//ESP32 Camera Surveillance Car

#include "esp\_camera.h"

#include <WiFi.h>

//

// WARNING!!! Make sure that you have either selected ESP32 Wrover Module,

// or another board which has PSRAM enabled

//

// Adafruit ESP32 Feather

// Select camera model

//#define CAMERA\_MODEL\_WROVER\_KIT

//#define CAMERA\_MODEL\_M5STACK\_PSRAM

#define CAMERA\_MODEL\_AI\_THINKER

const char\* ssid = "hac"; //Enter SSID WIFI Name

const char\* password = "hac123456"; //Enter WIFI Password

/\*#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

\*/

#if 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

// GPIO Setting

extern int gpLb = 2; // Left 1

extern int gpLf = 14; // Left 2

extern int gpRb = 15; // Right 1

extern int gpRf = 13; // Right 2

extern int gpLed = 4; // Light

extern String WiFiAddr ="";

void startCameraServer();

void setup() {

Serial.begin(115200);

Serial.setDebugOutput(true);

Serial.println();

pinMode(gpLb, OUTPUT); //Left Backward

pinMode(gpLf, OUTPUT); //Left Forward

pinMode(gpRb, OUTPUT); //Right Forward

pinMode(gpRf, OUTPUT); //Right Backward

pinMode(gpLed, OUTPUT); //Light

//initialize

digitalWrite(gpLb, LOW);

digitalWrite(gpLf, LOW);

digitalWrite(gpRb, LOW);

digitalWrite(gpRf, LOW);

digitalWrite(gpLed, LOW);

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;

//init with high specs to pre-allocate larger buffers

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;

}

// camera init

esp\_err\_t err = esp\_camera\_init(&config);

if (err != ESP\_OK) {

Serial.printf("Camera init failed with error 0x%x", err);

return;

}

//drop down frame size for higher initial frame rate

sensor\_t \* s = esp\_camera\_sensor\_get();

s->set\_framesize(s, FRAMESIZE\_CIF);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

startCameraServer();

Serial.print("Camera Ready! Use 'http://");

Serial.print(WiFi.localIP());

WiFiAddr = WiFi.localIP().toString();

Serial.println("' to connect");

}

void loop() {

// put your main code here, to run repeatedly:

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "esp\_http\_server.h"

#include "esp\_timer.h"

#include "esp\_camera.h"

#include "img\_converters.h"

#include "camera\_index.h"

#include "Arduino.h"

extern int gpLb;

extern int gpLf;

extern int gpRb;

extern int gpRf;

extern int gpLed;

extern String WiFiAddr;

void WheelAct(int nLf, int nLb, int nRf, int nRb);

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-mixed-replace;boundary=" PART\_BOUNDARY;

static const char\* \_STREAM\_BOUNDARY = "\r\n--" PART\_BOUNDARY "\r\n";

static const char\* \_STREAM\_PART = "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";

static ra\_filter\_t ra\_filter;

httpd\_handle\_t stream\_httpd = NULL;

httpd\_handle\_t camera\_httpd = NULL;

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 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.printf("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 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", (uint32\_t)(fb\_len), (uint32\_t)((fr\_end - fr\_start)/1000));

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];

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){

fb = esp\_camera\_fb\_get();

if (!fb) {

Serial.printf("Camera capture failed");

res = ESP\_FAIL;

} else {

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.printf("JPEG compression failed");

res = ESP\_FAIL;

}

} else {

\_jpg\_buf\_len = fb->len;

\_jpg\_buf = fb->buf;

}

}

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\_buf);

\_jpg\_buf = NULL;

}

if(res != ESP\_OK){

break;

}

int64\_t fr\_end = esp\_timer\_get\_time();

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)"

,(uint32\_t)(\_jpg\_buf\_len),

(uint32\_t)frame\_time, 1000.0 / (uint32\_t)frame\_time,

avg\_frame\_time, 1000.0 / avg\_frame\_time

);

}

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 {

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, "\"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, "\"hmirror\":%u,", s->status.hmirror);

p+=sprintf(p, "\"dcw\":%u,", s->status.dcw);

p+=sprintf(p, "\"colorbar\":%u", s->status.colorbar);

\*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");

String page = "";

page += "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0\">\n";

page += "<script>var xhttp = new XMLHttpRequest();</script>";

page += "<script>function getsend(arg) { xhttp.open('GET', arg +'?' + new Date().getTime(), true); xhttp.send() } </script>";

//page += "<p align=center><IMG SRC='http://" + WiFiAddr + ":81/stream' style='width:280px;'></p><br/><br/>";

page += "<p align=center><IMG SRC='http://" + WiFiAddr + ":81/stream' style='width:300px; transform:rotate(180deg);'></p><br/><br/>";

page += "<p align=center> <button style=background-color:lightgrey;width:90px;height:80px onmousedown=getsend('go') onmouseup=getsend('stop') ontouchstart=getsend('go') ontouchend=getsend('stop') ><b>Forward</b></button> </p>";

page += "<p align=center>";

page += "<button style=background-color:lightgrey;width:90px;height:80px; onmousedown=getsend('left') onmouseup=getsend('stop') ontouchstart=getsend('left') ontouchend=getsend('stop')><b>Left</b></button>&nbsp;";

page += "<button style=background-color:indianred;width:90px;height:80px onmousedown=getsend('stop') onmouseup=getsend('stop')><b>Stop</b></button>&nbsp;";

page += "<button style=background-color:lightgrey;width:90px;height:80px onmousedown=getsend('right') onmouseup=getsend('stop') ontouchstart=getsend('right') ontouchend=getsend('stop')><b>Right</b></button>";

page += "</p>";

page += "<p align=center><button style=background-color:lightgrey;width:90px;height:80px onmousedown=getsend('back') onmouseup=getsend('stop') ontouchstart=getsend('back') ontouchend=getsend('stop') ><b>Backward</b></button></p>";

page += "<p align=center>";

page += "<button style=background-color:yellow;width:140px;height:40px onmousedown=getsend('ledon')><b>Light ON</b></button>";

page += "<button style=background-color:yellow;width:140px;height:40px onmousedown=getsend('ledoff')><b>Light OFF</b></button>";

page += "</p>";

return httpd\_resp\_send(req, &page[0], strlen(&page[0]));

}

static esp\_err\_t go\_handler(httpd\_req\_t \*req){

WheelAct(HIGH, LOW, HIGH, LOW);

Serial.println("Go");

httpd\_resp\_set\_type(req, "text/html");

return httpd\_resp\_send(req, "OK", 2);

}

static esp\_err\_t back\_handler(httpd\_req\_t \*req){

WheelAct(LOW, HIGH, LOW, HIGH);

Serial.println("Back");

httpd\_resp\_set\_type(req, "text/html");

return httpd\_resp\_send(req, "OK", 2);

}

static esp\_err\_t left\_handler(httpd\_req\_t \*req){

WheelAct(HIGH, LOW, LOW, HIGH);

Serial.println("Left");

httpd\_resp\_set\_type(req, "text/html");

return httpd\_resp\_send(req, "OK", 2);

}

static esp\_err\_t right\_handler(httpd\_req\_t \*req){

WheelAct(LOW, HIGH, HIGH, LOW);

Serial.println("Right");

httpd\_resp\_set\_type(req, "text/html");

return httpd\_resp\_send(req, "OK", 2);

}

static esp\_err\_t stop\_handler(httpd\_req\_t \*req){

WheelAct(LOW, LOW, LOW, LOW);

Serial.println("Stop");

httpd\_resp\_set\_type(req, "text/html");

return httpd\_resp\_send(req, "OK", 2);

}

static esp\_err\_t ledon\_handler(httpd\_req\_t \*req){

digitalWrite(gpLed, HIGH);

Serial.println("LED ON");

httpd\_resp\_set\_type(req, "text/html");

return httpd\_resp\_send(req, "OK", 2);

}

static esp\_err\_t ledoff\_handler(httpd\_req\_t \*req){

digitalWrite(gpLed, LOW);

Serial.println("LED OFF");

httpd\_resp\_set\_type(req, "text/html");

return httpd\_resp\_send(req, "OK", 2);

}

void startCameraServer(){

httpd\_config\_t config = HTTPD\_DEFAULT\_CONFIG();

httpd\_uri\_t go\_uri = {

.uri = "/go",

.method = HTTP\_GET,

.handler = go\_handler,

.user\_ctx = NULL

};

httpd\_uri\_t back\_uri = {

.uri = "/back",

.method = HTTP\_GET,

.handler = back\_handler,

.user\_ctx = NULL

};

httpd\_uri\_t stop\_uri = {

.uri = "/stop",

.method = HTTP\_GET,

.handler = stop\_handler,

.user\_ctx = NULL

};

httpd\_uri\_t left\_uri = {

.uri = "/left",

.method = HTTP\_GET,

.handler = left\_handler,

.user\_ctx = NULL

};

httpd\_uri\_t right\_uri = {

.uri = "/right",

.method = HTTP\_GET,

.handler = right\_handler,

.user\_ctx = NULL

};

httpd\_uri\_t ledon\_uri = {

.uri = "/ledon",

.method = HTTP\_GET,

.handler = ledon\_handler,

.user\_ctx = NULL

};

httpd\_uri\_t ledoff\_uri = {

.uri = "/ledoff",

.method = HTTP\_GET,

.handler = ledoff\_handler,

.user\_ctx = NULL

};

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);

Serial.printf("Starting web server on port: '%d'", 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, &go\_uri);

httpd\_register\_uri\_handler(camera\_httpd, &back\_uri);

httpd\_register\_uri\_handler(camera\_httpd, &stop\_uri);

httpd\_register\_uri\_handler(camera\_httpd, &left\_uri);

httpd\_register\_uri\_handler(camera\_httpd, &right\_uri);

httpd\_register\_uri\_handler(camera\_httpd, &ledon\_uri);

httpd\_register\_uri\_handler(camera\_httpd, &ledoff\_uri);

}

config.server\_port += 1;

config.ctrl\_port += 1;

Serial.printf("Starting stream server on port: '%d'", config.server\_port);

if (httpd\_start(&stream\_httpd, &config) == ESP\_OK) {

httpd\_register\_uri\_handler(stream\_httpd, &stream\_uri);

}

}

void WheelAct(int nLf, int nLb, int nRf, int nRb)

{

digitalWrite(gpLf, nLf);

digitalWrite(gpLb, nLb);

digitalWrite(gpRf, nRf);

digitalWrite(gpRb, nRb);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Camera Calibration\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//File: index.html.gz, Size: 3635

#define index\_html\_gz\_len 3635

const uint8\_t index\_html\_gz[] = {

0x1F, 0x8B, 0x08, 0x08, 0x8A, 0xF8, 0xFE, 0x5B, 0x00, 0x03, 0x69, 0x6E, 0x64, 0x65, 0x78, 0x2E,

0x68, 0x74, 0x6D, 0x6C, 0x00, 0xDD, 0x5C, 0xFD, 0x72, 0xDA, 0xB8, 0x16, 0xFF, 0x7F, 0x9F, 0xC2,

0x71, 0x77, 0x8B, 0x3D, 0x6B, 0x08, 0x10, 0x92, 0xA6, 0x26, 0x90, 0x0D, 0x84, 0xB6, 0x3B, 0xD3,

0xAF, 0x6D, 0xF6, 0xEE, 0xEE, 0xCC, 0xCE, 0x4E, 0x2B, 0x6C, 0x19, 0xD4, 0x18, 0x8B, 0xDA, 0x72,

0x80, 0xB2, 0x7E, 0x8E, 0xFB, 0x40, 0xF7, 0xC5, 0xEE, 0x91, 0x64, 0x1B, 0x9B, 0x8F, 0x10, 0xA0,

0x85, 0x4E, 0x9B, 0x19, 0x90, 0xE5, 0xA3, 0xA3, 0x73, 0xCE, 0xEF, 0x7C, 0x48, 0xC6, 0xEA, 0xC5,

0x91, 0x4D, 0x2D, 0x36, 0x19, 0x62, 0xA5, 0xCF, 0x06, 0x6E, 0xF3, 0x87, 0x0B, 0xF9, 0xA5, 0xC0,

0xBF, 0x8B, 0x3E, 0x46, 0xB6, 0x6C, 0x8A, 0xCB, 0x01, 0x66, 0x48, 0xB1, 0xFA, 0xC8, 0x0F, 0x30,

0x6B, 0xA8, 0x21, 0x73, 0x8A, 0xE7, 0xEA, 0xFC, 0x6D, 0x0F, 0x0D, 0x70, 0x43, 0xBD, 0x23, 0x78,

0x34, 0xA4, 0x3E, 0x53, 0x15, 0x8B, 0x7A, 0x0C, 0x7B, 0x40, 0x3E, 0x22, 0x36, 0xEB, 0x37, 0x6C,

0x7C, 0x47, 0x2C, 0x5C, 0x14, 0x17, 0x06, 0xF1, 0x08, 0x23, 0xC8, 0x2D, 0x06, 0x16, 0x72, 0x71,

0xA3, 0x92, 0xE5, 0xC5, 0x08, 0x73, 0x71, 0xB3, 0x73, 0xF3, 0xF6, 0xA4, 0xAA, 0xBC, 0xF9, 0xA3,

0x5A, 0x3B, 0x2B, 0x5F, 0x1C, 0xCB, 0xBE, 0x19, 0x4D, 0xC0, 0x26, 0xFC, 0xBA, 0x4B, 0xED, 0xC9,

0xD4, 0x81, 0x69, 0x8A, 0x0E, 0x1A, 0x10, 0x77, 0x62, 0x5E, 0xF9, 0xC0, 0xD4, 0x78, 0x81, 0xDD,

0x3B, 0xCC, 0x88, 0x85, 0x8C, 0x00, 0x79, 0x41, 0x31, 0xC0, 0x3E, 0x71, 0xEA, 0x5D, 0x64, 0xDD,

0xF6, 0x7C, 0x1A, 0x7A, 0xB6, 0xF9, 0xA8, 0x72, 0xCE, 0xFF, 0xEA, 0x16, 0x75, 0xA9, 0x6F, 0x3E,

0xEA, 0x3C, 0xE3, 0x7F, 0x75, 0xC1, 0x27, 0x20, 0x9F, 0xB1, 0x59, 0x39, 0x1B, 0x8E, 0xA3, 0x7E,

0x75, 0x9A, 0xE9, 0x39, 0x87, 0x9E, 0x00, 0x5B, 0x8C, 0x50, 0xAF, 0x34, 0x40, 0xC4, 0x9B, 0xDA,

0x24, 0x18, 0xBA, 0x68, 0x62, 0x3A, 0x2E, 0x1E, 0x47, 0x8F, 0x06, 0xD8, 0x0B, 0x8D, 0xDC, 0x7D,

0xDE, 0x5F, 0xB4, 0x89, 0x2F, 0xFB, 0x4C, 0x98, 0x2A, 0x1C, 0x78, 0x92, 0x30, 0x1D, 0xEB, 0x51,

0x0F, 0xD7, 0x05, 0xE1, 0xC8, 0x47, 0x43, 0xB8, 0xE4, 0x5F, 0xF5, 0x01, 0xF1, 0xA4, 0x91, 0xCC,

0x93, 0x5A, 0x79, 0x38, 0xCE, 0x09, 0x7E, 0x72, 0xC6, 0xFF, 0xEA, 0x43, 0x64, 0xDB, 0xC4, 0xEB,

0x99, 0xE7, 0xFC, 0x36, 0xF5, 0x6D, 0xEC, 0x17, 0x7D, 0x64, 0x93, 0x30, 0x30, 0x6B, 0xD0, 0x33,

0x40, 0x7E, 0x0F, 0x78, 0x30, 0x3A, 0x34, 0x8B, 0x95, 0xF2, 0xAC, 0xC3, 0x27, 0xBD, 0x3E, 0x33,

0x79, 0x4F, 0xF4, 0x28, 0xC6, 0x26, 0xA7, 0x46, 0x46, 0x14, 0x21, 0x08, 0x72, 0x49, 0xCF, 0x2B,

0x12, 0x86, 0x07, 0x81, 0x19, 0x30, 0x1F, 0x33, 0xAB, 0x1F, 0x39, 0xA4, 0x17, 0xFA, 0x78, 0x9A,

0x08, 0x50, 0x8E, 0x79, 0x43, 0xA3, 0x38, 0xC2, 0xDD, 0x5B, 0xC2, 0x8A, 0xF1, 0x64, 0x5D, 0xEC,

0x50, 0x1F, 0xA7, 0x04, 0xC5, 0xAE, 0x4B, 0xAD, 0xDB, 0x62, 0xC0, 0x90, 0xCF, 0x16, 0x89, 0x91,

0xC3, 0xB0, 0x3F, 0x4F, 0x8B, 0x41, 0xE1, 0x05, 0xCA, 0x84, 0x41, 0x7C, 0x49, 0x3C, 0x97, 0x78,

0x78, 0x15, 0x5B, 0xC9, 0x21, 0x4F, 0x2A, 0xFA, 0x62, 0x35, 0x14, 0x32, 0xE8, 0xA5, 0x16, 0x10,

0x93, 0xD6, 0xA5, 0xE1, 0x2B, 0xE5, 0xF2, 0x4F, 0xF5, 0x3E, 0x16, 0xF6, 0x42, 0x21, 0xA3, 0xF7,

0x1B, 0x99, 0xFB, 0xC6, 0x2F, 0x03, 0x6C, 0x13, 0xA4, 0x68, 0x33, 0xF0, 0x94, 0xF3, 0x32, 0x58,

0x5A, 0x57, 0x90, 0x67, 0x2B, 0x1A, 0xF5, 0x09, 0x58, 0x1B, 0x09, 0x57, 0x70, 0xA1, 0x07, 0xDC,

0x7E, 0x88, 0xF5, 0xE9, 0x3A, 0x18, 0x62, 0x8F, 0x58, 0x0D, 0xC4, 0x12, 0x0D, 0x06, 0x68, 0x5C,

0xCC, 0x68, 0xC1, 0x2F, 0x63, 0x4D, 0x20, 0xD4, 0x2C, 0x0D, 0x3A, 0xEF, 0xFA, 0x4A, 0x51, 0xE1,

0xAE, 0xA5, 0xC7, 0xEA, 0x0A, 0x15, 0x33, 0xEA, 0x7E, 0x2F, 0x28, 0x27, 0x11, 0xFB, 0xA8, 0x1B,

0x32, 0x46, 0xBD, 0x60, 0x8D, 0x99, 0x3F, 0x86, 0x01, 0x23, 0xCE, 0xA4, 0x18, 0x83, 0x62, 0x06,

0x43, 0x04, 0xF9, 0xAA, 0x8B, 0xD9, 0x08, 0x63, 0x08, 0x5D, 0x0F, 0xDD, 0x01, 0xDC, 0xBD, 0x9E,

0x8B, 0xA7, 0x56, 0xE8, 0x07, 0x90, 0x39, 0x86, 0x94, 0x00, 0xA5, 0x5F, 0xCF, 0x01, 0x90, 0x25,

0x2C, 0x5A, 0xDD, 0x29, 0x0D, 0x19, 0x17, 0x09, 0x44, 0xA4, 0xC0, 0x8F, 0xB0, 0x09, 0xB4, 0xA4,

0xD9, 0xCB, 0x89, 0xCD, 0xCB, 0x73, 0x63, 0x4C, 0xAB, 0x8F, 0xAD, 0x5B, 0x6C, 0xFF, 0x9C, 0x4F,

0x17, 0x22, 0xD5, 0x94, 0x88, 0x37, 0x0C, 0x59, 0x91, 0x27, 0x84, 0xE1, 0x1A, 0x7D, 0x84, 0x25,

0xE2, 0x29, 0xAA, 0xD5, 0xD4, 0x67, 0xCD, 0xD3, 0xE1, 0x58, 0x29, 0xE7, 0x18, 0x35, 0x5D, 0xD4,

0xC5, 0x6E, 0xCA, 0x2E, 0x36, 0xA2, 0xF4, 0xA7, 0xD8, 0x09, 0x32, 0xD9, 0x23, 0x93, 0xA1, 0x6A,

0x4F, 0x7E, 0xCA, 0x31, 0x52, 0x44, 0xDB, 0xC8, 0x75, 0x05, 0xD8, 0x05, 0x18, 0x64, 0x42, 0x84,

0x9E, 0x91, 0x59, 0x89, 0x4A, 0x3E, 0xF2, 0x7A, 0x18, 0x00, 0x1C, 0x1B, 0x49, 0x33, 0x93, 0x52,

0x97, 0x4D, 0x6F, 0x96, 0x15, 0x10, 0x3B, 0x92, 0x40, 0x2E, 0x78, 0x7C, 0xA2, 0x56, 0x86, 0xBA,

0x52, 0x4D, 0x73, 0x23, 0x18, 0x3A, 0x67, 0x0A, 0x9E, 0x35, 0xE7, 0x10, 0x8C, 0x2B, 0x81, 0xE3,

0xE4, 0xEB, 0x84, 0xE3, 0x9C, 0x94, 0x4F, 0x6A, 0x73, 0xD1, 0xCF, 0xE7, 0xC9, 0xD7, 0x8A, 0x7A,

0x8A, 0x71, 0x2C, 0xA0, 0xD9, 0xA7, 0x77, 0xD8, 0x9F, 0xE6, 0x59, 0xD5, 0x9E, 0xD6, 0xEC, 0xE4,

0x3E, 0x02, 0xBF, 0xBC, 0xC3, 0x79, 0x82, 0x6A, 0xC5, 0xAA, 0x56, 0x62, 0x82, 0x12, 0x68, 0x88,

0xBA, 0x2E, 0xB6, 0x13, 0x57, 0xB3, 0xB1, 0x83, 0x42, 0x97, 0xE5, 0xA4, 0x43, 0x65, 0xFE, 0x17,

0x09, 0x5B, 0xFF, 0xCD, 0xCB, 0x78, 0x43, 0xD8, 0xF2, 0x9F, 0x69, 0x12, 0x20, 0x68, 0x38, 0xC4,

0x08, 0xFA, 0x2C, 0x2C, 0x4B, 0xCD, 0x62, 0x72, 0x13, 0x6E, 0xB1, 0xA4, 0xC0, 0xCC, 0x99, 0x27,

0x09, 0xFF, 0xC5, 0xB9, 0x4C, 0x87, 0x5A, 0x61, 0x30, 0x73, 0xF2, 0x25, 0x14, 0x66, 0x22, 0x4E,

0xE0, 0x12, 0x61, 0xC6, 0xD0, 0xF3, 0xB8, 0x6E, 0x45, 0xE6, 0xC3, 0xC4, 0xD3, 0x25, 0x42, 0x2D,

0xE2, 0x93, 0x15, 0x31, 0x2E, 0xD7, 0x79, 0x50, 0xCA, 0x29, 0xD6, 0x4A, 0x40, 0x61, 0x1E, 0x25,

0x26, 0x7B, 0x80, 0x3C, 0xAC, 0x1F, 0x0E, 0xBA, 0xD3, 0x78, 0x78, 0x05, 0x62, 0x43, 0x32, 0xF0,

0x7B, 0x5D, 0xA4, 0x95, 0x8D, 0xB2, 0x71, 0x02, 0x1F, 0x7A, 0xCE, 0x60, 0x52, 0xE4, 0x6A, 0x75,

0xA1, 0xFA, 0x9E, 0xCE, 0xD7, 0xEB, 0xD8, 0x81, 0xE6, 0xB4, 0x59, 0x85, 0x4F, 0xAE, 0x70, 0x57,

0x4A, 0xDC, 0xE1, 0x57, 0x18, 0x7C, 0x9D, 0x51, 0x17, 0xED, 0xB5, 0xD4, 0x10, 0x03, 0xFA, 0xB9,

0x28, 0xE3, 0xEF, 0x60, 0x58, 0x64, 0x44, 0xD8, 0x37, 0x0E, 0xCB, 0xE5, 0x09, 0xB6, 0xB4, 0x45,

0x59, 0x49, 0xF4, 0x2E, 0xCA, 0x6C, 0x02, 0x6C, 0x3C, 0x28, 0x21, 0x3E, 0x94, 0x92, 0xFA, 0x42,

0xCF, 0xAA, 0xB9, 0x1D, 0xE2, 0xBA, 0x45, 0x97, 0x8E, 0xE6, 0xB2, 0x47, 0xCE, 0xCE, 0xF3, 0x76,

0x9D, 0x37, 0xFF, 0xBD, 0xBC, 0x43, 0xF0, 0xB9, 0xAF, 0xC0, 0x7B, 0xFF, 0x41, 0x34, 0x03, 0xE5,

0x9E, 0x20, 0x59, 0x67, 0xD1, 0x07, 0x0C, 0x5D, 0x34, 0x98, 0xCC, 0x91, 0x51, 0x29, 0x18, 0x11,

0x58, 0x89, 0xCD, 0x15, 0xA3, 0x21, 0x0D, 0x88, 0x58, 0xE6, 0xF9, 0xD8, 0x45, 0x3C, 0xC9, 0x2F,

0x96, 0xE1, 0xB9, 0xE2, 0x91, 0xB9, 0x95, 0xF0, 0x94, 0x65, 0xF4, 0x61, 0x4B, 0x87, 0x92, 0xCC,

0x00, 0xB1, 0xBF, 0x0A, 0xE3, 0xE5, 0x92, 0x7B, 0xCE, 0xB6, 0xD5, 0x7B, 0x7D, 0x38, 0x76, 0xDC,

0x9E, 0x8F, 0x27, 0x09, 0x5B, 0x23, 0xFE, 0x36, 0xE5, 0x4A, 0x6F, 0x79, 0x8D, 0x16, 0x7E, 0x2D,

0xB5, 0x2E, 0xD5, 0x82, 0x68, 0x6E, 0xC8, 0xA2, 0x45, 0x92, 0x05, 0x96, 0xAA, 0x2E, 0x40, 0x9F,

0x06, 0x9B, 0x30, 0x4D, 0x1C, 0x83, 0xBC, 0xE9, 0x62, 0x87, 0x89, 0x85, 0x37, 0xCF, 0x8E, 0x27,

0x39, 0x0F, 0x29, 0xCE, 0xAA, 0xB7, 0xC4, 0x33, 0x5D, 0x3F, 0x25, 0xB6, 0x59, 0x46, 0xCB, 0x7D,

0x6A, 0x39, 0x79, 0x22, 0x78, 0x92, 0x62, 0x85, 0x7A, 0xD0, 0x33, 0x90, 0x01, 0x0C, 0x4A, 0xE0,

0xBF, 0xB4, 0xEA, 0x19, 0x5F, 0x3F, 0xAF, 0xBE, 0x15, 0xC5, 0xCB, 0x9E, 0x85, 0x90, 0x48, 0x4A,

0x6C, 0xC6, 0x0B, 0x6A, 0x73, 0x98, 0xCD, 0x70, 0x5F, 0x58, 0x79, 0xC0, 0x6A, 0x6B, 0x80, 0x20,

0x59, 0x72, 0x13, 0xC2, 0x36, 0x13, 0x74, 0x5B, 0x34, 0xEF, 0x6C, 0x79, 0x56, 0x39, 0xE3, 0x9B,

0xBD, 0x92, 0xE5, 0xD2, 0x20, 0x83, 0x03, 0xEA, 0x82, 0x24, 0x21, 0xC3, 0x75, 0xB9, 0xA4, 0x3B,

0x8D, 0x8D, 0x7A, 0xBA, 0x3C, 0xEC, 0x32, 0x18, 0x64, 0xA1, 0xC9, 0x4B, 0x56, 0xE1, 0x7B, 0x9D,

0xEC, 0x2A, 0x8A, 0xE1, 0x31, 0xD4, 0x37, 0xBE, 0x6F, 0x31, 0x2D, 0x2C, 0xDC, 0x2C, 0x1B, 0x06,

0x95, 0xC5, 0x25, 0x58, 0x54, 0xEA, 0x13, 0xDB, 0xC6, 0x5E, 0x6E, 0x73, 0x1C, 0xCD, 0x76, 0xFC,

0xC7, 0xF1, 0x96, 0x5F, 0x5E, 0xCC, 0x9E, 0x4E, 0x5C, 0xF0, 0x67, 0x00, 0xD9, 0x27, 0x03, 0x72,

0xC9, 0xAF, 0x58, 0x2E, 0x0A, 0x82, 0x86, 0xCA, 0xF7, 0xE2, 0x99, 0x87, 0x0B, 0x82, 0xC4, 0x26,

0x77, 0x0A, 0xB1, 0x1B, 0xAA, 0x4B, 0x7B, 0x74, 0xEE, 0x9E, 0xB8, 0x2F, 0x16, 0xC3, 0x0A, 0xA0,

0xDA, 0x50, 0x73, 0xCB, 0x72, 0x55, 0x8C, 0x9A, 0x75, 0xA9, 0xCD, 0xC7, 0x8F, 0x9E, 0x3E, 0x79,

0x72, 0x56, 0x7F, 0xEC, 0x75, 0x83, 0x61, 0xFC, 0xF9, 0xBB, 0xB8, 0x05, 0x8B, 0x5E, 0xC6, 0x60,

0x21, 0x1A, 0x5C, 0x1C, 0x0B, 0x6E, 0x73, 0x12, 0x1C, 0x83, 0x08, 0x2B, 0x84, 0x8A, 0x63, 0x63,

0x99, 0x5C, 0x09, 0x49, 0x00, 0x4E, 0xDA, 0x45, 0xFE, 0x12, 0x12, 0x41, 0x26, 0x7C, 0x5A, 0x11,

0x29, 0x4D, 0x15, 0x9E, 0xDD, 0xA5, 0xE3, 0x79, 0xD1, 0x85, 0x36, 0xB1, 0xDB, 0xC7, 0x54, 0xD8,

0x5E, 0xC5, 0x10, 0x86, 0x89, 0xE1, 0x7C, 0x33, 0xB2, 0x82, 0x26, 0x95, 0x2F, 0x36, 0x7B, 0x66,

0xFD, 0x2F, 0xA7, 0x76, 0x7C, 0x34, 0xC0, 0xDC, 0xDB, 0xE3, 0xCE, 0xD5, 0x6C, 0xE6, 0x21, 0x48,

0x47, 0xAA, 0xCD, 0x77, 0x58, 0x38, 0x2E, 0xC0, 0xBB, 0xD4, 0xAC, 0x0B, 0x5C, 0x64, 0x08, 0xE6,

0xE7, 0x57, 0x13, 0x11, 0xE3, 0x15, 0x75, 0x11, 0x09, 0x7F, 0x59, 0x23, 0x90, 0x60, 0x47, 0x87,

0xC2, 0xB3, 0xEE, 0x90, 0x1B, 0x82, 0x69, 0x2B, 0x65, 0xB5, 0xF9, 0x9F, 0xBF, 0x9E, 0x5F, 0x69,

0x10, 0x64, 0xE5, 0x71, 0xA5, 0x5A, 0x2E, 0xEB, 0x17, 0xC7, 0x92, 0x64, 0x63, 0x5E, 0x4F, 0xD5,

0xE6, 0x8D, 0x60, 0x55, 0x3D, 0x07, 0x56, 0xE5, 0x6A, 0x6D, 0x7B, 0x56, 0xE7, 0x6A, 0x53, 0x70,

0x02, 0x26, 0xE3, 0x27, 0x67, 0xE7, 0xDB, 0x33, 0x7A, 0x02, 0x32, 0xFD, 0x01, 0x9C, 0xCE, 0x41,

0xBB, 0xB3, 0x5D, 0x94, 0x3B, 0x53, 0x9B, 0x9C, 0xCF, 0x59, 0xAD, 0x3C, 0xAE, 0x9D, 0xEF, 0xC0,

0xE7, 0x54, 0x8D, 0xB7, 0x92, 0xDC, 0x65, 0x93, 0x96, 0xDA, 0x6C, 0xFF, 0xFA, 0x4C, 0xAB, 0x81,

0x8C, 0xD5, 0xA7, 0x67, 0xDB, 0xF3, 0xAE, 0xA9, 0xCD, 0xDF, 0xB8, 0x90, 0x27, 0x55, 0x60, 0x54,

0xDB, 0x41, 0xC8, 0x13, 0xB5, 0xF9, 0x42, 0x70, 0x02, 0x2E, 0xE3, 0xCA, 0x93, 0x1D, 0x44, 0x02,

0xF7, 0xFA, 0x4D, 0x70, 0x02, 0xFF, 0xE2, 0xEE, 0xF5, 0x40, 0x4E, 0x90, 0x28, 0x85, 0x69, 0xEE,

0x89, 0xD3, 0xC5, 0xEC, 0x93, 0xBB, 0x7D, 0x5F, 0x18, 0x7F, 0x0A, 0x21, 0xA7, 0xB3, 0xC9, 0xC6,

0x41, 0x1C, 0x8F, 0x03, 0x95, 0x64, 0xE3, 0x61, 0xF1, 0x9B, 0x91, 0x24, 0x7D, 0x4A, 0xA0, 0x36,

0x2B, 0xE5, 0x35, 0x1A, 0x88, 0xB1, 0xD9, 0x2C, 0x28, 0x06, 0xE7, 0x14, 0x50, 0x15, 0x60, 0x25,

0x62, 0x58, 0x19, 0xA0, 0x31, 0xF8, 0xE8, 0x89, 0x9A, 0x89, 0xEB, 0xAD, 0x52, 0xC4, 0x12, 0x69,

0xD1, 0x58, 0x6D, 0x9E, 0x9D, 0xAC, 0xB3, 0xF7, 0x0E, 0x70, 0x74, 0x45, 0x05, 0xF7, 0x70, 0x10,

0x6C, 0x8C, 0xC8, 0x6C, 0xA8, 0xDA, 0x6C, 0xA5, 0xED, 0x5D, 0x70, 0x29, 0x56, 0x77, 0xC0, 0x25,

0x23, 0x8E, 0x84, 0xA6, 0x58, 0x8D, 0xA1, 0xA9, 0xAA, 0xB3, 0x88, 0xF8, 0x92, 0xC0, 0xAC, 0x93,

0x76, 0x17, 0x5C, 0x78, 0x11, 0xF7, 0x51, 0xC0, 0x36, 0x46, 0x25, 0x19, 0x08, 0x69, 0x2D, 0x6E,

0x1D, 0x0C, 0x91, 0x54, 0x94, 0xEF, 0x00, 0x8F, 0x00, 0xB1, 0xD0, 0x17, 0x4F, 0xDF, 0x37, 0x46,

0x64, 0x36, 0x14, 0xEA, 0x61, 0xDA, 0x3E, 0x18, 0x2A, 0x19, 0x71, 0xBE, 0x07, 0x5C, 0x86, 0xD8,

0x22, 0xC8, 0x7D, 0x8F, 0x1D, 0x07, 0x4A, 0xD6, 0xE6, 0xD8, 0xE4, 0x86, 0x03, 0x3E, 0xF2, 0x5A,

0xE9, 0x88, 0xEB, 0x8D, 0xD7, 0x88, 0x73, 0xEC, 0xBE, 0xD4, 0x42, 0xB1, 0xBC, 0x7C, 0xDD, 0xF2,

0x9A, 0xA6, 0x72, 0x6E, 0xB9, 0x42, 0xA8, 0x00, 0x13, 0xDC, 0x13, 0x7B, 0xBE, 0xAD, 0x79, 0x54,

0xD5, 0xE6, 0x73, 0x1F, 0x4D, 0xC4, 0xCF, 0xB0, 0xBB, 0x2C, 0x7A, 0xDE, 0x61, 0x5B, 0xF9, 0x1D,

0x36, 0x72, 0xBB, 0xAC, 0xC0, 0x9E, 0xFB, 0x18, 0x7B, 0xBB, 0x71, 0x39, 0x85, 0x62, 0x06, 0x8D,

0xDD, 0x98, 0xC0, 0x82, 0xF5, 0x06, 0x0F, 0x09, 0xFA, 0x16, 0x16, 0x5C, 0x68, 0xD4, 0xDD, 0x38,

0x2C, 0x60, 0x8C, 0xDA, 0xBC, 0xFA, 0xB3, 0xB5, 0x71, 0x92, 0x92, 0x0F, 0x9F, 0x1E, 0xE2, 0xE1,

0x32, 0x3B, 0xC5, 0x02, 0xAA, 0x0B, 0x9B, 0xCD, 0xE5, 0x91, 0xF3, 0xD0, 0x0D, 0xE7, 0x12, 0xBD,

0x12, 0x01, 0xC5, 0xF3, 0x19, 0x35, 0xA3, 0xE6, 0xC3, 0x74, 0xFC, 0x7A, 0x19, 0x0C, 0x84, 0x78,

0xDF, 0x43, 0x64, 0xF3, 0xBA, 0x92, 0x0C, 0x14, 0x48, 0x29, 0xCF, 0xA1, 0xB5, 0x2F, 0xB8, 0xE4,

0xB4, 0x07, 0xC3, 0x2C, 0xD6, 0xFA, 0xD0, 0xC0, 0x81, 0x20, 0x03, 0x6A, 0x6F, 0xFE, 0x38, 0x22,

0x1E, 0xA7, 0x36, 0x01, 0xB5, 0x57, 0xD0, 0xD8, 0xB8, 0xCA, 0x24, 0x0C, 0xBE, 0x72, 0x79, 0xB9,

0x0A, 0x19, 0xDD, 0xA5, 0xB2, 0xDC, 0x84, 0x9E, 0x37, 0xD9, 0xA5, 0xAC, 0xB4, 0x5D, 0x1A, 0xDA,

0xDB, 0x73, 0x80, 0x9A, 0xF2, 0xC6, 0x71, 0x88, 0xB5, 0x7D, 0x55, 0x82, 0x8A, 0xF2, 0x82, 0x0E,

0x1E, 0x38, 0xFE, 0x2B, 0x67, 0x71, 0x6C, 0x6D, 0x9E, 0x20, 0xB0, 0x05, 0x28, 0x76, 0xDA, 0xCA,

0x4D, 0xE7, 0xF5, 0xCD, 0x9B, 0x77, 0xFB, 0xC9, 0x0E, 0x30, 0xE7, 0x81, 0x12, 0x03, 0xD7, 0xF6,

0xD0, 0x39, 0x01, 0x84, 0xA8, 0x6E, 0x83, 0x53, 0x55, 0x02, 0x75, 0x7D, 0xF3, 0x76, 0x5F, 0x28,

0x55, 0x0F, 0x07, 0x53, 0xF5, 0x5B, 0xC0, 0xE9, 0xBD, 0x8B, 0xEF, 0xB0, 0xBB, 0x05, 0x56, 0x72,

0x20, 0xC7, 0x4B, 0x79, 0xC9, 0x5B, 0x07, 0xDB, 0xC8, 0xA5, 0xA2, 0x7C, 0x07, 0xDB, 0x38, 0xF0,

0x8A, 0xF7, 0x42, 0xE8, 0x6D, 0x82, 0x47, 0x8E, 0x54, 0x9B, 0x9D, 0xF1, 0x90, 0x06, 0xA1, 0xFF,

0xC0, 0x82, 0xBA, 0x1C, 0x91, 0x5D, 0x9E, 0x0C, 0xCE, 0x44, 0x91, 0x88, 0x24, 0x8F, 0x06, 0xF9,

0x93, 0xFD, 0x14, 0x93, 0x6A, 0xB9, 0xF6, 0x45, 0x51, 0xE1, 0xCC, 0xBF, 0x26, 0x30, 0xBD, 0x2D,

0xEA, 0x4E, 0x8F, 0xD7, 0x9D, 0xE7, 0xED, 0xFD, 0xA4, 0xB2, 0xDE, 0xC1, 0x0A, 0x4E, 0xEF, 0xA0,

0x05, 0x47, 0x91, 0xBF, 0x76, 0xA6, 0x30, 0x6D, 0xB9, 0x89, 0x88, 0x07, 0xC2, 0xDE, 0x79, 0x9B,

0x0D, 0x44, 0xF6, 0xA1, 0xFA, 0x78, 0x97, 0xD0, 0x49, 0xC4, 0xC8, 0x47, 0xCE, 0xC9, 0x2C, 0x6E,

0x4E, 0xBF, 0x68, 0xD4, 0x9C, 0xAC, 0x95, 0x76, 0x97, 0xA0, 0xE1, 0x9A, 0x58, 0x98, 0xB8, 0xFC,

0xA5, 0xC7, 0x4D, 0x01, 0xC9, 0x8C, 0x95, 0x98, 0x28, 0x6D, 0x79, 0xB5, 0x0B, 0x36, 0xD5, 0x5D,

0xB0, 0xC9, 0x4A, 0x94, 0x87, 0xE7, 0xEC, 0x2B, 0x55, 0x9A, 0x4A, 0xF5, 0xFC, 0x6B, 0xC2, 0xD3,

0x1D, 0x6E, 0x9E, 0xD3, 0x60, 0x8C, 0xDA, 0x6C, 0xBD, 0xDD, 0x4F, 0x4E, 0xE3, 0x93, 0x3D, 0x30,

0xA7, 0xED, 0x94, 0xC1, 0x84, 0x52, 0x87, 0x5E, 0x8A, 0x8D, 0xB6, 0x40, 0x63, 0xC4, 0x05, 0xFF,

0x73, 0x4F, 0x68, 0x8C, 0x1E, 0x8E, 0xC6, 0x17, 0xAE, 0x30, 0xA3, 0x6F, 0x01, 0x1F, 0x1F, 0x8D,

0xDE, 0xF7, 0x06, 0x68, 0x63, 0x8C, 0xE2, 0x71, 0x6A, 0xF3, 0x1D, 0x1A, 0x29, 0xCF, 0x5F, 0x5D,

0xED, 0x05, 0xAB, 0x64, 0xD2, 0xC3, 0xE0, 0x95, 0xAA, 0x7C, 0x68, 0xCC, 0x5C, 0xEC, 0x6D, 0x1E,

0x54, 0x7C, 0x90, 0xDA, 0x7C, 0x89, 0xBD, 0x40, 0x69, 0x53, 0x3F, 0x3E, 0x76, 0xB4, 0x17, 0xD4,

0xC4, 0xCC, 0x87, 0x81, 0x4C, 0x2A, 0x7D, 0x68, 0xBC, 0xFA, 0x03, 0xE2, 0xFB, 0xD4, 0xDF, 0x18,

0xB2, 0x78, 0x9C, 0xDA, 0x7C, 0x51, 0x7C, 0x25, 0x5A, 0x7B, 0x81, 0x2B, 0x99, 0xF5, 0x30, 0x88,

0xA5, 0x3A, 0x1F, 0x1A, 0x34, 0xDB, 0x1A, 0x6D, 0x0C, 0x18, 0x8C, 0x51, 0x9B, 0xD7, 0xED, 0x3F,

0x15, 0xED, 0x9A, 0x8E, 0x3C, 0xFE, 0x36, 0x99, 0xD2, 0x79, 0xAD, 0xEF, 0x05, 0x35, 0x3E, 0xF5,

0x61, 0x10, 0x13, 0x4A, 0x1F, 0x1A, 0x2D, 0xF1, 0x66, 0x69, 0x17, 0x6D, 0x1E, 0x63, 0xC9, 0x40,

0xFE, 0x42, 0x05, 0xB4, 0x94, 0x16, 0xDA, 0x4F, 0x94, 0xA5, 0xF3, 0xEE, 0x63, 0x25, 0x38, 0x53,

0xF2, 0xD0, 0x38, 0x39, 0xC8, 0xC2, 0xEF, 0x6D, 0xCC, 0xB6, 0xF9, 0x35, 0x3F, 0x33, 0x56, 0x6D,

0x3E, 0x83, 0x0B, 0xE5, 0x5A, 0x5C, 0xEC, 0xAB, 0x8E, 0x65, 0xE7, 0xDF, 0x07, 0x6A, 0x39, 0x7D,

0xBF, 0x09, 0xE0, 0x60, 0xD5, 0x40, 0x7B, 0xDE, 0x56, 0x2F, 0xE9, 0xE6, 0x86, 0xC7, 0xF0, 0xBD,

0x93, 0xD7, 0xFB, 0x05, 0x70, 0x26, 0xC4, 0xDE, 0x30, 0xCC, 0xE8, 0xBD, 0x0F, 0x18, 0x93, 0x37,

0xDC, 0xC5, 0x5E, 0x53, 0x1E, 0x6C, 0x5D, 0x87, 0x94, 0x24, 0x93, 0xCF, 0x03, 0x30, 0x2B, 0x06,

0x8C, 0xB8, 0xAE, 0xDA, 0x7C, 0x8E, 0x99, 0x72, 0xC3, 0x9B, 0x17, 0xC7, 0x92, 0xE0, 0xE1, 0x5C,

0xE2, 0xB7, 0xC8, 0xF9, 0x61, 0x64, 0x34, 0x50, 0x9B, 0x37, 0xFC, 0x64, 0x2E, 0xF0, 0xE2, 0x57,

0x9B, 0x33, 0x13, 0x46, 0xC4, 0x9E, 0x4F, 0x41, 0xA8, 0x14, 0xA4, 0xF8, 0xFC, 0xA3, 0xAA, 0x24,

0xAD, 0x4C, 0x5F, 0xB3, 0x23, 0x88, 0x15, 0xEE, 0x65, 0xEB, 0xA7, 0xE3, 0x3F, 0xED, 0x59, 0xAB,

0x7F, 0x01, 0xBC, 0x38, 0xF6, 0xD0, 0x12, 0x73, 0xAF, 0x40, 0xE1, 0x42, 0x1E, 0x8D, 0x5E, 0xC1,

0x2A, 0x7D, 0x43, 0x5F, 0x58, 0x62, 0x76, 0x48, 0x23, 0x55, 0x6B, 0xEE, 0xF0, 0x46, 0xF2, 0x14,

0xF0, 0x61, 0x41, 0x2B, 0x8E, 0x71, 0xC4, 0xF5, 0x90, 0x37, 0x53, 0xF3, 0xFF, 0xEF, 0xBF, 0xEB,

0x7C, 0x86, 0x0C, 0x7A, 0x19, 0xC1, 0x54, 0x25, 0xF0, 0xAD, 0x86, 0xBA, 0xEA, 0x7D, 0xFF, 0x15,

0x9A, 0x1F, 0x2F, 0x53, 0x7D, 0x8E, 0x78, 0x89, 0xAD, 0x2F, 0x02, 0xCB, 0x27, 0x43, 0xD6, 0xFC,

0xC1, 0xA6, 0x56, 0x38, 0xC0, 0x1E, 0x2B, 0x21, 0xDB, 0xEE, 0xDC, 0x41, 0xE3, 0x25, 0x09, 0x18,

0x06, 0x2B, 0x68, 0x85, 0xEB, 0x37, 0xAF, 0xDA, 0xF2, 0xDC, 0xC3, 0x4B, 0x8A, 0x6C, 0x6C, 0x17,

0x0C, 0x27, 0xF4, 0x04, 0x1F, 0x4D, 0x9F, 0x26, 0x4D, 0xA5, 0xAB, 0xB5, 0xF4, 0xA9, 0x0B, 0x4E,

0xDB, 0xAE, 0xCB, 0xF4, 0xA0, 0xB5, 0x4A, 0x3C, 0xC6, 0xF5, 0xA9, 0x85, 0x02, 0x5C, 0x48, 0x02,

0xBD, 0x60, 0xB6, 0x1B, 0xAD, 0x52, 0xBC, 0xF6, 0xB9, 0xAC, 0xF0, 0x53, 0x34, 0xA0, 0xF4, 0x6D,

0x5D, 0x10, 0x89, 0xE7, 0x54, 0x05, 0x53, 0xB4, 0xE5, 0x2F, 0xBE, 0x45, 0xEA, 0x61, 0x39, 0x44,

0x3C, 0x0D, 0xCB, 0x12, 0x4B, 0xCF, 0x4A, 0xA8, 0xC3, 0xEE, 0x80, 0x30, 0x4E, 0x59, 0xA8, 0x14,

0x62, 0xAA, 0x38, 0x95, 0x98, 0x3E, 0x66, 0xA1, 0xEF, 0xD5, 0x23, 0x00, 0x36, 0x60, 0xCA, 0x75,

0xE3, 0xC3, 0x8F, 0x53, 0x2B, 0x3A, 0x16, 0x6F, 0x50, 0x52, 0xF7, 0xF2, 0x0E, 0xF9, 0x8D, 0x1F,

0xA7, 0xAD, 0x12, 0xB1, 0xA3, 0xC7, 0x30, 0x07, 0xB4, 0xDB, 0xD1, 0x87, 0xBA, 0xC3, 0x8F, 0xF1,

0x6B, 0xD7, 0x7A, 0x89, 0xF5, 0xB1, 0xA7, 0x75, 0x1A, 0xCD, 0x29, 0x1F, 0x4D, 0x5D, 0x5C, 0x72,

0x69, 0x4F, 0xFB, 0xE0, 0xE3, 0x4F, 0x21, 0x06, 0x66, 0x8C, 0x2A, 0x3F, 0x4E, 0xAF, 0x23, 0xC5,

0x21, 0x1E, 0x09, 0xFA, 0xD8, 0x36, 0x94, 0x80, 0x21, 0x16, 0x06, 0x26, 0x74, 0x77, 0x4A, 0xB2,

0x1D, 0x7D, 0xD0, 0x23, 0x3D, 0x82, 0x69, 0x14, 0xAB, 0x91, 0x5A, 0xD9, 0xA5, 0x96, 0x78, 0x4F,

0xB0, 0x44, 0x7D, 0xD2, 0x23, 0x5E, 0x5D, 0xCA, 0x86, 0x1B, 0x2D, 0x98, 0x09, 0xCC, 0xC3, 0x5D,

0x8A, 0x03, 0xC0, 0xD1, 0xD0, 0x0A, 0xD2, 0x0F, 0x0B, 0x7A, 0x64, 0x38, 0x0B, 0x04, 0x3E, 0x1E,

0xD0, 0x3B, 0x9C, 0xA5, 0xE9, 0x2D, 0x67, 0x92, 0xC4, 0x67, 0x41, 0x37, 0x5A, 0xE9, 0x01, 0xE6,

0xC6, 0x51, 0x39, 0x32, 0xFA, 0x2B, 0x99, 0xAE, 0x18, 0x53, 0x89, 0x0C, 0xD2, 0xD0, 0x5A, 0x46,

0xDB, 0xB8, 0xD6, 0x61, 0xE4, 0x75, 0xE3, 0x48, 0xF3, 0x42, 0xD7, 0x3D, 0x6A, 0x5C, 0xEB, 0xFF,

0xFE, 0x7B, 0x5D, 0xE7, 0x4E, 0xD0, 0xA9, 0xCF, 0x10, 0x6F, 0x34, 0x1A, 0xD2, 0x15, 0x2E, 0xC1,

0x90, 0x29, 0xF6, 0x46, 0xBB, 0x71, 0x74, 0xD4, 0x36, 0xD2, 0xEB, 0x46, 0x5B, 0x37, 0xC5, 0x7D,

0x01, 0xB4, 0x11, 0x7F, 0x43, 0xAF, 0x71, 0xFD, 0xF8, 0x71, 0xE7, 0xA8, 0xD1, 0x68, 0x5F, 0x72,

0x17, 0x33, 0x8F, 0xE0, 0x52, 0x2B, 0x20, 0x6C, 0x49, 0xBE, 0xC4, 0xBE, 0x6C, 0x5F, 0x62, 0xED,

0x4E, 0x37, 0x1D, 0xFE, 0x51, 0x40, 0xBD, 0xEC, 0x0D, 0xCD, 0xD1, 0x98, 0x6E, 0x60, 0x2D, 0xD0,

0x81, 0x39, 0xE6, 0x6D, 0x47, 0xB4, 0x0B, 0xC9, 0xAB, 0x2E, 0x19, 0x5A, 0x47, 0x1B, 0xEB, 0x26,

0xE6, 0x1F, 0x85, 0x7C, 0xE1, 0x48, 0x68, 0x60, 0xDE, 0xF6, 0x65, 0x5F, 0xF3, 0x74, 0xB3, 0x07,

0x1F, 0xBA, 0x1E, 0xD5, 0x53, 0x38, 0xC1, 0x1B, 0xFC, 0xC9, 0x8D, 0xF0, 0x58, 0xEA, 0x5F, 0xB9,

0xAE, 0x56, 0x90, 0xC7, 0xBA, 0x0A, 0x7A, 0x09, 0x2A, 0x51, 0x07, 0xF1, 0x68, 0x10, 0x36, 0xA6,

0x9E, 0xE5, 0x12, 0xEB, 0xB6, 0xA1, 0x71, 0xC3, 0x61, 0x08, 0x11, 0x79, 0xE0, 0xF4, 0x35, 0xB5,

0xB1, 0x1E, 0x45, 0x20, 0x9E, 0xF0, 0x3B, 0xE9, 0xA1, 0xD2, 0x7D, 0x3E, 0xC4, 0x3E, 0x98, 0xC6,

0x1C, 0x84, 0x99, 0xF4, 0x68, 0xA5, 0x55, 0xFA, 0x18, 0xF0, 0x20, 0x8C, 0x96, 0x90, 0xDC, 0x27,

0x5A, 0xBE, 0xC6, 0x66, 0x64, 0x6C, 0x83, 0x50, 0x44, 0x03, 0x50, 0xFE, 0x6E, 0x83, 0xBE, 0xFF,

0x18, 0x47, 0x15, 0xEE, 0xBA, 0x7A, 0xEC, 0x9D, 0x1F, 0x67, 0xEE, 0x0B, 0x75, 0xAA, 0xE3, 0x62,

0xDE, 0x6C, 0x4D, 0x7E, 0x05, 0xE7, 0x92, 0x99, 0x0B, 0xDC, 0xE4, 0x76, 0x1D, 0xCD, 0x2C, 0xBD,

0x02, 0xB5, 0xBB, 0x9A, 0x3A, 0xAD, 0x84, 0x40, 0x36, 0x58, 0x4D, 0x96, 0x2B, 0x75, 0x40, 0xEA,

0xAD, 0x26, 0xCD, 0x14, 0x32, 0x20, 0xA4, 0xAB, 0x09, 0xB3, 0xE9, 0x1B, 0x28, 0x87, 0x12, 0xAC,

0x11, 0xF1, 0x6C, 0x3A, 0x82, 0x98, 0xA6, 0x43, 0x0D, 0x44, 0x2A, 0x11, 0x0F, 0x74, 0x78, 0xF1,

0xFB, 0xAB, 0x97, 0x8D, 0x42, 0xB6, 0xC0, 0x16, 0x22, 0xE3, 0x93, 0x1C, 0xF0, 0xB1, 0xC4, 0xF3,

0x38, 0x87, 0xF2, 0xE7, 0x82, 0x79, 0x5E, 0x29, 0x70, 0x40, 0x39, 0xC5, 0x07, 0xF0, 0xC1, 0xDB,

0x05, 0x0E, 0x74, 0x98, 0x32, 0xA8, 0xBB, 0x79, 0x37, 0xE1, 0xF3, 0xCD, 0x98, 0x41, 0xE6, 0x42,

0x43, 0x80, 0x1F, 0x5F, 0xBE, 0xB7, 0xBA, 0x90, 0xAD, 0xAE, 0x11, 0xC3, 0x25, 0x8F, 0x8E, 0xC0,

0x0D, 0x24, 0xE7, 0xC8, 0xA0, 0x8B, 0xE3, 0xB1, 0xB8, 0x31, 0xC8, 0xDF, 0x90, 0xB0, 0xB6, 0xF2,

0xD3, 0x83, 0xB3, 0x67, 0x44, 0xAB, 0xB7, 0x2E, 0x61, 0xB8, 0xF9, 0x09, 0xB8, 0x1B, 0x5E, 0x7E,

0x74, 0x17, 0x82, 0x20, 0x32, 0xB6, 0xF2, 0xB3, 0x34, 0x16, 0xFA, 0x3C, 0xE1, 0x0B, 0x76, 0x3C,

0xB6, 0x53, 0x4F, 0xF3, 0x57, 0x83, 0xC3, 0xE3, 0x5B, 0x37, 0x82, 0x7B, 0x09, 0x32, 0x3F, 0xD7,

0x01, 0x2D, 0xBB, 0xC7, 0xC9, 0xE6, 0x7F, 0x4C, 0x2A, 0xE8, 0x75, 0x3F, 0x2F, 0x17, 0xA8, 0xE9,

0xEB, 0x86, 0x9F, 0x56, 0xAC, 0x15, 0x19, 0x25, 0x8A, 0x25, 0x0F, 0xEF, 0x11, 0x0C, 0x73, 0xC9,

0xEF, 0xEE, 0x25, 0xC8, 0xFE, 0x50, 0x0F, 0xB2, 0x84, 0x0B, 0xB2, 0x84, 0xBA, 0x11, 0xA6, 0xB2,

0xA4, 0x69, 0x2F, 0x99, 0x7D, 0x74, 0x0F, 0xF3, 0x24, 0xE1, 0xE9, 0xC6, 0x78, 0x35, 0x55, 0xEE,

0xBD, 0x3B, 0x10, 0x60, 0xB4, 0x20, 0xC0, 0x48, 0x37, 0x46, 0xA9, 0x00, 0x69, 0xCA, 0x4C, 0x04,

0x98, 0xAC, 0x09, 0x3F, 0xB9, 0xA1, 0x02, 0x19, 0x3E, 0xAF, 0x21, 0x9C, 0x25, 0x5F, 0xDD, 0xB8,

0xBA, 0x87, 0x36, 0x39, 0x38, 0x08, 0xB2, 0x5E, 0x2D, 0xC8, 0x7A, 0xA5, 0x1B, 0xA7, 0x17, 0x57,

0xB2, 0x90, 0x40, 0xF2, 0x26, 0xDA, 0x84, 0x67, 0x34, 0x83, 0x68, 0x9F, 0xF9, 0x37, 0x38, 0xEF,

0x64, 0x6E, 0x48, 0x9C, 0x57, 0xD3, 0x41, 0x97, 0x1A, 0x72, 0xB1, 0xCF, 0xB4, 0xC2, 0x5B, 0x17,

0xC3, 0x2A, 0x23, 0x7E, 0x95, 0x4F, 0x69, 0xFF, 0xFA, 0x4C, 0xA1, 0xBE, 0x22, 0x8E, 0xAD, 0x2B,

0x7E, 0x7A, 0xEC, 0x51, 0x91, 0x27, 0x93, 0x15, 0xCC, 0xFF, 0xAF, 0x07, 0x70, 0x29, 0x85, 0xF5,

0x49, 0xA0, 0x38, 0x98, 0x1F, 0x0A, 0xC0, 0x47, 0x1C, 0x7B, 0x4A, 0x6C, 0x25, 0x96, 0x42, 0x37,

0xF9, 0x95, 0xD6, 0xD5, 0x26, 0xBA, 0x71, 0x34, 0x49, 0x2C, 0x0A, 0x52, 0xF2, 0xDA, 0x92, 0x8A,

0x08, 0x32, 0x7E, 0x3E, 0x88, 0x8C, 0x9F, 0x73, 0x32, 0x7E, 0x06, 0xC0, 0x66, 0x11, 0xD0, 0x97,

0x12, 0x82, 0x1A, 0x65, 0x3D, 0xAE, 0x85, 0x50, 0xBA, 0xEA, 0xD9, 0x65, 0x66, 0xBC, 0xA8, 0x94,

0x57, 0xF2, 0x0C, 0xF0, 0xC5, 0xB1, 0xFC, 0xFF, 0xCB, 0xFE, 0x0F, 0x86, 0xED, 0x24, 0xF8, 0xD7,

0x4C, 0x00, 0x00

};