ДОДАТОК А

Лістинг програми з файлу формата .ino для ESP-32-CAM

```
const char* ssid = "Tenda_E24A88";
const char* password = "55555555";

#include "esp_wifi.h"
#include "esp_camera.h"
#include <WiFi.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"

#include "esp32-hal-ledc.h"

#define 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

void startCameraServer();

// Forward motor
#define pinServoF  12

// Up motor
#define pinServoUp 2

// Control servomotor
#define pinServoC  15

// Sensor
#define pinSensor  14

void initMotors()
{

  ledcSetup(3, 50, 16); // 50 hz PWM, 16-bit resolution Forward motor
  ledcSetup(4, 50, 16); // 50 hz PWM, 16-bit resolution Up motor

  ledcAttachPin(pinServoF, 3);
  ledcAttachPin(pinServoUp, 4);

  ledcWrite(3, 6554);
```

```
  ledcWrite(4, 6554);
  delay(2000);

  ledcWrite(3, 3277);
  ledcWrite(4, 3277);
  delay(2000);
}

void initSensor()
{
  ledcSetup(5, 5000, 8);
  ledcAttachPin(pinSensor, 5);
}

void initServo()
{
  ledcSetup(8, 50, 16); // 50 hz PWM, 16-bit resolution, range from 3250 to 6500.
  ledcAttachPin(pinServoC, 8);
}

void setup(){
   WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);

  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  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;

if(psramFound()){
  config.frame_size = FRAMESIZE_QVGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_QVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}
```

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

sensor_t * s = esp_camera_sensor_get();
s->set_framesize(s, FRAMESIZE_QVGA);
s->set_vflip(s, 1);
s->set_hmirror(s, 1);

initMotors();
initSensor();
initServo();

ledcSetup(7, 5000, 8);
ledcAttachPin(4, 7);  //pin4 is LED

Serial.println("ssid: " + (String)ssid);
Serial.println("password: " + (String)password);

WiFi.begin(ssid, password);
delay(500);

long int StartTime=millis();
while (WiFi.status() != WL_CONNECTED)
{
    delay(500);
```

```
    if ((StartTime+10000) < millis()) break;
  }

  startCameraServer();

  if (WiFi.status() == WL_CONNECTED)
  {
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.print("Camera Ready! Use 'http://");
    Serial.print(WiFi.localIP());
    Serial.println("' to connect");
  } else {
    Serial.println("");
    Serial.println("WiFi disconnected");
    Serial.print("Camera Ready! Use 'http://");
    Serial.print(WiFi.softAPIP());
    Serial.println("' to connect");
    char* apssid = "ESP32-CAM";
    char* appassword = "12345678";
    WiFi.softAP((WiFi.softAPIP().toString()+"_"+(String)apssid).c_str(),
appassword);
  }

  for (int i=0;i<5;i++)
  {
    ledcWrite(7,10);  // flash led
    delay(50);
    ledcWrite(7,0);
```

```
    delay(50);
  }
}
void loop() {
  delay(1000);
  Serial.printf("RSSi: %ld dBm\n", WiFi.RSSI());
}
```

ДОДАТОК Б

Лістинг програми з файлу формата .cpp для ESP-32-CAM

```
#include "dl_lib_matrix3d.h"
#include <esp32-hal-ledc.h>

#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "img_converters.h"
#include "Arduino.h"

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";

httpd_handle_t stream_httpd = NULL;
httpd_handle_t camera_httpd = NULL;
```

```
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;
  {
    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;
  }

  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();
  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;

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.println("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.println("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;
    Serial.printf("MJPG: %uB %ums (%.1ffps)\n",
                  (uint32_t)(_jpg_buf_len),
                  (uint32_t)frame_time, 1000.0 / (uint32_t)frame_time
                 );
  }

  last_frame = 0;
  return res;
}

enum state {fwd, rev, stp};
state actstate = stp;

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")) {
  Serial.println("framesize");
```

```
    if (s->pixformat == PIXFORMAT_JPEG) res = s->set_framesize(s,
(framesize_t)val);
  } else if (!strcmp(variable, "quality")) {
    Serial.println("quality");
    res = s->set_quality(s, val);
  }     else if (!strcmp(variable, "flash")) {
    ledcWrite(7, val);
  }  else if (!strcmp(variable, "servo")) {
    ledcWrite(8, 10 * val);
  }  else if (!strcmp(variable, "speedF"))  {
    ledcWrite(3, val);
  }  else if (!strcmp(variable, "speedUp")) {
    ledcWrite(4, val);
  }  else if (!strcmp(variable, "car")) {
    if (val == 6) {
      Serial.println("Restart");
      ESP.restart();
    }
  }  else  {
    Serial.println("variable");
    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++ = '}';
  *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 pwm_handler(httpd_req_t *req) {
    char pwmStr[10];
    snprintf(pwmStr, sizeof(pwmStr), "%d", ledcRead(5));

    httpd_resp_set_type(req, "text/plain");
    httpd_resp_send(req, pwmStr, strlen(pwmStr));
    return ESP_OK;
}
void updatePWMValue(httpd_req_t *req) {
    char pwmStr[10];
    snprintf(pwmStr, sizeof(pwmStr), "%d", ledcRead(5));
```

```
    String script = "document.getElementById('pwm').innerText = '" + String(pwmStr)
+ "';";
    httpd_resp_set_hdr(req, "Content-Type", "text/javascript");
    httpd_resp_send(req, script.c_str(), strlen(script.c_str()));
}
static const char PROGMEM INDEX_HTML[] = R"rawliteral(
<!doctype html>
<html>
<head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,initial-scale=1">
        <title>Control panel</title>
</head>
  <body>
    <figure>
      <div id="stream-container" class="image-container hidden">
        <div class="close" id="close-stream">×</div>
        <img id="stream" src="">
      </div>
    </figure>
    <section class="main">
    <section id="buttons">
      <table>
        <tr>
          <td align="center"><button id="get-still">Photo</button></td>
          <td align="center"><button id="restart"
onclick="fetch(document.location.origin+'/control?var=car&val=6');">Restart</butto
n></td>
          <td align="center"><button id="toggle-stream">Start Stream</button></td>
```

```
    </tr>
    <tr>
      <td>L/R</td>
      <td align="center" colspan="2"><input type="range" id="servo" min="350"
max="650" value="487"
onchange="try{fetch(document.location.origin+'/control?var=servo&val='+this.value
);}catch(e){}"></td>
    </tr>
    <tr>
      <td>Flash</td>
      <td align="center" colspan="2"><input type="range" id="flash" min="0"
max="255" value="0"
onchange="try{fetch(document.location.origin+'/control?var=flash&val='+this.value)
;}catch(e){}"></td>
    </tr>
    <tr>
      <td>Speed_F</td>
      <td align="center" colspan="2"><input type="range" id="speedF"
min="3277" max="6554" value="3277"
onchange="try{fetch(document.location.origin+'/control?var=speedF&val='+this.val
ue);}catch(e){}"></td>
    </tr>
     <tr>
      <td>Speed_Up</td>
      <td align="center" colspan="2"><input type="range" id="speedUp"
min="3277" max="6554" value="3277"
onchange="try{fetch(document.location.origin+'/control?var=speedUp&val='+this.va
lue);}catch(e){}"></td>
    </tr>
```

```
        <tr>
          <td>Quality</td>
          <td align="center" colspan="2"><input type="range" id="quality" min="10"
max="63" value="10"
onchange="try{fetch(document.location.origin+'/control?var=quality&val='+this.val
ue);}catch(e){}"></td>
        </tr>
        <tr>
          <td>Resolution</td>
          <td align="center" colspan="2"><input type="range" id="framesize" min="0"
max="6" value="5"
onchange="try{fetch(document.location.origin+'/control?var=framesize&val='+this.v
alue);}catch(e){}"></td>
        </tr>
        <tr>
          <td></td>
          <td align="center"><span id="pwm">0</span></td>
                <td></td>
        </tr>
     </table>
   </section>
   </section>
    <script>

    function updatePWM() {
      fetch('/pwm')
        .then(response => response.text())
        .then(data => {
          document.getElementById('pwm').innerText = data;
```

```
        });
    }
    etInterval(updatePWM, 500);

document.addEventListener('DOMContentLoaded', function() {
    function b(B) {
        let C;
        switch (B.type) {
            case 'checkbox':
                C = B.checked ? 1 : 0;
                break;
            case 'range':
            case 'select-one':
                C = B.value;
                break;
            case 'button':
            case 'submit':
                C = '1';
                break;
            default:
                return;
        }
        const D = `${c}/control?var=${B.id}&val=${C}`;
        fetch(D).then(E => {
            console.log(`request to ${D} finished, status: ${E.status}`)
        })
    }
    var c = document.location.origin;
    const e = B => {
```

```
            B.classList.add('hidden')
        },
        f = B => {
            B.classList.remove('hidden')
        },
        g = B => {
            B.classList.add('disabled'), B.disabled = !0
        },
        h = B => {
            B.classList.remove('disabled'), B.disabled = !1
        },
        i = (B, C, D) => {
            D = !(null != D) || D;
            let E;
            'checkbox' === B.type ? (E = B.checked, C = !!C, B.checked = C) : (E =
B.value, B.value = C), D && E !== C ? b(B) : !D && ('aec' === B.id ? C ? e(v) : f(v)
: 'agc' === B.id ? C ? (f(t), e(s)) : (e(t), f(s)) : 'awb_gain' === B.id ? C ? f(x) : e(x) :
'face_recognize' === B.id && (C ? h(n) : g(n)))
        };
    document.querySelectorAll('.close').forEach(B => {
        B.onclick = () => {
            e(B.parentNode)
        }
    }), fetch(`${c}/status`).then(function(B) {
        return B.json()
    }).then(function(B) {
        document.querySelectorAll('.default-action').forEach(C => {
            i(C, B[C.id], !1)
        })
```

```
});
const j = document.getElementById('stream'),
    k = document.getElementById('stream-container'),
    l = document.getElementById('get-still'),
    m = document.getElementById('toggle-stream'),
    n = document.getElementById('face_enroll'),
    o = document.getElementById('close-stream'),
    p = () => {
        window.stop(), m.innerHTML = 'Start Stream'
    },
    q = () => {
        j.src = `${c+':81'}/stream`, f(k), m.innerHTML = 'Stop Stream'
    };
l.onclick = () => {
    p(), j.src = `${c}/capture?_cb=${Date.now()}`, f(k)
}, o.onclick = () => {
    p(), e(k)
}, m.onclick = () => {
    const B = 'Stop Stream' === m.innerHTML;
    B ? p() : q()
}, n.onclick = () => {
    b(n)
}, document.querySelectorAll('.default-action').forEach(B => {
    B.onchange = () => b(B)
});
const r = document.getElementById('agc'),
    s = document.getElementById('agc_gain-group'),
    t = document.getElementById('gainceiling-group');
r.onchange = () => {
```

```
      b(r), r.checked ? (f(t), e(s)) : (e(t), f(s))
    };
    const u = document.getElementById('aec'),
      v = document.getElementById('aec_value-group');
    u.onchange = () => {
      b(u), u.checked ? e(v) : f(v)
    };
    const w = document.getElementById('awb_gain'),
      x = document.getElementById('wb_mode-group');
    w.onchange = () => {
      b(w), w.checked ? f(x) : e(x)
    };
      A = document.getElementById('framesize');
    A.onchange = () => {
      b(A), 5 < A.value && (i(y, !1), i(z, !1))
    }, y.onchange = () => {
      return 5 < A.value ? (alert('Please select CIF or lower resolution before enabling this feature!'), void i(y, !1)) : void(b(y), !y.checked && (g(n), i(z, !1)))
    }, z.onchange = () => {
      return 5 < A.value ? (alert('Please select CIF or lower resolution before enabling this feature!'), void i(z, !1)) : void(b(z), z.checked ? (h(n), i(y, !0)) : g(n))
    }
});
});
    </script>
  </body>
</html>
)rawliteral";
```

```
static esp_err_t index_handler(httpd_req_t *req){
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, (const char *)INDEX_HTML,
strlen(INDEX_HTML));
}

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
};

httpd_uri_t pwm_uri = {
        .uri       = "/pwm",
        .method    = HTTP_GET,
        .handler   = pwm_handler,
        .user_ctx  = NULL
    };

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);
```

```
    httpd_register_uri_handler(camera_httpd, &pwm_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);
  }
}
```

ДОДАТОК В

Демонстраційні матеріали у вигляді презентацій

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК.210315.ххх ПЗ	Пояснювальна записка	А4, 66 с.
		Додаткові матеріали	
2		Демонстраційні матеріал у	А4, 11 с.
		вигляді презентації	

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК.210315.ххх ВД			
Розроб.		Кузьменко О.С.			Розробка мінно-пошукового мобільного робота на повітряній подушці	Літера	Аркуш	Аркушів
Перевір.		Євсєєв В.В.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Стародубцев М.Г.						
Затв.		Невлюдов І.Ш.						