LAMPIRAN

Smartdoorbell.ino

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_camera.h"
// memasukan library telegram bot
#include <UniversalTelegramBot.h>
// memasukan library arduino json (digunakan jika ingin menggunakan telegram
bot)
#include <ArduinoJson.h>

// Pilih model kamera
#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"

const char* ssid = "POCO X3 NFC";
const char* password = "55555555";

// Definisi token bot Telegram yang didapat di BotFather
String BOTtoken = "7380400749:AAG0ljme8Nc2fvKw0fVTQ2ueNnJN-
Tx4BqY";

// Definisi ID dari pengguna Telegram
String CHAT_ID = "702477593";
bool sendPhoto = false; bool
sendPhotoReq = false;
bool motionDetected = false;

// Menjadikan ESP32-CAM sebagai client untuk dapat terhubung ke router
WiFiClientSecure clientTCP;
// Inisialisasi bot Telegram oleh ESP32-CAM
UniversalTelegramBot bot(BOTtoken, clientTCP);

// Variabel untuk menentukan nyala atau matinya LED flash bool
flashState = LOW;

// Variabel delay untuk mengecek pesan yang masuk dari Telegram setiap 1 detik
int botRequestDelay = 1000;
```

```
unsigned long lastTimeBotRan;


filename=\"esp32-cam.jpg\"\r\nContent-Type: image/jpeg\r\n\r\n";
   String tail = "\r\n--RandomNerdTutorials--\r\n";

   uint16_t imageLen = fb->len;     uint16_t
extraLen = head.length() + tail.length();
   uint16_t totalLen = imageLen + extraLen;

   clientTCP.println("POST /bot" + BOTtoken + "/sendPhoto HTTP/1.1");
clientTCP.println("Host: " + String(myDomain));     clientTCP.println("Content-
Length: " + String(totalLen));     clientTCP.println("Content-Type: multipart/form-
data;
boundary=RandomNerdTutorials");
clientTCP.println();
   clientTCP.print(head);

   uint8_t *fbBuf = fb->buf;
size_t fbLen = fb->len;
   for (size_t n = 0; n < fbLen; n += 1024)
{      if (n + 1024 < fbLen)
{        clientTCP.write(fbBuf, 1024);
fbBuf += 1024;
     } else if (fbLen % 1024 > 0)
{        size_t remainder = fbLen %
1024;
       clientTCP.write(fbBuf, remainder);
     }
   }

   clientTCP.print(tail);

   esp_camera_fb_return(fb);

   int waitTime = 10000;
long startTimer = millis();
   boolean state = false;

   while ((startTimer + waitTime) > millis()) {
     delay(100);
```

```
     while (clientTCP.available())
{        char c = clientTCP.read();
if (state) getBody += String(c);
       if (c == '\n') {
         if (getAll.length() == 0) state = true;
         getAll =
"";         } else if (c !=
'\r') {
         getAll += String(c);
       }
       startTimer = millis();
     }
     if (getBody.length() > 0) break;
   }

clientTCP.stop();   }
else {
   getBody = "Koneksi ke api.telegram.org gagal.";
  }

  return getBody;
}

void setup() {
  // Fungsi bawaan ESP32-CAM untuk menghindari program crash
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);

  // Memanggil serial monitor untuk keperluan debugging
  Serial.begin(115200);
pinMode(13, INPUT_PULLUP);
pinMode(14, INPUT_PULLUP);
  pinMode(4, OUTPUT);
  digitalWrite(4, 0);
Serial.setDebugOutput(true);
  Serial.println();

  // Konfigurasi kamera di ESP32-CAM
camera_config_t config;   config.ledc_channel
= LEDC_CHANNEL_0;   config.ledc_timer =
LEDC_TIMER_0;   config.pin_d0 =
Y2_GPIO_NUM;   config.pin_d1 =
Y3_GPIO_NUM;   config.pin_d2 =
Y4_GPIO_NUM;   config.pin_d3 =
```

```
Y5_GPIO_NUM;   config.pin_d4 =
Y6_GPIO_NUM;   config.pin_d5 =
Y7_GPIO_NUM;   config.pin_d6 =
Y8_GPIO_NUM;   config.pin_d7 =
Y9_GPIO_NUM;   config.pin_xclk =
XCLK_GPIO_NUM;   config.pin_pclk =
PCLK_GPIO_NUM;   config.pin_vsync =
VSYNC_GPIO_NUM;   config.pin_href =
HREF_GPIO_NUM;   config.pin_sscb_sda =
SIOD_GPIO_NUM;   config.pin_sscb_scl =
SIOC_GPIO_NUM;   config.pin_pwdn =
PWDN_GPIO_NUM;   config.pin_reset =
RESET_GPIO_NUM;   config.xclk_freq_hz =
20000000;   config.pixel_format =
PIXFORMAT_JPEG;

  // Inisialisasi dengan spesifikasi tinggi untuk pre-allocate buffer yang lebih besar
if (psramFound()) {
    config.frame_size = FRAMESIZE_UXGA;
config.jpeg_quality = 10;     config.fb_count
= 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
config.jpeg_quality = 12;
    config.fb_count = 1;
  }



  sensor_t * s = esp_camera_sensor_get();
  // Sensor awal terbalik secara vertikal dan warnanya sedikit jenuh
  if (s->id.PID == OV3660_PID) {     s->set_vflip(s, 1); //
Membalik vertikal     s->set_brightness(s, 1); //
Menaikkan kecerahan sedikit     s->set_saturation(s, -2);
// Menurunkan saturasi
  }
  // Turunkan ukuran frame untuk meningkatkan frame rate awal
  s->set_framesize(s, FRAMESIZE_QVGA);

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

  Serial.println("Kamera siap");
```

```
  // Menghubungkan ke internet
  WiFi.mode(WIFI_STA);
  Serial.println();
  Serial.print("Menghubungkan ke ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  clientTCP.setCACert(TELEGRAM_CERTIFICATE_ROOT);

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

  startCameraServer();

  Serial.print("Kamera Siap! Gunakan 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' untuk terhubung");
}

void loop() {   if
(digitalRead(13) == 0) {     while
(digitalRead(13) == 0) {
      delay(100);
    }
    sendPhoto = true;
  }
  String localIP = WiFi.localIP().toString();
if (sendPhoto) {
    Serialprintln("MENIRIM FOTO");
    bot.sendMessage(CHAT_ID, "Ada yang menekan Bel! Gunakan link http://" +
localIP + " untuk pemantauan");     sendPhotoTelegram();
    sendPhoto = false;
  }

if (sendPhotoReq) {
    Serial.println("MENGIRIM FOTO");
    sendPotoTelegram();
    sendPhooReq = false;
  }
```

```
  if (millis() > lastTieBotRan + botRequestDelay) {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
while (numNewMessages) {
      Serial.println("Mendapatkan respon");
handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
  }
delay(100); }
```

app_httpd.cpp

```
// limitations under the License.
#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "img_converters.h"
#include "camera_index.h"
#include "Arduino.h"

#include "fb_gfx.h"
#include "fd_forward.h"
#include "dl_lib.h"
#include "fr_forward.h"

#define ENROLL_CONFIRM_TIMES 5
#define FACE_ID_SAVE_NUMBER 7

#define FACE_COLOR_WHITE  0x00FFFFFF
#define FACE_COLOR_BLACK  0x00000000
#define FACE_COLOR_RED    0x000000FF
#define FACE_COLOR_GREEN  0x0000FF00
#define FACE_COLOR_BLUE   0x00FF0000
#define FACE_COLOR_YELLOW (FACE_COLOR_RED |
FACE_COLOR_GREEN)
#define FACE_COLOR_CYAN   (FACE_COLOR_BLUE |
FACE_COLOR_GREEN)
#define FACE_COLOR_PURPLE (FACE_COLOR_BLUE |
FACE_COLOR_RED)

typedef struct {
```

```
        size_t size; //number of values used for filtering
size_t index; //current value index         size_t count;
//value count
        int sum;
        int * values; //array to be filled with values
} ra_filter_t;

typedef struct
{         httpd_req_t *req;
size_t len;
} jpg_chunking_t;

#define PART_BOUNDARY "123456789000000000000987654321" static
const char* _STREAM_CONTENT_TYPE = "multipart/x-
mixedreplace;boundary=" PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY
"\r\n";
static const char* _STREAM_PART = "Content-Type:
image/jpeg\r\nContentLength: %u\r\n\r\n";

static ra_filter_t ra_filter;
httpd_handle_t stream_httpd = NULL; httpd_handle_t
camera_httpd = NULL; static mtmn_config_t
mtmn_config = {0}; static int8_t detection_enabled =
0; static int8_t recognition_enabled = 0; static int8_t
is_enrolling = 0;
static face_id_list id_list = {0};

static ra_filter_t * ra_filter_init(ra_filter_t * filter, size_t
sample_size){     memset(filter, 0, sizeof(ra_filter_t));

   filter->values = (int *)malloc(sample_size * sizeof(int));     if(!filter-
>values){
      return NULL;
   }
   memset(filter->values, 0, sample_size * sizeof(int));

   filter->size = sample_size;
   return filter;
}

static int ra_filter_run(ra_filter_t * filter, int
value){     if(!filter->values){         return value;
```

```
    }
    filter->sum -= filter->values[filter->index];
filter->values[filter->index] = value;     filter->sum
+= filter->values[filter->index];     filter-
>index++;
    filter->index = filter->index % filter->size;
if (filter->count < filter->size) {         filter-
>count++;
    }
    return filter->sum / filter->count;
}

static void rgb_print(dl_matrix3du_t *image_matrix, uint32_t color, const char *
str){     fb_data_t fb;
    fb.width = image_matrix->w;
fb.height = image_matrix->h;
fb.data = image_matrix->item;
fb.bytes_per_pixel = 3;     fb.format
= FB_BGR888;
    fb_gfx_print(&fb, (fb.width - (strlen(str) * 14)) / 2, 10, color, str);
}

static int rgb_printf(dl_matrix3du_t *image_matrix, uint32_t color, const char
*format, ...){     cha
r loc_buf[64];
    char * temp = loc_buf;
int len;     va_list arg;
va_list copy;
va_start(arg, format);
va_copy(copy, arg);
    len = vsnprintf(loc_buf, sizeof(loc_buf), format, arg);
va_end(copy);     if(len >= sizeof(loc_buf)){         temp
= (char*)malloc(len+1);
        if(temp == NULL) {
            return 0;
        }
    }
    vsnprintf(temp, len+1, format, arg);
va_end(arg);
    rgb_print(image_matrix, color, temp);
    if(len >
64){         free(temp);
```

```
    }     return
len;
}

static void draw_face_boxes(dl_matrix3du_t *image_matrix, box_array_t *boxes,
int face_id){
    int x, y, w, h, i;
    uint32_t color = FACE_COLOR_YELLOW;
if(face_id < 0){
        color = FACE_COLOR_RED;
    } else if(face_id > 0){
        color = FACE_COLOR_GREEN;
    }
fb_data_t fb;
    fb.width = image_matrix->w;
fb.height = image_matrix->h;
fb.data = image_matrix->item;
fb.bytes_per_pixel = 3;
fb.format = FB_BGR888;     for (i
= 0; i < boxes->len; i++){
        // rectangle box
        x = (int)boxes->box[i].box_p[0];         y
= (int)boxes->box[i].box_p[1];         w =
(int)boxes->box[i].box_p[2] - x + 1;         h =
(int)boxes->box[i].box_p[3] - y + 1;
fb_gfx_drawFastHLine(&fb, x, y, w, color);
fb_gfx_drawFastHLine(&fb, x, y+h-1, w,
color);         fb_gfx_drawFastVLine(&fb, x,
y, h, color);
fb_gfx_drawFastVLine(&fb, x+w-1, y, h,
color);
#if 0
        // landmark         int x0, y0, j;         for (j = 0; j <
10; j+=2) {             x0 = (int)boxes-
>landmark[i].landmark_p[j];             y0 = (int)boxes-
>landmark[i].landmark_p[j+1];
fb_gfx_fillRect(&fb, x0, y0, 3, 3, color);
        }
#endif
    }
}

static int run_face_recognition(dl_matrix3du_t *image_matrix, box_array_t
```

```
*net_boxes){
  dl_matrix3du_t *aligned_face = NULL;
  int matched_id = 0;

  aligned_face = dl_matrix3du_alloc(1, FACE_WIDTH, FACE_HEIGHT, 3);
if(!aligned_face){
    Serial.println("Could not allocate face recognition buffer");
return matched_id;
  }
  if (align_face(net_boxes, image_matrix, aligned_face) ==
ESP_OK){      if (is_enrolling == 1){
      int8_t left_sample_face = enroll_face(&id_list, aligned_face);

      if(left_sample_face == (ENROLL_CONFIRM_TIMES - 1)){
        Serial.printf("Enrolling Face ID: %d\n", id_list.tail);
      }
      Serial.printf("Enrolling Face ID: %d sample %d\n", id_list.tail,
ENROLL_CONFIRM_TIMES - left_sample_face);
      rgb_printf(image_matrix, FACE_COLOR_CYAN, "ID[%u]
Sample[%u]", id_list.tail, ENROLL_CONFIRM_TIMES - left_sample_face);
if (left_sample_face == 0){              is_enrolling = 0;
        Serial.printf("Enrolled Face ID: %d\n", id_list.tail);
      }
} else {
      matched_id = recognize_face(&id_list, aligned_face);
if (matched_id >= 0) {
        Serial.printf("Match Face ID: %u\n", matched_id);
rgb_printf(image_matrix, FACE_COLOR_GREEN, "Hello Subject
%u", matched_id);
      } else {
        Serial.println("No Match Found");
        rgb_print(image_matrix, FACE_COLOR_RED, "Intruder Alert!");
matched_id = -1;
      }
    }
  } else {
    Serial.println("Face Not Aligned");
    //rgb_print(image_matrix, FACE_COLOR_YELLOW, "Human Detected");
  }

  dl_matrix3du_free(aligned_face);
return matched_id;
}
```

```
static size_t jpg_encode_stream(void * arg, size_t index, const void* data, size_t
len){
   jpg_chunking_t *j = (jpg_chunking_t *)arg;
if(!index){        j->len = 0;
   }
   if(httpd_resp_send_chunk(j->req, (const char *)data, len) !=
ESP_OK){        return 0;
   }     j->len +=
len;     return
len;
}

static esp_err_t capture_handler(httpd_req_t
*req){     camera_fb_t * fb = NULL;     esp_err_t
res = ESP_OK;
   int64_t fr_start = esp_timer_get_time();

   fb = esp_camera_fb_get();
   if (!fb) {
      Serial.println("Camera capture failed");
httpd_resp_send_500(req);
      return ESP_FAIL;
   }

   httpd_resp_set_type(req, "image/jpeg");
   httpd_resp_set_hdr(req, "Content-Disposition", "inline; filename=capture.jpg");

   size_t out_len, out_width, out_height;
   uint8_t * out_buf;
   bool s;     bool
detected = false;     int
face_id = 0;
   if(!detection_enabled || fb->width >
400){        size_t fb_len = 0;
      if(fb->format == PIXFORMAT_JPEG){
         fb_len = fb->len;
         res = httpd_resp_send(req, (const char *)fb->buf, fb->len);
      } else {
         jpg_chunking_t jchunk = {req, 0};
         res = frame2jpg_cb(fb, 80, jpg_encode_stream,
&jchunk)?ESP_OK:ESP_FAIL;
         httpd_resp_send_chunk(req, NULL, 0);
```

```
        fb_len = jchunk.len;
      }
      esp_camera_fb_return(fb);
      int64_t fr_end = esp_timer_get_time();
      Serial.printf("JPG: %uB %ums\n", (uint32_t)(fb_len), (uint32_t)((fr_end -
fr_start)/1000));       return res;
   }

   dl_matrix3du_t *image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height,
3);
   if (!image_matrix)
{       esp_camera_fb_return(fb);
      Serial.println("dl_matrix3du_alloc failed");
httpd_resp_send_500(req);
      return ESP_FAIL;
   }

   out_buf = image_matrix->item;
out_len = fb->width * fb->height * 3;
out_width = fb->width;
   out_height = fb->height;

   s = fmt2rgb888(fb->buf, fb->len, fb->format, out_buf);
esp_camera_fb_return(fb);
   if(!s){
      dl_matrix3du_free(image_matrix);
Serial.println("to rgb888 failed");
httpd_resp_send_500(req);
      return ESP_FAIL;
   }

   box_array_t *net_boxes = face_detect(image_matrix, &mtmn_config);

   if
(net_boxes){       detected =
true;
if(recognition_enabled){
        face_id = run_face_recognition(image_matrix, net_boxes);
      }
      draw_face_boxes(image_matrix, net_boxes, face_id);
      free(net_boxes->box);       free(net_boxes-
>landmark);
      free(net_boxes);
```

```
    }

    jpg_chunking_t jchunk = {req, 0};
    s = fmt2jpg_cb(out_buf, out_len, out_width, out_height,
PIXFORMAT_RGB888, 90, jpg_encode_stream, &jchunk);
    dl_matrix3du_free(image_matrix);
    if(!s){
        Serial.println("JPEG compression failed");
return ESP_FAIL;
    }

    int64_t fr_end = esp_timer_get_time();
    Serial.printf("FACE: %uB %ums %s%d\n", (uint32_t)(jchunk.len),
(uint32_t)((fr_end - fr_start)/1000), detected?"DETECTED ":"", face_id);
return res;
}

static esp_err_t stream_handler(httpd_req_t
*req){     camera_fb_t * fb = NULL;     esp_err_t
res = ESP_OK;     size_t _jpg_buf_len = 0;
uint8_t * _jpg_buf = NULL;     char *
part_buf[64];
    dl_matrix3du_t *image_matrix = NULL;
    bool detected = false;
int face_id = 0;     int64_t
fr_start = 0;     int64_t
fr_ready = 0;     int64_t
fr_face = 0;     int64_t
fr_recognize = 0;
    int64_t fr_encode = 0;

    static int64_t last_frame = 0;
if(!last_frame) {
        last_frame = esp_timer_get_time();
    }

    res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
if(res != ESP_OK){
        return res;
    }
```

```
   while(true){        detected =
false;        face_id = 0;        fb
= esp_camera_fb_get();
      if (!fb) {
         Serial.println("Camera capture failed");
res = ESP_FAIL;
      } else {
         fr_start = esp_timer_get_time();
         fr_ready = fr_start;           fr_face =
fr_start;           fr_encode = fr_start;
fr_recognize = fr_start;
if(!detection_enabled || fb->width >
400){               if(fb->format !=
PIXFORMAT_JPEG){
                  bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf,
&_jpg_buf_len);
                  esp_camera_fb_return(fb);
                  fb = NULL;
if(!jpeg_converted){
                     Serial.println("JPEG compression failed");
res = ESP_FAIL;
                  }
               } else {
                  _jpg_buf_len = fb->len;
                  _jpg_buf = fb->buf;
               }
            } else {

               image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height, 3);

               if (!image_matrix) {
                  Serial.println("dl_matrix3du_alloc failed");
res = ESP_FAIL;
               } else {
                  if(!fmt2rgb888(fb->buf, fb->len, fb->format, image_matrix->item)){
                     Serial.println("fmt2rgb888 failed");
res = ESP_FAIL;
                  } else {
                     fr_ready = esp_timer_get_time();
box_array_t *net_boxes = NULL;
                     if(detection_enabled){
                        net_boxes = face_detect(image_matrix, &mtmn_config);
                     }
```

```
                fr_face = esp_timer_get_time();
fr_recognize = fr_face;
                if (net_boxes || fb->format !=
PIXFORMAT_JPEG){                    if(net_boxes){
  detected = true;                        if(recognition_enabled){
                      face_id = run_face_recognition(image_matrix, net_boxes);
                    }
                    fr_recognize = esp_timer_get_time();
                    draw_face_boxes(image_matrix, net_boxes, face_id);
free(net_boxes->box);                       free(net_boxes->landmark);
                    free(net_boxes);
                  }
                  if(!fmt2jpg(image_matrix->item, fb->width*fb->height*3,
fb>width, fb->height, PIXFORMAT_RGB888, 90, &_jpg_buf, &_jpg_buf_len)){
                    Serial.println("fmt2jpg failed");
res = ESP_FAIL;
                  }
                  esp_camera_fb_return(fb);
                  fb = NULL;
                } else {
                  _jpg_buf = fb->buf;
                  _jpg_buf_len = fb->len;
                }
                fr_encode = esp_timer_get_time();
              }
              dl_matrix3du_free(image_matrix);
            }
          }
        }
        if(res == ESP_OK){
          size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART,
_jpg_buf_len);
          res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
        }
        if(res == ESP_OK){
          res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
        }
        if(res == ESP_OK){
          res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
        }
        if(fb){
```

```
        esp_camera_fb_return(fb);
fb = NULL;            _jpg_buf =
NULL;       } else
if(_jpg_buf){           free(_jpg_b
uf);           _jpg_buf = NULL;
      }
      if(res !=
ESP_OK){           break;
      }
      int64_t fr_end = esp_timer_get_time();

      int64_t ready_time = (fr_ready - fr_start)/1000;
int64_t face_time = (fr_face - fr_ready)/1000;       int64_t
recognize_time = (fr_recognize - fr_face)/1000;       int64_t
encode_time = (fr_encode - fr_recognize)/1000;       int64_t
process_time = (fr_encode - fr_start)/1000;

      int64_t frame_time = fr_end - last_frame;       last_frame = fr_end;
frame_time /= 1000;       uint32_t avg_frame_time =
ra_filter_run(&ra_filter, frame_time);
Serial.printf("MJPG: %uB %ums (%.1ffps), AVG: %ums (%.1ffps),
%u+%u+%u+%u=%u %s%d\n",
        (uint32_t)(_jpg_buf_len),
        (uint32_t)frame_time, 1000.0 / (uint32_t)frame_time,
avg_frame_time, 1000.0 / avg_frame_time,
        (uint32_t)ready_time, (uint32_t)face_time, (uint32_t)recognize_time,
(uint32_t)encode_time, (uint32_t)process_time,
        (detected)?"DETECTED ":"", face_id
      );
   }

   last_frame = 0;
return res;
}

static      esp_err_t      cmd_handler(httpd_req_t
*req){     char*   buf;     size_t buf_len;     char
variable[32] = {0,};
   char value[32] = {0,};

   buf_len = httpd_req_get_url_query_len(req) + 1;
   if (buf_len > 1) {
```

```
      buf = (char*)malloc(buf_len);
if(!buf){
          httpd_resp_send_500(req);
          return ESP_FAIL;
      }
      if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK)
{           if (httpd_query_key_value(buf, "var", variable, sizeof(variable))
==
ESP_OK &&
              httpd_query_key_value(buf, "val", value, sizeof(value)) == ESP_OK) {
          } else {
              free(buf);
              httpd_resp_send_404(req);
              return ESP_FAIL;
          }
      } else {
          free(buf);
          httpd_resp_send_404(req);
          return ESP_FAIL;
      }
free(buf);    }
else {
      httpd_resp_send_404(req);
      return ESP_FAIL;
   }

   int val = atoi(value);
   sensor_t * s = esp_camera_sensor_get();
int res = 0;

   if(!strcmp(variable, "framesize")) {
      if(s->pixformat == PIXFORMAT_JPEG) res = s->set_framesize(s,
(framesize_t)val);
   }
   else if(!strcmp(variable, "quality")) res = s->set_quality(s, val);
else if(!strcmp(variable, "contrast")) res = s->set_contrast(s, val);    else
if(!strcmp(variable, "brightness")) res = s->set_brightness(s, val);    else
if(!strcmp(variable, "saturation")) res = s->set_saturation(s, val);    else
if(!strcmp(variable, "gainceiling")) res = s->set_gainceiling(s,
(gainceiling_t)val);    else if(!strcmp(variable, "colorbar")) res = s-
>set_colorbar(s, val);    else if(!strcmp(variable, "awb")) res = s-
>set_whitebal(s, val);    else if(!strcmp(variable, "agc")) res = s-
>set_gain_ctrl(s, val);    else if(!strcmp(variable, "aec")) res = s-
```

```
>set_exposure_ctrl(s, val);     else if(!strcmp(variable, "hmirror"))
res = s->set_hmirror(s, val); else if(!strcmp(variable, "vflip")) res =
s->set_vflip(s, val); else if(!strcmp(variable, "awb_gain")) res = s-
>set_awb_gain(s, val);
   else if(!strcmp(variable, "agc_gain")) res = s->set_agc_gain(s, val);     else
if(!strcmp(variable, "aec_value")) res = s->set_aec_value(s, val);     else
if(!strcmp(variable, "aec2")) res = s->set_aec2(s, val);     else
if(!strcmp(variable, "dcw")) res = s->set_dcw(s, val);     else
if(!strcmp(variable, "bpc")) res = s->set_bpc(s, val);     else if(!strcmp(variable,
"wpc")) res = s->set_wpc(s, val);     else if(!strcmp(variable, "raw_gma")) res =
s->set_raw_gma(s, val);     else if(!strcmp(variable, "lenc")) res = s-
>set_lenc(s, val);     else if(!strcmp(variable, "special_effect")) res = s-
>set_special_effect(s, val);     else if(!strcmp(variable, "wb_mode")) res = s-
>set_wb_mode(s, val);     else if(!strcmp(variable, "ae_level")) res = s-
>set_ae_level(s, val);     else if(!strcmp(variable, "face_detect"))
{         detection_enabled = val;         if(!detection_enabled) {
           recognition_enabled = 0;
       }
   }
   else if(!strcmp(variable, "face_enroll")) is_enrolling = val;
else if(!strcmp(variable, "face_recognize"))
{         recognition_enabled = val;
if(recognition_enabled){
           detection_enabled = val;
       }     }
else
{         res = -
1;
   }

   if(res){
      return httpd_resp_send_500(req);
   }

   httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
return httpd_resp_send(req, NULL, 0);
}

static esp_err_t status_handler(httpd_req_t
*req){     static char json_response[1024];

   sensor_t * s = esp_camera_sensor_get();
char * p = json_response;
```

```
    *p++ = '{';

    p+=sprintf(p, "\"framesize\":%u,", s->status.framesize);
    p+=sprintf(p, "\"quality\":%u,", s->status.quality);
    p+=sprintf(p, "\"brightness\":%d,", s->status.brightness);
    p+=sprintf(p, "\"contrast\":%d,", s->status.contrast);
p+=sprintf(p, "\"saturation\":%d,", s->status.saturation);
p+=sprintf(p, "\"sharpness\":%d,", s->status.sharpness);
p+=sprintf(p, "\"special_effect\":%u,", s->status.special_effect);
p+=sprintf(p, "\"wb_mode\":%u,", s->status.wb_mode);
p+=sprintf(p, "\"awb\":%u,", s->status.awb);     p+=sprintf(p,
"\"awb_gain\":%u,", s->status.awb_gain);     p+=sprintf(p,
"\"aec\":%u,", s->status.aec);     p+=sprintf(p, "\"aec2\":%u,", s-
>status.aec2);     p+=sprintf(p, "\"ae_level\":%d,", s-
>status.ae_level);     p+=sprintf(p, "\"aec_value\":%u,", s-
>status.aec_value);     p+=sprintf(p, "\"agc\":%u,", s->status.agc);
p+=sprintf(p, "\"agc_gain\":%u,", s->status.agc_gain);
p+=sprintf(p, "\"gainceiling\":%u,", s->status.gainceiling);
p+=sprintf(p, "\"bpc\":%u,", s->status.bpc);     p+=sprintf(p,
"\"wpc\":%u,", s->status.wpc);     p+=sprintf(p,
"\"raw_gma\":%u,", s->status.raw_gma);     p+=sprintf(p,
"\"lenc\":%u,", s->status.lenc);     p+=sprintf(p, "\"vflip\":%u,", s-
>status.vflip);     p+=sprintf(p, "\"hmirror\":%u,", s-
>status.hmirror);     p+=sprintf(p, "\"dcw\":%u,", s->status.dcw);
p+=sprintf(p, "\"colorbar\":%u,", s->status.colorbar);
p+=sprintf(p, "\"face_detect\":%u,", detection_enabled);
p+=sprintf(p, "\"face_enroll\":%u,", is_enrolling);     p+=sprintf(p,
"\"face_recognize\":%u", recognition_enabled);
    *p++ = '}';
*p++ = 0;
    httpd_resp_set_type(req, "application/json");
httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");     return
httpd_resp_send(req, json_response, strlen(json_response));
}

static esp_err_t index_handler(httpd_req_t
*req){     httpd_resp_set_type(req, "text/html");
httpd_resp_set_hdr(req, "Content-Encoding", "gzip");
sensor_t * s = esp_camera_sensor_get();     if (s->id.PID
== OV3660_PID) {
        return httpd_resp_send(req, (const char *)index_ov3660_html_gz,
index_ov3660_html_gz_len);
    }
```

```
   return httpd_resp_send(req, (const char *)index_ov2640_html_gz,
index_ov2640_html_gz_len);
}
void startCameraServer(){
   httpd_config_t config = HTTPD_DEFAULT_CONFIG();

   httpd_uri_t index_uri = {
      .uri       = "/",
      .method    = HTTP_GET,
      .handler   = index_handler,
      .user_ctx  = NULL
   };

   httpd_uri_t status_uri = {
      .uri       = "/status",
      .method    = HTTP_GET,
      .handler   = status_handler,
      .user_ctx  = NULL
   };

   httpd_uri_t cmd_uri = {
      .uri       = "/control",
      .method    =
HTTP_GET,        .handler   =
cmd_handler,
      .user_ctx  = NULL
   };

   httpd_uri_t capture_uri = {
      .uri       = "/capture",
      .method    = HTTP_GET,
      .handler   = capture_handler,
      .user_ctx  = NULL
   };

  httpd_uri_t stream_uri = {
      .uri       = "/stream",
      .method    = HTTP_GET,
      .handler   = stream_handler,
      .user_ctx  = NULL
   };
```

```
    ra_filter_init(&ra_filter, 20);

    mtmn_config.min_face = 80;
mtmn_config.pyramid = 0.7;
mtmn_config.p_threshold.score = 0.6;
mtmn_config.p_threshold.nms = 0.7;
mtmn_config.r_threshold.score = 0.7;
mtmn_config.r_threshold.nms = 0.7;
mtmn_config.r_threshold.candidate_number = 4;
mtmn_config.o_threshold.score = 0.7;
mtmn_config.o_threshold.nms = 0.4;
mtmn_config.o_threshold.candidate_number = 1;

    face_id_init(&id_list, FACE_ID_SAVE_NUMBER,
ENROLL_CONFIRM_TIMES);

    Serial.printf("Starting web server on port: '%d'\n", config.server_port);
    if (httpd_start(&camera_httpd, &config) == ESP_OK)
{        httpd_register_uri_handler(camera_httpd, &index_uri);
httpd_register_uri_handler(camera_httpd, &cmd_uri);
httpd_register_uri_handler(camera_httpd, &status_uri);
httpd_register_uri_handler(camera_httpd, &capture_uri);
    }

    config.server_port += 1;
config.ctrl_port += 1;
    Serial.printf("Starting stream server on port: '%d'\n", config.server_port);
    if (httpd_start(&stream_httpd, &config) == ESP_OK)
{        httpd_register_uri_handler(stream_httpd, &stream_uri);
    }
}
```

Camera_index.h

```
//File: index_ov2640.html.gz, Size: 4316
#define index_ov2640_html_gz_len 4316 const
uint8_t index_ov2640_html_gz[] = {
 0x1F, 0x8B, 0x08, 0x08, 0x50, 0x5C, 0xAE, 0x5C, 0x00, 0x03, 0x69, 0x6E,
0x64, 0x65, 0x78, 0x5F,
 0x6F, 0x76, 0x32, 0x36, 0x34, 0x30, 0x2E, 0x68, 0x74, 0x6D, 0x6C, 0x00,
0xE5, 0x5D, 0x7B, 0x73,
```

0xD3, 0xC6, 0x16, 0xFF, 0x9F, 0x4F, 0x21, 0x04, 0x25, 0xF6, 0x34, 0x76, 0x6C, 0xC7, 0x84, 0xE0,
0xDA, 0xE2, 0x42, 0x08, 0xD0, 0x19, 0x5E, 0x25, 0x2D, 0x74, 0xA6, 0xD3, 0x81, 0xB5, 0xB4, 0xB2,
0x55, 0x64, 0xC9, 0x95, 0x56, 0x76, 0x52, 0x26, 0x9F, 0xE3, 0x7E, 0xA0, 0xFB, 0xC5, 0xEE, 0xD9,
0x87, 0xA4, 0x95, 0xBC, 0x7A, 0xD8, 0x26, 0x36, 0x97, 0xEB, 0xCC, 0x14, 0xD9, 0xDA, 0x73, 0xF6,
0x9C, 0xF3, 0x3B, 0xAF, 0x5D, 0x3D, 0x3A, 0xBC, 0x6D, 0xF9, 0x26, 0xB9, 0x9A, 0x63, 0x6D, 0x4A,
0x66, 0xAE, 0x71, 0x6B, 0xC8, 0xFF, 0xD1, 0xE0, 0x33, 0x9C, 0x62, 0x64, 0xF1, 0x43, 0xF6, 0x75,
0x86, 0x09, 0xD2, 0xCC, 0x29, 0x0A, 0x42, 0x4C, 0x46, 0x7A, 0x44, 0xEC, 0xD6, 0xA9, 0x9E, 0x3F,
0xED, 0xA1, 0x19, 0x1E, 0xE9, 0x0B, 0x07, 0x2F, 0xE7, 0x7E, 0x40, 0x74, 0xCD, 0xF4, 0x3D, 0x82,
0x3D, 0x18, 0xBE, 0x74, 0x2C, 0x32, 0x1D, 0x59, 0x78, 0xE1, 0x98, 0xB8, 0xC5, 0xBE, 0x1C, 0x3A,
0x9E, 0x43, 0x1C, 0xE4, 0xB6, 0x42, 0x13, 0xB9, 0x78, 0xD4, 0x95, 0x79, 0x11, 0x87, 0xB8, 0xD8,
0x38, 0xBF, 0x78, 0x7B, 0xDC, 0xD3, 0xDE, 0xBC, 0xEF, 0xF5, 0x4F, 0x3A, 0xC3, 0x23, 0xFE, 0x5B,
0x3A, 0x26, 0x24, 0x57, 0xF2, 0x77, 0xFA, 0x19, 0xFB, 0xD6, 0x95, 0xF6, 0x25, 0xF3, 0x13, 0xFD,
0xD8, 0x20, 0x44, 0xCB, 0x46, 0x33, 0xC7, 0xBD, 0x1A, 0x68, 0x8F, 0x03, 0x98, 0xF3, 0xF0, 0x05,
0x76, 0x17, 0x98, 0x38, 0x26, 0x3A, 0x0C, 0x91, 0x17, 0xB6, 0x42, 0x1C, 0x38, 0xF6, 0x4F, 0x2B,
0x84, 0x63, 0x64, 0x7E, 0x9E, 0x04, 0x7E, 0xE4, 0x59, 0x03, 0xED, 0x4E, 0xF7, 0x94, 0xFE, 0xAD,
0x0E, 0x32, 0x7D, 0xD7, 0x0F, 0xE0, 0xFC, 0xF9, 0x33, 0xFA, 0xB7, 0x7A, 0x9E, 0xCD, 0x1E, 0x3A,
0xFF, 0xE0, 0x81, 0xD6, 0x3D, 0x99, 0x5F, 0x66, 0xCE, 0x5F, 0xDF, 0xCA, 0x7C, 0x9D, 0xF6, 0x8A,
0xA4, 0x17, 0xF4, 0xA7, 0xE5, 0xF4, 0x21, 0x36, 0x89, 0xE3, 0x7B, 0xED, 0x19, 0x72, 0x3C, 0x05,
0x27, 0xCB, 0x09, 0xE7, 0x2E, 0x02, 0x1B, 0xD8, 0x2E, 0x2E, 0xE5, 0x73, 0x67, 0x86, 0xBD, 0xE8,
0xB0, 0x82, 0x1B, 0x65, 0xD2, 0xB2, 0x9C, 0x80, 0x8F, 0x1A, 0x50, 0x3B, 0x44, 0x33, 0xAF, 0x92,
0x6D, 0x99, 0x5C, 0x9E, 0xEF, 0x61, 0x85, 0x01, 0xE9, 0x44, 0xCB, 0x00, 0xCD, 0xE9, 0x00, 0xFA,

0xEF, 0xEA, 0x90, 0x99, 0xE3, 0x71, 0xA7, 0x1A, 0x68, 0xC7, 0xFD, 0xCE,
0xFC, 0xB2, 0x02, 0xCA,
0xE3, 0x13, 0xFA, 0xB7, 0x3A, 0x68, 0x8E, 0x2C, 0xCB, 0xF1, 0x26, 0x03,
0xED, 0x54, 0xC9, 0xC2,
0x0F, 0x2C, 0x1C, 0xB4, 0x02, 0x64, 0x39, 0x51, 0x38, 0xD0, 0xFA, 0xAA,
0x31, 0x33, 0x14, 0x4C,
0x40, 0x16, 0xE2, 0x83, 0xB0, 0xAD, 0xAE, 0x52, 0x12, 0x31, 0x24, 0x70,
0x26, 0x53, 0x02, 0x90,
0xAE, 0x8C, 0xC9, 0x1B, 0x4D, 0x84, 0x50, 0x15, 0x9E, 0xA5, 0x76, 0x53,
0x5B, 0x0D, 0xB9, 0xCE,
0xC4, 0x6B, 0x39, 0x04, 0xCF, 0x40, 0x9D, 0x90, 0x04, 0x98, 0x98, 0xD3,
0x32, 0x51, 0x6C, 0x67,
0x12, 0x05, 0x58, 0x21, 0x48, 0x62, 0xB7, 0x12, 0x85, 0xE1, 0xE4, 0xEA,
0xA9, 0xD6, 0x12, 0x8F,
0x3F, 0x3B, 0xA4, 0x25, 0x6C, 0x32, 0xC6, 0xB6, 0x1F, 0x60, 0xE5, 0xC8,
0x78, 0x84, 0xEB, 0x9B,
0x9F, 0x5B, 0x21, 0x41, 0x01, 0xA9, 0xC3, 0x10, 0xD9, 0x04, 0x07, 0xD5,
0xFC, 0x30, 0xF5, 0x8A,
0x6A, 0x6E, 0xC5, 0xD3, 0x8A, 0x01, 0x8E, 0xE7, 0x3A, 0x1E, 0xAE, 0x2F,
0x5E, 0xD1, 0xBC, 0x59,
0x76, 0x7C, 0x54, 0x0D, 0x60, 0x9C, 0xD9, 0xA4, 0xCC, 0x4B, 0x98, 0xAE,
0xAB, 0x93, 0x89, 0xB8,
0xE9, 0x76, 0x3A, 0x3F, 0xAC, 0x9E, 0x9C, 0x62, 0xEE, 0xA6, 0x28, 0x22,
0xFE, 0xF6, 0x11, 0xB1,
0x12, 0x56, 0x39, 0x3D, 0xFE, 0x35, 0xC3, 0x96, 0x83, 0xB4, 0x86, 0x14,
0xCE, 0xA7, 0x1D, 0xF0,
0xA9, 0xA6, 0x86, 0x3C, 0x4B, 0x6B, 0xF8, 0x81, 0x03, 0x81, 0x80, 0x58,
0xBA, 0x71, 0xE1, 0x17,
0x28, 0x1C, 0x73, 0xDC, 0x54, 0xA8, 0x5C, 0x12, 0x33, 0xB2, 0x45, 0xD4,
0x61, 0x43, 0x3F, 0x35,
0x52, 0x0E, 0xFD, 0x54, 0x06, 0x90, 0x42, 0x47, 0xC6, 0xBE, 0x0C, 0x2F,
0x59, 0xC2, 0x22, 0xCC,
0xE8, 0x67, 0x86, 0x2E, 0x5B, 0xA5, 0xD8, 0xC5, 0x83, 0x62, 0x0C, 0xA1,
0xCC, 0x9A, 0x0D, 0x18,
0xBA, 0x98, 0x6A, 0x2D, 0x8D, 0x66, 0xC9, 0xA6, 0x9A, 0x46, 0x30, 0x55,
0x43, 0x4E, 0x3F, 0xB2,
0x53, 0xAC, 0xA1, 0xAE, 0x5A, 0xD5, 0x34, 0x77, 0xF0, 0x3F, 0x95, 0x0F,
0x71, 0x4D, 0x0A, 0xB3,
0x08, 0xFD, 0xD4, 0xCF, 0x24, 0x29, 0xB3, 0xCA, 0x6C, 0xA2, 0x60, 0x5C,
0x9C, 0x51, 0x56, 0xF8,
0x16, 0x45, 0xB7, 0x82, 0x6B, 0xB9, 0x08, 0x75, 0xB3, 0x8B, 0x82, 0x71,
0x99, 0x0C, 0x95, 0x59,

0x86, 0x7E, 0xAE, 0x6B, 0xF4, 0x1B, 0x77, 0xC6, 0x11, 0x21, 0xBE, 0x17,
0x6E, 0x55, 0xA2, 0x8A,
0xE2, 0xEC, 0xAF, 0x28, 0x24, 0x8E, 0x7D, 0xD5, 0x12, 0x21, 0x0D, 0x71,
0x36, 0x47, 0xD0, 0x42,
0x8E, 0x31, 0x59, 0x62, 0x5C, 0xDE, 0x6E, 0x78, 0x68, 0x01, 0x79, 0x67,
0x32, 0x71, 0x55, 0xBE,
0x67, 0x46, 0x41, 0x48, 0xFB, 0xB6, 0xB9, 0xEF, 0x00, 0xE3, 0x60, 0x75,
0xE2, 0x6C, 0x0C, 0xD6,
0x9C, 0xA8, 0x65, 0x8E, 0x15, 0x73, 0xF9, 0x11, 0xA1, 0x36, 0x56, 0x22,
0xE1, 0x83, 0x3A, 0x0E,
0xB9, 0x52, 0x9E, 0x13, 0x91, 0xA8, 0x38, 0x13, 0x87, 0x60, 0x69, 0x59,
0xC8, 0xCA, 0x35, 0x30,
0xA7, 0xD8, 0xFC, 0x8C, 0xAD, 0x1F, 0x2B, 0xDB, 0xB0, 0xAA, 0xF6, 0xB0,
0xED, 0x78, 0xF3, 0x88,
0xB4, 0x68, 0x3B, 0x35, 0xBF, 0x11, 0xCC, 0x99, 0x43, 0xC6, 0x2A, 0xF6,
0x7A, 0x65, 0x4D, 0xC5,
0xFD, 0xF9, 0x65, 0xB9, 0x11, 0x64, 0x61, 0x0D, 0x17, 0x8D, 0xB1, 0x5B,
0x26, 0xB2, 0x08, 0x86,
0x82, 0xB4, 0x2B, 0x72, 0x55, 0x71, 0xEF, 0xC6, 0x24, 0x4B, 0x8B, 0x57,
0xFF, 0xC1, 0x0F, 0xB5,
0xED, 0xC8, 0x8E, 0x0F, 0x33, 0x3F, 0x85, 0xD8, 0x85, 0x00, 0x2B, 0x6A,
0xBD, 0x61, 0xCC, 0x12,
0x64, 0x28, 0x9D, 0x20, 0x40, 0xDE, 0x04, 0x43, 0x2E, 0xB8, 0x3C, 0x8C,
0x0F, 0xCB, 0x17, 0x06,
0xB5, 0xD4, 0xA7, 0xA9, 0xFA, 0x7E, 0xF9, 0x42, 0x84, 0x27, 0x84, 0x0D,
0x9A, 0x11, 0x09, 0xD6,
0xD2, 0xF9, 0xBB, 0x4A, 0xA7, 0xE0, 0xFD, 0x88, 0x32, 0x60, 0xB2, 0x2E,
0xA5, 0xEC, 0xEF, 0x2B,
0x33, 0x42, 0xBC, 0xD2, 0xB3, 0xED, 0xAA, 0xB5, 0xA2, 0x6D, 0x1F, 0x77,
0x8E, 0xFB, 0x95, 0x0D,
0x93, 0x52, 0xCB, 0xDC, 0x7A, 0x51, 0x91, 0x31, 0x92, 0x6C, 0x52, 0x0D,
0xC1, 0x60, 0xEA, 0x2F,
0x70, 0xA0, 0x00, 0x22, 0x27, 0x6E, 0xFF, 0x61, 0xDF, 0xAA, 0xC1, 0x0D,
0x41, 0xBE, 0x5F, 0xA8,
0xB2, 0x69, 0x96, 0x5D, 0xAF, 0x6B, 0xF6, 0x4A, 0x1D, 0x93, 0xB3, 0x6B,
0x83, 0x37, 0xA0, 0xB1,
0x8B, 0xAD, 0x92, 0xF4, 0x6C, 0x61, 0x1B, 0x45, 0x2E, 0xA9, 0xB0, 0x37,
0xEA, 0xD0, 0xBF, 0xB2,
0x19, 0x59, 0x5C, 0xFD, 0x41, 0x37, 0x3A, 0x46, 0x2C, 0x12, 0xFE, 0x54,
0xCC, 0x19, 0xD7, 0x4E,
0x34, 0x9F, 0x63, 0x04, 0xA3, 0x4C, 0x5C, 0xB4, 0x24, 0xAD, 0xD5, 0x33,
0xAB, 0x13, 0x57, 0xAD,

0x85, 0x68, 0xA5, 0x2B, 0x26, 0xDD, 0xD0, 0x5A, 0x3A, 0x0F, 0x6C, 0xDF,
0x8C, 0x54, 0x65, 0xBA,
0x9E, 0x4B, 0xAD, 0xF2, 0x1B, 0xC4, 0x26, 0x0B, 0x5D, 0x87, 0x39, 0x76,
0xE4, 0x79, 0x14, 0xD1,
0x16, 0x09, 0x40, 0x4D, 0xC5, 0x44, 0xF5, 0x0C, 0xB7, 0x51, 0x74, 0x66,
0x0C, 0x5B, 0xB4, 0x19,
0x93, 0x0B, 0x40, 0x45, 0xA2, 0x48, 0x72, 0x88, 0x16, 0xFA, 0xA0, 0x54,
0xCC, 0x6A, 0x3B, 0xBB,
0x90, 0x69, 0x34, 0x53, 0x35, 0x06, 0xF1, 0x64, 0x5D, 0xA8, 0x62, 0x7C,
0xBA, 0x60, 0x32, 0x46,
0x8D, 0xCE, 0x61, 0xE7, 0xF0, 0x18, 0xFE, 0xA3, 0x68, 0xD0, 0xCB, 0x9D,
0x4B, 0x98, 0xB7, 0xC0,
0xF3, 0x72, 0xC9, 0xA7, 0x7A, 0x9F, 0xA4, 0x28, 0x8D, 0x55, 0x62, 0x51,
0x3F, 0x92, 0xB2, 0x1B,
0x26, 0xDD, 0x76, 0x45, 0x61, 0x29, 0x70, 0xE9, 0xF5, 0x1D, 0x51, 0xE1,
0x2D, 0xEB, 0x42, 0x3C,
0xF3, 0xFF, 0x69, 0xF1, 0xAA, 0xFA, 0x7F, 0xEF, 0xED, 0x92, 0x29, 0xBE,
0x6B, 0x4F, 0x5F, 0xDB,
0x2E, 0xE1, 0xBE, 0x7D, 0xA3, 0x53, 0x8C, 0x7A, 0x4B, 0xF4, 0x33, 0x20,
0xA1, 0x07, 0x8B, 0xAA,
0x00, 0x56, 0x57, 0x85, 0x3D, 0x8F, 0x34, 0x66, 0x03, 0x1B, 0xD8, 0x8E,
0xEB, 0xB6, 0x5C, 0x7F,
0x59, 0xDD, 0x89, 0x94, 0x7B, 0xF2, 0x8A, 0x9F, 0x56, 0xBB, 0xFC, 0xA6,
0xD2, 0x46, 0x90, 0xB9,
0xFE, 0x27, 0xA4, 0xFD, 0xBE, 0x03, 0xAE, 0x34, 0x34, 0x36, 0x2B, 0x14,
0x1B, 0xF8, 0xE3, 0x76,
0x13, 0xD5, 0x72, 0x25, 0xDE, 0x09, 0x96, 0x2E, 0xE6, 0xC2, 0xA5, 0x43,
0xCC, 0xE9, 0x06, 0x8B,
0xAA, 0xB9, 0x1F, 0x3A, 0xFC, 0x1A, 0x4D, 0x80, 0x5D, 0x44, 0x3B, 0xF8,
0x8D, 0x96, 0xDC, 0x95,
0x0B, 0x13, 0x99, 0xBC, 0x8E, 0x26, 0xCC, 0x74, 0xDF, 0xCE, 0x76, 0x49,
0x9B, 0xF7, 0x0E, 0xC5,
0xB9, 0x5A, 0xED, 0xD6, 0x15, 0xED, 0x7E, 0x36, 0x32, 0xD4, 0x83, 0xD6,
0xC8, 0xE8, 0x71, 0xD2,
0x9E, 0x04, 0xF8, 0xAA, 0x86, 0x32, 0x87, 0xE2, 0xDF, 0x01, 0xDF, 0x10,
0xDD, 0x7C, 0xED, 0xCF,
0x0A, 0x80, 0xF0, 0xA2, 0x76, 0x3F, 0xAC, 0x31, 0x75, 0xF1, 0x94, 0x75,
0xFC, 0x31, 0xD9, 0xEE,
0xD3, 0xF5, 0x1A, 0xE9, 0xA6, 0xA4, 0x84, 0xAA, 0x5D, 0x35, 0xAE, 0xBE,
0xCA, 0x93, 0x2E, 0xB6,
0x49, 0xC1, 0xD5, 0x0C, 0xD6, 0xA7, 0x1E, 0x97, 0x67, 0xB7, 0x96, 0xB4,
0x4F, 0x50, 0x99, 0x39,

0x92, 0x5D, 0xB9, 0x62, 0xEF, 0x53, 0x72, 0xA6, 0xD9, 0x73, 0x6D, 0xE6, 0xC5, 0x90, 0xC4, 0xED,
0x33, 0x83, 0x19, 0xC6, 0xCC, 0x44, 0xC9, 0x07, 0x78, 0xF0, 0xEF, 0x8D, 0xDE, 0x89, 0xF2, 0x62,
0x41, 0xC9, 0xE0, 0x32, 0xD1, 0x0A, 0xB7, 0xB5, 0x56, 0x4B, 0x56, 0xE1, 0x02, 0x59, 0xCE, 0x45,
0x4A, 0xA0, 0xCA, 0xA3, 0xB2, 0x2C, 0xC3, 0xAC, 0xEE, 0xD1, 0x94, 0x3A, 0xBB, 0x33, 0x43, 0xD0,
0xF6, 0x52, 0x77, 0x45, 0xC0, 0x51, 0x85, 0x5F, 0x1D, 0x77, 0x97, 0x36, 0x0D, 0xBB, 0x27, 0x9D,
0x8A, 0x29, 0x4D, 0xD7, 0x0F, 0xCB, 0xE3, 0x0A, 0x8D, 0xC1, 0x7E, 0x11, 0x51, 0x4C, 0x24, 0xB6,
0x2E, 0x95, 0x3B, 0x4F, 0xCC, 0xB9, 0x95, 0x67, 0x6A, 0x95, 0xEE, 0xD2, 0x98, 0x2A, 0x0F, 0xC7,
0x9C, 0xCD, 0xBB, 0x1D, 0x65, 0xA6, 0x2D, 0xDD, 0x7F, 0x23, 0xF8, 0x12, 0xD6, 0x9B, 0xF4, 0x82,
0xDC, 0x40, 0x33, 0xB1, 0x3A, 0x8D, 0x66, 0x8A, 0x5C, 0xB7, 0xCE, 0x26, 0x60, 0x29, 0x0E, 0x53,
0xC7, 0xB2, 0x70, 0xE9, 0x2E, 0x27, 0x5D, 0xF3, 0xE6, 0x58, 0xC4, 0x47, 0xC3, 0x23, 0xE9, 0x06,
0x96, 0xE1, 0x51, 0x7A, 0xAF, 0xCD, 0x90, 0xDE, 0xC5, 0x22, 0xDF, 0xE7, 0xC2, 0x2F, 0xB2, 0x68,
0xA6, 0x8B, 0xC2, 0x70, 0xA4, 0xD3, 0xBB, 0x31, 0xF4, 0xEC, 0x6D, 0x2F, 0x43, 0xCB, 0x59, 0x68,
0x8E, 0x35, 0xD2, 0x5D, 0x7F, 0xE2, 0xE7, 0xCE, 0xB1, 0xF3, 0x7C, 0xDB, 0x1B, 0x22, 0x75, 0xA4,
0x67, 0x2E, 0x09, 0xE8, 0x8C, 0x2A, 0xFD, 0x49, 0x37, 0xEE, 0xDD, 0x79, 0xF8, 0xE0, 0xC1, 0xC9,
0x4F, 0xF7, 0xBC, 0x71, 0x38, 0x17, 0xFF, 0xFD, 0x95, 0x5F, 0x41, 0x79, 0xF3, 0xBE, 0x77, 0xD2,
0x87, 0x86, 0x16, 0x13, 0xE2, 0x78, 0x93, 0x70, 0x78, 0xC4, 0x98, 0xE6, 0x04, 0x39, 0x02, 0x49,
0x0A, 0x64, 0x13, 0x09, 0x5D, 0x25, 0x5E, 0x3C, 0x24, 0x84, 0x1C, 0x35, 0x46, 0x81, 0x62, 0x08,
0x1B, 0xC6, 0xDB, 0x05, 0xD6, 0x69, 0xE9, 0x2C, 0xB1, 0x8D, 0xFD, 0xCB, 0xBC, 0x06, 0x4C, 0x29,
0x91, 0xF5, 0xC4, 0x28, 0x6C, 0x15, 0x31, 0x04, 0x32, 0x46, 0x4E, 0xAF, 0x87, 0x14, 0x8C, 0x49,
0xE4, 0x13, 0xD6, 0x97, 0xB6, 0xE7, 0xF9, 0xD4, 0x76, 0x80, 0x66, 0x98, 0x26, 0x22, 0xF1, 0x63,
0x31, 0x9B, 0x3C, 0x12, 0x09, 0xA5, 0x6E, 0xBC, 0xC3, 0x2C, 0x5C, 0x01, 0x65, 0xA5, 0x59, 0x57,

0xB8, 0x88, 0x0C, 0x9A, 0x99, 0x5F, 0x8F, 0x45, 0x14, 0x3B, 0xA6, 0x2D,
0xC4, 0xDC, 0xA6, 0x42,
0x20, 0xC6, 0xCE, 0x9F, 0x33, 0x07, 0x5B, 0x20, 0x37, 0x02, 0xD3, 0x76,
0x3B, 0xBA, 0xF1, 0xDB,
0xEF, 0xCF, 0x1F, 0x37, 0x20, 0x11, 0x75, 0x2E, 0xBB, 0xBD, 0x4E, 0xA7,
0x39, 0x3C, 0xE2, 0x43,
0xD6, 0xE6, 0xF5, 0x50, 0x37, 0x2E, 0x18, 0xAB, 0xDE, 0x29, 0xB0, 0xEA,
0xF4, 0xFA, 0x9B, 0xB3,
0x3A, 0xD5, 0x0D, 0xC6, 0x09, 0x98, 0x5C, 0x3E, 0x38, 0x39, 0xDD, 0x9C,
0xD1, 0x03, 0x90, 0xE9,
0x3D, 0x70, 0x3A, 0x05, 0xED, 0x4E, 0xB6, 0x51, 0xEE, 0x44, 0x37, 0x28,
0x1F, 0x88, 0x8A, 0xCB,
0xFE, 0xE9, 0x16, 0x7C, 0xEE, 0xEB, 0xA2, 0x24, 0x52, 0x97, 0x8D, 0x8F,
0x74, 0xE3, 0xEC, 0xE7,
0x67, 0x8D, 0x3E, 0xC8, 0xD8, 0x7B, 0x78, 0xB2, 0x39, 0xEF, 0xBE, 0x6E,
0xFC, 0x42, 0x85, 0x3C,
0xEE, 0x01, 0xA3, 0xFE, 0x16, 0x42, 0x1E, 0xEB, 0xC6, 0x0B, 0xC6, 0x09,
0xB8, 0x5C, 0x76, 0x1F,
0x6C, 0x21, 0x12, 0xB8, 0xD7, 0x2F, 0x8C, 0x13, 0xF8, 0x17, 0x75, 0xAF,
0x9A, 0x9C, 0x20, 0x5F,
0x32, 0xD3, 0x94, 0xC4, 0xE9, 0x6A, 0xF6, 0xC9, 0x9C, 0x2E, 0x0B, 0xE3,
0xBF, 0x23, 0x28, 0x1D,
0xE4, 0x6A, 0xED, 0x20, 0x16, 0x74, 0xA0, 0x12, 0x3F, 0xA8, 0x17, 0xBF,
0x92, 0x24, 0xC9, 0x65,
0x39, 0xDD, 0xE8, 0x76, 0x2A, 0x34, 0x60, 0xB4, 0x72, 0x16, 0x64, 0xC4,
0x19, 0x05, 0x74, 0xDA,
0x49, 0xB0, 0x18, 0xA6, 0xB7, 0x7E, 0x80, 0x8F, 0x1E, 0xEB, 0x52, 0x5C,
0x6F, 0x94, 0x22, 0x14,
0xD2, 0xA2, 0x4B, 0xDD, 0x38, 0x39, 0xAE, 0xB2, 0xF7, 0x16, 0x70, 0x8C,
0x59, 0x9B, 0xE2, 0xE1,
0x30, 0x5C, 0x1B, 0x91, 0x94, 0x54, 0x37, 0x9E, 0x24, 0xC7, 0xDB, 0xE0,
0xD2, 0xEA, 0x6D, 0x81,
0x8B, 0x24, 0x0E, 0x87, 0xA6, 0xD5, 0x13, 0xD0, 0xF4, 0xF4, 0x34, 0x22,
0xBE, 0x26, 0x30, 0x55,
0xD2, 0x6E, 0x83, 0x0B, 0x2D, 0xE2, 0x01, 0x0A, 0xC9, 0xDA, 0xA8, 0xC4,
0x84, 0x90, 0xD6, 0xC4,
0xD1, 0xDE, 0x10, 0x49, 0x44, 0xF9, 0x0E, 0xF0, 0x08, 0x11, 0x89, 0x02,
0x76, 0x43, 0xDC, 0xDA,
0x88, 0xA4, 0xA4, 0x50, 0x0F, 0x93, 0xE3, 0xBD, 0xA1, 0x22, 0x89, 0xF3,
0x3D, 0xE0, 0x32, 0xC7,
0xA6, 0x83, 0xDC, 0x8F, 0xD8, 0xB6, 0xA1, 0x64, 0xAD, 0x8F, 0x4D, 0x86,
0x1C, 0xF0, 0xE1, 0xDF,

0xB5, 0x73, 0xF6, 0x7D, 0xED, 0x1E, 0x31, 0xC7, 0xEE, 0x6B, 0x35, 0x8A, 0x1D, 0x75, 0xDF, 0xF2,
0xDA, 0x4F, 0xE4, 0xDC, 0xB0, 0x43, 0xE8, 0x02, 0x13, 0x3C, 0x61, 0x2B, 0xE5, 0x8D, 0x79, 0xF4,
0x74, 0xE3, 0x79, 0x80, 0xAE, 0xD8, 0xB3, 0x05, 0xDB, 0x34, 0x3D, 0xEF, 0xB0, 0xA5, 0xFD, 0x0A,
0x4B, 0xC1, 0x6D, 0x3A, 0xB0, 0xE7, 0x01, 0x86, 0x65, 0xE2, 0x56, 0x5C, 0xEE, 0x43, 0x31, 0x83,
0x83, 0xED, 0x98, 0x40, 0xC3, 0x7A, 0x81, 0xE7, 0x0E, 0xFA, 0x16, 0x1A, 0x2E, 0xB4, 0x1C, 0xAF,
0x1D, 0x16, 0x40, 0xA3, 0x1B, 0x8F, 0x3F, 0x3C, 0x59, 0x3B, 0x49, 0xF1, 0xFD, 0xE6, 0x3A, 0x1E,
0xCE, 0xB3, 0x93, 0x10, 0x50, 0x5F, 0x59, 0x6C, 0xAA, 0x23, 0xA7, 0xEE, 0x82, 0x53, 0xA1, 0x57,
0x2C, 0x20, 0xDB, 0x9E, 0xD3, 0x25, 0x35, 0xEB, 0xE9, 0x78, 0x73, 0x19, 0x0C, 0x84, 0xF8, 0x38,
0x41, 0xCE, 0xFA, 0x75, 0x25, 0x26, 0x64, 0x48, 0x69, 0xCF, 0xE1, 0x68, 0x57, 0x70, 0xF1, 0x69,
0xF7, 0x86, 0x99, 0xD0, 0x7A, 0xDF, 0xC0, 0x81, 0x20, 0x33, 0xDF, 0x5A, 0x7F, 0x3B, 0x42, 0xD0,
0xE9, 0x06, 0xA0, 0xF6, 0x0A, 0x0E, 0xD6, 0xAE, 0x32, 0x31, 0x83, 0x1B, 0x2E, 0x2F, 0x8F, 0x23,
0xE2, 0x6F, 0x53, 0x59, 0x2E, 0x22, 0xCF, 0xBB, 0xDA, 0xA6, 0xAC, 0x9C, 0xB9, 0x7E, 0x64, 0x6D,
0xCE, 0x01, 0x6A, 0xCA, 0x1B, 0xDB, 0x76, 0xCC, 0xCD, 0xAB, 0x12, 0x54, 0x94, 0x17, 0xFE, 0xAC,
0x26, 0xFD, 0x0D, 0x67, 0x71, 0x6C, 0xAE, 0x9F, 0x20, 0xB0, 0x09, 0x28, 0x9E, 0x9F, 0x69, 0x17,
0xE7, 0xAF, 0x2F, 0xDE, 0xBC, 0xDB, 0x4D, 0x76, 0x80, 0x39, 0xF7, 0x94, 0x18, 0xA8, 0xB6, 0xFB,
0xCE, 0x09, 0x20, 0x44, 0x6F, 0x13, 0x9C, 0x7A, 0x1C, 0xA8, 0xA7, 0x17, 0x6F, 0x77, 0x85, 0x52,
0x6F, 0x7F, 0x30, 0xF5, 0xBE, 0x05, 0x9C, 0x3E, 0xBA, 0x78, 0x81, 0xDD, 0x0D, 0xB0, 0xE2, 0x84,
0x14, 0x2F, 0xED, 0x25, 0x3D, 0xDA, 0xDB, 0x42, 0x2E, 0x11, 0xE5, 0x3B, 0x58, 0xC6, 0x81, 0x57,
0x7C, 0x64, 0x42, 0x6F, 0x12, 0x3C, 0x9C, 0x52, 0x37, 0xCE, 0x2F, 0xE7, 0x7E, 0x18, 0x05, 0x35,
0x0B, 0xAA, 0x1A, 0x91, 0x6D, 0x76, 0x06, 0x53, 0x51, 0x38, 0x22, 0xF1, 0xD6, 0x20, 0xDD, 0xD9,
0x4F, 0x30, 0xE9, 0x75, 0xFA, 0x5F, 0x15, 0x15, 0xCA, 0xFC, 0x26, 0x81, 0x99, 0x6C, 0x50, 0x77,

0x26, 0xB4, 0xEE, 0x3C, 0x3F, 0xDB, 0x4D, 0x2A, 0x9B, 0xEC, 0xAD, 0xE0, 0x4C, 0xF6, 0x5A, 0x70,
0x34, 0x7E, 0x51, 0x34, 0x81, 0x69, 0xC3, 0x45, 0x84, 0x20, 0x84, 0xB5, 0xF3, 0x26, 0x0B, 0x08,
0x79, 0x53, 0xFD, 0x72, 0x9B, 0xD0, 0x89, 0xC5, 0xC8, 0x46, 0xCE, 0x71, 0x1A, 0x37, 0xF7, 0xBF,
0x6A, 0xD4, 0x1C, 0x57, 0x4A, 0xBB, 0x4D, 0xD0, 0x50, 0x4D, 0x4C, 0xEC, 0xB8, 0xF4, 0x09, 0xA6,
0x75, 0x01, 0x91, 0x68, 0x39, 0x26, 0xDA, 0x19, 0xFF, 0xB6, 0x0D, 0x36, 0xBD, 0x6D, 0xB0, 0x91,
0x25, 0xCA, 0xC2, 0x73, 0x72, 0x43, 0x95, 0xA6, 0xDB, 0x3B, 0xBD, 0x49, 0x78, 0xC6, 0xF3, 0xF5,
0x73, 0x1A, 0xD0, 0xE8, 0xC6, 0x93, 0xB7, 0xBB, 0xC9, 0x69, 0x74, 0xB2, 0x9A, 0x39, 0x6D, 0xAB,
0x0C, 0xC6, 0x94, 0xDA, 0x77, 0x2B, 0xB6, 0xDC, 0x00, 0x8D, 0x25, 0x15, 0xFC, 0xC3, 0x8E, 0xD0,
0x58, 0xD6, 0x47, 0xE3, 0x2B, 0x57, 0x98, 0xE5, 0xB7, 0x80, 0x4F, 0x80, 0x96, 0x1F, 0x27, 0x33,
0xB4, 0x36, 0x46, 0x82, 0x4E, 0x37, 0xDE, 0xA1, 0xA5, 0xF6, 0xFC, 0xD5, 0xE3, 0x9D, 0x60, 0x15,
0x4F, 0xBA, 0x1F, 0xBC, 0x12, 0x95, 0xF7, 0x8D, 0x99, 0x8B, 0xBD, 0xF5, 0x83, 0x8A, 0x12, 0xE9,
0xC6, 0x4B, 0xEC, 0x85, 0xDA, 0x99, 0x1F, 0x88, 0xB7, 0xCD, 0xEC, 0x04, 0x35, 0x36, 0xF3, 0x7E,
0x20, 0xE3, 0x4A, 0xEF, 0x1B, 0xAF, 0xE9, 0xCC, 0x09, 0x02, 0x3F, 0x58, 0x1B, 0x32, 0x41, 0xA7,
0x1B, 0x2F, 0x5A, 0xAF, 0xD8, 0xD1, 0x4E, 0xE0, 0x8A, 0x67, 0xDD, 0x0F, 0x62, 0x89, 0xCE, 0xFB,
0x06, 0x6D, 0x61, 0xBB, 0xCE, 0x7C, 0x6D, 0xC8, 0x18, 0x95, 0x6E, 0xBC, 0x6F, 0x3D, 0x83, 0x7F,
0x77, 0x02, 0x17, 0x9F, 0x71, 0x3F, 0x60, 0x09, 0x6D, 0xF7, 0x0D, 0x95, 0x65, 0x2E, 0xD7, 0x06,
0x0A, 0x68, 0x74, 0xE3, 0xE9, 0xD9, 0x07, 0xAD, 0xF1, 0xD4, 0x5F, 0x7A, 0xF4, 0xC6, 0x3F, 0xED,
0xFC, 0x75, 0x73, 0x27, 0x88, 0xD1, 0xA9, 0xF7, 0x83, 0x17, 0x53, 0x7A, 0xDF, 0x68, 0xB1, 0xBB,
0x8F, 0xC7, 0x68, 0xFD, 0x74, 0x18, 0x13, 0xD2, 0x7B, 0x5F, 0xE0, 0x48, 0x7B, 0x82, 0x76, 0x93,
0x10, 0x93, 0x79, 0x77, 0xD1, 0xB4, 0xA7, 0x4A, 0xEE, 0x1B, 0x27, 0x1B, 0x99, 0xF8, 0xA3, 0x85,
0xC9, 0x26, 0x37, 0x5E, 0x48, 0xB4, 0xBA, 0xF1, 0x0C, 0xBE, 0x68, 0x4F, 0xD9, 0x97, 0x5D, 0xB5,

0x1C, 0xF2, 0xFC, 0xBB, 0x40, 0x2D, 0xA3, 0xEF, 0x37, 0x01, 0x1C, 0x34,
0x78, 0xFE, 0xC4, 0xDB,
0xE8, 0x7E, 0xEA, 0x0C, 0xB9, 0x80, 0xEF, 0x1D, 0xFF, 0xBE, 0x5B, 0x00,
0x53, 0x21, 0x76, 0x86,
0xA1, 0xA4, 0xF7, 0x2E, 0x60, 0x8C, 0x9F, 0x49, 0x60, 0xDB, 0x02, 0xFC,
0xE5, 0x4F, 0x55, 0x48,
0x89, 0x57, 0xC2, 0xB0, 0xAD, 0x1B, 0x4C, 0x5A, 0x21, 0x71, 0x5C, 0x57,
0x37, 0x9E, 0x63, 0xA2,
0x5D, 0xD0, 0xC3, 0xE1, 0x11, 0x1F, 0x50, 0x9F, 0x8B, 0xB8, 0xE1, 0x9F,
0xBE, 0x76, 0x0D, 0xCD,
0x74, 0xE3, 0x82, 0xBE, 0x16, 0x0B, 0x78, 0xD1, 0x6F, 0xEB, 0x33, 0x63,
0x46, 0xC4, 0x5E, 0xE0,
0x83, 0x50, 0x09, 0x48, 0xE2, 0xED, 0x24, 0xBA, 0x16, 0x1F, 0x49, 0xBF,
0x19, 0xE7, 0x6C, 0xB0,
0x46, 0xBD, 0xAC, 0x7A, 0x3A, 0x7A, 0x15, 0xD6, 0x2C, 0xBE, 0x58, 0x3B,
0x3C, 0xF2, 0x90, 0xC2,
0xDC, 0x05, 0x28, 0x0C, 0xF9, 0xFB, 0xD4, 0x0A, 0x58, 0x25, 0x0F, 0x53,
0x30, 0x4B, 0xA4, 0x0F,
0x26, 0x25, 0x6A, 0xE5, 0x1F, 0x58, 0x12, 0x1B, 0xB6, 0xF5, 0x82, 0x96,
0x3D, 0x7A, 0x24, 0xEA,
0x21, 0x3D, 0x4C, 0xCC, 0xFF, 0x9F, 0x7F, 0x57, 0xF9, 0x0C, 0x7D, 0xDB,
0x5D, 0x2A, 0x98, 0xAE,
0x85, 0x81, 0x39, 0xD2, 0x8B, 0x1E, 0xCD, 0x28, 0xD0, 0xFC, 0x48, 0xA5,
0x7A, 0x6E, 0xB0, 0xC2,
0xD6, 0xC3, 0xD0, 0x0C, 0x9C, 0x39, 0x31, 0x6E, 0x59, 0xBE, 0x19, 0xCD,
0xB0, 0x47, 0xDA, 0xC8,
0xB2, 0xCE, 0x17, 0x70, 0xF0, 0xD2, 0x09, 0x09, 0x06, 0x2B, 0x34, 0x0E,
0x9E, 0xBE, 0x79, 0x75,
0xC6, 0x1F, 0x51, 0x79, 0xE9, 0x23, 0x0B, 0x5B, 0x07, 0x87, 0x9A, 0x1D,
0x79, 0xDC, 0xCD, 0x1B,
0x98, 0x8E, 0xE5, 0x6F, 0x1A, 0x5C, 0xA0, 0x40, 0x1B, 0xA3, 0x10, 0xBF,
0xF0, 0x43, 0xA2, 0x8D,
0xB4, 0x84, 0xA3, 0xEB, 0x9B, 0xEC, 0xF6, 0xC5, 0xB6, 0x1F, 0x38, 0x13,
0xC7, 0x13, 0x23, 0xB9,
0xB2, 0xBF, 0x05, 0x2E, 0x0C, 0x4D, 0xA8, 0x7E, 0xD4, 0x0E, 0x06, 0xA7,
0xDD, 0x03, 0xFA, 0x34,
0x11, 0xC0, 0x00, 0x3F, 0x00, 0x04, 0x18, 0x06, 0x40, 0x80, 0x8F, 0x0C, 0xF1,
0x38, 0x11, 0x76,
0xDB, 0xCC, 0xE4, 0x54, 0x40, 0x2A, 0x6D, 0xE3, 0x80, 0xE3, 0x74, 0x40,
0x1F, 0xAD, 0xBB, 0x4E,
0x28, 0xC3, 0xA9, 0xBF, 0x2C, 0xA3, 0x0C, 0xF0, 0xCC, 0x5F, 0xE0, 0x1C,
0x71, 0x42, 0x2D, 0xBC,

0xB9, 0x72, 0xEA, 0xD8, 0xEB, 0x0F, 0x9A, 0xF1, 0x80, 0xE4, 0xCD, 0x3D, 0x23, 0x8D, 0x04, 0x11,
0xCE, 0xB2, 0xC5, 0x5E, 0x15, 0xD7, 0x58, 0xAC, 0x52, 0xC6, 0x36, 0x72, 0xC3, 0x1C, 0xE7, 0x68,
0x6E, 0x21, 0x82, 0xDF, 0xD3, 0xDD, 0x5D, 0x18, 0xD0, 0xC0, 0xEE, 0x21, 0xDF, 0xEA, 0x3D, 0x14,
0x67, 0xDE, 0x01, 0x5F, 0x82, 0x9B, 0xE9, 0xAC, 0xF2, 0xCF, 0x40, 0x91, 0xFD, 0x3A, 0xD2, 0xBC,
0x08, 0x42, 0xF8, 0x11, 0x53, 0x41, 0x1B, 0x64, 0xCE, 0x32, 0x6A, 0x17, 0xB2, 0x93, 0x78, 0x4B,
0x31, 0x9B, 0x93, 0xFD, 0xE8, 0xD8, 0x74, 0xE2, 0x36, 0x7B, 0x67, 0xF2, 0x08, 0x78, 0x1C, 0xC4,
0xD9, 0xFD, 0x20, 0x7D, 0x15, 0xA5, 0x4C, 0xC4, 0xEC, 0xD0, 0x16, 0x7D, 0xB0, 0x38, 0xBF, 0x10,
0x27, 0x6E, 0xDF, 0x5E, 0x24, 0x7C, 0x35, 0x69, 0x18, 0x9C, 0x4A, 0x4F, 0x5C, 0xC3, 0x09, 0xE9,
0x79, 0xBF, 0x55, 0xDE, 0x39, 0x1E, 0x31, 0x73, 0x89, 0xC3, 0xAD, 0x44, 0xF2, 0x8C, 0x05, 0xEE,
0xDD, 0xCB, 0x72, 0xBB, 0x3D, 0x12, 0x54, 0xA9, 0x26, 0x7C, 0x3C, 0x44, 0x06, 0x44, 0x1E, 0xA8,
0x2D, 0x9E, 0x02, 0x15, 0x22, 0x39, 0x76, 0xE3, 0x76, 0xC6, 0xF0, 0x89, 0x8C, 0x36, 0x35, 0x91,
0x63, 0x31, 0x03, 0xB1, 0x7B, 0x20, 0x9A, 0xE9, 0x53, 0x72, 0x5C, 0xBE, 0x47, 0xCC, 0xEB, 0x1B,
0x58, 0x5C, 0x1D, 0x6D, 0x82, 0xFD, 0xA9, 0x33, 0xA7, 0x3F, 0x88, 0xF1, 0xE9, 0x54, 0x32, 0xC7,
0x49, 0x86, 0x23, 0x55, 0x2C, 0x27, 0x37, 0xFD, 0x30, 0x7E, 0xF4, 0x3A, 0x81, 0xB8, 0x56, 0x21,
0x3F, 0x95, 0xCA, 0x26, 0x07, 0x36, 0xF4, 0x5A, 0x46, 0xFA, 0x7B, 0xCE, 0xD4, 0xC9, 0xC0, 0x02,
0x26, 0x6C, 0x82, 0x55, 0x26, 0xA5, 0x92, 0xC7, 0x37, 0x8A, 0x29, 0x0C, 0xC2, 0xD8, 0x2D, 0xC7,
0xD4, 0x14, 0x6C, 0x56, 0x38, 0x2C, 0x63, 0x95, 0x2B, 0xFC, 0x0A, 0x86, 0x3C, 0x10, 0x1B, 0xBC,
0xAE, 0x3D, 0x61, 0x35, 0x8A, 0x32, 0x17, 0x31, 0x96, 0xFD, 0xFD, 0x96, 0x2C, 0xFC, 0x75, 0x1C,
0x76, 0x49, 0x0A, 0x94, 0xFD, 0x80, 0xFA, 0x7F, 0x6C, 0x69, 0x1A, 0x22, 0xA9, 0xA3, 0x89, 0x07,
0xFB, 0xE3, 0xF8, 0x48, 0xE1, 0x30, 0x21, 0xF7, 0x49, 0x91, 0x32, 0xC8, 0x89, 0x2A, 0x87, 0x08,
0xC8, 0xDD, 0xD5, 0xE4, 0x47, 0xF5, 0xC7, 0x90, 0x42, 0x3F, 0x67, 0xF8, 0xB0, 0x8B, 0x32, 0x09,

0x13, 0xFE, 0x1B, 0xBF, 0xCD, 0xA9, 0xE5, 0x7B, 0x58, 0xCD, 0x5D, 0x0E,
0x12, 0x15, 0x4F, 0x5E,
0xC2, 0xF3, 0x4C, 0xA3, 0xF1, 0xCC, 0x21, 0x0A, 0x86, 0x07, 0x90, 0xBE,
0x55, 0xBC, 0x44, 0x63,
0x97, 0x12, 0x04, 0x98, 0x44, 0x81, 0x27, 0x47, 0x21, 0xCF, 0x64, 0x7F, 0x47,
0x38, 0xB8, 0x02,
0x46, 0x9F, 0xEE, 0x7E, 0x89, 0xEB, 0xC2, 0xF5, 0x11, 0x7B, 0x34, 0xC1,
0x77, 0x1F, 0x41, 0xE5,
0x18, 0xDD, 0xFD, 0xC2, 0xA0, 0xBE, 0xBE, 0x07, 0x53, 0xC2, 0x17, 0x36,
0xF1, 0xF5, 0x27, 0xCE,
0xC2, 0xA6, 0x2F, 0x9A, 0x6D, 0x30, 0x16, 0x31, 0x6E, 0x6D, 0x32, 0xC5,
0x5E, 0x23, 0xC0, 0xE1,
0x1C, 0xD8, 0xE3, 0x34, 0x01, 0xC6, 0x33, 0xFA, 0x2E, 0x86, 0x12, 0x35,
0x69, 0x7C, 0x0A, 0x30,
0xD0, 0x81, 0x00, 0xC4, 0xD7, 0xEE, 0x7E, 0x61, 0x2C, 0xAE, 0x35, 0x1B,
0xB2, 0x40, 0x38, 0xC5,
0xD6, 0x21, 0xD4, 0x2B, 0x44, 0xE8, 0x13, 0xB8, 0x77, 0xBF, 0xC4, 0xAC,
0xDA, 0xFC, 0xA7, 0xEB,
0x4F, 0x89, 0x87, 0x24, 0x45, 0x24, 0xAE, 0x7D, 0xEC, 0x44, 0x9B, 0xF1,
0xBA, 0x60, 0x28, 0xF8,
0xC1, 0x63, 0xD7, 0x6D, 0x1C, 0xF0, 0x07, 0x95, 0x45, 0x6E, 0x6F, 0x43,
0xB3, 0x7A, 0x8E, 0x40,
0x6C, 0xB9, 0x28, 0xB0, 0x7C, 0xE5, 0x7B, 0xA6, 0xEB, 0x98, 0x9F, 0x69,
0x42, 0x6F, 0x66, 0x05,
0xE7, 0x19, 0xC2, 0x6D, 0xF3, 0x17, 0xCF, 0xBC, 0xF6, 0x2D, 0x9C, 0x73,
0xD3, 0x26, 0x15, 0xE3,
0xE8, 0x08, 0xAC, 0x8C, 0xAC, 0x38, 0x95, 0x71, 0x8C, 0xE8, 0x1B, 0x0A,
0xB8, 0x99, 0x32, 0x16,
0xE6, 0xCA, 0x08, 0x5D, 0xB8, 0xCD, 0xD2, 0x2A, 0x1F, 0xAB, 0x9C, 0xBA,
0x2D, 0x47, 0x4F, 0x4B,
0x6C, 0xF1, 0x57, 0xE8, 0x7B, 0x8D, 0xE6, 0xAD, 0xC4, 0x0C, 0xAB, 0x3C,
0xE8, 0x04, 0x12, 0x83,
0x8C, 0x89, 0x8A, 0xCC, 0x94, 0x5D, 0x0D, 0x1C, 0xA4, 0x99, 0xA4, 0xC0,
0x66, 0xF4, 0x23, 0x55,
0x42, 0x56, 0x06, 0xD9, 0xBC, 0x7F, 0x30, 0x97, 0xF9, 0xF3, 0x90, 0x97,
0x4E, 0x29, 0x23, 0x35,
0x25, 0x73, 0x71, 0xFF, 0xA3, 0xAF, 0xE8, 0x97, 0xDB, 0x17, 0xE8, 0xC9,
0xCF, 0x5D, 0x4C, 0x0F,
0x9F, 0x5C, 0xFD, 0x0C, 0x25, 0x9F, 0x37, 0x2E, 0x4C, 0x96, 0x94, 0xE0,
0x2C, 0x69, 0x1A, 0x2B,
0x29, 0xD3, 0x06, 0x53, 0xE2, 0xC1, 0x9A, 0x7E, 0x9E, 0x6F, 0xCA, 0x38,
0x24, 0xEB, 0x83, 0x0C,

0x29, 0xE5, 0x5A, 0x4D, 0x9B, 0x59, 0x15, 0x48, 0xF4, 0x72, 0xAE, 0x2B, 0xA3, 0x97, 0x16, 0x02,
0x12, 0x35, 0x73, 0xE4, 0x6A, 0x62, 0xB9, 0x25, 0x3E, 0x90, 0x8C, 0x1D, 0x12, 0x7F, 0xCE, 0x57,
0x26, 0x39, 0x27, 0x5F, 0x3A, 0x9E, 0xE5, 0x2F, 0xDB, 0xF4, 0x7C, 0x43, 0x94, 0x56, 0x59, 0xD1,
0xB6, 0xE3, 0x81, 0x01, 0x5F, 0xFC, 0xFA, 0xEA, 0x25, 0x4D, 0x39, 0xF2, 0x0A, 0xE7, 0x20, 0xDB,
0x17, 0xB1, 0x77, 0x02, 0x2B, 0x67, 0xA0, 0xB0, 0xB5, 0xA1, 0xD5, 0xE6, 0xA9, 0x26, 0x69, 0x47,
0x69, 0x24, 0xD0, 0xC3, 0x4F, 0x7C, 0x4E, 0x5A, 0x78, 0x32, 0x00, 0x37, 0x2B, 0x65, 0xF1, 0xE7,
0x79, 0x51, 0x20, 0x0E, 0x1F, 0x13, 0x02, 0xEE, 0xAA, 0x71, 0x47, 0x0E, 0x69, 0x8E, 0x11, 0xAB,
0xC3, 0x5B, 0x9A, 0x0C, 0x7E, 0x41, 0xC8, 0xA7, 0x66, 0x12, 0x31, 0x96, 0x15, 0x5E, 0xCA, 0x93,
0x68, 0x0E, 0x71, 0x89, 0x1F, 0x7D, 0x34, 0xC7, 0x90, 0x1A, 0x9F, 0x82, 0xE7, 0xB7, 0x3D, 0xD0,
0xA0, 0x79, 0x5D, 0xA6, 0x0E, 0x37, 0x57, 0x0A, 0x64, 0x5D, 0x21, 0x58, 0x12, 0x52, 0x73, 0xCB,
0xD8, 0x47, 0xCD, 0x4E, 0xF6, 0xDE, 0x73, 0x2F, 0x6E, 0x6D, 0x8B, 0x0C, 0x3B, 0x5A, 0x35, 0x2D,
0xEF, 0x6E, 0x32, 0x0C, 0xD2, 0xF4, 0xB2, 0x22, 0x6C, 0xAE, 0x81, 0x91, 0xFC, 0x22, 0x1E, 0x10,
0xCB, 0x2E, 0x07, 0x44, 0x81, 0xEC, 0xD9, 0xDE, 0x2F, 0xD7, 0x2C, 0xE4, 0x20, 0x17, 0x39, 0x4C,
0xA3, 0x2F, 0x2A, 0x98, 0xD2, 0xF2, 0x2C, 0x9C, 0xA0, 0x4E, 0x99, 0x50, 0xE6, 0xBF, 0xD2, 0x7A,
0xC1, 0x67, 0x88, 0xA5, 0xCD, 0xF7, 0xA8, 0xD9, 0xDA, 0x70, 0x16, 0x81, 0x95, 0x66, 0xB1, 0x4F,
0xF2, 0xDF, 0x68, 0xC3, 0x96, 0x04, 0x0F, 0x34, 0x70, 0x65, 0x41, 0x0D, 0xA7, 0xA5, 0x4C, 0x20,
0xBA, 0xBD, 0x0A, 0x02, 0xE9, 0xAE, 0x27, 0x89, 0x56, 0xEA, 0x22, 0x4B, 0xD3, 0x5F, 0xFE, 0x3E,
0x1D, 0xC6, 0x02, 0xB8, 0xAE, 0x6A, 0xAE, 0xC0, 0x09, 0xC6, 0x35, 0x13, 0xB7, 0xA1, 0x44, 0xA2,
0xAD, 0x92, 0x9C, 0xA6, 0xA0, 0x2D, 0x5E, 0x6D, 0x89, 0x73, 0xDE, 0x54, 0xD4, 0x0A, 0xAF, 0xB6,
0xC1, 0xD7, 0x92, 0x83, 0xC4, 0xF7, 0x3F, 0xA6, 0x26, 0xC4, 0xE5, 0xF6, 0xC6, 0xB2, 0xBD, 0xE3,
0xE5, 0x40, 0x05, 0x85, 0x7C, 0x9B, 0x26, 0x37, 0x17, 0xAE, 0x69, 0x2E, 0x2C, 0xCC, 0x45, 0x09,

0xD2, 0x0E, 0xB4, 0x7A, 0x6D, 0x92, 0xF8, 0xFF, 0x87, 0x27, 0xA9, 0x66,
0xCB, 0x71, 0xA9, 0x9C,
0xA2, 0xF7, 0x97, 0xD4, 0x2B, 0x27, 0xC8, 0x3C, 0xCB, 0xC1, 0xD5, 0x5A,
0x8E, 0xEB, 0xA9, 0x15,
0xAF, 0x1D, 0x28, 0x41, 0xAA, 0x96, 0x7A, 0x85, 0x11, 0xAB, 0x92, 0xEC,
0x75, 0xB3, 0xFF, 0xDD,
0x42, 0xF2, 0x66, 0x89, 0x44, 0x58, 0xBE, 0x51, 0x5C, 0x59, 0x3D, 0xF9,
0x30, 0x49, 0xC9, 0x64,
0x8D, 0x52, 0x49, 0x9A, 0x8C, 0x94, 0xA8, 0x13, 0x39, 0x4A, 0xA9, 0xE3,
0x41, 0xBC, 0xEC, 0x26,
0x5F, 0x6B, 0x19, 0x2B, 0x19, 0x9D, 0x06, 0x4E, 0xCA, 0x80, 0x77, 0xFC,
0x86, 0x76, 0x3F, 0xBF,
0x26, 0xE6, 0xBD, 0x17, 0x57, 0x36, 0xD7, 0x71, 0xC9, 0x03, 0x12, 0x95,
0x32, 0x63, 0x92, 0x00,
0xE1, 0xF4, 0x45, 0x62, 0x56, 0x8A, 0x82, 0x5C, 0x1C, 0x90, 0x86, 0xFE,
0xD6, 0xC5, 0x74, 0xBD,
0x22, 0x9E, 0xC6, 0x39, 0xFB, 0xF9, 0x99, 0xE6, 0x07, 0x1A, 0x7F, 0xC1,
0x5D, 0x90, 0xBC, 0x5B,
0x44, 0x13, 0x6F, 0x7F, 0x62, 0xAB, 0x42, 0x9A, 0x83, 0xC8, 0xD4, 0x09,
0xA1, 0x49, 0xA6, 0x4F,
0xDE, 0xE2, 0xDB, 0x7A, 0xF2, 0x82, 0xA7, 0x4A, 0xF5, 0x78, 0x57, 0xFC,
0x53, 0xA2, 0x48, 0xCE,
0x9C, 0x9C, 0x26, 0xB5, 0xE5, 0x6D, 0xA1, 0xE3, 0x4A, 0x22, 0x2A, 0x5B,
0x87, 0xAE, 0x61, 0xC2,
0xE4, 0xF4, 0x37, 0x6B, 0x45, 0xB5, 0x02, 0x95, 0x86, 0x4C, 0xC8, 0x52,
0x5B, 0xA6, 0xBA, 0xAE,
0x58, 0x53, 0xB5, 0xD8, 0x2F, 0x41, 0x94, 0xEE, 0x79, 0x29, 0xB3, 0x7C,
0x31, 0x2A, 0xDC, 0xE2,
0xBC, 0xB0, 0xF2, 0xCF, 0xF0, 0x28, 0xDE, 0x59, 0xE5, 0xDF, 0xF8, 0xAB,
0x8B, 0x86, 0x47, 0xFC,
0x7F, 0x22, 0xF6, 0x5F, 0x04, 0x9C, 0x39, 0x76, 0x5C, 0x6C, 0x00, 0x00
};

//File: index_ov3660.html.gz, Size: 4408
#define index_ov3660_html_gz_len 4408 const
uint8_t index_ov3660_html_gz[] = {
0x1F, 0x8B, 0x08, 0x08, 0x28, 0x5C, 0xAE, 0x5C, 0x00, 0x03, 0x69, 0x6E,
0x64, 0x65, 0x78, 0x5F,
0x6F, 0x76, 0x33, 0x36, 0x36, 0x30, 0x2E, 0x68, 0x74, 0x6D, 0x6C, 0x00,
0xE5, 0x5D, 0xEB, 0x92,
0xD3, 0xC6, 0x12, 0xFE, 0xCF, 0x53, 0x08, 0x41, 0x58, 0x6F, 0x65, 0xED,

0xF5, 0x6D, 0xCD, 0xE2,
0xD8, 0xE6, 0xC0, 0xB2, 0x84, 0x54, 0x01, 0x49, 0x20, 0x21, 0xA9, 0x4A,
0xA5, 0x60, 0x2C, 0x8D,
0xED, 0x09, 0xB2, 0xE4, 0x48, 0x23, 0x7B, 0x37, 0xD4, 0x3E, 0xC7, 0x79,
0xA0, 0xF3, 0x62, 0xA7,
0xE7, 0x22, 0x69, 0x24, 0x8F, 0x2E, 0xB6, 0x59, 0x9B, 0xC3, 0x31, 0x55,
0x20, 0x5B, 0xD3, 0x3D,
0xDD, 0xFD, 0xF5, 0x6D, 0x46, 0x17, 0x06, 0x77, 0x6D, 0xCF, 0xA2, 0xD7,
0x0B, 0x6C, 0xCC, 0xE8,
0xDC, 0x19, 0xDD, 0x19, 0x88, 0x7F, 0x0C, 0xF8, 0x0C, 0x66, 0x18, 0xD9,
0xE2, 0x90, 0x7F, 0x9D,
0x63, 0x8A, 0x0C, 0x6B, 0x86, 0xFC, 0x00, 0xD3, 0xA1, 0x19, 0xD2, 0x49,
0xFD, 0xDC, 0xCC, 0x9E,
0x76, 0xD1, 0x1C, 0x0F, 0xCD, 0x25, 0xC1, 0xAB, 0x85, 0xE7, 0x53, 0xD3,
0xB0, 0x3C, 0x97, 0x62,
0x17, 0x86, 0xAF, 0x88, 0x4D, 0x67, 0x43, 0x1B, 0x2F, 0x89, 0x85, 0xEB,
0xFC, 0xCB, 0x09, 0x71,
0x09, 0x25, 0xC8, 0xA9, 0x07, 0x16, 0x72, 0xF0, 0xB0, 0xA5, 0xF2, 0xA2,
0x84, 0x3A, 0x78, 0x74,
0xF9, 0xF6, 0xA7, 0x4E, 0xDB, 0xF8, 0xF1, 0x5D, 0xA7, 0xD7, 0x6B, 0x0E,
0x4E, 0xC5, 0x6F, 0xC9,
0x98, 0x80, 0x5E, 0xAB, 0xDF, 0xD9, 0x67, 0xEC, 0xD9, 0xD7, 0xC6, 0xA7,
0xD4, 0x4F, 0xEC, 0x33,
0x01, 0x21, 0xEA, 0x13, 0x34, 0x27, 0xCE, 0x75, 0xDF, 0x78, 0xE2, 0xC3,
0x9C, 0x27, 0x2F, 0xB0,
0xB3, 0xC4, 0x94, 0x58, 0xE8, 0x24, 0x40, 0x6E, 0x50, 0x0F, 0xB0, 0x4F,
0x26, 0xDF, 0xAD, 0x11,
0x8E, 0x91, 0xF5, 0x71, 0xEA, 0x7B, 0xA1, 0x6B, 0xF7, 0x8D, 0x7B, 0xAD,
0x73, 0xF6, 0x67, 0x7D,
0x90, 0xE5, 0x39, 0x9E, 0x0F, 0xE7, 0x2F, 0x9F, 0xB3, 0x3F, 0xEB, 0xE7,
0xF9, 0xEC, 0x01, 0xF9,
0x07, 0xF7, 0x8D, 0x56, 0x6F, 0x71, 0x95, 0x3A, 0x7F, 0x73, 0x27, 0xF5,
0x75, 0xD6, 0xCE, 0x93,
0x5E, 0xD2, 0x9F, 0x17, 0xD3, 0x07, 0xD8, 0xA2, 0xC4, 0x73, 0x1B, 0x73,
0x44, 0x5C, 0x0D, 0x27,
0x9B, 0x04, 0x0B, 0x07, 0x81, 0x0D, 0x26, 0x0E, 0x2E, 0xE4, 0x73, 0x6F,
0x8E, 0xDD, 0xF0, 0xA4,
0x84, 0x1B, 0x63, 0x52, 0xB7, 0x89, 0x2F, 0x46, 0xF5, 0x99, 0x1D, 0xC2,
0xB9, 0x5B, 0xCA, 0xB6,
0x48, 0x2E, 0xD7, 0x73, 0xB1, 0xC6, 0x80, 0x6C, 0xA2, 0x95, 0x8F, 0x16,
0x6C, 0x00, 0xFB, 0x77,
0x7D, 0xC8, 0x9C, 0xB8, 0xC2, 0xA9, 0xFA, 0x46, 0xA7, 0xDB, 0x5C, 0x5C,

0x95, 0x40, 0xD9, 0xE9,
0xB1, 0x3F, 0xEB, 0x83, 0x16, 0xC8, 0xB6, 0x89, 0x3B, 0xED, 0x1B, 0xE7,
0x5A, 0x16, 0x9E, 0x6F,
0x63, 0xBF, 0xEE, 0x23, 0x9B, 0x84, 0x41, 0xDF, 0xE8, 0xEA, 0xC6, 0xCC,
0x91, 0x3F, 0x05, 0x59,
0xA8, 0x07, 0xC2, 0xD6, 0x5B, 0x5A, 0x49, 0xE4, 0x10, 0x9F, 0x4C, 0x67,
0x14, 0x20, 0x5D, 0x1B,
0x93, 0x35, 0x9A, 0x0C, 0xA1, 0x32, 0x3C, 0x0B, 0xED, 0xA6, 0xB7, 0x1A,
0x72, 0xC8, 0xD4, 0xAD,
0x13, 0x8A, 0xE7, 0xA0, 0x4E, 0x40, 0x7D, 0x4C, 0xAD, 0x59, 0x91, 0x28,
0x13, 0x32, 0x0D, 0x7D,
0xAC, 0x11, 0x24, 0xB6, 0x5B, 0x81, 0xC2, 0x70, 0x72, 0xFD, 0x54, 0x7D,
0x85, 0xC7, 0x1F, 0x09,
0xAD, 0x4B, 0x9B, 0x8C, 0xF1, 0xC4, 0xF3, 0xB1, 0x76, 0x64, 0x34, 0xC2,
0xF1, 0xAC, 0x8F, 0xF5,
0x80, 0x22, 0x9F, 0x56, 0x61, 0x88, 0x26, 0x14, 0xFB, 0xE5, 0xFC, 0x30,
0xF3, 0x8A, 0x72, 0x6E,
0xF9, 0xD3, 0xCA, 0x01, 0xC4, 0x75, 0x88, 0x8B, 0xAB, 0x8B, 0x97, 0x37,
0x6F, 0x9A, 0x9D, 0x18,
0x55, 0x01, 0x18, 0x32, 0x9F, 0x16, 0x79, 0x09, 0xD7, 0x75, 0x7D, 0x32, 0x19,
0x37, 0xAD, 0x66,
0xF3, 0x9B, 0xF5, 0x93, 0x33, 0x2C, 0xDC, 0x14, 0x85, 0xD4, 0xDB, 0x3D,
0x22, 0xD6, 0xC2, 0x2A,
0xA3, 0xC7, 0xBF, 0xE6, 0xD8, 0x26, 0xC8, 0xA8, 0x29, 0xE1, 0x7C, 0xDE,
0x04, 0x9F, 0x3A, 0x36,
0x90, 0x6B, 0x1B, 0x35, 0xCF, 0x27, 0x10, 0x08, 0x88, 0xA7, 0x1B, 0x07,
0x7E, 0x81, 0xC2, 0xB1,
0xC0, 0xC7, 0x1A, 0x95, 0x0B, 0x62, 0x46, 0xB5, 0x88, 0x3E, 0x6C, 0xD8,
0xA7, 0x42, 0xCA, 0x61,
0x9F, 0xD2, 0x00, 0xD2, 0xE8, 0xC8, 0xD9, 0x17, 0xE1, 0xA5, 0x4A, 0x98,
0x87, 0x19, 0xFB, 0xCC,
0xD1, 0x55, 0xBD, 0x10, 0xBB, 0x68, 0x50, 0x84, 0x21, 0x94, 0x59, 0xAB,
0x06, 0x43, 0x97, 0x33,
0xA3, 0x6E, 0xB0, 0x2C, 0x79, 0xAC, 0xA7, 0x91, 0x4C, 0xF5, 0x90, 0xB3,
0x8F, 0xEA, 0x14, 0x1B,
0xA8, 0xAB, 0x57, 0x35, 0xC9, 0x1D, 0xE2, 0x8F, 0xCE, 0x87, 0x84, 0x26,
0xB9, 0x59, 0x84, 0x7D,
0xAA, 0x67, 0x92, 0x84, 0x59, 0x69, 0x36, 0xD1, 0x30, 0xCE, 0xCF, 0x28,
0x6B, 0x7C, 0xF3, 0xA2,
0x5B, 0xC3, 0xB5, 0x58, 0x84, 0xAA, 0xD9, 0x45, 0xC3, 0xB8, 0x48, 0x86,
0xD2, 0x2C, 0xC3, 0x3E,
0x37, 0x15, 0xFA, 0x8D, 0x7B, 0xE3, 0x90, 0x52, 0xCF, 0x0D, 0x76, 0x2A,

0x51, 0x79, 0x71, 0xF6,
0x57, 0x18, 0x50, 0x32, 0xB9, 0xAE, 0xCB, 0x90, 0x86, 0x38, 0x5B, 0x20, 0x68, 0x21, 0xC7, 0x98,
0xAE, 0x30, 0x2E, 0x6E, 0x37, 0x5C, 0xB4, 0x84, 0xBC, 0x33, 0x9D, 0x3A, 0x3A, 0xDF, 0xB3, 0x42,
0x3F, 0x60, 0x7D, 0xDB, 0xC2, 0x23, 0xC0, 0xD8, 0x5F, 0x9F, 0x38, 0x1D, 0x83, 0x15, 0x27, 0xAA,
0x5B, 0x63, 0xCD, 0x5C, 0x5E, 0x48, 0x99, 0x8D, 0xB5, 0x48, 0x78, 0xA0, 0x0E, 0xA1, 0xD7, 0xDA,
0x73, 0x32, 0x12, 0x35, 0x67, 0xA2, 0x10, 0x2C, 0x2C, 0x0B, 0x69, 0xB9, 0xFA, 0xD6, 0x0C, 0x5B,
0x1F, 0xB1, 0xFD, 0x6D, 0x69, 0x1B, 0x56, 0xD6, 0x1E, 0x36, 0x88, 0xBB, 0x08, 0x69, 0x9D, 0xB5,
0x53, 0x8B, 0x5B, 0xC1, 0x9C, 0x3B, 0x64, 0xA4, 0x62, 0xBB, 0x5D, 0xD4, 0x54, 0x9C, 0x2D, 0xAE,
0x8A, 0x8D, 0xA0, 0x0A, 0x3B, 0x72, 0xD0, 0x18, 0x3B, 0x45, 0x22, 0xCB, 0x60, 0xC8, 0x49, 0xBB,
0x32, 0x57, 0xE5, 0xF7, 0x6E, 0x5C, 0xB2, 0xA4, 0x78, 0x75, 0x1F, 0x7E, 0x53, 0xD9, 0x8E, 0xFC,
0xF8, 0x24, 0xF5, 0x53, 0x80, 0x1D, 0x08, 0xB0, 0xBC, 0xD6, 0x1B, 0xC6, 0xAC, 0x40, 0x86, 0xC2,
0x09, 0x7C, 0xE4, 0x4E, 0x31, 0xE4, 0x82, 0xAB, 0x93, 0xE8, 0xB0, 0x78, 0x61, 0x50, 0x49, 0x7D,
0x96, 0xAA, 0xCF, 0x8A, 0x17, 0x22, 0x22, 0x21, 0x6C, 0xD1, 0x8C, 0x28, 0xB0, 0x16, 0xCE, 0xDF,
0xD2, 0x3A, 0x85, 0xE8, 0x47, 0xB4, 0x01, 0x93, 0x76, 0x29, 0x6D, 0x7F, 0x5F, 0x9A, 0x11, 0xA2,
0x95, 0xDE, 0x64, 0x52, 0xB6, 0x56, 0x9C, 0x4C, 0x3A, 0xCD, 0x4E, 0xB7, 0xB4, 0x61, 0xD2, 0x6A,
0x99, 0x59, 0x2F, 0x6A, 0x32, 0x46, 0x9C, 0x4D, 0xCA, 0x21, 0xE8, 0xCF, 0xBC, 0x25, 0xF6, 0x35,
0x40, 0x64, 0xC4, 0xED, 0x3E, 0xEA, 0xDA, 0x15, 0xB8, 0x21, 0xC8, 0xF7, 0x4B, 0x5D, 0x36, 0x4D,
0xB3, 0x6B, 0xB7, 0xAC, 0x76, 0xA1, 0x63, 0x0A, 0x76, 0x0D, 0xF0, 0x06, 0x34, 0x76, 0xB0, 0x5D,
0x90, 0x9E, 0x6D, 0x3C, 0x41, 0xA1, 0x43, 0x4B, 0xEC, 0x8D, 0x9A, 0xEC, 0x4F, 0xD1, 0x8C, 0x3C,
0xAE, 0xFE, 0x60, 0x1B, 0x1D, 0x43, 0x1E, 0x09, 0x7F, 0x6A, 0xE6, 0x8C, 0x6A, 0x27, 0x5A, 0x2C,
0x30, 0x82, 0x51, 0x16, 0xCE, 0x5B, 0x92, 0x56, 0xEA, 0x99, 0xF5, 0x89, 0xAB, 0xD2, 0x42, 0xB4,
0xD4, 0x15, 0xE3, 0x6E, 0x68, 0x23, 0x9D, 0xFB, 0x13, 0xCF, 0x0A, 0x75,

0x65, 0xBA, 0x9A, 0x4B,
0xAD, 0xF3, 0xEB, 0x47, 0x26, 0x0B, 0x1C, 0xC2, 0x1D, 0x3B, 0x74, 0x5D,
0x86, 0x68, 0x9D, 0xFA,
0xA0, 0xA6, 0x66, 0xA2, 0x6A, 0x86, 0xDB, 0x2A, 0x3A, 0x53, 0x86, 0xCD,
0xDB, 0x8C, 0xC9, 0x04,
0xA0, 0x26, 0x51, 0xC4, 0x39, 0xC4, 0x08, 0x3C, 0x50, 0x2A, 0x62, 0xB5,
0x9B, 0x5D, 0xE8, 0x2C,
0x9C, 0xEB, 0x1A, 0x83, 0x68, 0xB2, 0x16, 0x54, 0x31, 0x31, 0x9D, 0x3F,
0x1D, 0xA3, 0x5A, 0xF3,
0xA4, 0x79, 0xD2, 0x81, 0xBF, 0x34, 0x0D, 0x7A, 0xB1, 0x73, 0x49, 0xF3,
0xE6, 0x78, 0x5E, 0x26,
0xF9, 0x94, 0xEF, 0x93, 0xE4, 0xA5, 0xB1, 0x52, 0x2C, 0xAA, 0x47, 0x52,
0x7A, 0xC3, 0xA4, 0xD5,
0x28, 0x29, 0x2C, 0x39, 0x2E, 0xBD, 0xB9, 0x23, 0x6A, 0xBC, 0x65, 0x53,
0x88, 0xE7, 0xDE, 0x3F,
0x75, 0x51, 0x55, 0xFF, 0xEF, 0xBD, 0x5D, 0x31, 0xC5, 0x57, 0xED, 0xE9,
0x1B, 0xDB, 0x25, 0x38,
0xB4, 0x6F, 0x34, 0xF3, 0x51, 0xAF, 0xCB, 0x7E, 0x06, 0x24, 0x74, 0x61,
0x51, 0xE5, 0xC3, 0xEA,
0x2A, 0xB7, 0xE7, 0x51, 0xC6, 0x6C, 0x61, 0x83, 0x09, 0x71, 0x9C, 0xBA,
0xE3, 0xAD, 0xCA, 0x3B,
0x91, 0x62, 0x4F, 0x5E, 0xF3, 0xD3, 0x72, 0x97, 0xDF, 0x56, 0xDA, 0x10,
0x32, 0xD7, 0xFF, 0x84,
0xB4, 0x5F, 0x77, 0xC0, 0x15, 0x86, 0xC6, 0x76, 0x85, 0x62, 0x0B, 0x7F,
0xDC, 0x6D, 0xA2, 0x4A,
0xAE, 0x24, 0x3A, 0xC1, 0xC2, 0xC5, 0x5C, 0xB0, 0x22, 0xD4, 0x9A, 0x6D,
0xB1, 0xA8, 0x5A, 0x78,
0x01, 0x11, 0xD7, 0x68, 0x7C, 0xEC, 0x20, 0xD6, 0xC1, 0x6F, 0xB5, 0xE4,
0x2E, 0x5D, 0x98, 0xA8,
0xE4, 0x55, 0x34, 0xE1, 0xA6, 0xFB, 0x72, 0xB6, 0x4B, 0x1A, 0xA2, 0x77,
0xC8, 0xCF, 0xD5, 0x7A,
0xB7, 0x2E, 0x69, 0xF7, 0xD3, 0x91, 0xA1, 0x1F, 0xB4, 0x41, 0x46, 0x8F,
0x92, 0xF6, 0xD4, 0xC7,
0xD7, 0x15, 0x94, 0x39, 0x91, 0xFF, 0xF6, 0xC5, 0x86, 0xE8, 0xF6, 0x6B,
0x7F, 0x5E, 0x00, 0xA4,
0x17, 0x35, 0xBA, 0x41, 0x85, 0xA9, 0xF3, 0xA7, 0xAC, 0xE2, 0x8F, 0xF1,
0x76, 0x9F, 0x69, 0x56,
0x48, 0x37, 0x05, 0x25, 0x54, 0xEF, 0xAA, 0x51, 0xF5, 0xD5, 0x9E, 0x74,
0xF0, 0x84, 0xE6, 0x5C,
0xCD, 0xE0, 0x7D, 0x6A, 0xA7, 0x38, 0xBB, 0xD5, 0x95, 0x7D, 0x82, 0xD2,
0xCC, 0x11, 0xEF, 0xCA,
0xE5, 0x7B, 0x9F, 0x96, 0x33, 0xCB, 0x9E, 0x1B, 0x33, 0xCF, 0x87, 0x24,

0x6A, 0x9F, 0x39, 0xCC,
0x30, 0x66, 0x2E, 0x4B, 0x3E, 0xC0, 0x83, 0x7F, 0xAF, 0xB5, 0x7B, 0xDA, 0x8B, 0x05, 0x05, 0x83,
0x8B, 0x44, 0xCB, 0xDD, 0xD6, 0x5A, 0x2F, 0x59, 0xB9, 0x0B, 0x64, 0x35, 0x17, 0x69, 0x81, 0x2A,
0x8E, 0xCA, 0xA2, 0x0C, 0xB3, 0xBE, 0x47, 0x53, 0xE8, 0xEC, 0x64, 0x8E, 0xA0, 0xED, 0x65, 0xEE,
0x8A, 0x80, 0xA3, 0x0E, 0xBF, 0x2A, 0xEE, 0xAE, 0x6C, 0x1A, 0xB6, 0x7A, 0xCD, 0x92, 0x29, 0x2D,
0xC7, 0x0B, 0x8A, 0xE3, 0x0A, 0x8D, 0xC1, 0x7E, 0x21, 0xD5, 0x4C, 0x24, 0xB7, 0x2E, 0xB5, 0x3B,
0x4F, 0xDC, 0xB9, 0xB5, 0x67, 0x2A, 0x95, 0xEE, 0xC2, 0x98, 0x2A, 0x0E, 0xC7, 0x8C, 0xCD, 0x5B,
0x4D, 0x6D, 0xA6, 0x2D, 0xDC, 0x7F, 0xA3, 0xF8, 0x0A, 0xD6, 0x9B, 0xEC, 0x82, 0x5C, 0xDF, 0xB0,
0xB0, 0x3E, 0x8D, 0xA6, 0x8A, 0x5C, 0xAB, 0xCA, 0x26, 0x60, 0x21, 0x0E, 0x33, 0x62, 0xDB, 0xB8,
0x70, 0x97, 0x93, 0xAD, 0x79, 0x2B, 0x36, 0x0F, 0x4C, 0x7E, 0xDD, 0xA6, 0xD4, 0xAD, 0x04, 0x45,
0xE1, 0x75, 0xFA, 0xD6, 0x6D, 0x47, 0x8C, 0x2C, 0x34, 0x79, 0x7B, 0xC4, 0xE9, 0x56, 0xA4, 0x50,
0x54, 0x6D, 0x70, 0xC7, 0xDB, 0xC4, 0xCC, 0x64, 0x60, 0x07, 0x36, 0x6A, 0x3D, 0x9B, 0x2B, 0x52,
0x0D, 0x4E, 0x95, 0x7B, 0x89, 0x06, 0xA7, 0xC9, 0x6D, 0x4F, 0x03, 0x76, 0x43, 0x91, 0x7A, 0xCB,
0x91, 0xB8, 0xDE, 0x65, 0x58, 0x0E, 0x0A, 0x82, 0xA1, 0xC9, 0x6E, 0x8C, 0x31, 0xD3, 0x77, 0x20,
0x0D, 0x6C, 0xB2, 0x34, 0x88, 0x3D, 0x34, 0x1D, 0x6F, 0xEA, 0x65, 0xCE, 0xF1, 0xF3, 0xE2, 0x0A,
0x04, 0x24, 0xCD, 0xA1, 0x99, 0xBA, 0x3A, 0x63, 0x72, 0xAA, 0xE4, 0x27, 0x73, 0xF4, 0xE0, 0xDE,
0xA3, 0x87, 0x0F, 0x7B, 0xDF, 0x3D, 0x70, 0xC7, 0xC1, 0x42, 0xFE, 0xFD, 0x8B, 0xB8, 0x98, 0x25,
0xEE, 0x88, 0x82, 0x3C, 0x4A, 0x29, 0xE8, 0x19, 0x0C, 0x4E, 0x39, 0xD3, 0x8C, 0x20, 0xA7, 0x20,
0x49, 0x8E, 0x6C, 0xB2, 0xB6, 0xEA, 0xC4, 0x8B, 0x86, 0x04, 0x50, 0x2E, 0xC6, 0xC8, 0xD7, 0x0C,
0xE1, 0xC3, 0x44, 0xE7, 0xC6, 0xFD, 0xD6, 0xE4, 0x35, 0x66, 0xEC, 0x5D, 0x65, 0x35, 0xE0, 0x4A,
0xC9, 0x02, 0x24, 0x47, 0x61, 0x3B, 0x8F, 0x21, 0x90, 0x71, 0x72, 0x76, 0x69, 0x2A, 0x67, 0x4C,
0x2C, 0x9F, 0xB4, 0xBE, 0x72, 0xA5, 0x44, 0x4C, 0x3D, 0xF1, 0xD1, 0x1C,

0x33, 0xF7, 0x97, 0x3F,
0xE6, 0xB3, 0xC9, 0x22, 0x11, 0x53, 0x9A, 0xA3, 0x37, 0x98, 0x67, 0x4E,
0x40, 0x59, 0x6B, 0xD6,
0x35, 0x2E, 0xB2, 0x98, 0xA5, 0xE6, 0x37, 0x23, 0x11, 0xE5, 0xE6, 0x75,
0x1D, 0x71, 0xB7, 0x29,
0x11, 0x88, 0xB3, 0xF3, 0x16, 0xDC, 0xC1, 0x96, 0xC8, 0x09, 0xC1, 0xB4,
0xAD, 0x96, 0x39, 0xFA,
0xF9, 0xF7, 0xEF, 0x9F, 0xD4, 0xDA, 0xCD, 0xEE, 0xF9, 0x55, 0xEB, 0xAC,
0xD7, 0x3D, 0x1E, 0x9C,
0x8A, 0x21, 0x9B, 0xF3, 0x6A, 0x9A, 0xA3, 0x5F, 0x19, 0x2F, 0xA8, 0x2F,
0xCD, 0xAB, 0x56, 0xBB,
0xD9, 0xDC, 0x9E, 0xD7, 0x23, 0x73, 0xF4, 0x96, 0xB3, 0x6A, 0x9F, 0x03,
0xAB, 0x66, 0x7B, 0x07,
0xB1, 0xCE, 0xCD, 0x11, 0xE7, 0x04, 0x4C, 0xAE, 0x1E, 0xF6, 0xCE, 0xB7,
0x67, 0xF4, 0x10, 0x64,
0x7A, 0x07, 0x9C, 0xCE, 0x41, 0xBB, 0xDE, 0x2E, 0xCA, 0xF5, 0xCC, 0x11,
0xE3, 0xD3, 0xEB, 0x36,
0xAF, 0xBA, 0xE7, 0x3B, 0xF0, 0x39, 0x33, 0x65, 0xA7, 0xC3, 0xDC, 0x3F,
0x3A, 0x32, 0x47, 0x17,
0x3F, 0x3C, 0xAF, 0x75, 0x41, 0xC6, 0xF6, 0xA3, 0xDE, 0xF6, 0xBC, 0xBB,
0xE0, 0x17, 0x4C, 0xC8,
0x4E, 0x1B, 0x18, 0x75, 0x77, 0x10, 0xB2, 0x63, 0x8E, 0x5E, 0x70, 0x4E,
0xC0, 0xE5, 0xAA, 0xF5,
0x70, 0x07, 0x91, 0xC0, 0xBD, 0x7E, 0xE6, 0x9C, 0xC0, 0xBF, 0x98, 0x7B,
0x55, 0xE4, 0x04, 0xB9,
0x97, 0x9B, 0xA6, 0x20, 0xE6, 0xD7, 0x33, 0x59, 0xEA, 0x74, 0x51, 0x4A,
0xF8, 0x3B, 0x84, 0x8E,
0x80, 0x5E, 0x6F, 0x9C, 0x10, 0x24, 0x1D, 0xA8, 0x24, 0x0E, 0xAA, 0xE5,
0x02, 0x45, 0x92, 0xF8,
0x6A, 0xAB, 0x39, 0xEA, 0x96, 0x28, 0xC0, 0x49, 0xD5, 0x84, 0xCA, 0x69,
0x53, 0xF2, 0x9B, 0xAC,
0x3F, 0x64, 0xA8, 0xB3, 0xFB, 0x79, 0xC0, 0x43, 0x3B, 0xA6, 0x12, 0xD5,
0x5B, 0x25, 0x1B, 0x8D,
0xAC, 0xE8, 0xCA, 0x1C, 0xF5, 0x3A, 0x65, 0xD6, 0xDE, 0x01, 0x8C, 0x31,
0xEF, 0x3D, 0x5D, 0x1C,
0x04, 0x1B, 0xE3, 0x91, 0x90, 0x9A, 0xA3, 0xA7, 0xF1, 0xF1, 0x2E, 0xA8,
0xD4, 0xCB, 0x34, 0xE5,
0xB4, 0x39, 0xB0, 0x28, 0xE2, 0x08, 0x64, 0xEA, 0x1D, 0x09, 0x4D, 0x82,
0xCC, 0xE7, 0x05, 0xE6,
0x36, 0x71, 0x61, 0xED, 0x80, 0x8F, 0x02, 0xBA, 0x31, 0x2A, 0x11, 0x21,
0x24, 0x35, 0x79, 0x74,
0x30, 0x44, 0x62, 0x51, 0xBE, 0x02, 0x3C, 0x02, 0x44, 0x43, 0x9F, 0xDF,

0xE5, 0xB8, 0x31, 0x22,
0x09, 0x29, 0x54, 0xC3, 0xF8, 0x78, 0x27, 0x54, 0x76, 0x49, 0x5F, 0x8A, 0x38,
0x12, 0x97, 0x28,
0x85, 0x75, 0x6F, 0x09, 0x97, 0x32, 0x69, 0x77, 0xC2, 0x65, 0x86, 0xFC,
0xC5, 0x56, 0xE9, 0x2B,
0xA6, 0x04, 0x54, 0xA2, 0xC3, 0x83, 0x85, 0x4A, 0x22, 0xCC, 0x57, 0x10,
0x2B, 0xB0, 0xFE, 0xF6,
0x48, 0xB0, 0x79, 0xC7, 0x2F, 0xE9, 0xCC, 0xD1, 0x33, 0x5C, 0x7F, 0xCD,
0x8E, 0x76, 0x81, 0xE3,
0x49, 0x48, 0xBD, 0x1D, 0x00, 0x89, 0x64, 0x11, 0x70, 0x34, 0x25, 0x1A,
0xE7, 0xB7, 0x84, 0xC6,
0xF9, 0x2D, 0xA2, 0x81, 0xF0, 0x7B, 0x07, 0x2F, 0xB1, 0xB3, 0x31, 0x1C,
0x11, 0xA1, 0x39, 0xBA,
0xBC, 0x5A, 0x78, 0x01, 0xBB, 0x5B, 0xF8, 0x25, 0xFB, 0xBE, 0x53, 0x90,
0x9C, 0xED, 0x80, 0x49,
0x2C, 0x90, 0x8C, 0x91, 0x33, 0x89, 0xCA, 0xD9, 0x2D, 0xA1, 0x52, 0x26,
0xEB, 0x2E, 0xA8, 0x4C,
0x11, 0x71, 0x2D, 0x4C, 0x1C, 0x76, 0xE7, 0xE2, 0xA6, 0xC0, 0x28, 0xB4,
0xE6, 0xE8, 0xFB, 0xE4,
0xCB, 0x2E, 0xC0, 0x34, 0x77, 0xC0, 0x45, 0x95, 0x27, 0x1D, 0x2F, 0x67,
0xB0, 0x58, 0xBE, 0x25,
0x6C, 0x5A, 0xAD, 0xDB, 0xAC, 0x2A, 0x0B, 0x6C, 0x11, 0xE4, 0xBC, 0xC7,
0x93, 0x09, 0x2C, 0x83,
0x36, 0x2F, 0x2D, 0x29, 0x72, 0xA8, 0x2F, 0xE2, 0xBB, 0x71, 0xC9, 0xBF,
0x6F, 0xBC, 0x87, 0x91,
0x61, 0xF7, 0xB9, 0x36, 0x32, 0x9A, 0xFA, 0xB5, 0xF0, 0x6B, 0x2F, 0x96,
0x73, 0xDB, 0x5D, 0x0D,
0x60, 0x82, 0xA7, 0x7C, 0x53, 0x7D, 0x6B, 0x1E, 0x6D, 0xF0, 0x6C, 0x1F,
0x5D, 0xF3, 0xC7, 0x10,
0x77, 0x59, 0x48, 0xBF, 0xC1, 0xB6, 0xF1, 0x0B, 0x71, 0xB7, 0x57, 0xA6,
0xCB, 0x04, 0xC1, 0xD8,
0xDD, 0x8D, 0xCB, 0x19, 0x2C, 0x91, 0xE0, 0x60, 0x37, 0x26, 0x3D, 0xF0,
0x24, 0xBC, 0x20, 0xE8,
0x4B, 0x58, 0xC4, 0xA3, 0xD5, 0x78, 0xF3, 0x82, 0xB2, 0x1A, 0x43, 0x5D,
0xFE, 0xED, 0xA9, 0x71,
0xC9, 0x6F, 0x03, 0xDB, 0x38, 0x5D, 0x89, 0x2B, 0xD4, 0x55, 0x1C, 0x5D,
0x24, 0x2A, 0x29, 0xA7,
0xB9, 0xB6, 0x27, 0xAA, 0x0F, 0xA0, 0xAA, 0xFB, 0xA2, 0x1A, 0xF5, 0x22,
0x01, 0xF9, 0x05, 0x3D,
0x53, 0xD1, 0xB6, 0x9A, 0x8E, 0xB7, 0xD8, 0x8A, 0x59, 0xAB, 0xCD, 0xDB,
0x30, 0x6B, 0x05, 0x30,
0xD9, 0x4B, 0x76, 0x87, 0xA0, 0x6D, 0x00, 0x5E, 0x7B, 0x01, 0x8A, 0xCD,

0x7A, 0x18, 0xA0, 0xB8,
0xBE, 0x87, 0x06, 0x0A, 0xBC, 0xE5, 0x3D, 0xAB, 0xA3, 0xDB, 0x04, 0x15, 0x27, 0x34, 0x47, 0xAF,
0x90, 0x1B, 0x42, 0x91, 0xD9, 0x17, 0x60, 0xF1, 0xC4, 0x07, 0x0B, 0x2F, 0xA9, 0xF7, 0xA1, 0xA1,
0x03, 0x41, 0xE6, 0x9E, 0xBD, 0xF9, 0x72, 0x47, 0xD2, 0x89, 0x94, 0xF8, 0x0A, 0x8E, 0x36, 0x6E,
0x0C, 0x22, 0x0E, 0xB7, 0xDC, 0x11, 0x88, 0xA5, 0xD4, 0xF6, 0xCD, 0xC0, 0xDB, 0xD0, 0x75, 0xAF,
0x77, 0xE9, 0x04, 0x2E, 0x1C, 0x2F, 0xB4, 0xB7, 0xE7, 0x00, 0x6D, 0xC0, 0x8F, 0x93, 0x09, 0xB1,
0xB6, 0x6F, 0x24, 0xA0, 0x09, 0x78, 0xE1, 0xCD, 0x2B, 0xD2, 0xDF, 0x72, 0xE1, 0xC5, 0xD6, 0x16,
0x2B, 0x39, 0x0B, 0x50, 0xBC, 0xBC, 0xD8, 0x6B, 0xE1, 0x85, 0x39, 0x0F, 0x94, 0x19, 0x98, 0xB6,
0x87, 0x4E, 0x0A, 0x20, 0xC4, 0x7B, 0xEE, 0x3C, 0xDB, 0x80, 0x25, 0x28, 0xE3, 0x8C, 0x1E, 0x2D,
0xBF, 0x0F, 0xB5, 0xBE, 0x4B, 0x24, 0x4A, 0xAF, 0xEE, 0x5A, 0x67, 0x9D, 0x5E, 0xBC, 0xBC, 0xEB,
0xB4, 0x3F, 0xEF, 0x02, 0x8F, 0x31, 0xBF, 0x5D, 0x7C, 0xDA, 0xDB, 0x40, 0x03, 0xD9, 0xE8, 0x35,
0xBB, 0xCE, 0xB0, 0x41, 0xC2, 0xDE, 0x3D, 0x90, 0xDA, 0x87, 0x8B, 0xA4, 0xF6, 0x17, 0x10, 0x4A,
0xD3, 0x2D, 0x32, 0xDE, 0x94, 0x65, 0xBC, 0xEF, 0x2F, 0xF6, 0x83, 0xD0, 0xF4, 0x60, 0xA9, 0x6E,
0x7A, 0xD0, 0x54, 0x67, 0x88, 0x9B, 0xAD, 0x62, 0x98, 0xB6, 0xEC, 0x60, 0x25, 0xA1, 0xD8, 0xCB,
0xDA, 0x25, 0xC9, 0xB5, 0xAE, 0x76, 0xC9, 0x72, 0x91, 0x18, 0xE9, 0x24, 0xD7, 0x4B, 0xAE, 0x8A,
0x9C, 0x7D, 0xDE, 0xCB, 0xBA, 0xDD, 0x32, 0x69, 0x77, 0x09, 0x1A, 0x1F, 0xAD, 0xDE, 0x4F, 0xE7,
0x68, 0x63, 0x30, 0x24, 0x1D, 0x60, 0xF1, 0xEA, 0xC9, 0x3E, 0xDB, 0x85, 0x68, 0xDE, 0xC3, 0xC4,
0x51, 0xAC, 0xF5, 0xA1, 0x73, 0x9D, 0x83, 0xDD, 0xCD, 0x93, 0x1D, 0x23, 0x32, 0x47, 0x2F, 0xB1,
0x1B, 0x18, 0x17, 0x9E, 0x2F, 0xDF, 0xFD, 0xB4, 0x17, 0xD4, 0xF8, 0xCC, 0x87, 0x81, 0x4C, 0x28,
0x7D, 0x68, 0xBC, 0x66, 0x73, 0xE2, 0xFB, 0x9E, 0xBF, 0x31, 0x64, 0x92, 0x0E, 0x96, 0x15, 0xF5,
0x57, 0xFC, 0x68, 0x2F, 0x70, 0x45, 0xB3, 0x1E, 0x06, 0xB1, 0x58, 0xE7, 0x43, 0x83, 0xB6, 0x9C,
0x38, 0x64, 0xB1, 0x31, 0x64, 0x9C, 0xCA, 0x1C, 0xBD, 0xAB, 0x3F, 0x87,

0x7F, 0xF7, 0x02, 0x97,
0x98, 0xF1, 0x30, 0x60, 0x49, 0x6D, 0x0F, 0x0D, 0xD5, 0x78, 0xB1, 0x79,
0x3A, 0x04, 0x1A, 0x73,
0xF4, 0xF4, 0xA7, 0xFD, 0xF4, 0x7E, 0x6C, 0xB2, 0x8A, 0x08, 0xED, 0x84,
0x07, 0x57, 0xEA, 0xD0,
0x68, 0xAC, 0xB6, 0x40, 0x63, 0xC5, 0x04, 0xFF, 0x6D, 0x4F, 0x68, 0xAC,
0xAA, 0xA3, 0xF1, 0x99,
0xE3, 0x65, 0xF5, 0x25, 0xE0, 0xC3, 0x9F, 0xC5, 0x18, 0xA3, 0xCD, 0xCB,
0x51, 0x44, 0xC8, 0x6E,
0x1A, 0x83, 0x23, 0xE3, 0x29, 0xDA, 0x4F, 0x41, 0x8A, 0xE7, 0xDD, 0x47,
0x08, 0x25, 0x4A, 0x1E,
0x1A, 0xA7, 0x09, 0xB2, 0xF0, 0x7B, 0x1B, 0xD3, 0x6D, 0xAE, 0x2D, 0x2B,
0xB4, 0xE6, 0xE8, 0x39,
0x7C, 0x31, 0x9E, 0xF1, 0x2F, 0xFB, 0x6A, 0xF9, 0xD4, 0xF9, 0xF7, 0x81,
0x5A, 0x4A, 0xDF, 0x2F,
0x02, 0x38, 0x68, 0xB0, 0xBD, 0xA9, 0xBB, 0xD5, 0x23, 0x0D, 0x29, 0x72,
0x09, 0xDF, 0x1B, 0xF1,
0x7D, 0xBF, 0x00, 0x26, 0x42, 0xEC, 0x0D, 0x43, 0x45, 0xEF, 0x7D, 0xC0,
0x18, 0x3D, 0x16, 0xC4,
0x8B, 0xB4, 0x78, 0x15, 0x5E, 0x19, 0x52, 0xF2, 0xE1, 0x27, 0x7E, 0x4B,
0x0B, 0xA6, 0xF5, 0x80,
0x12, 0xC7, 0x81, 0x85, 0x30, 0xA6, 0xC6, 0x5B, 0x76, 0x38, 0x38, 0x15,
0x03, 0xAA, 0x73, 0x91,
0xCF, 0xDC, 0xB0, 0x97, 0x50, 0xA2, 0xB9, 0x39, 0x7A, 0xCB, 0x5E, 0x12,
0x08, 0xBC, 0xD8, 0xB7,
0xCD, 0x99, 0x71, 0x23, 0x62, 0xD7, 0xF7, 0x40, 0xA8, 0x18, 0x24, 0xF9,
0xAE, 0x26, 0xD3, 0x88,
0x8E, 0x94, 0xDF, 0x46, 0x97, 0x7C, 0xB0, 0xC1, 0xBC, 0xAC, 0x7C, 0x3A,
0x76, 0xD5, 0xC2, 0xCA,
0xBF, 0xB8, 0x31, 0x38, 0x75, 0x91, 0xC6, 0xDC, 0x39, 0x28, 0x0C, 0xC4,
0xDB, 0x25, 0x73, 0x58,
0xC5, 0xCF, 0x33, 0x71, 0x4B, 0x24, 0x8F, 0x69, 0xC6, 0x6A, 0x65, 0x1F,
0xDF, 0x94, 0xDB, 0x4C,
0xD5, 0x82, 0x96, 0x3F, 0x88, 0x29, 0xEB, 0x21, 0x3B, 0x8C, 0xCD, 0xFF,
0x9F, 0x7F, 0x97, 0xF9,
0x0C, 0x7B, 0xF7, 0x67, 0x22, 0x98, 0x69, 0x04, 0xBE, 0x35, 0x34, 0xF3,
0x9E, 0x8E, 0xCA, 0xD1,
0xFC, 0x54, 0xA7, 0x7A, 0x66, 0xB0, 0xC6, 0xD6, 0x83, 0xC0, 0xF2, 0xC9,
0x82, 0x8E, 0xEE, 0xD8,
0x9E, 0x15, 0xCE, 0xB1, 0x4B, 0x1B, 0xC8, 0xB6, 0x2F, 0x97, 0x70, 0xF0,
0x92, 0x04, 0x14, 0x83,
0x15, 0x6A, 0x47, 0xCF, 0x7E, 0x7C, 0x75, 0x21, 0x9E, 0x12, 0x7B, 0xE9,

0x21, 0x1B, 0xDB, 0x47,
0x27, 0xC6, 0x24, 0x74, 0x85, 0x9B, 0xD7, 0x30, 0x1B, 0x2B, 0xDE, 0xBB, 0xBA, 0x44, 0xBE, 0x31,
0x46, 0x01, 0x7E, 0xE1, 0x05, 0xD4, 0x18, 0x1A, 0x31, 0x47, 0xC7, 0xB3, 0xF8, 0x7D, 0xBF, 0x0D,
0xCF, 0x27, 0x53, 0xE2, 0xCA, 0x91, 0x42, 0xD9, 0x5F, 0x7D, 0x07, 0x86, 0xC6, 0x54, 0xDF, 0x1A,
0x47, 0xFD, 0xF3, 0xD6, 0x11, 0x7B, 0x1C, 0x0F, 0x60, 0x80, 0x1F, 0x00, 0x02, 0x0C, 0x03, 0x20,
0xC0, 0x87, 0x23, 0xF9, 0x78, 0x20, 0x76, 0x1A, 0xDC, 0xE4, 0x4C, 0x40, 0x26, 0x6D, 0xED, 0x48,
0xE0, 0x74, 0xC4, 0x1E, 0x34, 0xBE, 0x89, 0x29, 0x83, 0x99, 0xB7, 0x2A, 0xA2, 0xF4, 0xF1, 0xDC,
0x5B, 0xE2, 0x0C, 0x71, 0x4C, 0x2D, 0xBD, 0xB9, 0x74, 0xEA, 0xC8, 0xEB, 0x8F, 0x8E, 0xA3, 0x01,
0xF1, 0x7B, 0xCC, 0x86, 0x06, 0xF5, 0x43, 0x9C, 0x66, 0x8B, 0xDD, 0x32, 0xAE, 0x91, 0x58, 0x85,
0x8C, 0x27, 0xC8, 0x09, 0x32, 0x9C, 0xC3, 0x85, 0x8D, 0x28, 0x7E, 0xC7, 0x76, 0x0C, 0x61, 0x40,
0x0D, 0x3B, 0x27, 0x62, 0xFB, 0xF0, 0x44, 0x9E, 0x79, 0x03, 0x7C, 0x29, 0x3E, 0x4E, 0x66, 0x55,
0x7F, 0x06, 0x8A, 0xF4, 0xD7, 0xA1, 0xE1, 0x86, 0x10, 0xC2, 0x8F, 0xB9, 0x0A, 0x46, 0x3F, 0x75,
0x96, 0x53, 0x3B, 0x90, 0x9D, 0xE4, 0x3B, 0xDB, 0xF9, 0x9C, 0xFC, 0x47, 0x32, 0x61, 0x13, 0x37,
0xF8, 0x1B, 0xE4, 0x87, 0xC0, 0xE3, 0x28, 0xCA, 0xEE, 0x47, 0xC9, 0x8B, 0x79, 0x55, 0x22, 0x6E,
0x87, 0x86, 0xEC, 0x83, 0xE5, 0xF9, 0xA5, 0x3C, 0x71, 0xF7, 0xEE, 0x32, 0xE6, 0x6B, 0x28, 0xC3,
0xE0, 0x54, 0x72, 0xE2, 0x06, 0x4E, 0x28, 0x4F, 0x3F, 0xAF, 0xF3, 0xCE, 0xF0, 0x88, 0x98, 0x2B,
0x1C, 0xEE, 0xC4, 0x92, 0xA7, 0x2C, 0xF0, 0xE0, 0x41, 0x9A, 0xDB, 0xDD, 0xA1, 0xA4, 0x4A, 0x34,
0x11, 0xE3, 0x21, 0x32, 0x20, 0xF2, 0x40, 0x6D, 0xF9, 0x4C, 0xBC, 0x14, 0x89, 0x4C, 0x6A, 0x77,
0x53, 0x86, 0x8F, 0x65, 0x9C, 0x30, 0x13, 0x11, 0x9B, 0x1B, 0x88, 0x5F, 0x33, 0x3C, 0x4E, 0x9E,
0x7A, 0x15, 0xF2, 0x3D, 0xE6, 0x5E, 0x5F, 0xC3, 0xF2, 0xF2, 0xDB, 0x31, 0xD8, 0x9F, 0x39, 0x73,
0xF2, 0x83, 0x1C, 0x9F, 0x4C, 0xA5, 0x72, 0x9C, 0xA6, 0x38, 0x32, 0xC5, 0x32, 0x72, 0xB3, 0x0F,
0x9F, 0x00, 0x86, 0xB2, 0x9D, 0xEF, 0xE4, 0xF9, 0xFC, 0x8C, 0x39, 0xD9,

0x87, 0x4F, 0xBC, 0x3E,
0xB0, 0x50, 0x82, 0xE8, 0x0E, 0x09, 0x8D, 0x62, 0x9C, 0xDD, 0x6A, 0xCC, 0x54, 0xE2, 0x22, 0xC0,
0x61, 0x11, 0xAB, 0x4C, 0x01, 0xD7, 0x30, 0x14, 0x01, 0x55, 0x13, 0xF5, 0xE9, 0x29, 0xAF, 0x35,
0x8C, 0xB9, 0x8C, 0x95, 0xF4, 0xEF, 0x77, 0x54, 0xE1, 0x6F, 0xA2, 0xF0, 0x89, 0x53, 0x99, 0x8A,
0x27, 0xF3, 0xE3, 0xC8, 0x62, 0xCC, 0xD5, 0x13, 0x87, 0x91, 0xAF, 0x2B, 0x89, 0xFC, 0x3C, 0x31,
0xAB, 0x05, 0x39, 0x4C, 0xF1, 0xF8, 0x7E, 0x46, 0x54, 0xD5, 0xD5, 0x41, 0xEE, 0x96, 0xA1, 0xBE,
0x80, 0x64, 0x0C, 0xA9, 0xF0, 0x63, 0x8A, 0x0F, 0xDF, 0xB0, 0x8F, 0x99, 0x88, 0xDF, 0xC4, 0xE5,
0xFD, 0xBA, 0xE7, 0x62, 0x3D, 0x77, 0xD5, 0xD9, 0x75, 0x3C, 0x45, 0x29, 0xCE, 0x32, 0x0D, 0xC7,
0x73, 0x42, 0x35, 0x0C, 0x8F, 0x20, 0x0D, 0xEB, 0x78, 0xC9, 0x06, 0x2D, 0x21, 0xF0, 0x31, 0x0D,
0x7D, 0x57, 0x8D, 0x26, 0x91, 0x91, 0xFE, 0x0E, 0xB1, 0x7F, 0x0D, 0x8C, 0x3E, 0xDC, 0xFF, 0x14,
0xE5, 0xF7, 0x9B, 0x53, 0xFE, 0x6C, 0x8E, 0xE7, 0x3C, 0x86, 0x0A, 0x30, 0xBC, 0xFF, 0x89, 0x43,
0x7D, 0xF3, 0x00, 0xA6, 0x84, 0x2F, 0x7C, 0xE2, 0x9B, 0x0F, 0x82, 0xC5, 0x84, 0xBD, 0x3E, 0xBB,
0xC6, 0x59, 0x44, 0xB8, 0x35, 0xE8, 0x0C, 0xBB, 0x35, 0x1F, 0x07, 0x0B, 0x60, 0x8F, 0x93, 0x44,
0x16, 0xCD, 0xE8, 0x39, 0x18, 0x4A, 0xCD, 0xB4, 0xF6, 0xC1, 0xC7, 0x40, 0x07, 0x02, 0x50, 0xCF,
0xB8, 0xFF, 0x89, 0xB3, 0xB8, 0x31, 0x26, 0x10, 0xCD, 0xC1, 0x0C, 0xDB, 0x27, 0x50, 0x77, 0x10,
0x65, 0x4F, 0xA6, 0xDF, 0xFF, 0x14, 0xB1, 0x6A, 0x88, 0x9F, 0x6E, 0x3E, 0xC4, 0x1E, 0x12, 0x17,
0x83, 0xA8, 0x86, 0xF1, 0x13, 0x0D, 0xCE, 0xEB, 0x2D, 0x47, 0xC1, 0xF3, 0x9F, 0x38, 0x4E, 0xED,
0x48, 0xBC, 0x7E, 0x41, 0xE6, 0xE8, 0x06, 0x34, 0x9D, 0x97, 0x08, 0xC4, 0x56, 0x93, 0x3B, 0xCF,
0x3B, 0x9E, 0x6B, 0x39, 0xC4, 0xFA, 0xC8, 0x12, 0xF3, 0x71, 0x5A, 0x70, 0x11, 0xE9, 0x4E, 0x43,
0xBC, 0x4E, 0xEB, 0xB5, 0x67, 0xE3, 0x8C, 0x9B, 0x1E, 0x33, 0x31, 0x4E, 0x4F, 0xC1, 0xCA, 0xC8,
0x8E, 0x52, 0x92, 0xC0, 0x88, 0xBD, 0x77, 0x45, 0x98, 0x29, 0x65, 0x61, 0xA1, 0x8C, 0xD4, 0x45,
0xD8, 0x2C, 0xA9, 0xD6, 0x91, 0xCA, 0x89, 0xDB, 0x0A, 0xF4, 0x8C, 0xD8,

0x16, 0x7F, 0x05, 0x9E,
0x5B, 0x3B, 0xBE, 0x13, 0x9B, 0x61, 0x9D, 0x07, 0x9B, 0x40, 0x61, 0x90,
0x32, 0x51, 0x9E, 0x99,
0xD2, 0x5D, 0xFD, 0x51, 0x92, 0x49, 0x72, 0x6C, 0x26, 0x3E, 0x4A, 0x4D,
0xE3, 0x05, 0x8D, 0xCF,
0xFC, 0x07, 0x77, 0x9A, 0x3F, 0x4F, 0x44, 0x11, 0x54, 0x72, 0xD2, 0xB1,
0x62, 0x30, 0xE1, 0x81,
0xEC, 0xBF, 0x1E, 0x51, 0x1B, 0x11, 0xE8, 0xAE, 0x2F, 0x1D, 0xCC, 0x0E,
0x9F, 0x5E, 0xFF, 0x00,
0xC5, 0x5B, 0xB4, 0x20, 0x5C, 0x9A, 0x84, 0xE0, 0x22, 0x6E, 0xFF, 0x4A,
0x29, 0x93, 0x56, 0x51,
0xE1, 0xC1, 0xDB, 0x77, 0x91, 0x71, 0x8A, 0x38, 0xC4, 0x9D, 0x7E, 0x8A,
0x94, 0x71, 0x2D, 0xA7,
0x4D, 0xF5, 0xF7, 0x0A, 0xBD, 0x9A, 0xED, 0x8A, 0xE8, 0x95, 0x96, 0x5E,
0xA1, 0xE6, 0xAE, 0x5C,
0x4E, 0xAC, 0x36, 0xB7, 0x47, 0x8A, 0xB1, 0x03, 0xEA, 0x2D, 0xC4, 0x1A,
0x23, 0xE3, 0xE6, 0x2B,
0xE2, 0xDA, 0xDE, 0xAA, 0xC1, 0xCE, 0xD7, 0x64, 0x91, 0x54, 0x15, 0x6D,
0x10, 0x17, 0x0C, 0xF8,
0xE2, 0x97, 0x57, 0x2F, 0x59, 0xD2, 0x51, 0xD7, 0x2A, 0x47, 0xE9, 0x0E,
0x87, 0xBF, 0xEB, 0x5C,
0x3B, 0x03, 0x83, 0xAD, 0x01, 0x4D, 0xB3, 0x48, 0x36, 0x71, 0x63, 0xC9,
0x62, 0x81, 0x1D, 0x7E,
0x10, 0x73, 0xB2, 0xD2, 0x93, 0x02, 0xF8, 0xB8, 0x54, 0x16, 0x6F, 0x91,
0x15, 0x05, 0x22, 0xF1,
0x09, 0xA5, 0xE0, 0xB0, 0x86, 0x70, 0xE5, 0x80, 0x65, 0x19, 0xB9, 0xCE,
0xBB, 0x63, 0xA8, 0xE0,
0xE7, 0x04, 0x7D, 0x62, 0x26, 0x19, 0x65, 0x69, 0xE1, 0x95, 0x4C, 0x89,
0x16, 0x10, 0x99, 0xF8,
0xF1, 0x7B, 0x6B, 0x0C, 0xC9, 0xF1, 0x19, 0x78, 0x7E, 0xC3, 0x05, 0x0D,
0x8E, 0x6F, 0x8A, 0xD4,
0x11, 0xE6, 0x4A, 0x80, 0xAC, 0x2A, 0x04, 0x4F, 0x43, 0x7A, 0x6E, 0x29,
0xFB, 0xE8, 0xD9, 0xA9,
0xDE, 0x2B, 0xAE, 0xDD, 0xB2, 0x36, 0x2D, 0xCF, 0xB0, 0xC3, 0x75, 0xD3,
0x8A, 0x3E, 0x25, 0xC5,
0x20, 0x49, 0x30, 0x6B, 0xC2, 0x66, 0xDA, 0x14, 0xC5, 0x2F, 0xA2, 0x01,
0x91, 0xEC, 0x6A, 0x40,
0xE4, 0xC8, 0x9E, 0xEE, 0xE2, 0x32, 0xED, 0x42, 0x06, 0x72, 0x99, 0xC5,
0x0C, 0xF6, 0xD6, 0x8F,
0x19, 0x2B, 0xD0, 0xD2, 0x09, 0xAA, 0x14, 0x0A, 0x6D, 0x06, 0x2C, 0xAC,
0x18, 0x62, 0x86, 0x48,
0xDA, 0x6C, 0xB7, 0x99, 0xAE, 0x0E, 0x17, 0x21, 0x58, 0x69, 0x1E, 0xF9,

0xA4, 0xF8, 0x8D, 0xB5,
0x6C, 0x71, 0xF0, 0x40, 0x0B, 0x57, 0x14, 0xD4, 0x70, 0x5A, 0xC9, 0x04,
0xB2, 0xDF, 0x2B, 0x21,
0x50, 0xEE, 0xBA, 0xE0, 0xB4, 0xF0, 0xD3, 0xBA, 0xD8, 0x1A, 0x23, 0xC3,
0xB8, 0xE3, 0x18, 0x73,
0x46, 0x24, 0xBB, 0xA2, 0x04, 0xF1, 0xF5, 0xEE, 0x34, 0x0B, 0xF9, 0x5A,
0x57, 0x7A, 0xA3, 0xA0,
0x15, 0xDD, 0xB7, 0x96, 0xE8, 0x83, 0x8B, 0x95, 0xC7, 0xAA, 0xF2, 0x51,
0x97, 0x5D, 0x42, 0xA1,
0xDE, 0x65, 0x27, 0xD4, 0xC7, 0x15, 0xD5, 0xC7, 0x52, 0x7D, 0x46, 0x90,
0x34, 0x84, 0xE5, 0x2D,
0x7F, 0xEC, 0x8C, 0xBF, 0x3D, 0x4D, 0x34, 0x5B, 0x8D, 0x0B, 0xE5, 0x94,
0xAD, 0xB8, 0xA2, 0x5E,
0x31, 0x41, 0xEA, 0x9E, 0x62, 0xA1, 0xD6, 0x6A, 0x5C, 0x4D, 0xAD, 0xA8,
0x95, 0x67, 0x04, 0x89,
0x5A, 0xFA, 0x86, 0x3F, 0x52, 0x25, 0xDE, 0x42, 0xE6, 0xFF, 0xA7, 0x4B,
0xFC, 0xCE, 0x94, 0x58,
0x58, 0xB1, 0xFF, 0x5A, 0x5A, 0xCA, 0xC4, 0x30, 0x45, 0xC9, 0x78, 0xC9,
0x50, 0x4A, 0x1A, 0x8F,
0x54, 0xA8, 0x63, 0x39, 0x0A, 0xA9, 0xA3, 0x41, 0xA2, 0x06, 0xC6, 0x5F,
0x2B, 0x19, 0x2B, 0x1E,
0x9D, 0x04, 0x42, 0xC2, 0x40, 0x34, 0xE0, 0x23, 0xE3, 0x2C, 0xBB, 0xD4,
0x14, 0x8D, 0x90, 0x50,
0x36, 0xD3, 0xFE, 0xA8, 0x03, 0x62, 0x95, 0x52, 0x63, 0xE2, 0x00, 0x11,
0xF4, 0x79, 0x62, 0x96,
0x8A, 0x82, 0x1C, 0xEC, 0xD3, 0x9A, 0xF9, 0x93, 0x83, 0xD9, 0xF2, 0x41,
0xDE, 0x14, 0x7E, 0xF1,
0xC3, 0x73, 0xC3, 0xF3, 0x0D, 0xF1, 0x16, 0x4D, 0x3F, 0x7E, 0x6B, 0x8E,
0x21, 0x5F, 0x31, 0xC7,
0x17, 0x69, 0xC4, 0x9D, 0x1A, 0x74, 0x46, 0x02, 0xE8, 0x59, 0xD9, 0x93,
0xE0, 0xF8, 0xAE, 0x19,
0xBF, 0x45, 0xAE, 0x54, 0x3D, 0xD1, 0xA4, 0x7E, 0x17, 0x2B, 0x92, 0x31,
0xA7, 0xA0, 0x49, 0x6C,
0x79, 0x57, 0xEA, 0xB8, 0x96, 0x58, 0x8A, 0x96, 0x85, 0x1B, 0x98, 0x30,
0x3E, 0xFD, 0xC5, 0x5A,
0x51, 0xAF, 0x40, 0xA9, 0x21, 0x63, 0xB2, 0xC4, 0x96, 0x89, 0xAE, 0x6B,
0xD6, 0xD4, 0xAD, 0xBD,
0x0B, 0x10, 0x65, 0x5B, 0x49, 0xDA, 0x6C, 0x9E, 0x8F, 0x8A, 0xB0, 0xB8,
0xA8, 0x72, 0xE2, 0x33,
0x38, 0x8D, 0x36, 0x2C, 0xC5, 0x37, 0xF1, 0x52, 0xAE, 0xC1, 0xA9, 0xF8,
0x9F, 0x0A, 0xFF, 0x0B,
0x9B, 0xFC, 0x8E, 0x51, 0xC1, 0x70, 0x00, 0x00

```
};
```

Camera_pins.h

```
#if defined(CAMERA_MODEL_WROVER_KIT)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM    21
#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      19
#define Y4_GPIO_NUM      18
#define Y3_GPIO_NUM       5
#define Y2_GPIO_NUM       4
#define VSYNC_GPIO_NUM   25
#define HREF_GPIO_NUM    23
#define PCLK_GPIO_NUM    22
#elif defined(CAMERA_MODEL_ESP_EYE)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM    4
#define SIOD_GPIO_NUM    18
#define SIOC_GPIO_NUM    23

#define Y9_GPIO_NUM      36
#define Y8_GPIO_NUM      37
#define Y7_GPIO_NUM      38
#define Y6_GPIO_NUM      39
#define Y5_GPIO_NUM      35
#define Y4_GPIO_NUM      14
#define Y3_GPIO_NUM      13
#define Y2_GPIO_NUM      34
#define VSYNC_GPIO_NUM   5
#define HREF_GPIO_NUM    27
#define PCLK_GPIO_NUM    25
```

```
#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
#define PWDN_GPIO_NUM     -1 #define
RESET_GPIO_NUM    15
#define XCLK_GPIO_NUM     27
#define SIOD_GPIO_NUM     25
#define SIOC_GPIO_NUM     23

#define Y9_GPIO_NUM       19
#define Y8_GPIO_NUM       36
#define Y7_GPIO_NUM       18
#define Y6_GPIO_NUM       39
#define Y5_GPIO_NUM        5
#define Y4_GPIO_NUM       34
#define Y3_GPIO_NUM       35
#define Y2_GPIO_NUM       32
#define VSYNC_GPIO_NUM    22
#define HREF_GPIO_NUM     26
#define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_M5STACK_WIDE)
#define PWDN_GPIO_NUM     -1 #define
RESET_GPIO_NUM    15
#define XCLK_GPIO_NUM     27
#define SIOD_GPIO_NUM     22
#define SIOC_GPIO_NUM     23

#define Y9_GPIO_NUM       19
#define Y8_GPIO_NUM       36
#define Y7_GPIO_NUM       18
#define Y6_GPIO_NUM       39
#define Y5_GPIO_NUM        5
#define Y4_GPIO_NUM       34
#define Y3_GPIO_NUM       35
#define Y2_GPIO_NUM       32
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     26
#define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_AI_THINKER)
#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

#else
#error "Camera model not selected"
#endif
```

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA JURUSAN TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

REKOMENDASI UJIAN TUGAS AKHIR

Pembimbing Laporan Tugas Akhir, memberikan rekomendasi ujian laporan tugas akhir kepada,

Nama Mahasiswa	: Novri Rahmadhan
NIM	: 062130701724
Jurusan/Program Studi	: Teknik Komputer/D3 Teknik Komputer
Judul Tugas Akhir	: Rancang Bangun *Smart Doorbel* Pemantau Tamu Rumah Menggunakan ESP32CAM Berbasis *Internet of Things*

Mahasiswa tersebut telah memenuhi persyaratan dan dapat mengikuti Ujian Laporan Tugas Akhir, pada Tahun Akademik 2023/ 2024

Palembang, 12 Juli 2024

Disetujui oleh,

Pembimbing I	Pembimbing II
Indarto, S.T.,M.Cs.	**Mustaziri, S.T.,M.Kom**
NIP 197307062005011003	NIP 196909282005011002

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI

POLITEKNIK NEGERI SRIWIJAYA

JURUSAN TEKNIK KOMPUTER

PELAKSANAAN REVISI UJIAN TUGAS AKHIR

Nama Mahasiswa : Novri Rahmadhan

NIM : 062130701724

Jurusan/Program Studi : Teknik Komputer/D3 Teknik Komputer

Judul Tugas Akhir : Rancang Bangun *Smart Doorbell* Pemantau Tamu Rumah Menggunakan ESP32CAM Berbasis *Internet of Things*

Telah melaksanakan revisi terhadap Laporan Tugas Akhir yang diujikan pada hari Senin, tanggal 15 Juli tahun 2024. Pelaksanaan revisi terhadap Laporan Tugas Akhir tersebut telah disetujui oleh Dosen Penguji yang memberikan revisi:

No	Komentar	Nama Dosen Penguji	Tanggal	Tanda Tangan
1.	ACC	Slamet Widodo, M.Kom.	1/8/24	
2.	OK	Adi Sutrisman, S.Kom., M.Kom.	31/7-24	
3.	OK	Isnainy Azro, M.Kom.	31/7.2024	
4.	Ok.	Arsia Rini, S.Kom., M.Kom.	1/8 24	

Palembang,

Ketua Penguji

Slamet Widodo, M.Kom.

NIP 197305162002121001

No. Dok. :	Tgl. Berlaku :	No. Rev. :
	KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI POLITEKNIK NEGERI SRIWIJAYA Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414 Website : www.polsri.ac.id E-mail : info@polsri.ac.id	
	PENILAIAN UJIAN TUGAS AKHIR (TA)	

Dosen Penguji : Slamet Widodo, M.Kom.

Nama Mahasiswa : Novri Rahmadhan

NIM : 062130701724

Jurusan /Program Studi : Teknik Komputer/D3 Teknik Komputer

Judul LA/ Skripsi : Rancang Bangun *Smart Doorbell* Pemantau Tamu Rumah Menggunakan ESP32CAM Berbasis *Internet of Things*

No	Uraian Revisi	Paraf
	Saran Buat Buka Logger Revisi Button pakai Sirene	

Palembang,
Dosen Penguji,

Slamet Widodo, M.Kom.
NIP 197305162002121001

No. Dok. :	Tgl. Berlaku :	No. Rev. :
	KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI POLITEKNIK NEGERI SRIWIJAYA Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414 **Website :** www.polsri.ac.id **E-mail :** info@polsri.ac.id	
	PENILAIAN UJIAN TUGAS AKHIR (TA)	

Dosen Penguji : Isnainy azro, M.Kom

Nama Mahasiswa : Novri Rahmadhan

NIM : 062130701724

Jurusan /Program Studi : Teknik Komputer/D3 Teknik Komputer

Judul LA/ Skripsi : Rancang Bangun *Smart Doorbell* Pemantau Tamu Rumah Menggunakan ESP32CAM Berbasis *Internet of Things*

No	Uraian Revisi	Paraf
1.	Perbaikan Tujuan dan manfaat	
2.	Buat flowchart program nya / Flowchart diperbaiki	
3.	Tampilkan coding program di bab III bila kurang dari 5 lembar bila lebih dari 5 lembar letakkan dilampiran	
4	Abstrak diperbaiki	

Palembang,
Dosen Penguji,

Isnainy azro, M.Kom
NIP 197310012002122007

No. Dok. :	Tgl. Berlaku :	No. Rev. :
	KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI POLITEKNIK NEGERI SRIWIJAYA Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414 Website : www.polsri.ac.id E-mail : info@polsri.ac.id	
	PENILAIAN UJIAN TUGAS AKHIR (TA)	

Dosen Penguji : Adi Sutrisman, S.Kom., M.Kom.

Nama Mahasiswa : Novri Rahmadhan

NIM : 062130701724

Jurusan /Program Studi : Teknik Komputer/D3 Teknik Komputer

Judul LA/ Skripsi : Rancang Bangun *Smart Doorbell* Pemantau Tamu Rumah Menggunakan ESP32CAM Berbasis *Internet of Things*

No	Uraian Revisi	Paraf
–	Abstrak → 1 Spasi	
–	Tambahkan Lampiran (Kode program)	

Palembang,
Dosen Penguji,

Adi Sutrisman, S.Kom., M.Kom.
NIP 197503052001121005

No. Dok. :	Tgl. Berlaku :	No. Rev. :
	KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI POLITEKNIK NEGERI SRIWIJAYA Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414 **Website :** www.polsri.ac.id **E-mail :** info@polsri.ac.id	
	PENILAIAN UJIAN TUGAS AKHIR (TA)	

Dosen Penguji : Arsia Rini, S.Kom., M.Kom.

Nama Mahasiswa : Novri Rahmadhan

NIM : 062130701724

Jurusan /Program Studi : Teknik Komputer/D3 Teknik Komputer

Judul LA/ Skripsi : Rancang Bangun *Smart Doorbell* Pemantau Tamu Rumah Menggunakan ESP32CAM Berbasis *Internet of Things*

No	Uraian Revisi	Paraf
1.	Penulisan	
2.	format kutipan / referensi → APA	
3.	Langkah pembuatan Alat (Bab IV)	
4.	Abstrak. (permasalahan)	

Palembang,
Dosen Penguji,

Arsia Rini, S.Kom., M.Kom.
NIP 198809222020122014