Инженерно-техническая разработка в сфере БАС: Проектирование топологии сети БПЛА

Техническая документация

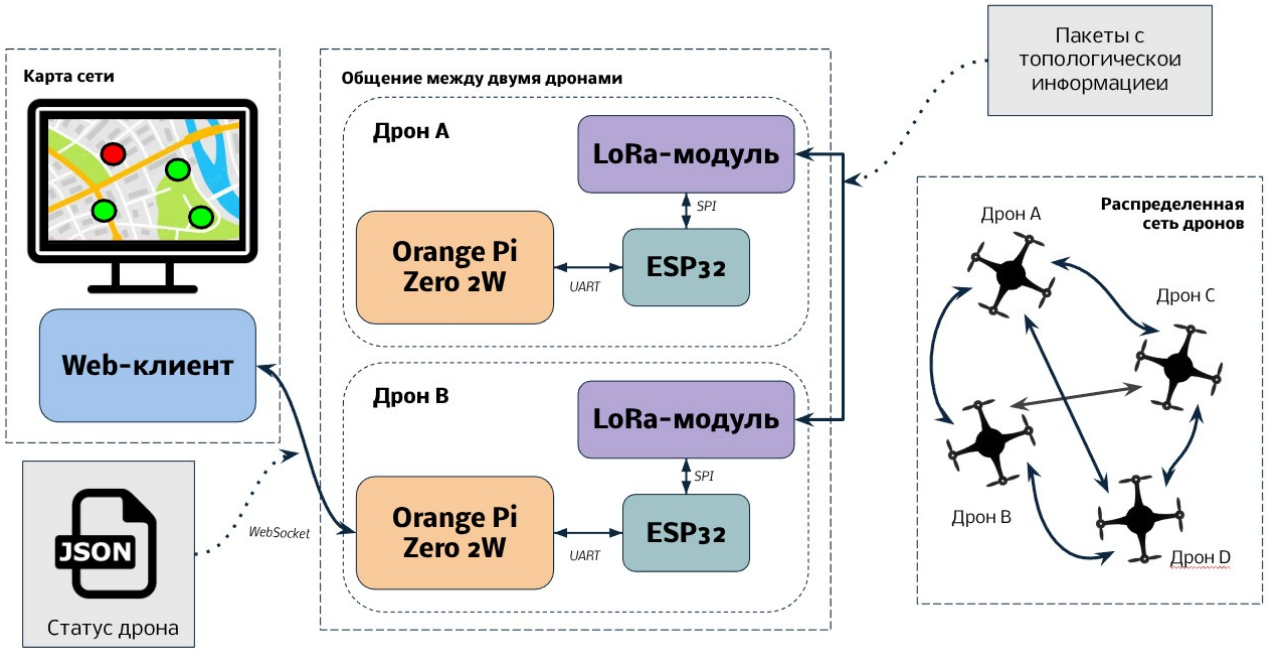
Российский новый университет (РосНОУ)
Команда: Мифические барсы

Содержание

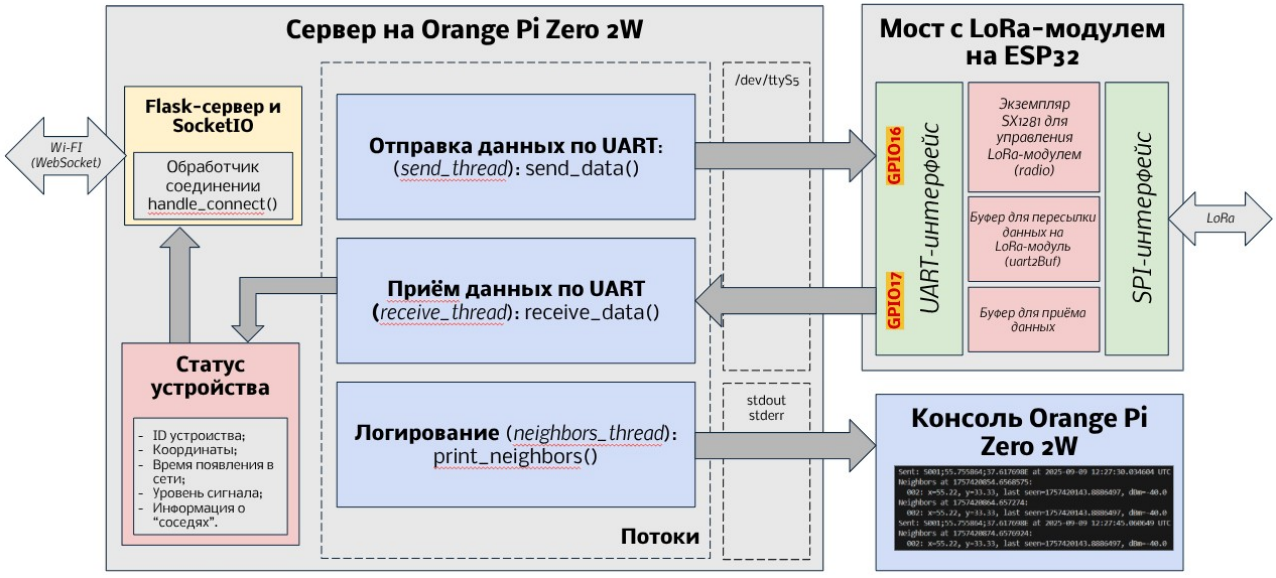
1. Архитектура решения

Готовое решение представляет собой ПО для организации распределённой сети БПЛА с использованием LoRa-модулей для передачи данных. В данном прототипе в качестве вычислительного узла применяется Orange Pi Zero 2W, что позволяет каждому устройству в сети принимать и отправлять топологические данные. На их основе формируется карта сети, доступная через веб-клиент.

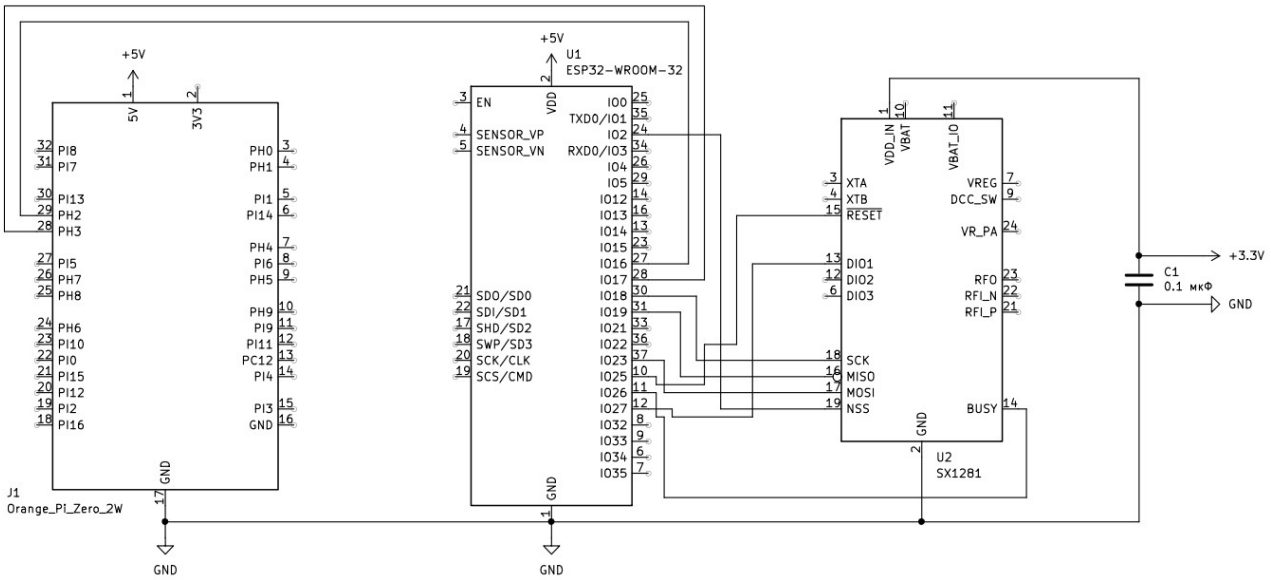
1.1. Общая функциональная схема



1.2. Функциональная схема сервера



1.3. Принципиальная схема



1.4. Программный код

1.4.1. Сервер на Orange Pi Zero 2W (orange_pi.py)

```
import serial
import threading
import time
import fcntl
import os
import signal
import sys
from typing import Dict, List
from flask import Flask
from flask_socketio import SocketIO
import json

# Параметры UART
SERIAL_PORT = '/dev/ttyS5'
BAUD_RATE = 115200
SEND_INTERVAL = 1.0
LOCK_FILE = "/tmp/drone_communication.lock"

# Параметры сервера
HOST = '0.0.0.0'
PORT = 5000

# Данные текущего дрона
DRONE_ID = "001"
drone_data = {
    "id": DRONE_ID,
    "x": 55.755864,
    "y": 37.617698
}

# Список для хранения данных соседних дронов
neighbors: Dict[str, Dict] = {}

# Флаг для завершения потоков
running = True
```

```
# Путь к файлу JSON
try:
    SCRIPT_DIR = os.path.dirname(os.path.abspath(__file__))
except NameError:
    SCRIPT_DIR = os.getcwd()  # Fallback to current working directory
JSON_FILE = os.path.join(SCRIPT_DIR, "drone_data.json")

# Инициализация Flask и SocketIO
app = Flask(__name__)
socketio = SocketIO(app, cors_allowed_origins="*")

# Проверка и установка блокировки
lock_fd = None
try:
    lock_fd = open(LOCK_FILE, 'w')
    fcntl.flock(lock_fd, fcntl.LOCK_EX | fcntl.LOCK_NB)
except IOError:
    print("Another instance of the script is already running. Exiting.")
    sys.exit(1)

# Инициализация UART
ser = None
try:
    ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=0.2)
    ser.setDTR(False)
    ser.setRTS(False)
    ser.flushInput()
    ser.flushOutput()
    print(f"Opened serial port: {ser.name}")
except serial.SerialException as e:
    print(f"Failed to open serial port: {e}")
    if lock_fd:
        fcntl.flock(lock_fd, fcntl.LOCK_UN)
        lock_fd.close()
    sys.exit(1)

# Функция для сохранения данных в JSON файл
def save_drone_data():
    data = get_drone_data()
    try:
        with open(JSON_FILE, 'w') as f:
            json.dump(data, f, indent=4)
        print(f"Saved drone data to {JSON_FILE} at {time.time()}")
    except Exception as e:
        print(f"Error saving JSON: {e}")

# Функция для отправки данных через UART
def send_data():
    while running:
        try:
            ser.flushOutput()
            message = f"S{drone_data['id']};{drone_data['x']};{drone_data['y']}E"
            ser.write(message.encode('utf-8'))
            print(f"Sent: {message} at {time.time()}")
            time.sleep(SEND_INTERVAL)
            time.sleep(0.01)
        except serial.SerialException as e:
            print(f"Error sending data: {e} at {time.time()}")
            time.sleep(1)

# Функция для приёма данных через UART
def receive_data():
    buffer = ""
```

```
    processed_messages = set()
    while running:
        try:
            if ser.in_waiting > 0:
                data = ser.read(ser.in_waiting).decode('utf-8', errors='ignore')
                buffer += data
                print(f"Raw data received: {data} at {time.time()}")

                while True:
                    start_idx = buffer.find('S')
                    end_idx = buffer.find('E', start_idx + 1)

                    if start_idx == -1 or end_idx == -1:
                        break

                    message = buffer[start_idx:end_idx + 1]
                    buffer = buffer[end_idx + 1:]

                    # Для избежания дубликатов используем хеш без dBm, если он есть
                    content = message[1:-1]
                    parts_for_hash = content.split(';')[:3]  # Берем только первые 3 части для хеша
                    message_for_hash = 'S' + ';'.join(parts_for_hash) + 'E'
                    message_hash = hash(message_for_hash)
                    if message_hash in processed_messages:
                        print(f"Duplicate message skipped: {message} at {time.time()}")
                        continue
                    processed_messages.add(message_hash)

                    if message.startswith('S') and message.endswith('E'):
                        try:
                            parts = content.split(';')
                            if len(parts) >= 3:
                                drone_id = parts[0]
                                x = float(parts[1])
                                y = float(parts[2])
                                dBm = None
                                if len(parts) > 3:
                                    try:
                                        dBm = float(parts[3])  # Сохраняем dBm, если нужно
                                        print(f"Received dBm from {drone_id}: {dBm}")
                                    except ValueError:
                                        print(f"Invalid dBm in: {message}")

                                if drone_id != DRONE_ID:
                                    neighbors[drone_id] = {
                                        "id": drone_id,
                                        "x": x,
                                        "y": y,
                                        "timestamp": time.time(),
                                        "dBm": dBm  # Добавляем поле для dBm, если оно есть
                                    }
                                    print(f"Received from {drone_id}: {neighbors[drone_id]} at
{time.time()}")
                                    socketio.emit('drone_data', get_drone_data())
                                    save_drone_data()  # Сохраняем JSON при получении новой
информации
                                else:
                                    print(f"Ignored own message: {message} at {time.time()}")
                            else:
                                print(f"Invalid format: {message} at {time.time()}")
                        except ValueError:
                            print(f"Invalid coordinates in: {message} at {time.time()}")
                    else:
                        print(f"Invalid message received: {message} at {time.time()}")
```

```
            else:
                # Отладка: сообщаем, если не поступают данные
                time.sleep(0.1)
                # print(f"No data in UART buffer at {time.time()}")  # Раскомментируйте для отладки

        except serial.SerialException as e:
            print(f"Error receiving data: {e} at {time.time()}")
            time.sleep(1)

# Функция для формирования JSON с данными дронов
def get_drone_data():
    return {
        "self": drone_data,
        "neighbors": list(neighbors.values())
    }

# Функция для отображения списка соседей (консоль)
def print_neighbors():
    while running:
        print(f"Neighbors at {time.time()}:")
        if neighbors:
            for drone_id, data in neighbors.items():
                dBm_str = f", dBm={data.get('dBm', 'N/A')}" if data.get('dBm') is not None else ""
                print(f"  {drone_id}: x={data['x']}, y={data['y']}, last seen={data['timestamp']}
{dBm_str}")
        else:
            print("  No neighbors detected.")
        time.sleep(10)

# WebSocket: отправка данных при подключении клиента
@socketio.on('connect')
def handle_connect():
    print(f"Client connected at {time.time()}")
    socketio.emit('drone_data', get_drone_data())

# Обработчик сигнала для корректного завершения
def signal_handler(sig, frame):
    global running
    print("\nShutting down...")
    running = False
    if ser:
        ser.close()
        print("Serial port closed.")
    if lock_fd:
        fcntl.flock(lock_fd, fcntl.LOCK_UN)
        lock_fd.close()
        if os.path.exists(LOCK_FILE):
            os.remove(LOCK_FILE)
    sys.exit(0)

# Регистрируем обработчик сигналов
signal.signal(signal.SIGINT, signal_handler)
signal.signal(signal.SIGTERM, signal_handler)

# Запуск потоков
try:
    send_thread = threading.Thread(target=send_data, daemon=True)
    receive_thread = threading.Thread(target=receive_data, daemon=True)
    neighbors_thread = threading.Thread(target=print_neighbors, daemon=True)

    send_thread.start()
    receive_thread.start()
```

```
    neighbors_thread.start()

    # Запуск Flask-SocketIO сервера
    print(f"Starting WebSocket server on http://{HOST}:{PORT}")
    socketio.run(app, host=HOST, port=PORT, allow_unsafe_werkzeug=True)

except KeyboardInterrupt:
    signal_handler(signal.SIGINT, None)

except Exception as e:
    print(f"Unexpected error: {e}")
    signal_handler(signal.SIGTERM, None)
```

1.4.2. Прошивка на ESP32 (esp32_code.ino)

```
#include <RadioLib.h>

// ====== Радио (SX1281) пины ======
#define CS_PIN    2   // NSS_CTS
#define IRQ_PIN   27  // DIO1
#define RST_PIN   25  // NRESET
#define BUSY_PIN  26  // BUSY

SX1281 radio = new Module(CS_PIN, IRQ_PIN, RST_PIN, BUSY_PIN);

// ====== UART2 (поменяй пины если нужно) ======
#define UART2_RX_PIN 16
#define UART2_TX_PIN 17
#define UART2_BAUD   115200

String uart2Buf = "";

void setup() {
  Serial.begin(115200);
  Serial.println(F("[SYS] start"));

  Serial2.begin(UART2_BAUD, SERIAL_8N1, UART2_RX_PIN, UART2_TX_PIN);
  Serial.println(F("[UART2] started"));

  Serial.print(F("[SX1281] Initializing ... "));
  int state = radio.begin();
  if (state == RADIOLIB_ERR_NONE) {
    Serial.println(F("success!"));
  } else {
    Serial.print(F("failed, code "));
    Serial.println(state);
    while (true) { delay(1000); }
  }

  Serial.println(F("[SX1281] Ready."));
}

void loop() {
  // ---------- 1) Обработка входящего с UART2, пересылка на радио ----------
  while (Serial2.available()) {
    char c = (char)Serial2.read();
    if (c == '\r') continue;
    uart2Buf += c;
    if (c == '\n' || c == 'E') {
      uart2Buf.trim();
      if (uart2Buf.length() > 0) {
        Serial.print(F("[UART2 -> RADIO] Sending: "));
```

```
      Serial.println(uart2Buf);
      int txState = radio.transmit(uart2Buf);
      if (txState == RADIOLIB_ERR_NONE) {
        Serial.println(F("[RADIO TX] success"));
      } else {
        Serial.print(F("[RADIO TX] failed, code "));
        Serial.println(txState);
      }
    }
    uart2Buf = "";
  }
}

// 2) Приём с радио — ищем все S...E и отправляем по UART2 с добавлением ;dBm перед E
String rxData;
// Таймаут при receive — подбери под своё приложение (здесь 500 ms)
int rxState = radio.receive(rxData, 500);

if (rxState == RADIOLIB_ERR_NONE) {
  rxData.trim();
  Serial.print(F("[RADIO RX] raw: "));
  Serial.println(rxData);

  // Получим RSSI для принятого пакета (будем использовать для всех найденных S..E)
  float rssi_f = radio.getRSSI();
  int rssi_int = (int)rssi_f; // целое dBm

  int start = rxData.indexOf('S');
  while (start >= 0) {
    int end = rxData.indexOf('E', start + 1);
    if (end < 0) break; // нет завершающей E — выходим

    String packet = rxData.substring(start, end + 1); // S...E включительно

    // Лог
    Serial.print(F("[RADIO] Found S..E: "));
    Serial.println(packet);

    // Вставляем ;<dBm> прямо перед конечной 'E'
    if (packet.length() >= 2) {
      String withoutE = packet.substring(0, packet.length() - 1); // без 'E'
      String out = withoutE + ";" + String(rssi_int) + "E";

      // Отправляем по UART2 без перевода строки
      Serial.print(F("[RADIO -> UART2] sending with dBm: "));
      Serial.println(out);
      Serial2.print(out); // <-- НЕТ '\n' в конце, как просили
    } else {
      Serial.println(F("[RADIO] Packet too short, ignored."));
    }

    // продолжаем искать следующий S после текущего end
    start = rxData.indexOf('S', end + 1);
  }

} else if (rxState == RADIOLIB_ERR_RX_TIMEOUT) {
  // нормально — пакетов не было
} else {
  Serial.print(F("[RADIO RX] error code "));
  Serial.println(rxState);
}

delay(10);
```

```
}
```

1.4.3. Веб-клиент

1.4.3.1. HTML-страница (client.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Drone Mesh Dashboard</title>
    <style>
        body {
            margin: 0;
            font-family: Arial, sans-serif;
            background: #1a1a1a;
            color: #fff;
        }
        #dashboard {
            display: flex;
            height: 100vh;
        }
        #map-container {
            flex: 1;
            position: relative;
            background: #2a2a2a;
            overflow: hidden;
            border: 2px solid #444;
        }
        #map-canvas {
            width: 100%;
            height: 100%;
            cursor: grab;
            background: #1e3a5f;
        }
        #map-canvas:active {
            cursor: grabbing;
        }
        #table-container {
            width: 450px;
            background: #2a2a2a;
            border-left: 2px solid #444;
            padding: 20px;
            box-sizing: border-box;
            overflow-y: auto;
        }
        h2 {
            margin-top: 0;
            color: #4CAF50;
            border-bottom: 2px solid #4CAF50;
            padding-bottom: 10px;
        }
        table {
            width: 100%;
            border-collapse: collapse;
            margin-top: 20px;
```

```
}
th, td {
    border: 1px solid #555;
    padding: 8px;
    text-align: left;
}
th {
    background-color: #3a3a3a;
    color: #4CAF50;
}
tr:nth-child(even) {
    background-color: #333;
}
.controls {
    position: absolute;
    top: 10px;
    left: 10px;
    z-index: 100;
    background: rgba(0,0,0,0.8);
    padding: 15px;
    border-radius: 8px;
    border: 1px solid #444;
}
.zoom-btn, .connect-btn {
    background: #4CAF50;
    border: none;
    color: white;
    padding: 8px 12px;
    margin: 2px;
    border-radius: 5px;
    cursor: pointer;
    font-size: 14px;
    font-weight: bold;
}
.zoom-btn:hover, .connect-btn:hover {
    background: #45a049;
}
.zoom-btn:active, .connect-btn:active {
    transform: scale(0.95);
}
.status {
    position: absolute;
    top: 10px;
    right: 10px;
    background: rgba(0,0,0,0.8);
    padding: 15px;
    border-radius: 8px;
    font-size: 14px;
    border: 1px solid #444;
}
.status.connected {
    border-left: 4px solid #4CAF50;
}
.status.disconnected {
    border-left: 4px solid #f44336;
}
```

```
        #ip-selector {
            margin-bottom: 10px;
        }
        #ip-input {
            padding: 5px;
            margin-right: 5px;
            border-radius: 5px;
            border: 1px solid #555;
            background: #333;
            color: #fff;
        }
    </style>
</head>
<body>
    <div id="dashboard">
        <div id="map-container">
            <canvas id="map-canvas"></canvas>
            <div class="controls">
                <div style="color: #ccc; margin-bottom: 5px; font-size: 12px;">Управление</div>
                <button class="zoom-btn" onclick="zoomIn()">Zoom +</button>
                <button class="zoom-btn" onclick="zoomOut()">Zoom -</button>
                <button class="zoom-btn" onclick="resetView()">Reset</button>
                <div style="color: #ccc; margin-top: 5px; font-size: 11px;">
                    Zoom: <span id="zoom-level">10</span>
                </div>
                <div id="ip-selector">
                    <input id="ip-input" type="text" placeholder="Enter Orange Pi IP"
value="192.168.0.104">
                    <button class="connect-btn" onclick="connectToServer()">Connect</button>
                </div>
            </div>
            <div id="connection-status" class="status disconnected">
                Disconnected
            </div>
        </div>
        <div id="table-container">
            <h2>Drone Network Status</h2>
            <table id="drone-table">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Latitude</th>
                        <th>Longitude</th>
                        <th>Last Seen</th>
                        <th>Signal (dBm)</th>
                    </tr>
                </thead>
                <tbody id="drone-table-body"></tbody>
            </table>
        </div>
    </div>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.5.0/socket.io.min.js"></script>
    <script>
        const CONFIG = {
            TILE_PATH: './tiles',
```

```
    TILE_SIZE: 256,
    INITIAL_ZOOM: 10,
    MIN_ZOOM: 1,
    MAX_ZOOM: 15,
    INITIAL_LAT: 55.7558,
    INITIAL_LON: 37.6176
};

let appState = {
    zoom: CONFIG.INITIAL_ZOOM,
    centerLat: CONFIG.INITIAL_LAT,
    centerLon: CONFIG.INITIAL_LON,
    offsetX: 0,
    offsetY: 0,
    isDragging: false,
    lastMouseX: 0,
    lastMouseY: 0,
    tilesLoaded: 0,
    tilesTotal: 0,
    loadedTiles: new Map(),
    drones: { self: null, neighbors: [] },
    socket: null
};

const canvas = document.getElementById('map-canvas');
const ctx = canvas.getContext('2d');
const connectionStatus = document.getElementById('connection-status');
const droneTableBody = document.getElementById('drone-table-body');

function connectToServer() {
    const ip = document.getElementById('ip-input').value;
    if (appState.socket) {
        appState.socket.disconnect();
    }
    console.log(`Attempting to connect to ws://${ip}:5000`);
    appState.socket = io(`ws://${ip}:5000`, { transports: ['websocket'] });
    appState.socket.on('connect', () => {
        console.log('Connected to server');
        connectionStatus.textContent = `Connected to ${ip}`;
        connectionStatus.className = 'status connected';
    });
    appState.socket.on('connect_error', (error) => {
        console.error('WebSocket connection error:', error);
        connectionStatus.textContent = `Connection failed: ${error.message}`;
        connectionStatus.className = 'status disconnected';
    });
    appState.socket.on('disconnect', () => {
        console.log('Disconnected from server');
        connectionStatus.textContent = 'Disconnected';
        connectionStatus.className = 'status disconnected';
    });
    appState.socket.on('drone_data', (data) => {
        console.log('Received drone data:', data);
        appState.drones = data;
        updateDroneTable();
        drawMap();
```

```
            });
        }

        function resizeCanvas() {
            const container = canvas.parentElement;
            canvas.width = container.clientWidth;
            canvas.height = container.clientHeight;
            drawMap();
        }

        function deg2num(lat_deg, lon_deg, zoom) {
            const lat_rad = lat_deg * Math.PI / 180;
            const n = Math.pow(2, zoom);
            const x = Math.floor((lon_deg + 180) / 360 * n);
            const y = Math.floor((1 - Math.asinh(Math.tan(lat_rad)) / Math.PI) / 2 * n);
            return [x, y];
        }

        // Новая функция для конвертации координат в пиксели на экране
        function latLonToPixels(lat, lon) {
            const lat_rad = lat * Math.PI / 180;
            const n = Math.pow(2, appState.zoom);

            // Получаем точные tile координаты (не округленные)
            const tileX = (lon + 180) / 360 * n;
            const tileY = (1 - Math.asinh(Math.tan(lat_rad)) / Math.PI) / 2 * n;

            // Координаты центра карты в tile coordinates
            const centerTileCoords = deg2num(appState.centerLat, appState.centerLon, appState.zoom);

            // Пиксельные координаты на экране
            const pixelX = canvas.width / 2 + (tileX - centerTileCoords[0]) * CONFIG.TILE_SIZE +
appState.offsetX;
            const pixelY = canvas.height / 2 + (tileY - centerTileCoords[1]) * CONFIG.TILE_SIZE +
appState.offsetY;

            return [pixelX, pixelY];
        }

        function loadTile(z, x, y) {
            const tileKey = `${z}-${x}-${y}`;
            if (appState.loadedTiles.has(tileKey)) {
                return appState.loadedTiles.get(tileKey);
            }
            const img = new Image();
            const tileUrl = `${CONFIG.TILE_PATH}/${z}/${x}/${y}.png`;
            img.onload = () => {
                appState.tilesLoaded++;
                drawMap();
            };
            img.onerror = () => {
                console.log(`Tile not found: ${tileUrl}`);
                const canvas = document.createElement('canvas');
                canvas.width = CONFIG.TILE_SIZE;
                canvas.height = CONFIG.TILE_SIZE;
                const ctx = canvas.getContext('2d');
                ctx.fillStyle = '#2a2a2a';
```

```
                ctx.fillRect(0, 0, CONFIG.TILE_SIZE, CONFIG.TILE_SIZE);
                ctx.strokeStyle = '#444';
                ctx.strokeRect(0, 0, CONFIG.TILE_SIZE, CONFIG.TILE_SIZE);
                ctx.fillStyle = '#666';
                ctx.font = '12px Arial';
                ctx.textAlign = 'center';
                ctx.fillText(`${z}/${x}/${y}`, CONFIG.TILE_SIZE/2, CONFIG.TILE_SIZE/2);
                img.src = canvas.toDataURL();
            };
            img.src = tileUrl;
            appState.loadedTiles.set(tileKey, img);
            appState.tilesTotal++;
            return img;
        }

        function drawMap() {
            ctx.fillStyle = '#1e3a5f';
            ctx.fillRect(0, 0, canvas.width, canvas.height);
            const centerTile = deg2num(appState.centerLat, appState.centerLon, appState.zoom);
            const tilesNeeded = Math.ceil(Math.max(canvas.width, canvas.height) / CONFIG.TILE_SIZE)
+ 2;
            for (let dx = -tilesNeeded; dx <= tilesNeeded; dx++) {
                for (let dy = -tilesNeeded; dy <= tilesNeeded; dy++) {
                    const tileX = centerTile[0] + dx;
                    const tileY = centerTile[1] + dy;
                    if (tileX < 0 || tileY < 0 || tileX >= Math.pow(2, appState.zoom) || tileY >=
Math.pow(2, appState.zoom)) {
                        continue;
                    }
                    const img = loadTile(appState.zoom, tileX, tileY);
                    if (img.complete && img.naturalWidth > 0) {
                        const pixelX = canvas.width / 2 + (tileX - centerTile[0]) * CONFIG.TILE_SIZE
+ appState.offsetX;
                        const pixelY = canvas.height / 2 + (tileY - centerTile[1]) *
CONFIG.TILE_SIZE + appState.offsetY;
                        ctx.drawImage(img, pixelX, pixelY, CONFIG.TILE_SIZE, CONFIG.TILE_SIZE);
                    }
                }
            }
            drawDrones();
        }

        function drawDrones() {
            if (!appState.drones.self) return;

            // Рисуем свой дрон (красная точка)
            const selfPixels = latLonToPixels(appState.drones.self.x, appState.drones.self.y);

            // Более приятный размер точки
            const dotSize = Math.max(4, Math.min(10, appState.zoom - 4));

            ctx.fillStyle = '#ff4444';
            ctx.beginPath();
            ctx.arc(selfPixels[0], selfPixels[1], dotSize, 0, 2 * Math.PI);
            ctx.fill();

            // Обводка для лучшей видимости
            ctx.strokeStyle = '#ffffff';
```

```
            ctx.lineWidth = 2;
            ctx.stroke();

            // Черный текст с белой обводкой для лучшей читаемости
            ctx.font = 'bold 12px Arial';
            ctx.textAlign = 'center';
            ctx.strokeStyle = '#ffffff';
            ctx.lineWidth = 3;
            ctx.strokeText(appState.drones.self.id, selfPixels[0], selfPixels[1] - dotSize - 5);
            ctx.fillStyle = '#000000';
            ctx.fillText(appState.drones.self.id, selfPixels[0], selfPixels[1] - dotSize - 5);

            // Рисуем соседей (зеленые точки)
            appState.drones.neighbors.forEach(neighbor => {
                const neighborPixels = latLonToPixels(neighbor.x, neighbor.y);

                ctx.fillStyle = '#44ff44';
                ctx.beginPath();
                ctx.arc(neighborPixels[0], neighborPixels[1], dotSize - 1, 0, 2 * Math.PI);
                ctx.fill();

                // Обводка
                ctx.strokeStyle = '#ffffff';
                ctx.lineWidth = 2;
                ctx.stroke();

                // Черный текст с белой обводкой
                ctx.font = 'bold 12px Arial';
                ctx.textAlign = 'center';
                ctx.strokeStyle = '#ffffff';
                ctx.lineWidth = 3;
                ctx.strokeText(neighbor.id, neighborPixels[0], neighborPixels[1] - dotSize - 5);
                ctx.fillStyle = '#000000';
                ctx.fillText(neighbor.id, neighborPixels[0], neighborPixels[1] - dotSize - 5);
            });
        }

        function updateDroneTable() {
            droneTableBody.innerHTML = '';
            if (appState.drones.self) {
                const row = document.createElement('tr');
                row.innerHTML = `
                    <td>${appState.drones.self.id} (Self)</td>
                    <td>${appState.drones.self.x.toFixed(6)}</td>
                    <td>${appState.drones.self.y.toFixed(6)}</td>
                    <td>-</td>
                    <td>N/A</td>
                `;
                droneTableBody.appendChild(row);
            }
            appState.drones.neighbors.forEach(neighbor => {
                const row = document.createElement('tr');
                const dBmValue = (neighbor.dBm !== undefined && neighbor.dBm !== null) ?
neighbor.dBm.toFixed(1) : 'N/A';
                row.innerHTML = `
                    <td>${neighbor.id}</td>
```

```
                <td>${neighbor.x.toFixed(6)}</td>
                <td>${neighbor.y.toFixed(6)}</td>
                <td>${new Date(neighbor.timestamp * 1000).toLocaleTimeString()}</td>
                <td>${dBmValue}</td>
            `;
            droneTableBody.appendChild(row);
        });
    }

    canvas.addEventListener('mousedown', (e) => {
        appState.isDragging = true;
        appState.lastMouseX = e.clientX;
        appState.lastMouseY = e.clientY;
        canvas.style.cursor = 'grabbing';
    });

    canvas.addEventListener('mousemove', (e) => {
        const rect = canvas.getBoundingClientRect();
        const x = e.clientX - rect.left;
        const y = e.clientY - rect.top;
        if (appState.isDragging) {
            const deltaX = e.clientX - appState.lastMouseX;
            const deltaY = e.clientY - appState.lastMouseY;
            appState.offsetX += deltaX;
            appState.offsetY += deltaY;
            appState.lastMouseX = e.clientX;
            appState.lastMouseY = e.clientY;
            drawMap();
        }
    });

    canvas.addEventListener('mouseup', () => {
        appState.isDragging = false;
        canvas.style.cursor = 'grab';
    });

    canvas.addEventListener('mouseleave', () => {
        appState.isDragging = false;
        canvas.style.cursor = 'grab';
    });

    canvas.addEventListener('wheel', (e) => {
        e.preventDefault();
        const zoomDelta = e.deltaY > 0 ? -1 : 1;
        const newZoom = Math.max(CONFIG.MIN_ZOOM, Math.min(CONFIG.MAX_ZOOM, appState.zoom +
zoomDelta));
        if (newZoom !== appState.zoom) {
            appState.zoom = newZoom;
            appState.loadedTiles.clear();
            appState.tilesLoaded = 0;
            appState.tilesTotal = 0;
            drawMap();
            document.getElementById('zoom-level').textContent = appState.zoom;
        }
    });
```

```
        function zoomIn() {
            if (appState.zoom < CONFIG.MAX_ZOOM) {
                appState.zoom++;
                appState.loadedTiles.clear();
                appState.tilesLoaded = 0;
                appState.tilesTotal = 0;
                drawMap();
                document.getElementById('zoom-level').textContent = appState.zoom;
            }
        }

        function zoomOut() {
            if (appState.zoom > CONFIG.MIN_ZOOM) {
                appState.zoom--;
                appState.loadedTiles.clear();
                appState.tilesLoaded = 0;
                appState.tilesTotal = 0;
                drawMap();
                document.getElementById('zoom-level').textContent = appState.zoom;
            }
        }

        function resetView() {
            appState.offsetX = 0;
            appState.offsetY = 0;
            appState.zoom = CONFIG.INITIAL_ZOOM;
            appState.centerLat = CONFIG.INITIAL_LAT;
            appState.centerLon = CONFIG.INITIAL_LON;
            appState.loadedTiles.clear();
            appState.tilesLoaded = 0;
            appState.tilesTotal = 0;
            drawMap();
            document.getElementById('zoom-level').textContent = appState.zoom;
        }

        window.addEventListener('resize', resizeCanvas);
        resizeCanvas();
        connectToServer(); // Автоподключение при загрузке
        console.log('Drone Dashboard initialized');
    </script>
</body>
</html>
```

1.4.3.2. Скрипт для загрузки тайлов (site/tiles_download.py)

```
#!/usr/bin/env python3
import os
import requests
import time
import math
from urllib.parse import urlparse


class TileDownloader:
    def __init__(self, base_url="https://tile.openstreetmap.org"):
        self.base_url = base_url
        self.session = requests.Session()
        self.session.headers.update({
```

```
            'User-Agent': 'Drone Mesh Dashboard/1.0 (Educational Use)'
        })

    def deg2num(self, lat_deg, lon_deg, zoom):
        """Конвертация координат в номера тайлов"""
        lat_rad = math.radians(lat_deg)
        n = 2.0 ** zoom
        xtile = int((lon_deg + 180.0) / 360.0 * n)
        ytile = int((1.0 - math.asinh(math.tan(lat_rad)) / math.pi) / 2.0 * n)
        return (xtile, ytile)

    def download_tile(self, z, x, y, output_dir="tiles"):
        """Загрузка одного тайла"""
        url = f"{self.base_url}/{z}/{x}/{y}.png"

        # Создание директорий
        tile_dir = os.path.join(output_dir, str(z), str(x))
        os.makedirs(tile_dir, exist_ok=True)

        file_path = os.path.join(tile_dir, f"{y}.png")

        # Если файл уже существует, пропускаем
        if os.path.exists(file_path):
            print(f"Тайл {z}/{x}/{y} уже существует")
            return True

        try:
            response = self.session.get(url, timeout=10)
            response.raise_for_status()

            with open(file_path, 'wb') as f:
                f.write(response.content)

            print(f"Загружен: {z}/{x}/{y}")
            return True

        except requests.exceptions.RequestException as e:
            print(f"Ошибка загрузки {z}/{x}/{y}: {e}")
            return False

    def download_region(self, lat_min, lat_max, lon_min, lon_max,
                        zoom_min=1, zoom_max=15, output_dir="tiles"):
        """Загрузка региона карты"""

        print(f"Загрузка региона:")
        print(f"  Широта: {lat_min} до {lat_max}")
        print(f"  Долгота: {lon_min} до {lon_max}")
        print(f"  Зум: {zoom_min} до {zoom_max}")

        total_tiles = 0
        downloaded = 0

        for zoom in range(zoom_min, zoom_max + 1):
            # Вычисляем границы тайлов для данного зума
            x_min, y_max = self.deg2num(lat_min, lon_min, zoom)
            x_max, y_min = self.deg2num(lat_max, lon_max, zoom)

            for x in range(x_min, x_max + 1):
                for y in range(y_min, y_max + 1):
                    if self.download_tile(zoom, x, y, output_dir):
                        downloaded += 1
                    total_tiles += 1
```

```
                    # Небольшая задержка чтобы не перегружать сервер
                    time.sleep(0.1)

        print(f"\n3агрузка завершена: {downloaded}/{total_tiles} тайлов")


def main():
    downloader = TileDownloader()

    # Пример: Загрузка Московской области
    # Измените координаты под вашу область
    lat_center = 55.7558  # Москва
    lon_center = 37.6176

    # Радиус в градусах (примерно 50км)
    radius = 0.5

    downloader.download_region(
        lat_min=lat_center - radius,
        lat_max=lat_center + radius,
        lon_min=lon_center - radius,
        lon_max=lon_center + radius,
        zoom_min=14,  # Мелкий масштаб
        zoom_max=20,  # Крупный масштаб
        output_dir="tiles"
    )


if __name__ == "__main__":
    main()
```

2. Установка Armbian и необходимых зависимостей для сервера на Orange Pi Zero 2W

2.1. Установка образа

1. Загрузить образ Armbian по следующей ссылке: https://www.armbian.com/orange-pi-zero-2w/;
2. Записать загрузочный образ на microSD-карту такими утилитами, как Rufus или USBImager;
3. Отредактировать на microSD-карте файл `/root/.not_logged_in_yet` так, чтобы устройство подключалось к нужной сети при запуске:

```
PRESET_NET_CHANGE_DEFAULTS=1

# Wi-Fi будет приоритетней Ethernet, если они включены вместе
PRESET_NET_WIFI_ENABLED=1
PRESET_NET_ETHERNET_ENABLED=1
PRESET_NET_WIFI_SSID=<SSID сети>
PRESET_NET_WIFI_KEY=<пароль сети>
PRESET_NET_WIFI_COUNTRYCODE='CA'
PRESET_CONNECT_WIRELESS=n
```

4. Вставить microSD-карту в Orange Pi Zero 2W и включить устройство в сеть. Первый запуск может занять долгое время;

5. Подключиться к устройству по SSH (можно также подключиться к устройству с помощью USB/TTL-адаптера и такой утилиты, как PuTTY) и обновить на нём пакеты:

```
sudo apt update && sudo apt full-upgrade --yes
sudo reboot now
```

6. Также нужно установить Python:

```
sudo apt install python3 python3-dev python3-venv
```

Пароль для пользователя root по умолчанию: “root”.

2.2. Установка wiringOP

wiringOP – утилита и библиотека для управления GPIO Orange Pi Zero 2W, являющаяся аналогом WiringPI для данного устройства. Установить wiringOP можно, подключившись к Orange Pi Zero 2W и выполнив следующие команды:

```
# Клонирование репозитория wiringOP
cd ~
sudo apt update && sudo apt install git
git clone https://github.com/orangepi-xunlong/wiringOP.git -b next
# Сборка библиотеки
cd ~/wiringOP
sudo ./build clean
sudo ./build
```

Проверить работу библиотеки можно следующей командой:

```
gpio readall
```

Должен быть следующий результат:

```
+------+-----+----------+--------+---+  ZERO2W  +---+--------+----------+-----+------+
| GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
+------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
|      |     |     3.3V |        |   |  1 || 2  |   |        | 5V       |     |      |
|  264 |   0 |    SDA.1 |   ALT2 | 0 |  3 || 4  |   |        | 5V       |     |      |
|  263 |   1 |    SCL.1 |    OUT | 1 |  5 || 6  |   |        | GND      |     |      |
|  269 |   2 |     PWM3 |   ALT2 | 0 |  7 || 8  | 0 | ALT2   | TXD.0    | 3   | 224  |
|      |     |      GND |        |   |  9 || 10 | 0 | ALT2   | RXD.0    | 4   | 225  |
|  226 |   5 |    TXD.5 |    OFF | 0 | 11 || 12 | 0 | ALT2   | PI01     | 6   | 257  |
|  227 |   7 |    RXD.5 |    OFF | 0 | 13 || 14 |   |        | GND      |     |      |
|  261 |   8 |    TXD.2 |   ALT2 | 0 | 15 || 16 | 0 | ALT2   | PWM4     | 9   | 270  |
|      |     |     3.3V |        |   | 17 || 18 | 0 | OFF    | PH04     | 10  | 228  |
|  231 |  11 |   MOSI.1 |    OFF | 0 | 19 || 20 |   |        | GND      |     |      |
|  232 |  12 |   MISO.1 |    OFF | 0 | 21 || 22 | 0 | OFF    | RXD.2    | 13  | 262  |
|  230 |  14 |   SCLK.1 |    OFF | 0 | 23 || 24 | 0 | OFF    | CE.0     | 15  | 229  |
|      |     |      GND |        |   | 25 || 26 | 0 | OFF    | CE.1     | 16  | 233  |
|  266 |  17 |    SDA.2 |   ALT2 | 0 | 27 || 28 | 0 | ALT2   | SCL.2    | 18  | 265  |
|  256 |  19 |     PI00 |   ALT2 | 0 | 29 || 30 |   |        | GND      |     |      |
|  271 |  20 |     PI15 |   ALT2 | 0 | 31 || 32 | 0 | ALT2   | PWM1     | 21  | 267  |
|  268 |  22 |     PI12 |   ALT2 | 0 | 33 || 34 |   |        | GND      |     |      |
|  258 |  23 |     PI02 |   ALT2 | 0 | 35 || 36 | 0 | OFF    | PC12     | 24  | 76   |
|  272 |  25 |     PI16 |   ALT2 | 0 | 37 || 38 | 0 | ALT2   | PI04     | 26  | 260  |
|      |     |      GND |        |   | 39 || 40 | 1 | IN     | PI03     | 27  | 259  |
+------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
| GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
+------+-----+----------+--------+---+  ZERO2W  +---+--------+----------+-----+------+
```

2.3. Установка wiringOP-Python

wiringOP-Python – обёртка над wiringOP для использования данной библиотеки с языком программирования Python. Установить wiringOP-Python можно следующими командами:

```
# Установка зависимостей
sudo apt install swig python3-setuptools

# Клонирование репозитория
cd ~
git clone --recursive https://github.com/orangepi-xunlong/wiringOP-Python -b next
cd wiringOP-Python
git submodule update --init –remote

# Компиляция и установка
python3 generate-bindings.py > bindings.i
sudo python3 setup.py install
```

Проверить работу библиотеки можно следующей командой:

```
python3 -c "import wiringpi; help(wiringpi)"
```

В консоли должен вывестись следующий результат:

```
Help on module wiringpi:

NAME
    wiringpi

DESCRIPTION
    # This file was automatically generated by SWIG (https://www.swig.org).
    # Version 4.1.0
...
```

3. Настройка сервера на Orange Pi Zero 2W

3.1. Включение UART5

Для включения UART5 необходимо изменить `/boot/armbianEnv.txt` так, чтобы в строку `overlays` входил элемент `uart5`. Если строки `overlays` нет, нужно добавить её в конец файла следующим образом:

```
overlays=uart5
```

В противном случае нужно добавить `uart5` на конец через пробел примерно таким образом:

```
overlays=spi-spidev i2c3 uart5
```

После перезагрузки микрокомпьтера UART-устройство будет иметь путь `/dev/ttyS5`.

3.2. Настройка автозапуска сервера

Предварительно в домашний каталог `/root` в отдельную директорию (например `/root/server`) скопировать скрипт `orange_pi.py`.

Для работы сервера нужно в отдельной виртуальной Python-среде установить нужно следующие пакеты

```
# Под root
cd ~/server
```

```
# Создание виртуальной среды
python3 -m venv .venv
.venv/bin/pip install -U pip setuptools wheel python-socketio Flask Flask-SocketIO
```

Настроить автозапуск Python-скрипта можно, добавив следующие строки в `/etc/rc.local`:

```
cd /root/server
source .venv/bin/activate
python3 orange_pi.py &
exit 0
```

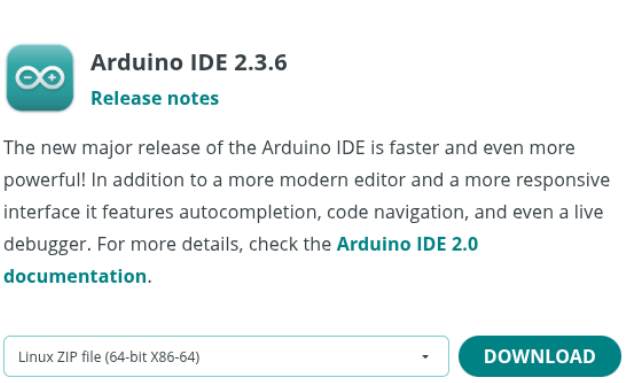
4. Прошивка ESP32

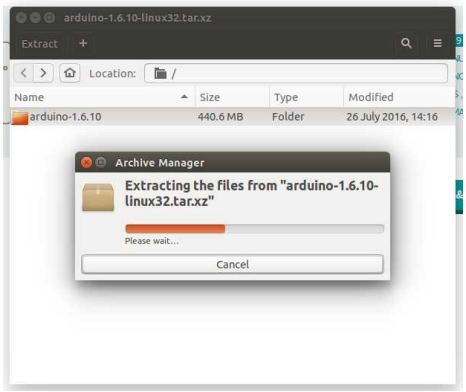
4.1. С помощью Arduino IDE

4.1.1. Установка Arduino IDE с поддержкой ESP32

4.1.1.1. Arduino IDE

1. Загрузить Arduino IDE по следующей ссылке: https://www.arduino.cc/en/software/;
2. Распаковать загруженный архив;





3. Запустить установщик из архива (install.sh):

4.1.1.2. Поддержка ESP32

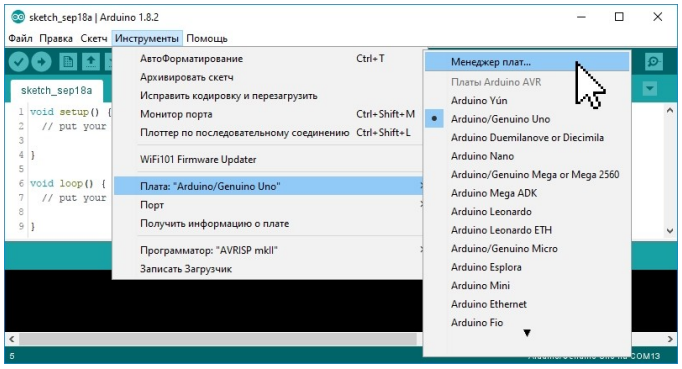
В консоли необходимо выполнить следующие команды:

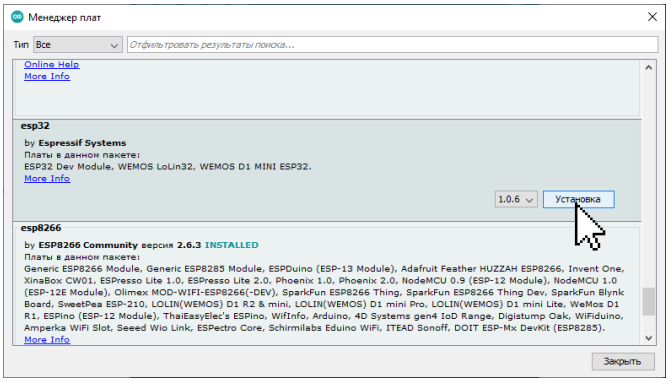
```
# Для доступа к серийному порту, пользователь дожен быть в группе dialout
sudo usermod -a -G dialout $USER

# Установка необхоимых зависимостей (Git, pipб pyserial)
sudo apt-get install git
wget https://bootstrap.pypa.io/get-pip.py
sudo python3 get-pip.py
sudo pip3 install pyserial
```

```
# Установка пакета поддержки ESP32 от Espressif
mkdir -p ~/Arduino/hardware/espressif
cd ~/Arduino/hardware/espressif
git clone https://github.com/espressif/arduino-esp32.git esp32
cd esp32/tools
python3 get.py
```

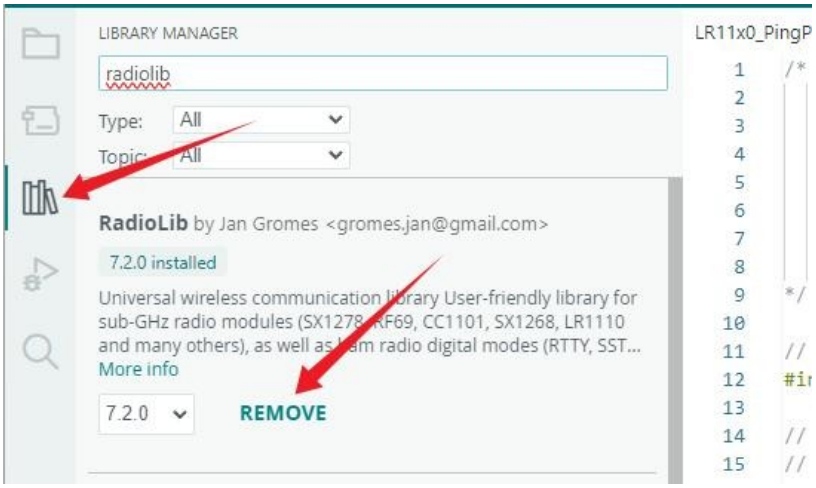
Также данную процедуру можно проделать с помощью менеджера плат Arduino IDE





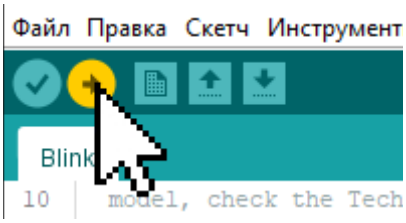
4.1.1.3. Установка RadioLib

Установить RadioLib можно через менеджер библиотек Arduino IDE:



4.1.2. Прошивка

1. Из репозитория 1t_topology_solution открыть файл esp32_code.ino в Arduino IDE;
2. Нужно выбрать плату в меню “Инструменты” → “Плата” → “ESP32 Arduino” → “AI-Thinker ESP32”;
3. Выбрать соответствующий порт в меню “Инструменты” → “Порт”;
4. Начать компиляцию и загрузку скетча.



4.2. С помощью arduino-cli

В консоли необходимо выполнить следующие команды:

```
# Первичная настройка arduino-cli и установка пакета для ESP32
arduino-cli config init
arduino-cli core update-index
arduino-cli board install
arduino-cli core install esp32:esp32
arduino-cli lib install RadioLib

# Сборка
cd <путь-к-репозиторию>/1t_topology_solution
arduino-cli compile --fqbn esp32:esp32:nodemcu-32s esp32_code

# Прошивка
arduino-cli upload esp32_code -p <путь-к-порту-USB> -b esp32:esp32:nodemcu-32s
```

5. Запуск клиента

Следующие действия производятся на ПК оператора:

1. В репозитории `1t_topology_solution` зайти в директорию `site/`;
2. Запустить Python-скрипт `tiles_download.py` для загрузки тайлов карты;
3. Подключиться к той же локальной сети, к которой подключены дроны (в данном случае локальная сеть образуется точкой доступа, к которой подключаются дроны при запуске);
4. В Web-браузере отрыть HTML-файл `1t_topology_solution/site/client.html`;
5. Ввести IP-адрес нужного сервера.

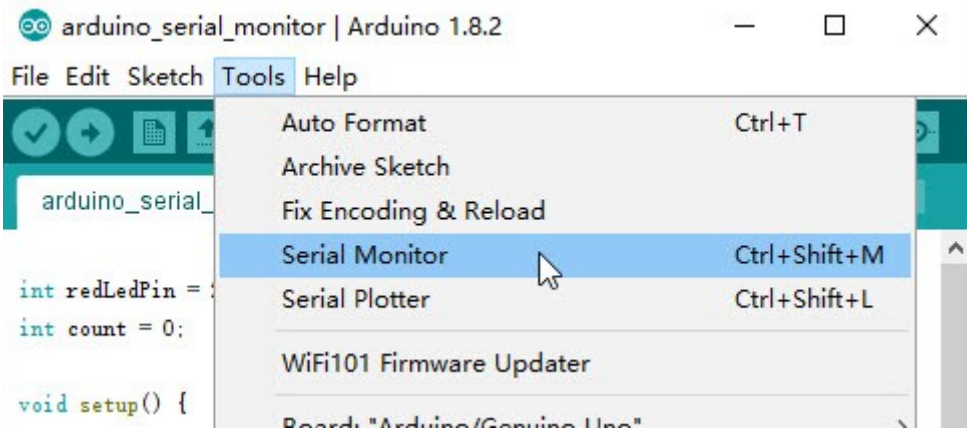
IP-адрес сервера можно узнать любой программой для мониторинга локальной сети. Одними из способов также являются:

- Подключение к консоли Orange Pi Zero 2W и использование команды `ip addr` для интерфейса `wlan0`;
- Сканирование локальной подстети утилитой `nmap` с ключами `-sn` (вывод списка IP-адресов всех устройств в сети).

6. Отладка и мониторинг

6.1. Вывод ESP32

Вывод по UART в консоль ESP32 можно посмотреть с помощью монитора Arduino.



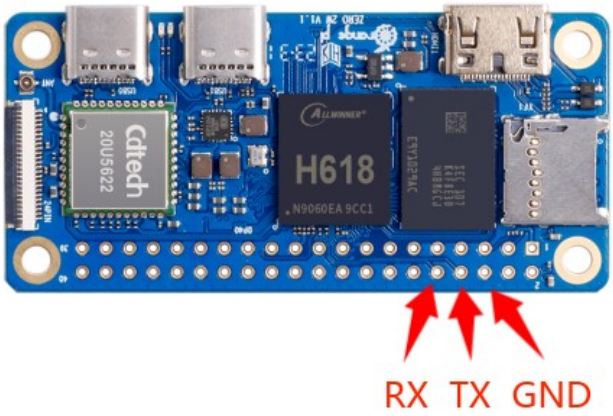
Соответствующая команда для arduino-cli:

```
arduino-cli monitor -p <путь-к-порту-USB> --config 115200
```

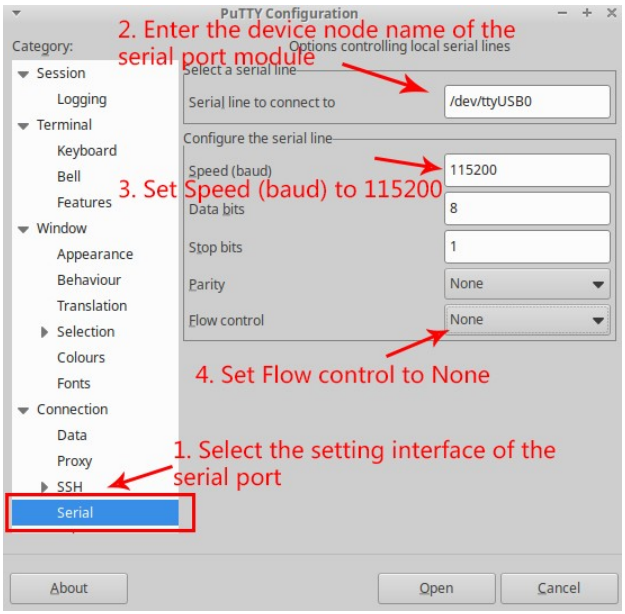
6.2. Подключение к Orange Pi Zero 2W через адаптер USB/TTL

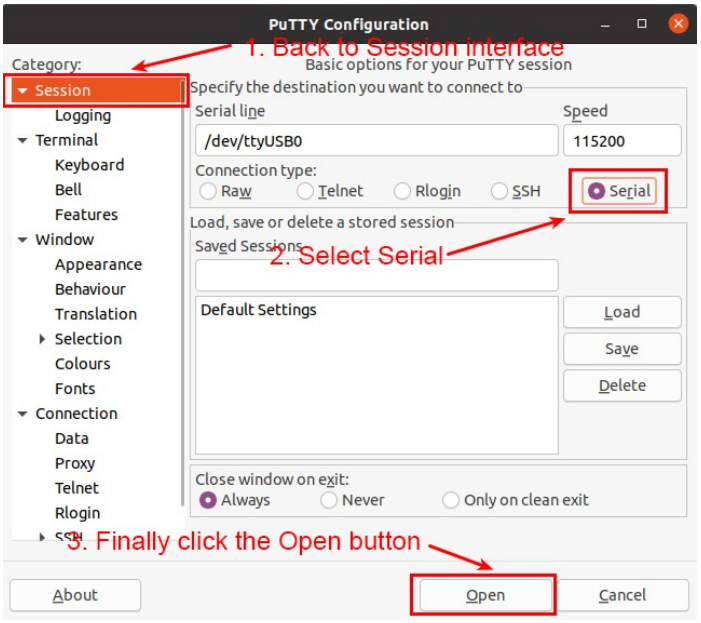
У Orange Pi Zero 2W есть UART-порт для отладки, к которому можно подключить USB/TTL-адаптер следующим образом:

Порт на плате расположен следующим образом:



После подключения, устройство будет отображено файлом с названием наподобие `/dev/ttyUSB0` на ПК. Подключиться к устройству можно с помощью PuTTY со следующими настройками:





После подключения Orange Pi Zero 2W нужно перезагрузить.