ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра «Електронних обчислювальних машин»

Кваліфікаційна робота на тему:

«Розробка алгоритму роботи та системи оманливих рацій для боротьби з ворожими структурами та захисту військових об'єктів»

Виконав: ст. Групи спм-22-4 Шаманов д.О.

Керівник: ст. викл. Сорокін А. Р.

2024

Актуальність проблеми

- Через технічний прогрес ведення війни, та жагу кожної зі сторін збільшувати свої шанси на перемогу з супротивником, все більше використовуються системи для виявлення місцезнаходження військових на лінії зіткнення та в глибокому тилу.
- Для цього ворог використовує системи РЕБ, РЕЗ та інші розробки для отримання інформації про місцезнаходження цілей чи скупчення особового складу ЗСУ.
- Через це є велика потреба в системах, які б вводили супротивника в оману, змушуючи витрачати час/боєприпаси та свою увагу на фіктивні цілі, пропускаючи при цьому реальні.

Мета та задачі

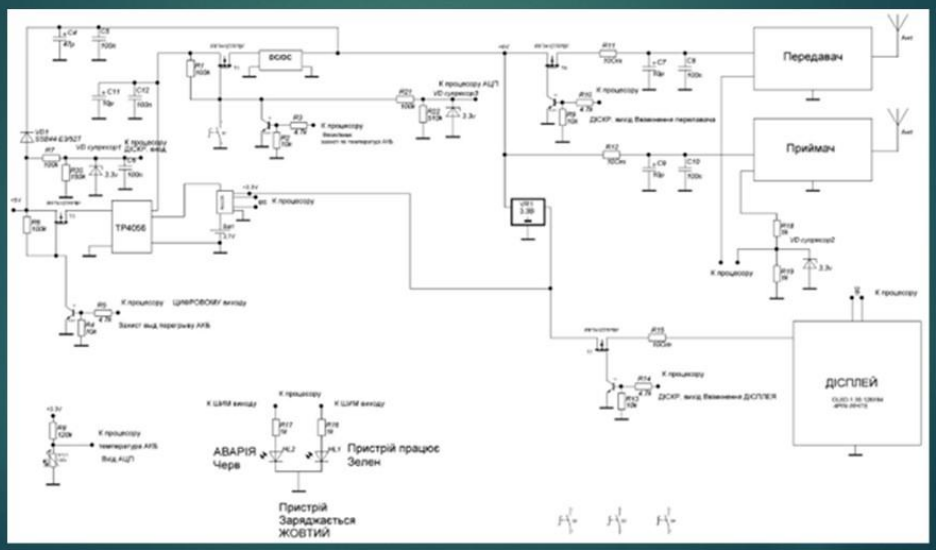
- Розробка пристрою для генерації оманливих радіо та Wi-Fi сигналів, для захисту військових об'єктів та особового складу ЗСУ від систем РЕБ, РЕЗ та інших систем розвідки.
- Дослідження найкращих варіантів для розміщення пристроїв, щоб вони повторювали звичні для ЗСУ розміщення на позиціях
- Ім макету пристрою для тестування та поліпшення алгоритмів генерації даних.

Порівняння існуючих технологій оманливих рацій

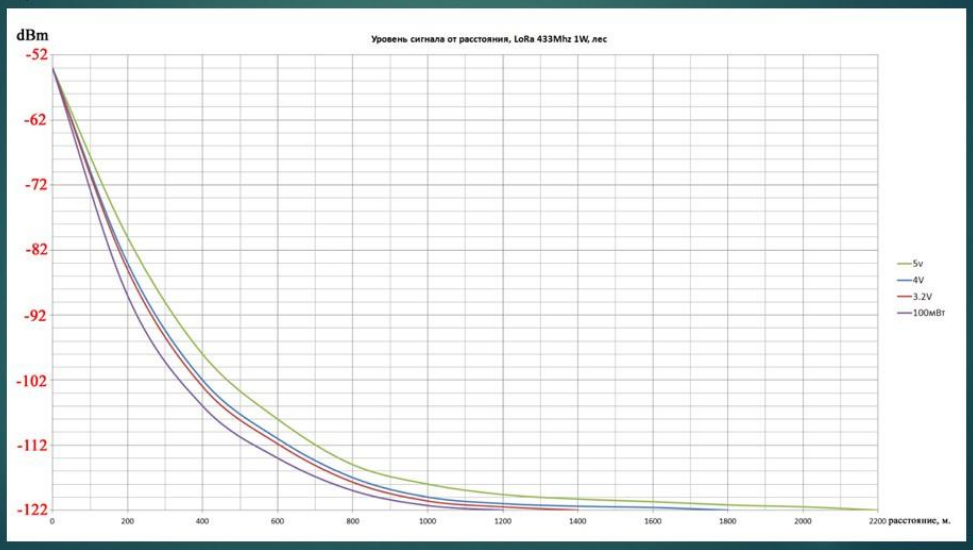
Наразі, не було знайдено жодних пристроїв на просторах інтернету, які б повторювали тему мого диплому, або були б схожі на нього. Проте, в історії є декілька прикладів, коли використовували оманливі дані для введення супротивника в оману:

- Висадка в Нормандії.
- В'єтнамська війна
- Війна в Іраку

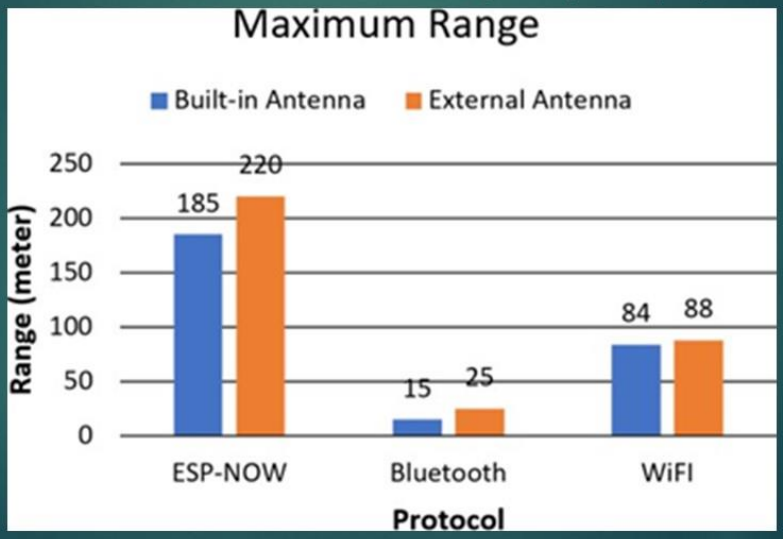
Функціонально-принципова схема



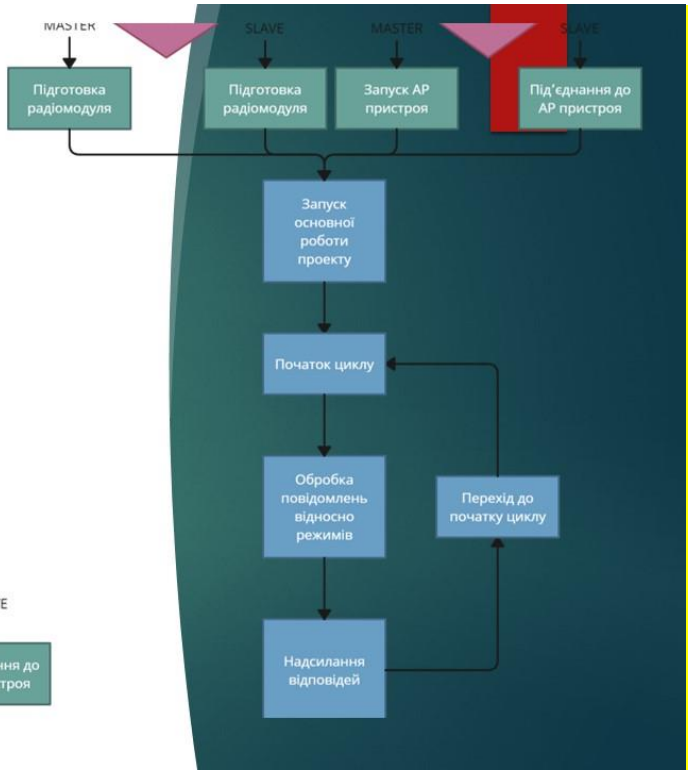
Якість сигналу радіо модуля 433МГц відносно відстані



Якість сигналу Wi-Fi відносно відстані від роутеру



Блок схема роботи системи



Порівняння програмованих платформ

Arduino

Переваги:

- Простота використання
- Дуже велика поширенність
- Велика спільнота

Недоліки:

- Маленька потужність
- Відсутність вбудов аних модулів

ESP32

Переваги:

- Мала ціна
- Висока продуктивність
- Низьке енергоспоживанн я
- Багатофункціона льність

Недоліки:

- Відсутність Wi-FI 5ГГц

STM32

Переваги:

- Широкий вибір контролерів
- Висока продуктивність
- Простота використання

Недоліки:

- Ціна
- Відсутність вбудов аних модулів

Порівняння радіо модулів 433 МГц

FS1000A + MX-RM-5V

Переваги:

- Дешевизна
- Простота при роботі
- Компактність

Недоліки:

- Маленька потужність

Si4432

Переваги:

- Універсальний модуль
- Висока потужність
- Чутливий приймач

Недоліки:

- Ціна

Переваги цієї системи оманливих рацій

- Швидкість налаштування
- Економність ресурсів
- Можливість автономної роботи
- Зниження ризиків для особового складу
- Зниження ризиків для військових об'єктів
- Ефективність проти різних типів загроз
- Можливість повторного використання

Місця використання пристроїв

1. Захист військових на позиціях

- 2. Захист систем ППО

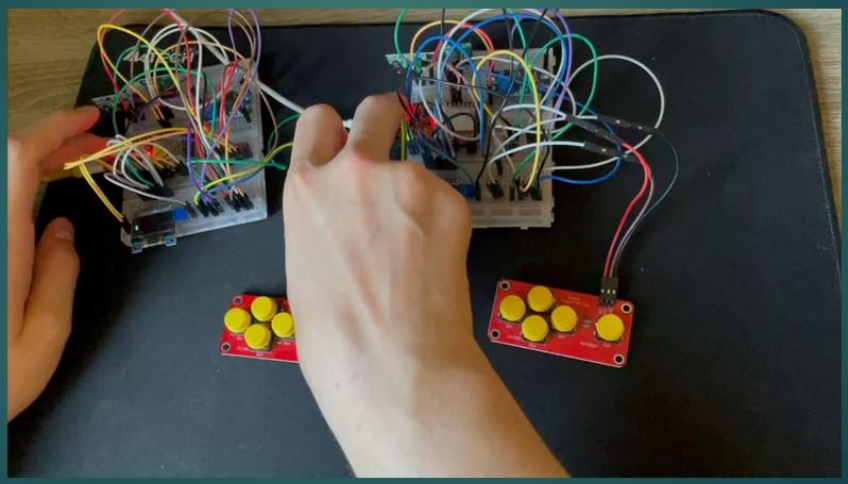
- 3. Захист аеродромів

- 4. Захист важливих об'єктів

План розвитку та покращення системи оманливих рацій

- До можливих покращень можна віднести:
  1. Додавання безпровідної зарядки, для збільшення вологозахисту.
- 2. Додавання сім карток з виходом до інтернету, для ще однієї можливості імітації сигналів
- 3. Додавання біомімікрії, нейромереж або штучного інтелекту для покращення алгоритмів генерації штучних сигналів спілкування.
- Розширення застосування для захисту інших видів систем, цілей.

Відеопрезентація роботи пристроя



Висновки

- Військові захищають нас від ворога. І ми маємо захищати їх, придумуючи нові системи для цього. Ця система оманливих рацій є унікальним продуктом, який може зберегти життя військових.
- Існує значний потенціал для вдосконалення системи шляхом інтеграції передових технологій, підвищення реалізму та обману, розширення сфери застосування та відповідального використання.
- Система оманливих рацій може стати цінним інструментом для військових, органів безпеки та інших організацій, які прагнуть захистити свої об'єкти та особовий склад.

ДОДАТОК Б КОД ПРОЕКТУ

```
#include <Arduino.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <iostream>
#include <WiFi.h>
#include "esp_random.h"
#include <Preferences.h>
#include <Gyver433.h>
#include <Preferences.h>
#include <INA226_WE.h>
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_MOSI 23
#define OLED_CLK 18
#define OLED_DC 16
#define OLED_CS 5
#define OLED_RESET 17
#define KEYBOARD_OUT 2
#define RECIEVER_RADIO_PIN 32  // GPIO pin for radio module
#define SENDER_RADIO_PIN 35    // GPIO pin for radio module
#define RADIO_FREQUENCY 433.92 // Frequency in MHz
using namespace std;
const char *ssid[7] = {"1", "2", "3", "4", "5", "6", "7"};
const char *password = "20142024";
String clientIPs[8];
String currLocalIP;
uint8_t queryCount = 0;
String masterSlaveAction;
String radioWifiAction;
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
                         OLED_MOSI,   OLED_CLK,   OLED_DC,   OLED_RESET,
OLED_CS);
WiFiServer server(80);
Gyver433_TX<SENDER_RADIO_PIN> tx;
Gyver433_RX<RECIEVER_RADIO_PIN, 64> rx;
Preferences preferences;
void TextDisplay(String data)
{
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(5, 28);
  display.println(data);
  display.display();
}
void WiFiEvent(WiFiEvent_t event)
{
  switch (event)
  {
  case WIFI_EVENT_STA_DISCONNECTED:
    if (!clientIPs[0].isEmpty())
    {
      if (currLocalIP.equals(clientIPs[0]))
      {
        TextDisplay("MASTER");
        delay(1000);
        preferences.putString("masterSlave", "MASTER");
      }
      else
```

```
      {
        TextDisplay("SLAVE");
        delay(1000);
        preferences.putString("masterSlave", "SLAVE");
      }
      preferences.putString("radioWifi", "WIFI");
      esp_restart();
    }
    break;
  }
}
String getAction(uint16_t action)
{
  if (action <= 99)
  {
    return "LEFT";
  }
  else if (action >= 100 && action <= 300)
  {
    return "TOP";
  }
  else if (action >= 300 && action <= 500)
  {
    return "BOTTOM";
  }
  else if (action >= 550 && action <= 700)
  {
    return "RIGHT";
  }
  else if (action >= 1000 && action <= 1200)
  {
    return "OK";
  }
  else
    return "DEFAULT";
};
String getKey(String action)
{
  return action.substring(0, 2);
};
String getValue(String action)
{
  return action.substring(3);
};
String getSampleProcessData()
{
  long rand = random(7);
  if (clientIPs[rand].isEmpty())
  {
    return getSampleProcessData();
  }
  String resData = clientIPs[rand];
  return resData.substring(random(resData.length())) + ";";
}
String processMasterCommand(String key, String action)
{
  if (key.equals("IP"))
  {
    bool isNew = true;
    for (int i = 0; i < 8; i++)
    {
      if (clientIPs[i].equals(action))
      {
        isNew = false;
```

```
        break;
      }
    }
    if (isNew)
    {
      for (int i = 0; i < 8; i++)
      {
        if (clientIPs[i].isEmpty())
        {
          clientIPs[i] = action;
          break;
        }
      }
    }
  }
  else if (key.equals("GT"))
  {
    String ips = "IG:";
    for (int i = 0; i < 8; i++)
    {
      if (!clientIPs[i].isEmpty())
      {
        ips += clientIPs[i] + ":";
      }
    }
    return ips + ";";
  }
  else if (key.equals("DT"))
  {
    return getSampleProcessData();
  }
  return "";
};
String processSlaveCommand(String key, String action)
{
  if (key.equals("IG"))
  {
    for (int i = 0; i < 8; i++)
    {
      String ipAddress = action.substring(0, action.indexOf(':'));
      TextDisplay("IP");
      delay(3000);
      TextDisplay(ipAddress);
      delay(3000);
      if (!ipAddress.isEmpty())
      {
        for (int i = 0; i < 8; i++)
        {
          if (clientIPs[i].isEmpty() || clientIPs[i].equals(ipAddress))
          {
            clientIPs[i] = ipAddress;
            break;
          }
        }
        action.remove(0, action.indexOf(':') + 1);
      }
      else
      {
        break;
      }
    }
  }
  else if (key.equals("GT"))
  {
```

```
    String ips = "IG:";
    for (int i = 0; i < 8; i++)
    {
      if (!clientIPs[i].isEmpty())
      {
        ips += clientIPs[i] + ":";
      }
    }
    ips.remove(ips.length() - 1);
    return ips;
  }
  return "";
};
void isr()
{
  rx.tickISR();
}
void setup()
{
  pinMode(KEYBOARD_OUT, INPUT);
  if (!display.begin(SSD1306_SWITCHCAPVCC))
  {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;)
      ;
  }
  preferences.begin("storeSettings", false);
  String masterSlaveStored = preferences.getString("masterSlave");
  String radioWifiStored = preferences.getString("radioWifi");
  display.clearDisplay();
  display.display();
  TextDisplay("BDS");
  delay(3000);
  bool isActionSetted = false;
  int cursor = 0;
  if (!masterSlaveStored.isEmpty() && !radioWifiStored.isEmpty())
  {
    TextDisplay("Clear old config?");
    int count = 0;
    while (count < 100)
    {
      uint16_t val = analogRead(KEYBOARD_OUT);
      String value = getAction(val);
      if (value.equals("OK"))
      {
        TextDisplay("");
        masterSlaveStored = "";
        radioWifiStored = "";
        preferences.clear();
        delay(1000);
        break;
      }
      delay(100);
      count++;
    }
    masterSlaveAction = masterSlaveStored;
    radioWifiAction = radioWifiStored;
  }
  if (masterSlaveAction.isEmpty() && radioWifiAction.isEmpty())
  {
    String masterSlaveMenu[2] = {"MASTER", "SLAVE"};
    String radioWifiMenu[2] = {"RADIO", "WIFI"};
    String currentMenu[2];
    String defaultAction = "DEFAULT";
```

```
  int currentPos = 0;
  isActionSetted = false;
  while (!isActionSetted)
  {
    if (currentPos == 0)
    {
      currentMenu[0] = masterSlaveMenu[0];
      currentMenu[1] = masterSlaveMenu[1];
    }
    else
    {
      currentMenu[0] = radioWifiMenu[0];
      currentMenu[1] = radioWifiMenu[1];
    }
    TextDisplay(currentMenu[cursor]);
    uint16_t val = analogRead(KEYBOARD_OUT);
    String value = getAction(val);
    if (value.equals(defaultAction))
    {
      delay(150);
      continue;
    }
    defaultAction = value;
    if (value.equals("OK"))
    {
      if (currentPos == 0)
      {
        masterSlaveAction = currentMenu[cursor];
        currentPos++;
        cursor = 0;
        delay(150);
      }
      else
      {
        radioWifiAction = currentMenu[cursor];
        isActionSetted = true;
      }
    }
    else if (value.equals("LEFT"))
    {
      cursor -= 1;
    }
    else if (value.equals("RIGHT"))
    {
      cursor += 1;
    }
    if (cursor > 1)
    {
      cursor = 0;
    }
    else if (cursor < 0)
    {
      cursor = 1;
    }
    delay(150);
  }
}
if (radioWifiAction.equals("WIFI"))
{
  Serial.begin(115200);
  if (masterSlaveAction.equals("MASTER"))
  {
    uint8_t rand = random(0, 7);
    WiFi.mode(WIFI_AP);
```

```
    TextDisplay(ssid[rand]);
    WiFi.softAP(ssid[rand], password);
    server.begin();
  }
  else if (masterSlaveAction.equals("SLAVE"))
  {
    WiFi.mode(WIFI_STA);
    WiFi.onEvent(WiFiEvent);
    bool isConnected = false;
    for (int i = 0; i < 7; i++)
    {
      TextDisplay("Connect to");
      TextDisplay(ssid[i]);
      delay(100);
      int counter = 0;
      WiFi.begin(ssid[i], password);
      while (counter < 100)
      {
        if (WiFi.status() == WL_CONNECTED)
        {
          isConnected = true;
          break;
        }
        delay(100);
        counter++;
      }
      if (isConnected)
      {
        break;
      }
    }
    if (isConnected)
    {
      WiFiClient client;
      if (client.connect("192.168.4.1", 80))
      {
        TextDisplay("SEND LOCAL IP");
        currLocalIP = WiFi.localIP().toString();
        String localIP = "IP:" + currLocalIP + ";";
        TextDisplay(localIP);

        client.print(localIP);
        delay(100);
        client.stop();
      }
    }
  }
  else
  {
    esp_restart();
  }
}
else if (radioWifiAction.equals("RADIO"))
{
  Serial.begin(9600);
  attachInterrupt(0, isr, CHANGE);
  if (masterSlaveAction.equals("MASTER"))
  {
    String helloWal = "GTVAL;";
    TextDisplay(helloWal);
    delay(3000);
  }
  else if (masterSlaveAction.equals("SLAVE"))
  {
```

```
      uint64_t chipId = ESP.getEfuseMac();
      String id = "IP:" + String(chipId, HEX);
      TextDisplay(id);
      delay(3000);
      tx.sendData(id);
    }
  }
  else
  {
    esp_restart();
  }
  preferences.putString("masterSlave", masterSlaveAction);
  preferences.putString("radioWifi", radioWifiAction);
}
void loop()
{
  if (radioWifiAction.equals("WIFI"))
  {
    if (masterSlaveAction.equals("MASTER"))
    {
      uint8_t nums = WiFi.softAPgetStationNum();
      TextDisplay(to_string(nums).c_str());
      WiFiClient client = server.available();
      if (client)
      {
        TextDisplay("accepted");
        String res = client.readStringUntil(';');
        if (!res.isEmpty())
        {
          String key = getKey(res);
          String value = getValue(res);
          String processRes = processMasterCommand(key, value);
          client.flush();
          client.println(processRes);
        }
        client.stop();
      }
    }
    else if (masterSlaveAction.equals("SLAVE"))
    {
      WiFiClient client;
      if (client.connect("192.168.4.1", 80))
      {
        if (queryCount == 10)
        {
          client.flush();
          client.print("GT;");
          queryCount = 0;
        }
        else
        {
          client.flush();
          client.print("DT;");
        }
        String res = client.readStringUntil(';');
        if (!res.isEmpty())
        {
          String key = getKey(res);
          String value = getValue(res);
          String processRes = processSlaveCommand(key, value);
          client.flush();
          client.println(processRes);
        }
        else
```

```
        {
          TextDisplay("EMPTY");
        }
        client.stop();
      }
      queryCount++;
      delay(random(500, 2000));
    }
  }
  else if (radioWifiAction.equals("RADIO"))
  {
    if (masterSlaveAction.equals("MASTER"))
    {
      if (rx.gotData())
      {
        String res;
        rx.readData(res);
        TextDisplay("res");
        delay(3000);
        TextDisplay(res);
        delay(3000);
        String key = getKey(res);
        String value = getValue(res);
        String processRes = processMasterCommand(key, value);
        tx.sendData(processRes);
      }
    }
    else if (masterSlaveAction.equals("SLAVE"))
    {
      if (queryCount == 10)
      {
        tx.sendData("GT;");
        queryCount = 0;
      }
      else
      {
        tx.sendData("DT;");
      }
      if (rx.gotData())
      {
        String res;
        rx.readData(res);
        TextDisplay("res");
        delay(3000);
        TextDisplay(res);
        delay(3000);
        String key = getKey(res);
        String value = getValue(res);
        String processRes = processSlaveCommand(key, value);
        tx.sendData(processRes);
      }
    }
    queryCount++;
    delay(random(500, 2000));
  }
}
```