ДОДАТОК А

Лістинг файлу esp32.ino

```
#include <esp32_identification.h>
#include "lib/dsp/image/image.hpp"

#include "esp_camera.h"

#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

/* Constant */
#define ESP32_CAMERA_RAW_FRAME_BUFFER_COLS           320
#define ESP32_CAMERA_RAW_FRAME_BUFFER_ROWS           240
#define ESP32_CAMERA_FRAME_BYTE_SIZE                 3

static bool debug_nn = false;
static bool is_initialised = false;
uint8_t *snapshot_buf; //points to the output of the capture

static camera_config_t camera_config = {
  .pin_pwdn = PWDN_GPIO_NUM,
  .pin_reset = RESET_GPIO_NUM,
  .pin_xclk = XCLK_GPIO_NUM,
  .pin_sscb_sda = SIOD_GPIO_NUM,
  .pin_sscb_scl = SIOC_GPIO_NUM,

  .pin_d7 = Y9_GPIO_NUM,
  .pin_d6 = Y8_GPIO_NUM,
  .pin_d5 = Y7_GPIO_NUM,
  .pin_d4 = Y6_GPIO_NUM,
```

```
        .pin_d3 = Y5_GPIO_NUM,
        .pin_d2 = Y4_GPIO_NUM,
        .pin_d1 = Y3_GPIO_NUM,
        .pin_d0 = Y2_GPIO_NUM,
        .pin_vsync = VSYNC_GPIO_NUM,
        .pin_href = HREF_GPIO_NUM,
        .pin_pclk = PCLK_GPIO_NUM,

        .xclk_freq_hz = 20000000,
        .ledc_timer = LEDC_TIMER_0,
        .ledc_channel = LEDC_CHANNEL_0,

        .pixel_format = PIXFORMAT_JPEG, //YUV422,GRAYSCALE,RGB565,JPEG
        .frame_size = FRAMESIZE_QVGA,

        .jpeg_quality = 12, //0-63 lower number means higher quality
        .fb_count = 1,       //if more than one, i2s runs in continuous mode
(only with JPEG)
        .fb_location = CAMERA_FB_IN_PSRAM,
        .grab_mode = CAMERA_GRAB_WHEN_EMPTY,
      };

      /* Function */
      bool esp32_camera_init(void);
      void esp32_camera_deinit(void);
      bool  esp32_camera_capture(uint32_t  img_width,  uint32_t  img_height,
uint8_t *out_buf) ;

      void setup()
      {
        Serial.begin(115200);
        while (!Serial);
        if (esp32_camera_init() == false) {
          esp32_printf("Failed to initialize Camera!\r\n");
        }
        else {
          esp32_printf("Camera initialized\r\n");
        }
        ledcSetup(7, 5000, 8);
        ledcAttachPin(4, 7);  //pin4 is LED
        ledcWrite(7, 200);
        esp32_printf("\nStarting continious inference in 2 seconds...\n");
        esp32_sleep(2000);
```

```
}

//  Get data and run identification
void loop()
{
  // instead of wait_ms, wait on the signal, this allows threads to
cancel
  if (esp32_sleep(5) != ESP32_IDENTIFICATION_OK) {
    return;
  }

  snapshot_buf = (uint8_t*)malloc(ESP32_CAMERA_RAW_FRAME_BUFFER_COLS *
ESP32_CAMERA_RAW_FRAME_BUFFER_ROWS * ESP32_CAMERA_FRAME_BYTE_SIZE);

  // check if allocation was successful
  if (snapshot_buf == nullptr) {
    esp32_printf("ERR: Failed to allocate snapshot buffer!\n");
    return;
  }

  id::signal_t signal;
  signal.total_length      =      ESP32_CLASSIFIER_INPUT_WIDTH      *
ESP32_CLASSIFIER_INPUT_HEIGHT;
  signal.get_data = &esp32_camera_get_data;

  if        (esp32_camera_capture((size_t)ESP32_CLASSIFIER_INPUT_WIDTH,
(size_t)ESP32_CLASSIFIER_INPUT_HEIGHT, snapshot_buf) == false) {
    esp32_printf("Failed to capture image\r\n");
    free(snapshot_buf);
    return;
  }

  // Run the classifier
  esp32_identification_result_t result = { 0 };

  ESP32_IDENTIFICATION_ERROR  err  =  run_classifier(&signal,  &result,
debug_nn);
  if (err != ESP32_IDENTIFICATION_OK) {
    esp32_printf("ERR: Failed to run classifier (%d)\n", err);
    return;
  }

#if ESP32_CLASSIFIER_OBJECT_DETECTION == 1
```

```
        bool bb_found = result.bounding_boxes[0].value > 0;
        for (size_t ix = 0; ix < result.bounding_boxes_count; ix++) {
          auto bb = result.bounding_boxes[ix];
          if (bb.value == 0) {
            continue;
          }
          esp32_printf("The %s was found (confidence is %u%%).\n", bb.label,
static_cast<unsigned int>(bb.value * 100));
        }
        if (!bb_found) {
          esp32_printf("No objects found\n");
        }
      #else
        for (size_t ix = 0; ix < ESP32_CLASSIFIER_LABEL_COUNT; ix++) {
          esp32_printf("    %s: %.5f\n", result.classification[ix].label,
                       result.classification[ix].value);
        }
      #endif

      #if ESP32_CLASSIFIER_HAS_ANOMALY == 1
        esp32_printf("    anomaly score: %.3f\n", result.anomaly);
      #endif
        free(snapshot_buf);
      }

      // Setup image sensor & start streaming
      bool esp32_camera_init(void) {

        if (is_initialised) return true;

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

        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, 0); // lower the saturation
      }

      is_initialised = true;
      return true;
    }

    /**
       @brief      Stop streaming of sensor data
    */
    void esp32_camera_deinit(void) {
      //deinitialize the camera
      esp_err_t err = esp_camera_deinit();

      if (err != ESP_OK)
      {
        esp32_printf("Camera deinit failed\n");
        return;
      }

      is_initialised = false;
      return;
    }


    /**
       @brief      Capture, rescale and crop image

       @param[in]  img_width     width of output image
       @param[in]  img_height    height of output image
       @param[in]  out_buf       pointer to store output image, NULL may be
used
                                 if esp32_camera_frame_buffer is to be used
for capture and resize/cropping.

       @retval      false if not initialised, image captured, rescaled or
cropped failed

    */
    bool  esp32_camera_capture(uint32_t  img_width,  uint32_t  img_height,
uint8_t *out_buf) {
      bool do_resize = false;
```

```
        if (!is_initialised) {
          esp32_printf("ERR: Camera is not initialized\r\n");
          return false;
        }

        camera_fb_t *fb = esp_camera_fb_get();

        if (!fb) {
          esp32_printf("Camera capture failed\n");
          return false;
        }

        bool converted = fmt2rgb888(fb->buf, fb->len, PIXFORMAT_JPEG,
snapshot_buf);

        esp_camera_fb_return(fb);

        if (!converted) {
          esp32_printf("Conversion failed\n");
          return false;
        }

        if ((img_width != ESP32_CAMERA_RAW_FRAME_BUFFER_COLS)
            || (img_height != ESP32_CAMERA_RAW_FRAME_BUFFER_ROWS)) {
          do_resize = true;
        }

        if (do_resize) {
          id::image::processing::crop_and_interpolate_rgb888(
            out_buf,
            ESP32_CAMERA_RAW_FRAME_BUFFER_COLS,
            ESP32_CAMERA_RAW_FRAME_BUFFER_ROWS,
            out_buf,
            img_width,
            img_height);
        }
        return true;
      }

      static int esp32_camera_get_data(size_t offset, size_t length, float
*out_ptr)
      {
```

```
        // already have a RGB888 buffer, so recalculate offset into pixel
index
        size_t pixel_ix = offset * 3;
        size_t pixels_left = length;
        size_t out_ptr_ix = 0;

        while (pixels_left != 0) {
          // Swap BGR to RGB
          out_ptr[out_ptr_ix]   =   (snapshot_buf[pixel_ix   +   2]   <<   16)   +
(snapshot_buf[pixel_ix + 1] << 8) + snapshot_buf[pixel_ix];

          // go to the next pixel
          out_ptr_ix++;
          pixel_ix += 3;
          pixels_left--;
        }
        return 0;
      }
```

ДОДАТОК Б

Лістинг файлу esp32_identification.h

```
#ifndef _INFERENCE_H
#define _INFERENCE_H

// Undefine min/max macros as these conflict with C++ std min/max
functions
// these are often included by Arduino cores
#include <Arduino.h>
#include <stdarg.h>
#ifdef min
#undef min
#endif // min
#ifdef max
#undef max
#endif // max
#ifdef round
#undef round
#endif // round
// Similar the ESP32 seems to define this, which is also used as an enum
value in TFLite
#ifdef DEFAULT
#undef DEFAULT
#endif // DEFAULT
//     Infineon     core     defines     this,     conflicts     with
CMSIS/DSP/Include/dsp/controller_functions.h
#ifdef A0
#undef A0
#endif // A0
#ifdef A1
#undef A1
#endif // A1
#ifdef A2
#undef A2
#endif // A2

/* Includes ----------------------------------------------------------
---- */
#include "lib/classifier/esp32_run_classifier.h"
#include "lib/dsp/numpy.hpp"
#include "model-parameters/model_metadata.h"
#include "lib/classifier/esp32_classifier_smooth.h"
extern void esp32_printf(const char *format, ...);

#endif // _INFERENCE_H
```

ДОДАТОК В

Демонстраційний матеріал

Формат	Позиція	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кільк.	Примітки
			Текстові документи		
А4	1	ГЮИК. 416616.003 ПЗ	Пояснювальна записка	98	с.
			Інші документи		
А4	2		Лістинг коду	8	с.
	3		Демонстраційний	14	с.
			матеріал у		
			вигляді презентації		

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК. 416616.003 ПЗ		
Розроб.		Єчевський А. Д.			Розробка програмного забезпечення для ідентифікації виробів на контрольних операціях технологічних процесів	Літера	Аркуш	Аркушів
Перевір.		Сотник С. В.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Демська Н. П.						
Затв.		Невлюдов І.Ш.						