COMPUTER VISION DAN TINYML

Modul VIII Praktikum Sistem Tertanam

INSTITUT TEKNOLOGI SUMATERA
LAMPUNG SELATAN
2022

I. Daftar Isi

II. Persiapan Praktikum

A. Alat & Bahan

1. ESP32-CAM
2. FTDI **(cukup salah satu)**
3. ESP32-CAM-MB **(cukup salah satu)**
4. Kabel jumper
5. Breadboard
6. **2 jenis objek** untuk model klasifikasi (misal buah apel dan jeruk)
7. Jaringan WiFi (LAN dan internet)

B. Catatan/Kebutuhan Lainnya

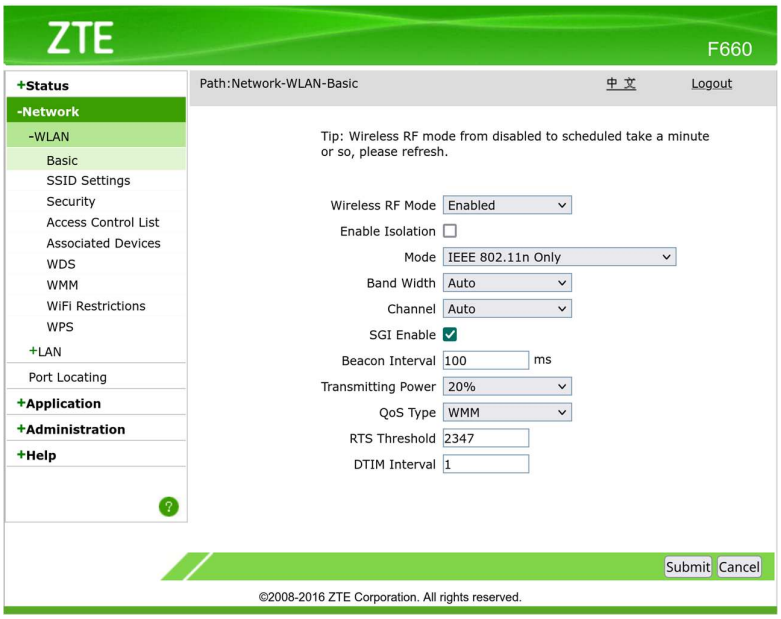
1. Edge Impulse

   Silahkan membuat akun Edge Impulse terlebih dahulu pada situs berikut ini. Edge Impulse akan digunakan untuk proses training model klasifikasi secara mudah tanpa perlu pengetahuan yang mendalam mengenai machine learning.

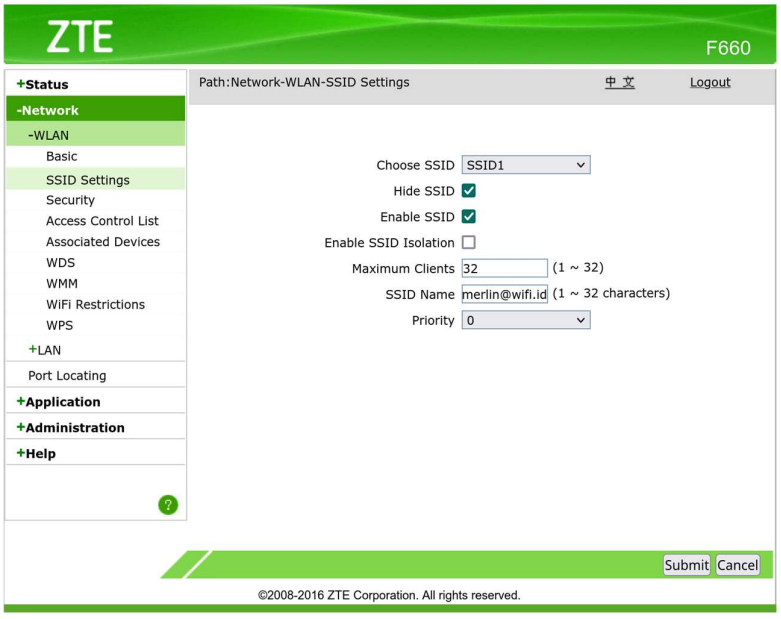
2. Jaringan WiFi

   Pastikan terdapat koneksi WiFi yang memadai. LAN digunakan untuk mengambil video/gambar dari ESP32-CAM (melalui IP lokal) dan internet digunakan untuk mengakses Edge Impulse dan/atau referensi lainnya. Bila IP lokal tidak dapat diakses oleh perangkat kalian, pastikan WiFi tidak dalam mode isolasi. Pada router ZTE F660 milik IndiHome, mode isolasi dapat dimatikan dengan cara berikut:

   - Akses halaman login WiFi (misal http://192.168.1.1) dan masukan credential kalian
   - Pada sidebar, navigasikan ke “network” -> “WLAN” -> “Basic” dan klik “Enable Isolation” agar tidak tercentang



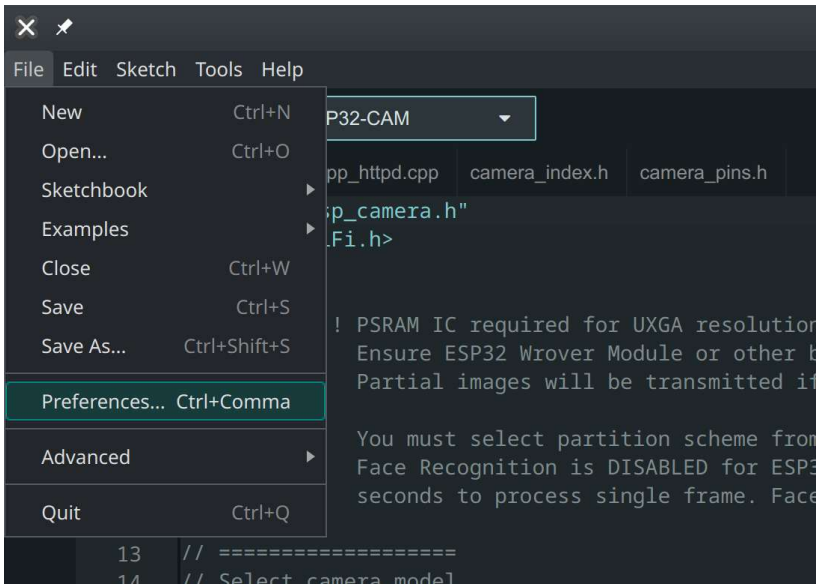
- Pada sidebar, navigasikan lagi ke “network” -> “WLAN” -> “SSID Settings” dan klik “Enable SSID Isolation” agar tidak tercentang



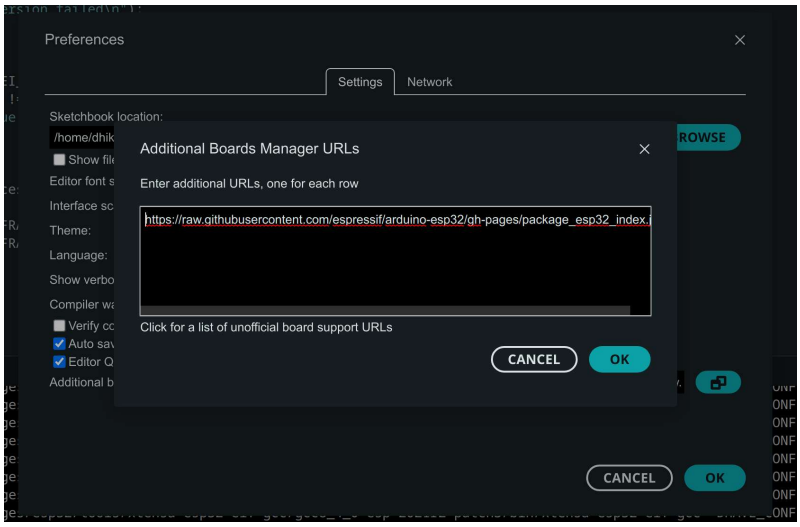
3. Pemasangan Board ESP32 di Arduino IDE

   Bila board ESP32 ada pada daftar board di Arduino IDE, silahkan lakukan langkah-langkah berikut:

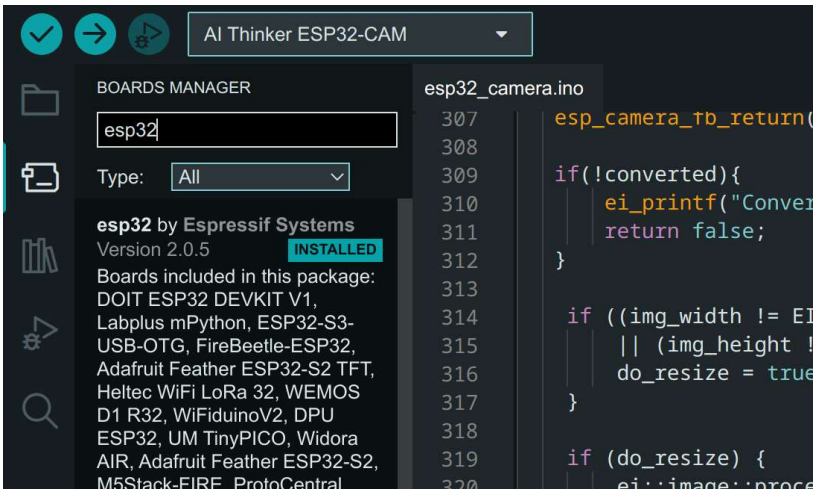
   - Pada bagian atas Arduino IDE, pilih menu “File” -> “Preferences”



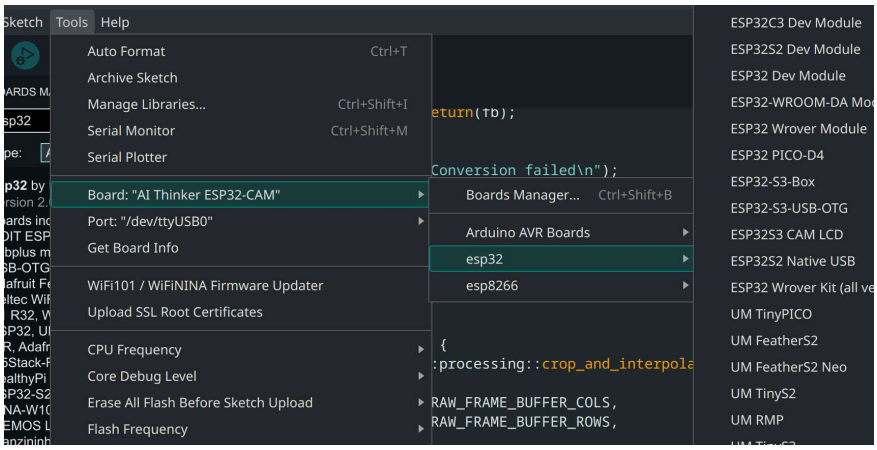
- Saat sudah terbuka popup “Preferences”, tambahkan URL baru di bagian “Additional boards manager URLs”, URL tersebut yaitu: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json



- Klik ok dan tunggu sampai proses download data board selesai
- Klik bagian “Boards Manager” pada sidebar Arduino IDE dan install board ESP32 (bila belum terpasang)



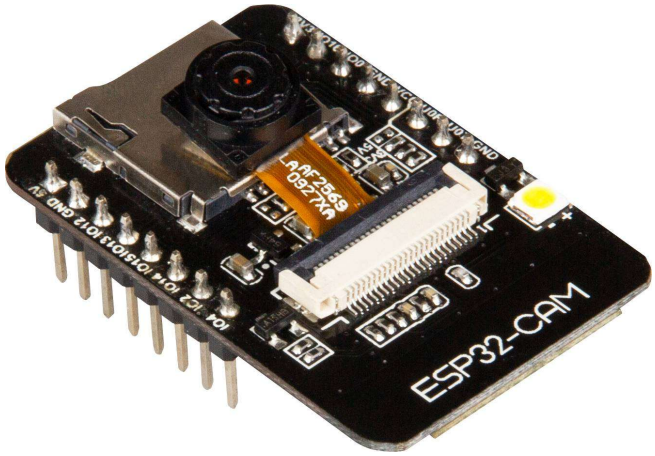
- Setelah board ESP32 terpasang, maka board tersebut akan dapat dipilih untuk proses compile dan upload kode



III. Pendahuluan

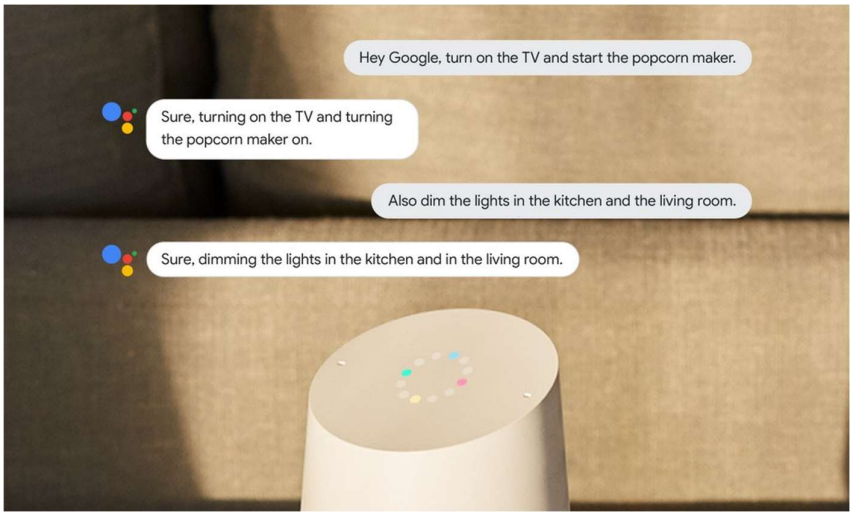
A. ESP32-CAM

ESP32-CAM merupakan sebuah modul sekaligus mikrokontroler terjangkau yang dapat mengambil gambar dan/atau video menggunakan kamera OV2640 [1]. Selain dapat mengambil gambar dan/atau video, modul ini juga dilengkapi dengan slot MicroSD dan transfer data via WiFi dan bluetooth [1]. Untuk membaca atau menulis kode ke ESP32-CAM, dapat digunakan modul bantuan seperti FTDI programmer ataupun mikrokontroler lainnya seperti Arduino Uno [2].



B. Machine Learning dan TinyML

Machine learning (ML) merupakan salah satu bidang ilmu yang berfokuskan pada pemanfaatan mesin untuk menyelesaikan masalah dan berimprovisasi seiring berjalannya waktu secara otomatis [3]. Melalui perkembangan teknologi, konsep ML ini terus berkembang dari yang tadinya hanya bersifat mahal dan eksperimental hingga bisa dimanfaatkan secara luas. TinyML merupakan salah satu cabang ML yang memungkinkan terjadinya ML melalui suatu perangkat (*embedded system*) dengan spesifikasi terbatas [4]. Karena spesifikasi dan arsitektur sistem yang lebih terbatas, TinyML umumnya juga memanfaatkan berbagai kombinasi ilmu dan teknologi seperti *edge/cloud computing* dan *transfer learning* [5]. Saat ini, TinyML telah banyak dimanfaatkan pada berbagai bidang seperti rumah tangga (asisten digital), transportasi (kendaraan otonom), layanan kesehatan, dll.



C. Computer Vision

Computer vision merupakan salah satu bidang ilmu yang melibatkan bagaimana suatu mesin/komputer dapat “melihat” [6]. Proses ini umumnya melibatkan konsep Artificial Intelligence dan Machine Learning agar mesin tersebut dapat bekerja seperti penglihatan manusia [7]. Computer vision memiliki beberapa tahapan yang meliputi 3 proses, yaitu object recognition, object detection, dan scene understanding [7]. Untuk dapat mengenali objek dengan deskripsi dan fitur-fitur yang luas (*feature extraction*), *deep learning* juga umum diimplementasikan pada *computer vision*. Salah satu algoritma deep learning yang paling banyak digunakan adalah CNN dengan berbagai modelnya, seperti MobileNet, AlexNet, VGGNet, GoogleNet, dll [7].

IV. Percobaan

A. Camera Web Server

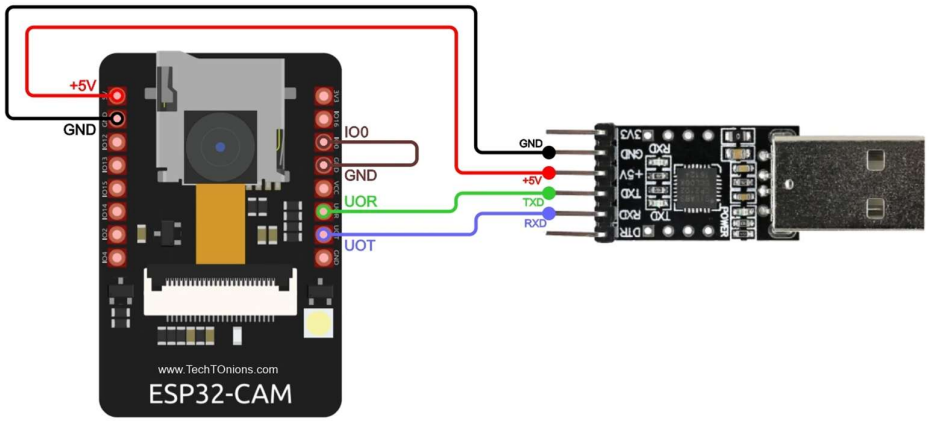
Pada percobaan ini, kita akan mencoba mencoba mengambil gambar atau video menggunakan ESP32-CAM.

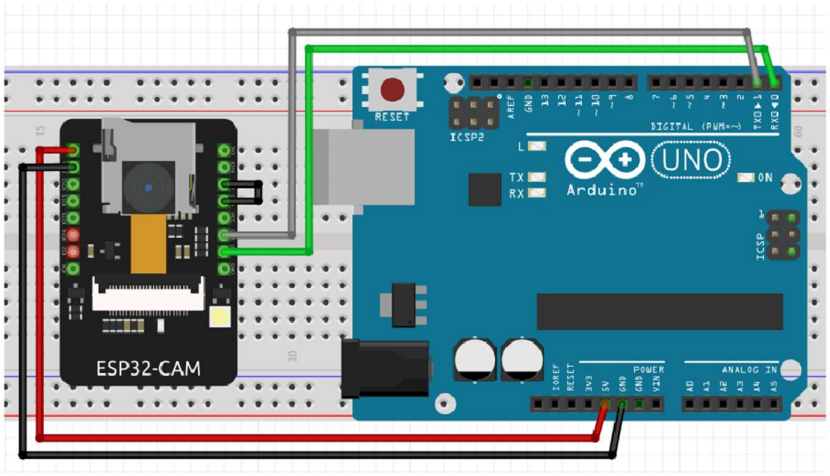
Persiapan

Pada percobaan ini, tidak ada library eksternal yang dibutuhkan. Namun, pastikan bahwa board ESP32 telah terpasang pada Arduino IDE agar dapat melakukan *upload* dan *flashing* kode.

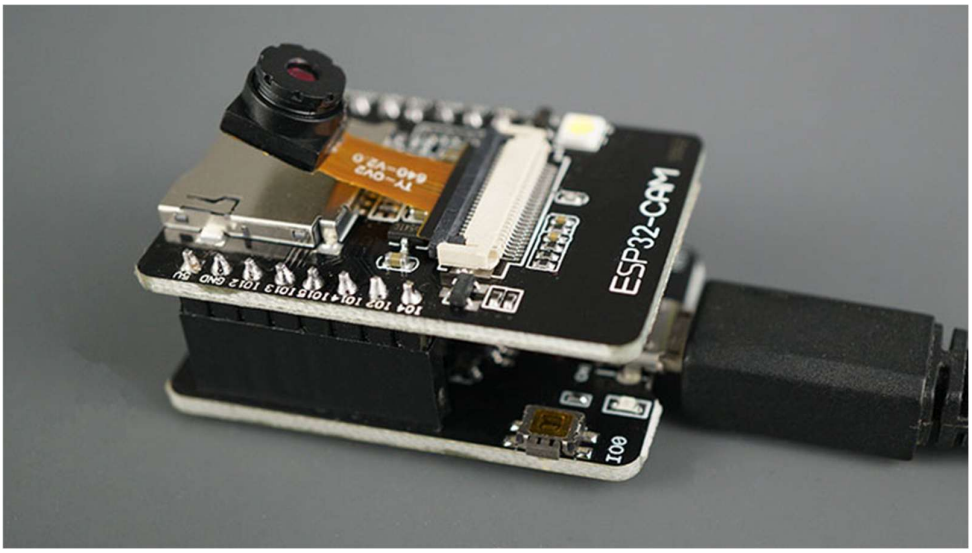
Rangkaian dan Kode

Mula-mula, buatlah rangkaian seperti gambar berikut bila menggunakan modul FTDI dan/atau mikrokontroler pada umumnya.





Bila menggunakan modul ESP32-CAM-MB, langsung saja pasangkan ke ESP32-CAM seperti pada gambar di bawah ini.



Selanjutnya, bila board ESP32 telah terpasang pada Arduino IDE, kode web server kamera dapat diakses dengan memilih **"File" -> "Examples" -> "ESP32" -> "Camera" -> "CameraWebServer"**. Kode utama (CameraWebServer.ino) kurang lebih akan terlihat seperti di bawah ini:

```
#include "esp_camera.h"
#include <WiFi.h>

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is selected
//            Partial images will be transmitted if image exceeds buffer size
//
//            You must select partition scheme from the board menu that has at least 3MB APP space.
//            Face Recognition is DISABLED for ESP32 and ESP32-S2, because it takes up from 15
//            seconds to process single frame. Face Detection is ENABLED if PSRAM is enabled as well

// ===================
// Select camera model
// ===================
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
//#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
// ** Espressif Internal Boards **
//#define CAMERA_MODEL_ESP32_CAM_BOARD
//#define CAMERA_MODEL_ESP32S2_CAM_BOARD
//#define CAMERA_MODEL_ESP32S3_CAM_LCD
```

```
30.
31. #include "camera_pins.h"
32.
33. // ===========================
34. // Enter your WiFi credentials
35. // ===========================
36. const char* ssid = "**********";
37. const char* password = "**********";
38.
39. void startCameraServer();
40.
41. void setup() {
42.  Serial.begin(115200);
43.  Serial.setDebugOutput(true);
44.  Serial.println();
45.
46.  camera_config_t config;
47.  config.ledc_channel = LEDC_CHANNEL_0;
48.  config.ledc_timer = LEDC_TIMER_0;
49.  config.pin_d0 = Y2_GPIO_NUM;
50.  config.pin_d1 = Y3_GPIO_NUM;
51.  config.pin_d2 = Y4_GPIO_NUM;
52.  config.pin_d3 = Y5_GPIO_NUM;
53.  config.pin_d4 = Y6_GPIO_NUM;
54.  config.pin_d5 = Y7_GPIO_NUM;
55.  config.pin_d6 = Y8_GPIO_NUM;
56.  config.pin_d7 = Y9_GPIO_NUM;
57.  config.pin_xclk = XCLK_GPIO_NUM;
58.  config.pin_pclk = PCLK_GPIO_NUM;
59.  config.pin_vsync = VSYNC_GPIO_NUM;
60.  config.pin_href = HREF_GPIO_NUM;
61.  config.pin_sscb_sda = SIOD_GPIO_NUM;
62.  config.pin_sscb_scl = SIOC_GPIO_NUM;
63.  config.pin_pwdn = PWDN_GPIO_NUM;
64.  config.pin_reset = RESET_GPIO_NUM;
65.  config.xclk_freq_hz = 20000000;
66.  config.frame_size = FRAMESIZE_UXGA;
67.  config.pixel_format = PIXFORMAT_JPEG; // for streaming
68.  //config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
69.  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
70.  config.fb_location = CAMERA_FB_IN_PSRAM;
71.  config.jpeg_quality = 12;
72.  config.fb_count = 1;
73.
74.  // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
75.  //                     for larger pre-allocated frame buffer.
76.  if(config.pixel_format == PIXFORMAT_JPEG){
77.    if(psramFound()){
78.      config.jpeg_quality = 10;
79.      config.fb_count = 2;
80.      config.grab_mode = CAMERA_GRAB_LATEST;
81.    } else {
82.      // Limit the frame size when PSRAM is not available
83.      config.frame_size = FRAMESIZE_SVGA;
84.      config.fb_location = CAMERA_FB_IN_DRAM;
85.    }
86.  } else {
87.    // Best option for face detection/recognition
88.    config.frame_size = FRAMESIZE_240X240;
89. #if CONFIG_IDF_TARGET_ESP32S3
90.    config.fb_count = 2;
91. #endif
92.  }
93.
94. #if defined(CAMERA_MODEL_ESP_EYE)
95.  pinMode(13, INPUT_PULLUP);
```

```
 pinMode(14, INPUT_PULLUP);
#endif

 // camera init
    esp_err_t err = esp_camera_init(&config);
    if (err != ESP_OK) {
      Serial.printf("Camera init failed with error 0x%x", err);
      return;
    }

    sensor_t * s = esp_camera_sensor_get();
    // initial sensors are flipped vertically and colors are a bit saturated
    if (s->id.PID == OV3660_PID) {
      s->set_vflip(s, 1); // flip it back
      s->set_brightness(s, 1); // up the brightness just a bit
      s->set_saturation(s, -2); // lower the saturation
    }
    // drop down frame size for higher initial frame rate
    if(config.pixel_format == PIXFORMAT_JPEG){
      s->set_framesize(s, FRAMESIZE_QVGA);
    }

   #if defined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)
    s->set_vflip(s, 1);
    s->set_hmirror(s, 1);
   #endif

   #if defined(CAMERA_MODEL_ESP32S3_EYE)
    s->set_vflip(s, 1);
   #endif

    WiFi.begin(ssid, password);
    WiFi.setSleep(false);

    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");

    startCameraServer();

    Serial.print("Camera Ready! Use 'http://");
    Serial.print(WiFi.localIP());
    Serial.println("' to connect");
   }

   void loop() {
    // Do nothing. Everything is done in another task by the web server
    delay(10000);
   }
```

Upload kode tersebut ke mikrokontroler (pilih **AI-Thinker ESP32-CAM**) dan akses alamat IP yang keluar di Serial Monitor dengan menggunakan web browser. Ambil beberapa gambar untuk 2 objek yang berbeda (minimal 10 buah untuk masing-masing objek) agar dapat digunakan pada percobaan berikutnya.

Pertanyaan / Asprak Bertanya

1. Bagaimana cara kerja ESP32-CAM?
2. Apa fungsi dari “WiFi.setSleep(false)” pada kode? Adakah pengaruhnya bila parameter fungsi tersebut diubah ke “true”?
3. Apa fungsi dari “startCameraServer()” pada kode? Apa yang terjadi bila fungsi tersebut dihapus?
4. Bagaimana cara mengatur ukuran gambar kamera yang digunakan pada kode program?
5. Adakah cara untuk mengambil gambar dari ESP32-CAM secara cepat tanpa melalui tombol “Get Still” pada halaman IP kamera?

B. Object Classification

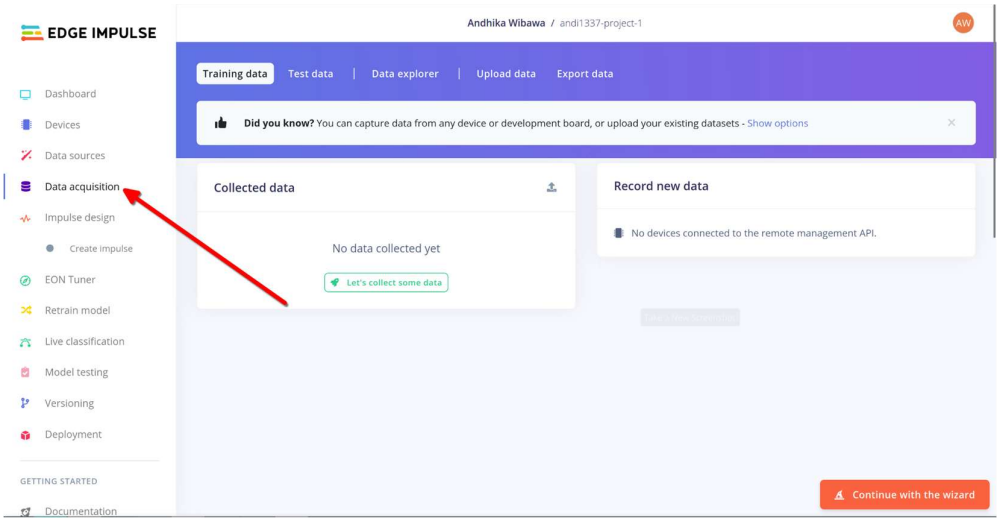
Pada percobaan ini, kita akan mencoba mengklasifikan objek pada gambar atau video menggunakan ESP32-CAM dan bantuan Edge Impulse. Referensi percobaan dapat juga dilihat pada link berikut ini.

Persiapan

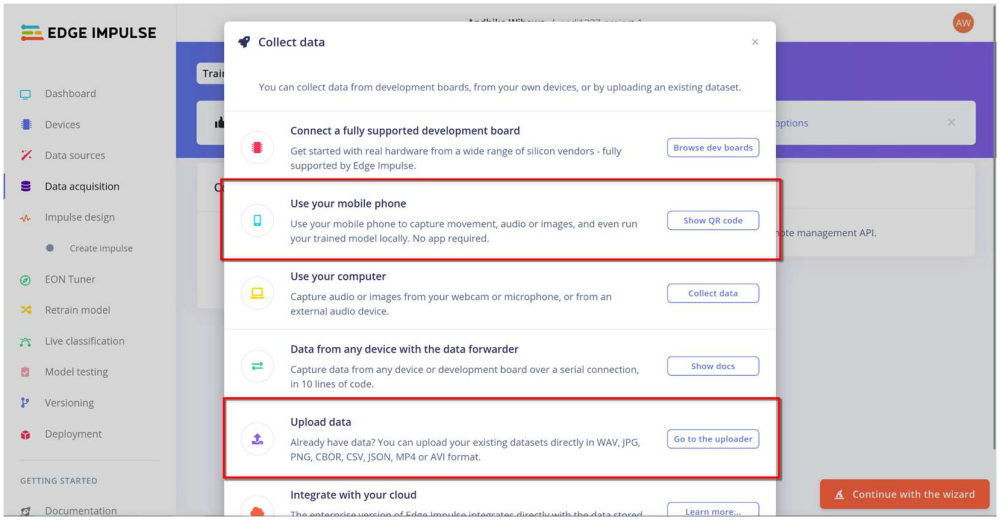
Buat akun Edge Impulse seperti yang telah dijelaskan sebelumnya dan akses situs tersebut. Buka Studio Edge Impulse untuk melakukan percobaan.

Langkah-Langkah

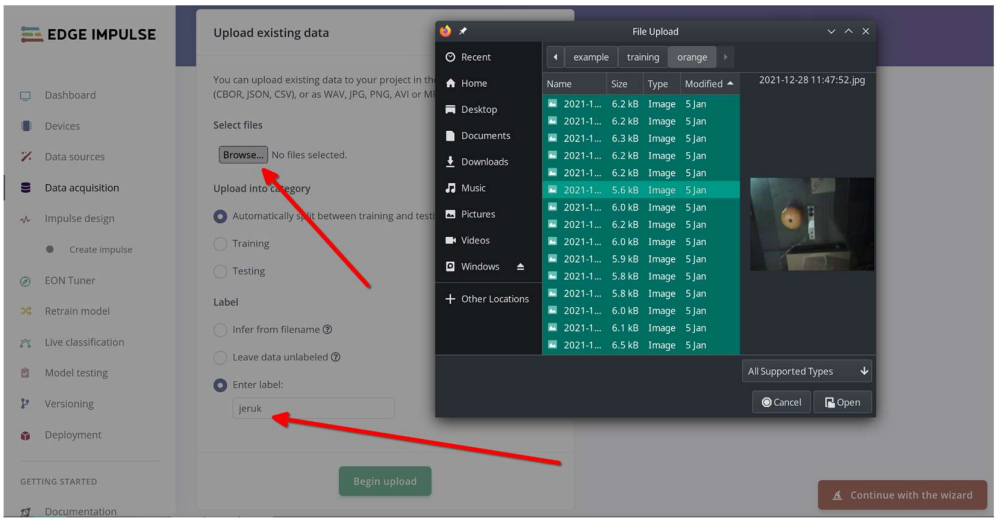
1. Setelah melakukan pendaftaran, kita akan dialihkan ke halaman studio Edge Impulse. Bila belum, akses dengan link berikut ini
2. Pada sidebar, pilih "Data acquisition" untuk memilih/mengambil data gambar yang akan digunakan oleh model machine learning



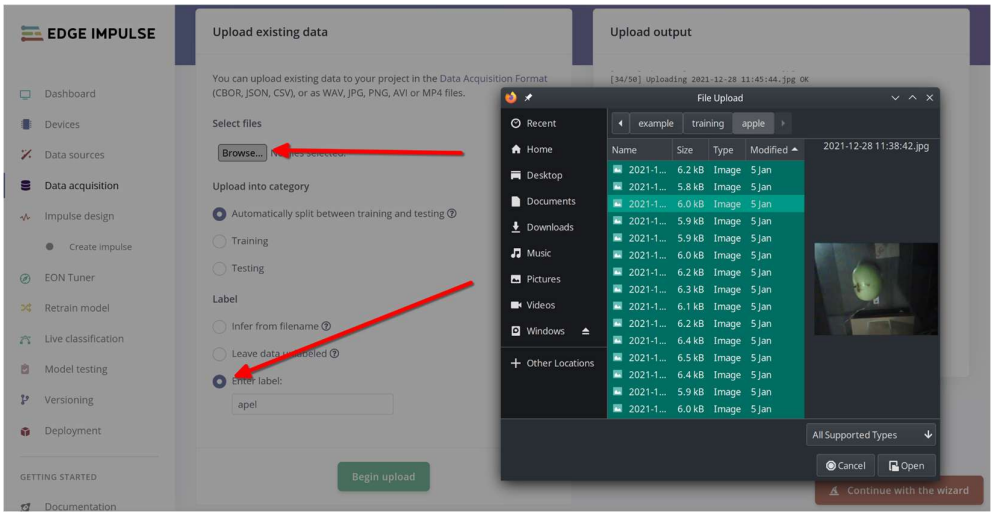
3. Pilih opsi yang ingin kalian gunakan untuk memperoleh data, misal melalui smartphone atau data gambar yang telah disimpan melalui ESP32-CAM pada percobaan sebelumnya



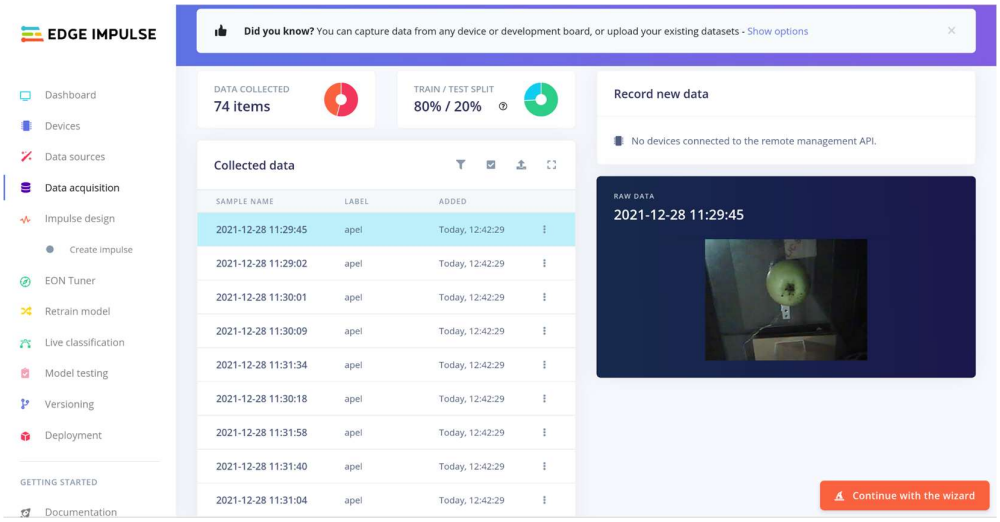
4. Mulai proses pengumpulan gambar. Gunakan 2 jenis objek yang berbeda (misal buah jeruk dan apel) dengan masing-masing jenis objek minimal memiliki gambar sebanyak 10 buah (lebih banyak lebih baik)



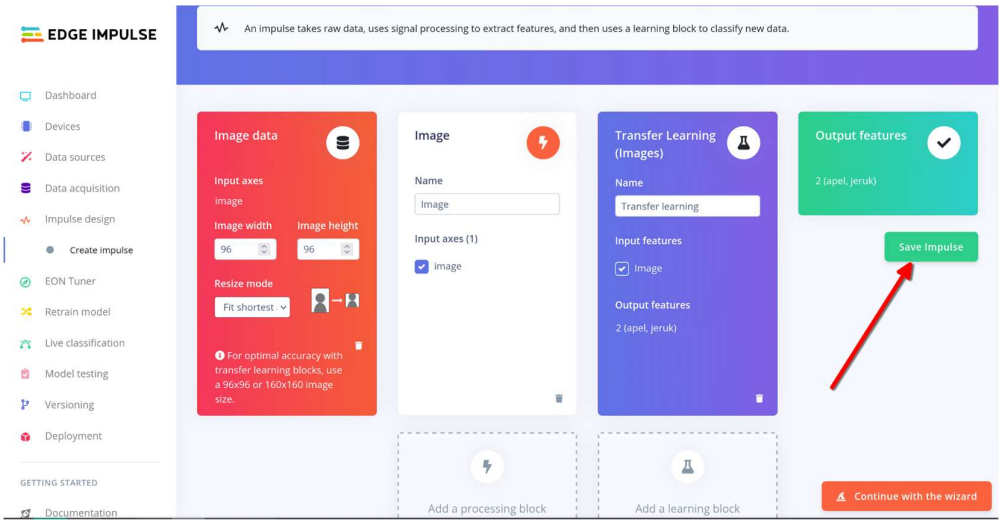
5. Upload kembali jenis objek kedua dengan label yang berbeda (misal bila sebelumnya jeruk maka sekarang apel)



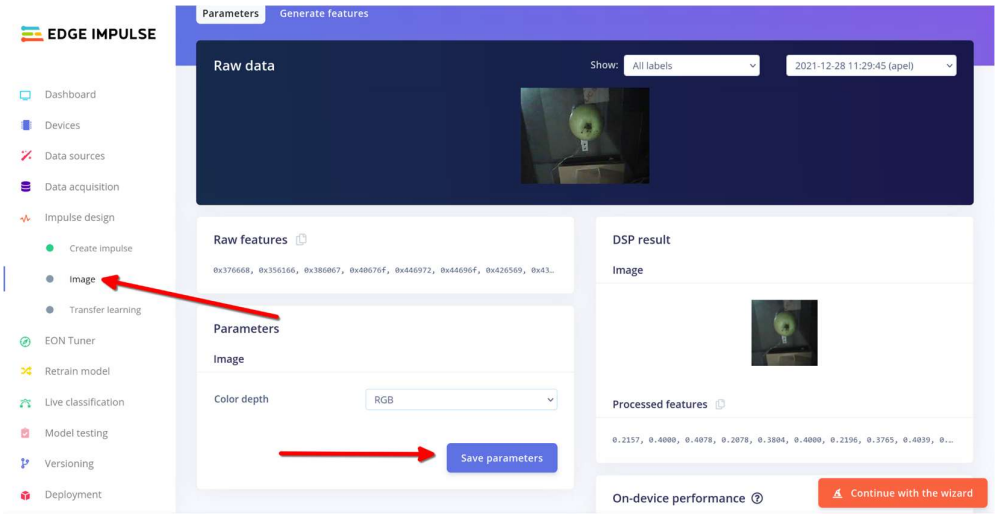
6. Cek data yang telah kalian ambil sebelum lanjut ke tahap berikutnya



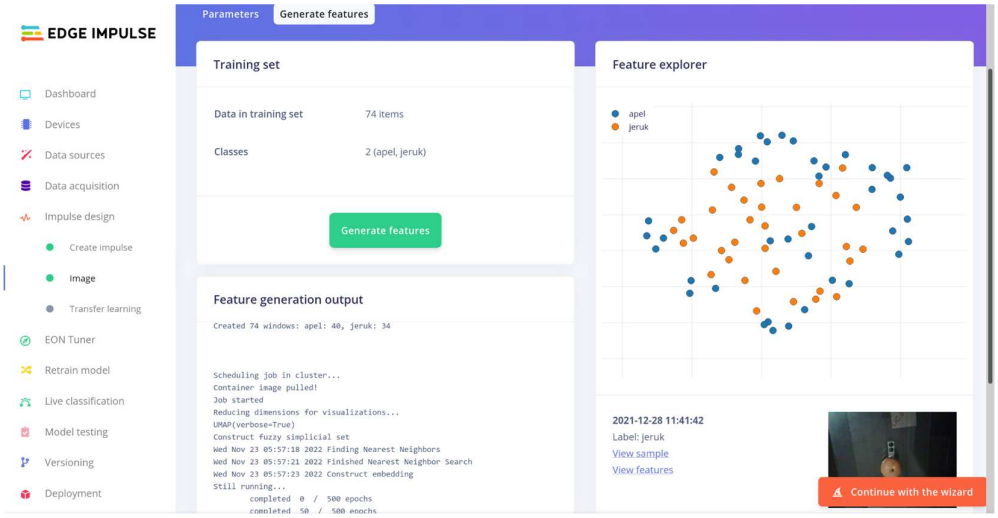
7. Selanjutnya, klik “Impulse design” -> “Create impulse” melalui sidebar dan tambahkan alur pemrosesan seperti gambar di bawah ini. Bila sudah, klik “Save impulse”



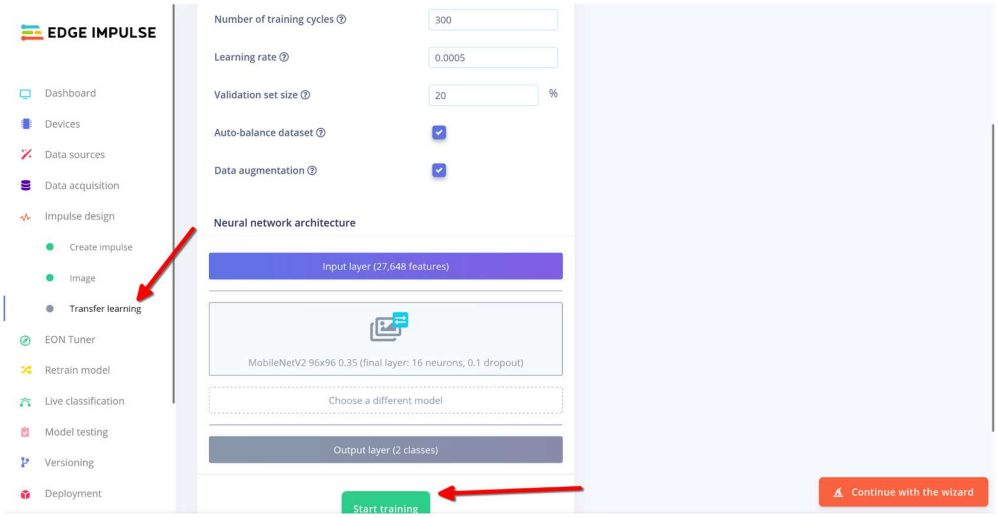
8. Setelah itu, pilih submenu baru yaitu “Image” pada bagian “Impulse design”. Pilih format warna gambar yang ingin dijadikan parameter. Karena buah berhubungan erat dengan warna maka pilih RGB. Bila ada proses selanjutnya, pilih juga “Generate features”



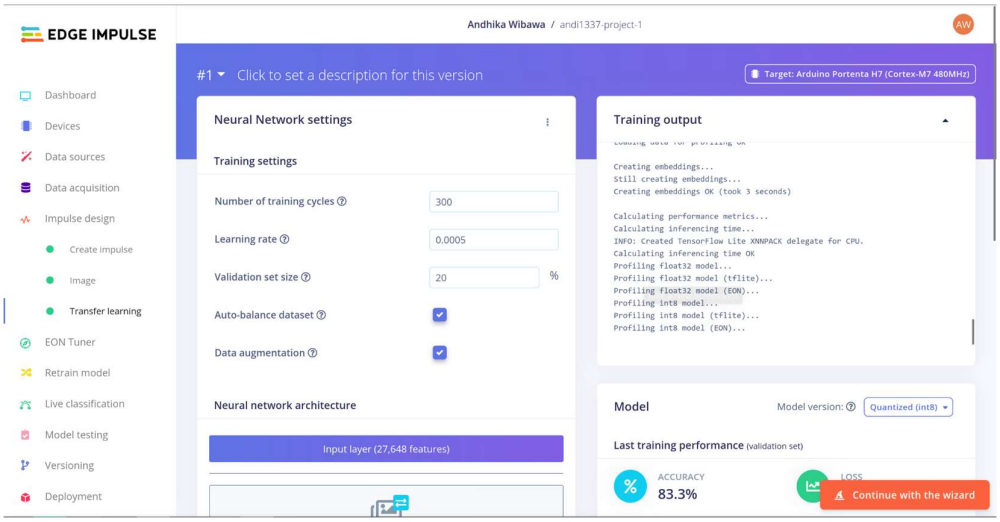
9. Bila ada proses selanjutnya, pilih “Generate features”. Hasil akan terlihat seperti di bawah ini



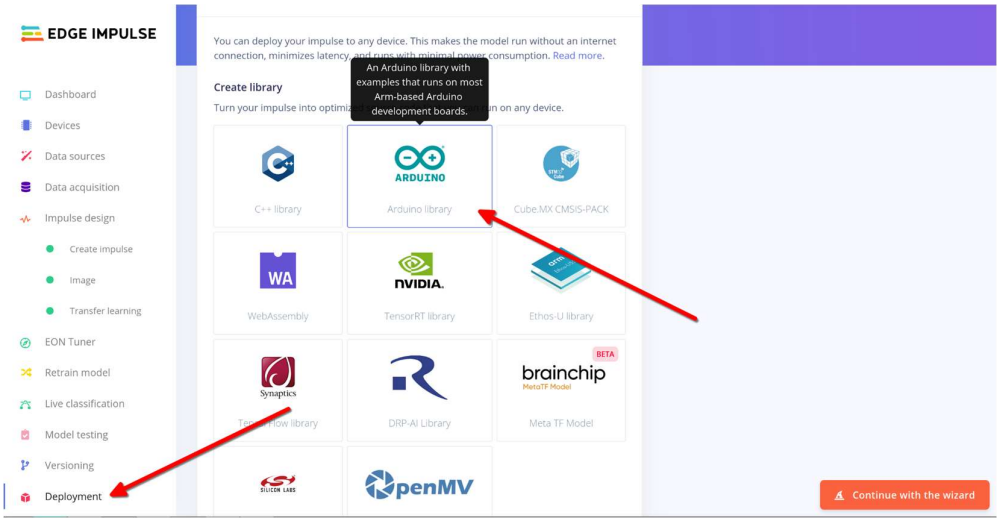
10. Selanjutnya, klik submenu “Transfer learning”. Konfigurasi dapat disesuaikan sendiri oleh kalian. Semakin tinggi nilai epoch, maka akan semakin baik hasilnya namun proses akan semakin lama



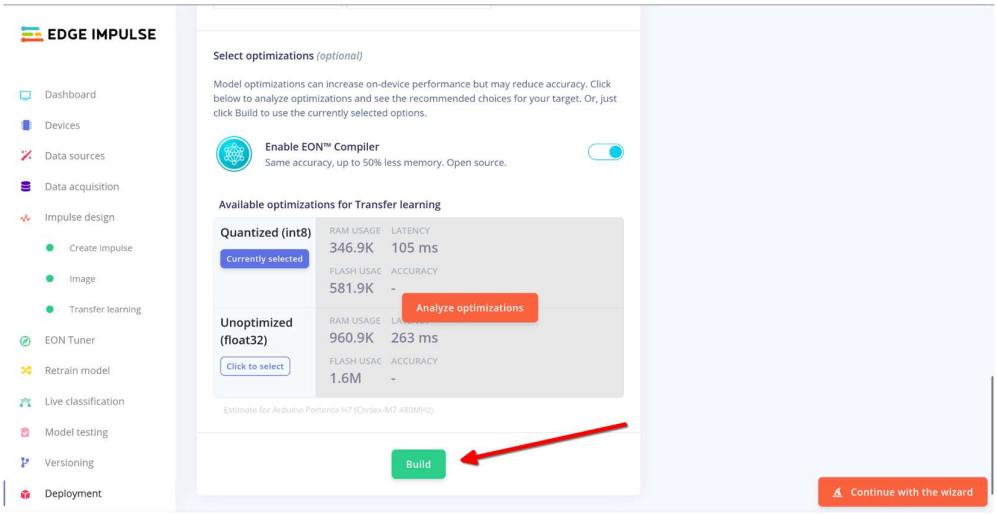
11. Tunggu beberapa saat sampai proses training selesai, hasilnya kira-kira akan seperti ini



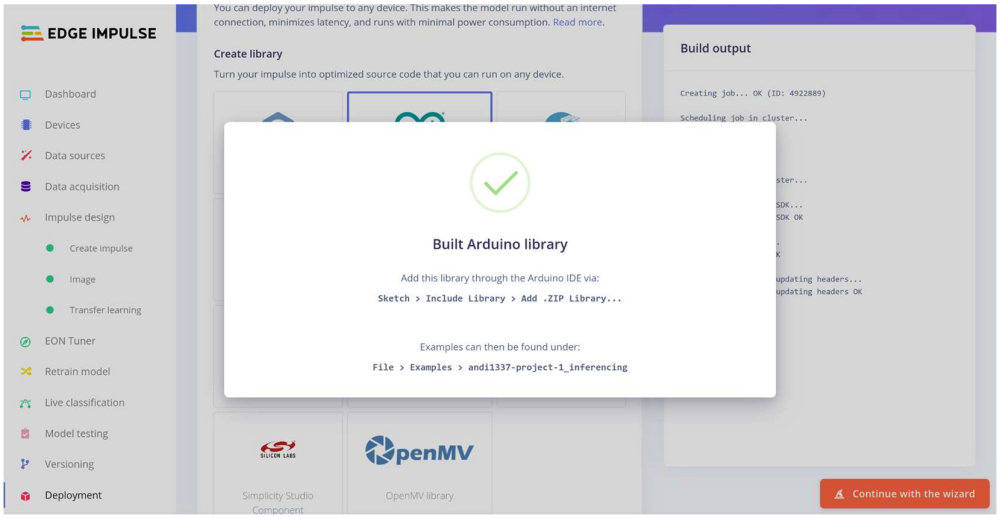
12. Bila akurasi dirasa cukup bagus, klik “Deployment” pada sidebar untuk mengekspor model ke dalam bentuk kode Arduino



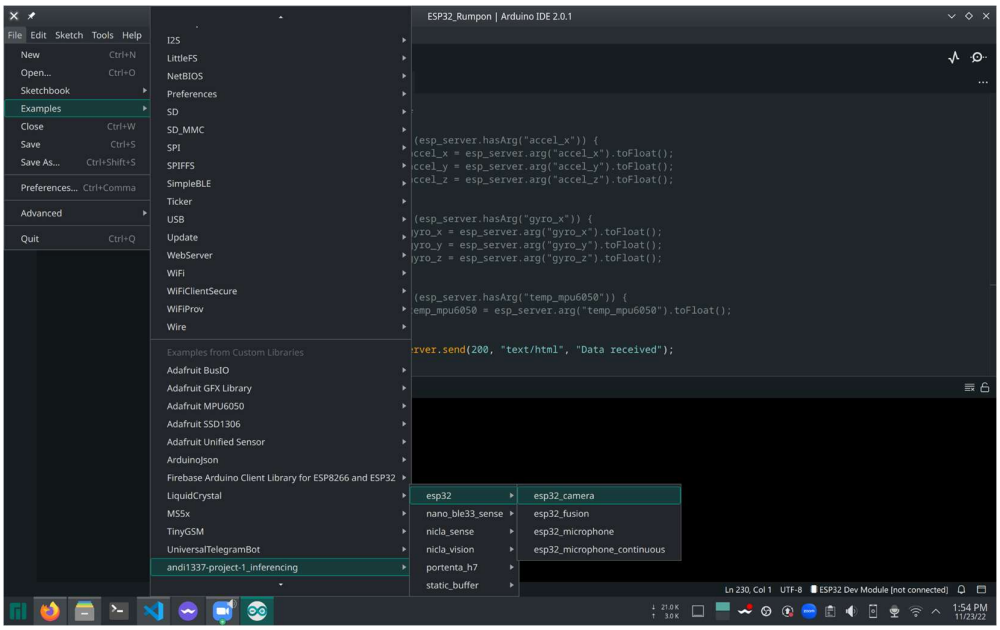
13. Scroll ke bawah dan klik “Build”



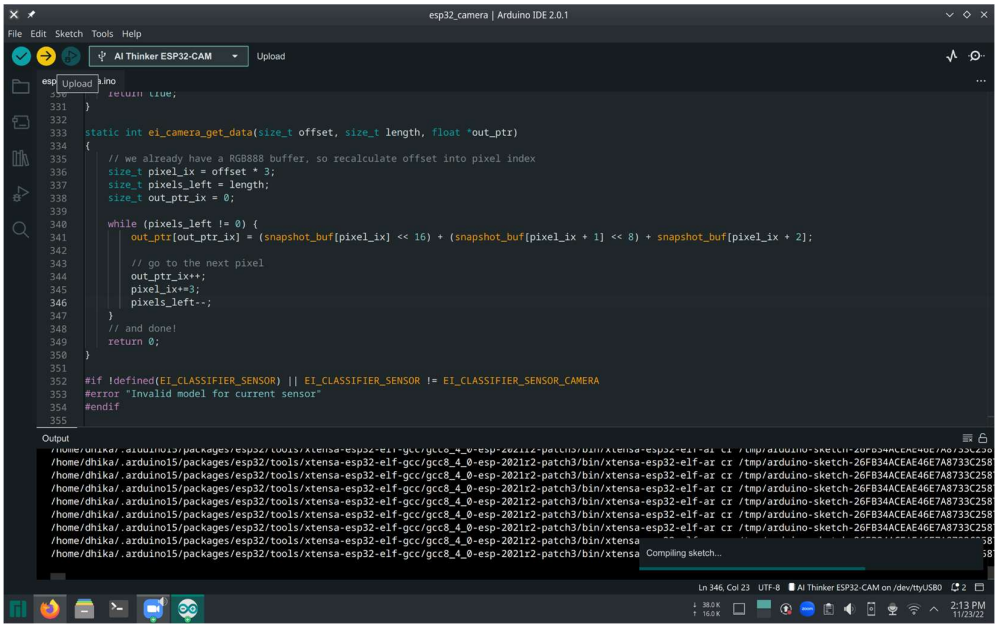
14. Bila sudah selesai, maka kode akan terunduh dan dapat dipasang ke Arduino IDE seperti intruksi gambar di bawah ini



15. Bila file ZIP library/example tersebut telah terpasang pada Arduino IDE. Akses file seperti tahapan berikut (nama *example* menyesuaikan)



16. Compile dan upload kode example tersebut



17. Bila tidak ada masalah, maka hasil klasifikasi akan muncul pada Serial Monitor ketika gambar ditangkap oleh ESP32-CAM

Pertanyaan / Asprak Bertanya

1. Bagaimana Edge Impulse bekerja? Jelaskan proses-proses dalam membangun model klasifikasi menggunakan Edge Impulse!
2. Adakah pengaruh jumlah data terhadap model machine learning?
3. Apakah epoch dari suatu model memiliki peran khusus dalam membangun model pembelajaran mesin? Apa perbedaan antara epoch tinggi dengan epoch rendah?

4. Membangun model training dan testing dari ML tidak efisien jika dilakukan secara langsung pada ESP32-CAM. menurutmu, mengapa demikian?

Alternatif Lampiran

Pada percobaan 2, jika kalian malas melampirkan gambar maka dapat juga melampirkan bukti berupa link video demonstrasi. Namun, pada video tidak perlu terdapat efek musik atau opening karena tidak akan ada nilai tambahan.

V. Challenge

Pada praktikum kali ini, semua **challenge ditiadakan** karena kompleksitas dari materi yang ada (namun tetap ada pengumpulan laporan). Silahkan manfaatkan waktu yang ada untuk mengerjakan tugas besar.

Catatan tambahan: Perlu diketahui juga bahwa terdapat banyak cara untuk melakukan machine learning dan/atau computer vision menggunakan mikrokontroler. Salah satu alternatif dari Edge Impulse yang juga populer adalah Teachable Machine (menghasilkan model dengan format TFLITE). Contoh proses klasifikasi dengan model TFLITE (TensorFlow Lite) dapat dilakukan seperti referensi berikut ini, dimana bridging antara Python dengan C++ dapat dilakukan melalui komunikasi serial atau bantuan protokol/framework lainnya seperti Firmata dan MicroPython.

VI. Referensi

[1] R. B. Salikhov, V. K. Abdrakhmanov, dan I. N. Safargalin, "Internet of Things (IoT) Security Alarms on ESP32-CAM," *J. Phys.: Conf. Ser.*, vol. 2096, no. 1, hlm. 012109, Nov 2021, doi: 10.1088/1742-6596/2096/1/012109.

[2] P. Dani Prasetyo Adi, "Performance evaluation of ESP32 Camera Face Recognition for various projects," Feb 2022, doi: 10.31763/iota.v2i1.512.

[3] M. I. Jordan dan T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, hlm. 255–260, 2015.

[4] R. David *dkk.*, "Tensorflow lite micro: Embedded machine learning for tinyml systems," *Proceedings of Machine Learning and Systems*, vol. 3, hlm. 800–811, 2021.

[5] V. J. Reddi *dkk.*, "Widening access to applied machine learning with tinyML," *arXiv preprint arXiv:2106.04008*, 2021.

[6] H. Tian, T. Wang, Y. Liu, X. Qiao, dan Y. Li, "Computer vision technology in agricultural automation—A review," *Information Processing in Agriculture*, vol. 7, no. 1, hlm. 1–19, 2020.

[7] X. Feng, Y. Jiang, X. Yang, M. Du, dan X. Li, "Computer vision algorithms and hardware implementations: A survey," *Integration*, vol. 69, hlm. 309–320, 2019.