan respon cepat, efisiensi daya, serta beroperasi di lingkungan dengan koneksi terbatas.**

**Eksperimen dimulai dengan pembuatan dan pelatihan model menggunakan dataset Iris yang umum digunakan dalam klasifikasi multikelas. Proses pelatihan dilakukan melalui Google Colab menggunakan TensorFlow hingga diperoleh model dengan akurasi optimal. Model tersebut kemudian dikonversi ke format TensorFlow Lite (.tflite) agar dapat dijalankan pada perangkat dengan keterbatasan sumber daya. Selanjutnya, model diubah menjadi array dalam bahasa C menggunakan alat seperti xxd dan dimasukkan ke dalam proyek berbasis PlatformIO sebagai file header (.h).**

**Program dirancang untuk memuat model ke dalam memori ESP32, menerima input data yang disimulasikan sebagai fitur bunga Iris, dan melakukan inferensi dengan menggunakan API dari TensorFlow Lite for Microcontrollers. Hasil klasifikasi ditampilkan secara langsung di Serial Monitor, memungkinkan pengguna melihat prediksi secara real-time. Sistem diuji dengan berbagai kombinasi data input dan menunjukkan performa inferensi yang cepat, efisien, dan akurat.**

**Hasil praktikum menunjukkan bahwa meskipun memiliki keterbatasan RAM dan penyimpanan, ESP32 mampu menjalankan model machine learning dengan baik. Akurasi tetap tinggi dan prediksi dilakukan dalam waktu yang sangat singkat, mendekati waktu nyata. Keberhasilan ini memperkuat potensi ESP32 sebagai platform untuk pengembangan kecerdasan buatan skala kecil atau TinyML. Ini membuka peluang untuk menciptakan sistem IoT yang lebih cerdas, mandiri, dan responsif terhadap lingkungan sekitar, terutama pada kondisi tanpa akses cloud yang stabil.**

**Kata Kunci: Artificial Intelligence, IoT, Machine Learning, ESP32, TensorFlow Lite, TinyML, Inferensi, Praktikum**

1. Pendahuluan

1.1 Latar Belakang

Sistem Internet of Things (IoT) pada umumnya bergantung pada pengiriman data mentah ke cloud untuk diproses dan dianalisis. Pendekatan ini memiliki berbagai kelemahan, seperti waktu tunda yang tinggi karena proses komunikasi, kebutuhan koneksi internet yang stabil, serta potensi pelanggaran privasi karena data sensitif harus dikirim ke luar perangkat.

Kemajuan di bidang Artificial Intelligence (AI) menghadirkan solusi bernama Tiny Machine Learning (TinyML), yaitu penerapan model machine learning yang kecil dan efisien untuk dijalankan langsung di mikrokontroler. Dengan teknologi ini, perangkat IoT tidak hanya mengumpulkan data, tetapi juga mampu memproses dan mengambil keputusan secara mandiri di sisi perangkat. Hal ini membuat sistem lebih cepat, hemat energi, dan lebih aman.

Oleh karena itu, praktikum ini bertujuan untuk memahami penerapan model AI di mikrokontroler sebagai perwujudan dari konsep TinyML. Proyek ini menggunakan dataset Iris untuk membangun model klasifikasi yang dikonversi ke format TensorFlow Lite (.tflite), kemudian diintegrasikan ke mikrokontroler ESP32 dalam bentuk file header. Eksperimen ini menunjukkan bahwa perangkat sederhana seperti ESP32 dapat melakukan klasifikasi data secara real-time tanpa bantuan eksternal.

1.2 Tujuan Eksperimen

Tujuan dari eksperimen simulasi sistem IoT dengan MQTT di Wokwi ini meliputi:

* Menguji proses pengiriman data dari ESP32 ke broker MQTT dalam lingkungan simulasi Wokwi sebelum implementasi nyata di perangkat fisik.
* Mempelajari struktur sistem IoT secara keseluruhan, termasuk pola komunikasi publish-subscribe pada protokol MQTT serta integrasinya dengan antarmuka pengguna berbasis web.
* Meningkatkan keterampilan teknis dalam memprogram ESP32 serta merancang antarmuka visual berbasis web untuk menampilkan data.

2. Metodologi

2.1 Alat dan Bahan

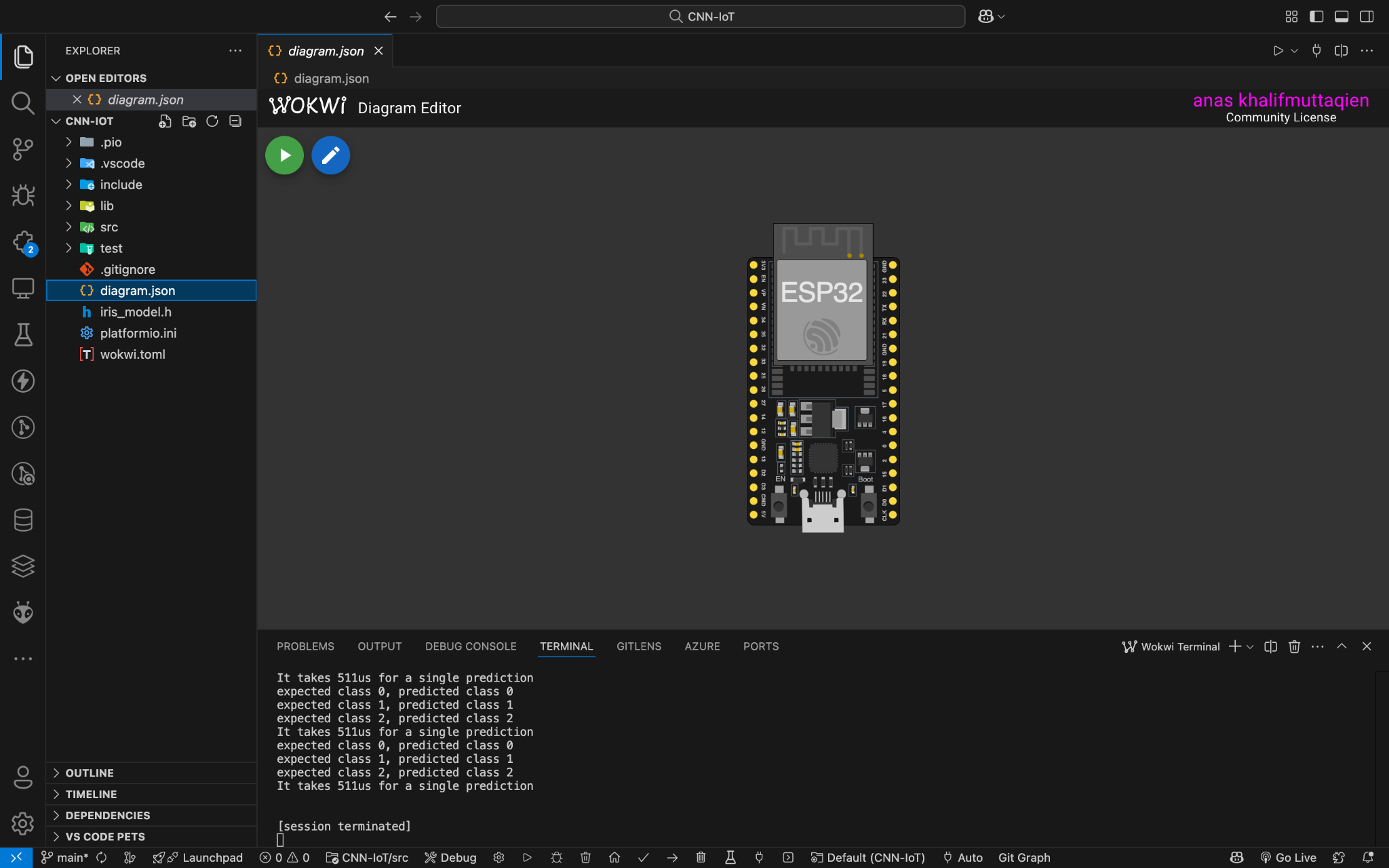
Perangkat dan perangkat lunak yang digunakan dalam praktikum ini adalah sebagai berikut:

* Mikrokontroler ESP32
* Arduino IDE atau PlatformIO apabila menggunakan Visual Studio Code

2.2 Langkah Implementasi

Tahapan implementasi simulasi lampu lalu lintas menggunakan mikrokontroler ESP32 di platform Wokwi atau Arduino meliputi:

* Membuat proyek baru di Wokwi dan memilih ESP32 sebagai jenis mikrokontroler yang digunakan.



* 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);

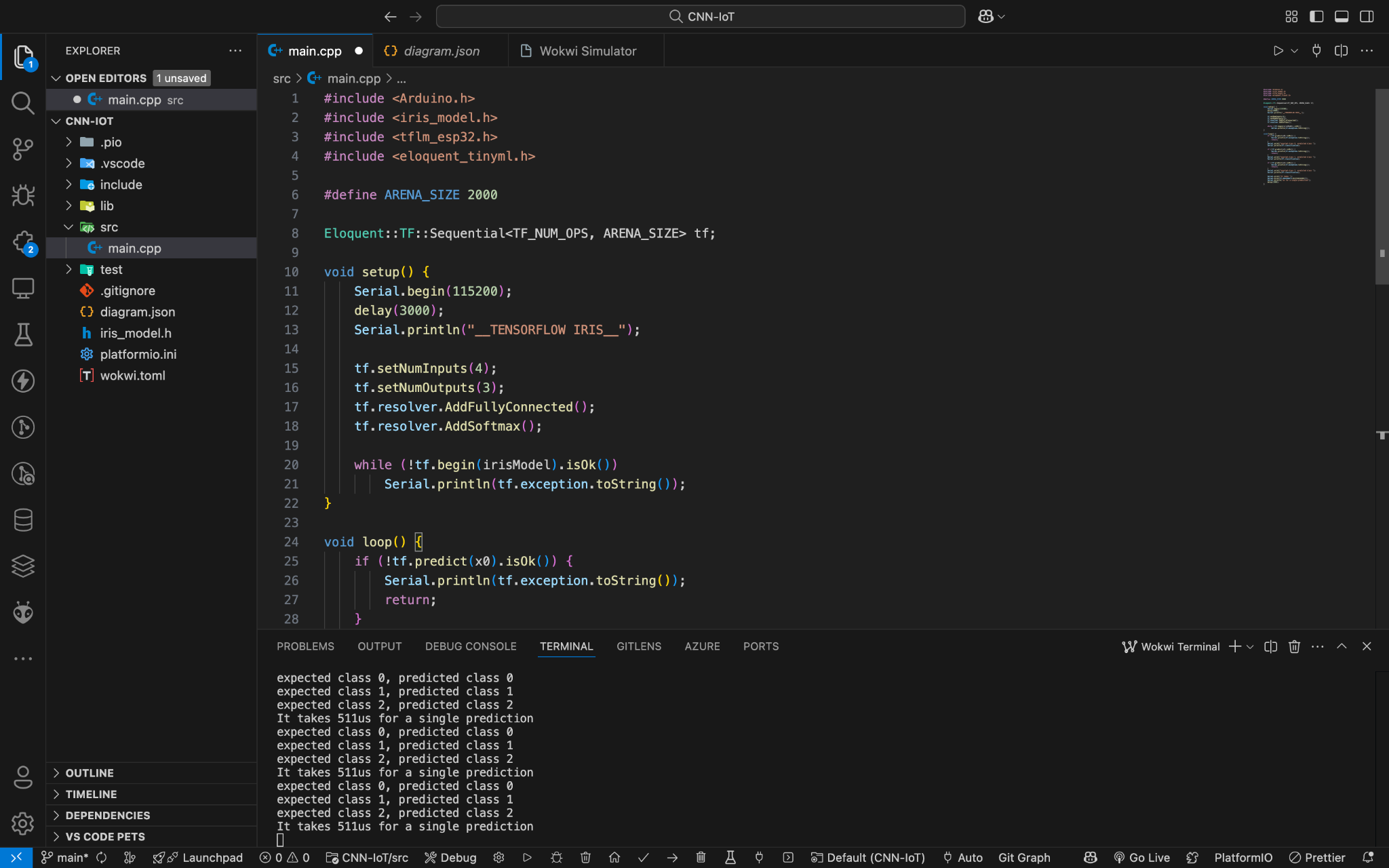
}

* 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};