ДОДАТОК А

Програмний код для інтерфейсу користувача

Код інтерфейсу користувача:

```
using GMap.NET;
using GMap.NET.WindowsForms;
using GMap.NET.WindowsForms.Markers;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Drawing.Imaging;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace dip2
{
   public partial class Form1 : Form
   {

      System.Drawing.Point lastPoint;

      public double Latitude { get; set; }
      public double Longitude { get; set; }
      public double Lat { get; set; }
```

```
        public double Lon { get; set; }

        private readonly string apiUrl =
@"https://api.thingspeak.com/channels/2525917/feeds/last.json?api_key=QTPMM25KFRP0VR2
W";
        private Timer timer;
        private GMapOverlay markersOverlay;
        private GMapRoute route;

        public Form1()
        {
            InitializeComponent();
            timer = new Timer();
            timer.Interval =  5000;
            timer.Tick += Timer_Tick;

            this.CenterToScreen();
            markersOverlay = new GMapOverlay("markers");
            gMap.Overlays.Add(markersOverlay);

            ToolTip toolTip = new ToolTip();
            toolTip.SetToolTip(location, "Indicates the location of an object");
            toolTip.SetToolTip(StartTracking, "Starts tracking the object");
            toolTip.SetToolTip(StopTracking, "Stops tracking the object");

            System.Drawing.Drawing2D.GraphicsPath gp = new
System.Drawing.Drawing2D.GraphicsPath();
            gp.AddEllipse(0, 0, satellite.Width - 3, satellite.Height - 3);
            Region rg = new Region(gp);
            satellite.Region = rg;
            System.Drawing.Drawing2D.GraphicsPath gp2 = new
System.Drawing.Drawing2D.GraphicsPath();
            gp2.AddEllipse(0, 0, ground.Width - 3, ground.Height - 3);
            Region rg2 = new Region(gp2);
```

```
    ground.Region = rg2;
    ground.Visible = false;

    StartTracking.Enabled = false;
    StopTracking.Enabled = false;
}

private async void Timer_Tick(object sender, EventArgs e)
{
    await FetchData();
}
private async Task FetchData()
{
    using (HttpClient client = new HttpClient())
    {
        try
        {
            HttpResponseMessage response = await client.GetAsync(apiUrl);
            response.EnsureSuccessStatusCode();

            string responseBody = await response.Content.ReadAsStringAsync();

            dynamic jsonObject = JsonConvert.DeserializeObject(responseBody);

            double latitude = Convert.ToDouble(jsonObject.field1);
            double longitude = Convert.ToDouble(jsonObject.field2);

            Latitude = latitude;
            Longitude = longitude;

            var markersToRemove = new List<GMapMarker>();
            foreach (var mark in markersOverlay.Markers)
            {
                if (mark is GMarkerGoogle m && m.Type == GMarkerGoogleType.red)
```

```
                    {
                        markersToRemove.Add(mark);
                    }
                }
                foreach (var mark in markersToRemove)
                {
                    markersOverlay.Markers.Remove(mark);
                }

                GMapMarker marker = new GMarkerGoogle(new PointLatLng(latitude,
longitude), GMarkerGoogleType.red);
                markersOverlay.Markers.Add(marker);

                if (route != null)
                {
                    route.Points.Add(marker.Position);
                }
                gMap.Position = marker.Position;
            }
            catch (HttpRequestException ex)
            {
                MessageBox.Show($"Request exception: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"An error occurred: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
    private void gMap_Load(object sender, EventArgs e)
    {
        GMap.NET.GMaps.Instance.Mode = GMap.NET.AccessMode.ServerAndCache;
```

```
            gMap.MapProvider = GMap.NET.MapProviders.GoogleMapProvider.Instance;
            gMap.MinZoom = 2;
            gMap.MaxZoom = 20;
            gMap.Zoom = 18;
            gMap.Position = new GMap.NET.PointLatLng(50.43968, 30.63042);
            gMap.MouseWheelZoomType =
GMap.NET.MouseWheelZoomType.MousePositionAndCenter;
            gMap.CanDragMap = true;
            gMap.DragButton = MouseButtons.Left;
            gMap.ShowCenter = false;
            gMap.ShowTileGridLines = false;
        }

        private void panelUp_MouseMove(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                this.Left += e.X - lastPoint.X;
                this.Top += e.Y - lastPoint.Y;
            }
        }
        private void panelUp_MouseDown(object sender, MouseEventArgs e)
        {
            lastPoint = new System.Drawing.Point(e.X, e.Y);
        }

        private void CloseButton_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void StartTracking_Click(object sender, EventArgs e)
        {
```

```
            GMapMarker marker1 = new GMarkerGoogle(new PointLatLng(Latitude,
Longitude), GMarkerGoogleType.blue);
            markersOverlay.Markers.Add(marker1);
            if (route == null)
            {
                route = new GMapRoute(new List<PointLatLng> { marker1.Position }, "Route");
                route.Stroke = new Pen(Color.Yellow, 4);
                markersOverlay.Routes.Add(route);
            }
        }

        private void StopTracking_Click(object sender, EventArgs e)
        {
            GMapMarker marker2 = new GMarkerGoogle(new PointLatLng(Latitude,
Longitude), GMarkerGoogleType.green);
            markersOverlay.Markers.Add(marker2);
            route = null;
        }

        private void location_Click(object sender, EventArgs e)
        {
            StartTracking.Enabled = true;
            StopTracking.Enabled = true;

            timer.Stop();
            FetchData();
            timer.Start();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Minimized;
        }
        int delay = 0;
```

```
private void gMap_MouseMove(object sender, MouseEventArgs e)
{
   try
   {
      delay++;
      if (delay == 10)
      {
         int x = e.X;
         int y = e.Y;
         double lat = gMap.FromLocalToLatLng(e.X, e.Y).Lat;
         double lon = gMap.FromLocalToLatLng(e.X, e.Y).Lng;
         labelLat.Text = "Lat: " + Convert.ToString(lat);
         labelLon.Text = "Lon: " + Convert.ToString(lon);
         delay = 0;
      }
   }
   catch (Exception ex)
   {

      MessageBox.Show(ex.Message, "Error!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
   }
}

List<PointLatLng> points = new List<PointLatLng>();
public GMapOverlay polygons = new GMapOverlay("polygons");
GMapPolygon polygon;
List<PointLatLng> points_route = new List<PointLatLng>();
public GMapOverlay routes = new GMapOverlay("Routes");

private void gMap_MouseDoubleClick(object sender, MouseEventArgs e)
{
   if (e.Button == System.Windows.Forms.MouseButtons.Right)
   {
```

```
                points.Add(new PointLatLng(gMap.FromLocalToLatLng(e.X, e.Y).Lat,
gMap.FromLocalToLatLng(e.X, e.Y).Lng));
                Lat = gMap.FromLocalToLatLng(e.X, e.Y).Lat;
                Lon = gMap.FromLocalToLatLng(e.X, e.Y).Lng;
                GMapMarker marker1 = new GMarkerGoogle(new PointLatLng(Lat, Lon),
GMarkerGoogleType.red_dot);
                markersOverlay.Markers.Add(marker1);
                if (points.Count >= 0)
                {
                   polygons.Clear();
                   gMap.Overlays.Remove(polygons);
                   polygon = new GMapPolygon(points, "boundary");
                   polygon.Fill = new SolidBrush(Color.FromArgb(30, Color.Red));
                   polygon.Stroke = new Pen(Color.Red, 2);
                   polygons.Polygons.Add(polygon);
                }
                gMap.Overlays.Add(polygons);
                gMap.Refresh();
             }

             if (e.Button == System.Windows.Forms.MouseButtons.Left)
             {
                points_route.Add(new PointLatLng(gMap.FromLocalToLatLng(e.X, e.Y).Lat,
gMap.FromLocalToLatLng(e.X, e.Y).Lng));
                Lat = gMap.FromLocalToLatLng(e.X, e.Y).Lat;
                Lon = gMap.FromLocalToLatLng(e.X, e.Y).Lng;
                GMapMarker marker1 = new GMarkerGoogle(new PointLatLng(Lat, Lon),
GMarkerGoogleType.blue_dot);
                markersOverlay.Markers.Add(marker1);

                if (points_route.Count >= 0)
                {

                   gMap.Overlays.Remove(routes);
```

```
            GMapRoute r = new GMapRoute(points_route, "Route line");
            r.Stroke = new System.Drawing.Pen(System.Drawing.Color.Blue, 2);
            gMap.Overlays.Add(routes);
            routes.Routes.Add(r);
        }
        gMap.Overlays.Add(routes);
        gMap.Refresh();
    }
}

private void ClearAll_Click(object sender, EventArgs e)
{
    markersOverlay.Markers.Clear();
    markersOverlay.Routes.Clear();
    polygons.Polygons.Clear();
    points.Clear();
    routes.Clear();
    points_route.Clear();

    gMap.Refresh();
    timer.Stop();

    StartTracking.Enabled = false;
    StopTracking.Enabled = false;
}

private void satellite_Click_1(object sender, EventArgs e)
{
    gMap.MapProvider = GMap.NET.MapProviders.BingHybridMapProvider.Instance;
    satellite.Visible = false;
    ground.Visible = true;
}

private void ground_Click(object sender, EventArgs e)
```

```
        {
            gMap.MapProvider = GMap.NET.MapProviders.GoogleMapProvider.Instance;
            ground.Visible = false;
            satellite.Visible = true;
        }

        private void Video_Click(object sender, EventArgs e)
        {
            System.Diagnostics.Process.Start(@"http://192.168.43.114/");
        }
    }
}
```

ДОДАТОК Б

Програмний код для апаратної чатини

Код апаратної частини:

```
#include <WiFi.h>
#include <TinyGPS++.h>
#include <LiquidCrystal_I2C.h>

const char* ssid = "Sanya_zam";
const char* password = "30082003";

const char* thingspeakHost = "api.thingspeak.com";
unsigned long myChannelNumber = 2525917;
const char* myWriteAPIKey = "5EBB7JKYYKYUQFR6";

#define RXD0 16
#define TXD0 17
#define GPS_BAUD 9600

TinyGPSPlus gps;

unsigned long lastUploadTime = 0;
const unsigned long uploadInterval = 5000;

LiquidCrystal_I2C lcd(0x27, 16, 2);

void sendDataToThingSpeak(float latitude, float longitude) {
  WiFiClient client;

  if (client.connect(thingspeakHost, 80)) {
    String postStr = String("/update?api_key=");
```

```
    postStr += myWriteAPIKey;
    postStr += "&field1=";
    postStr += String(latitude, 6);
    postStr += "&field2=";
    postStr += String(longitude, 6);
    postStr += "\r\n";

    client.print("GET " + postStr);
    delay(10);
  }
  client.stop();
}

void setup() {
  Serial.begin(115200);
  Serial2.begin(GPS_BAUD, SERIAL_8N1, RXD0, TXD0);
  lcd.init();
  lcd.backlight();
  lcd.print("Waiting for GPS");
  Serial.println("Connecting to WiFi...");
  WiFi.begin(ssid, password);

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

void loop() {
  while (Serial2.available() > 0) {
    if (gps.encode(Serial2.read())) {
      if (gps.location.isValid()) {
        float latitude = gps.location.lat();
```

```
      float longitude = gps.location.lng();
      Serial.print("Latitude: ");
      Serial.println(latitude, 6);
      Serial.print("Longitude: ");
      Serial.println(longitude, 6);

      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Lat: ");
      lcd.print(latitude, 6);
      lcd.setCursor(0, 1);
      lcd.print("Lng: ");
      lcd.print(longitude, 6);

      if (millis() - lastUploadTime >= uploadInterval) {
        sendDataToThingSpeak(latitude, longitude);
        lastUploadTime = millis();
      }
    } else {
      Serial.println("GPS data not valid");
    }
  }
 }
}
```

Модуляційний код для GPS:

```
#include "wokwi-api.h"
 #include <stdio.h>
 #include <stdlib.h>
 #include <string.h>    // contains strlen() function
```

```
#define LEN(arr) ((int)(sizeof(arr) / sizeof(arr)[0]))    // macro

#define SECOND 5000000    // micros

// A NMEA 0183 sentence can have a maximum of 80 characters plus a
// carriage return and a line feed

const char gps_tx_data[][80] = { // GPRMC & GPGGA (Hypothetical Data)
  "$GPGGA,155624.122,0653.171,S,10736.833,E,1,12,1.0,0.0,M,0.0,M,,*7F\r\n",
  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30\r\n",
  "$GPRMC,155624.122,A,0653.171,S,10736.833,E,037.4,270.0,200124,000.0,W*6B\r\n",
  "$GPGGA,155625.122,0653.171,S,10736.822,E,1,12,1.0,0.0,M,0.0,M,,*7E\r\n",
  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30\r\n",
  "$GPRMC,155625.122,A,0653.171,S,10736.822,E,079.0,182.9,200124,000.0,W*63\r\n",
  "$GPGGA,155626.122,0653.193,S,10736.821,E,1,12,1.0,0.0,M,0.0,M,,*72\r\n",
  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30\r\n",
  "$GPRMC,155626.122,A,0653.193,S,10736.821,E,133.6,181.0,200124,000.0,W*6C\r\n",
  "$GPGGA,155627.122,0653.230,S,10736.820,E,1,12,1.0,0.0,M,0.0,M,,*78\r\n",
  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30\r\n",
  "$GPRMC,155627.122,A,0653.230,S,10736.820,E,230.4,182.0,200124,000.0,W*67\r\n",
  "$GPGGA,155628.122,0653.294,S,10736.818,E,1,12,1.0,0.0,M,0.0,M,,*72\r\n",
  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30\r\n",
  "$GPRMC,155628.122,A,0653.294,S,10736.818,E,150.1,184.4,200124,000.0,W*6F\r\n",
  "$GPGGA,155629.122,0653.335,S,10736.815,E,1,12,1.0,0.0,M,0.0,M,,*74\r\n",
  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30\r\n",
  "$GPRMC,155629.122,A,0653.335,S,10736.815,E,171.4,186.2,200124,000.0,W*6B\r\n",
};

typedef struct {
  uart_dev_t uart0;
  uint32_t   gps_tx_index;
} chip_state_t;
```

```
static void chip_timer_event  (void *user_data);

void chip_init(void) {

  setvbuf(stdout, NULL, _IOLBF, 1024);

  chip_state_t *chip = malloc(sizeof(chip_state_t));

  const uart_config_t uart_config = {
    .tx        = pin_init("TX", INPUT_PULLUP),
    .rx        = pin_init("RX", INPUT),
    .baud_rate  = 9600,
    .user_data  = chip,
  };

  chip->uart0       = uart_init(&uart_config);

  chip->gps_tx_index = 0;

  const timer_config_t timer_config = {
    .callback  = chip_timer_event,
    .user_data = chip,
  };

  timer_t timer = timer_init(&timer_config);

  timer_start(timer, SECOND, true);

  printf("GPS Chip initialized!\n");

}

void chip_timer_event(void *user_data) {
```

```
  chip_state_t *chip = (chip_state_t*) user_data;

  printf("chip_timer_event\n");

  const char * message = gps_tx_data[chip->gps_tx_index++];

  uart_write(chip->uart0, (uint8_t *) message, strlen(message));

  if (chip->gps_tx_index >= LEN(gps_tx_data)) {
    chip->gps_tx_index = 0;
  }
}
```

ДОДАТОК В

Демонстраційний матеріал

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра КІТАР

АТЕСТАЦІЙНА РОБОТА

На тему: Розробка системи автоматизації для відстеження та відображення місцезнаходження мобільного робота на карті місцевості.

Виконав:
ст. гр. АКТАКІТ-20-1
Шевченко О. О.

Керівник:
доц. каф. КІТАР
Сичова О. В.

Мета атестаційної роботи

Мета роботи – забезпечити точне визначення та відображення місцезнаходження мобільного робота на карті місцевості.

Об'єкт розробки – процес відстеження та відображення місцезнаходження мобільного робота.

Предмет розробки – програмне та апаратне забезпечення системи відстеження та відображення мобільного робота.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз способів відстеження місцезнаходження;
– провести аналіз аналогічних систем ;
– розробити архітектуру взаємодії системи з мобільним роботом;
– розробити алгоритм роботи системи;
– вибрати відповідні апаратні компоненти системи;
– виконати віртуальне підключення компонентів;
– виконати практичне підключення компонентів;
– розробити інтерфейс користувача;
– виконати тестування отриманої системи.

Опис структурної схеми системи відстеження та відображення

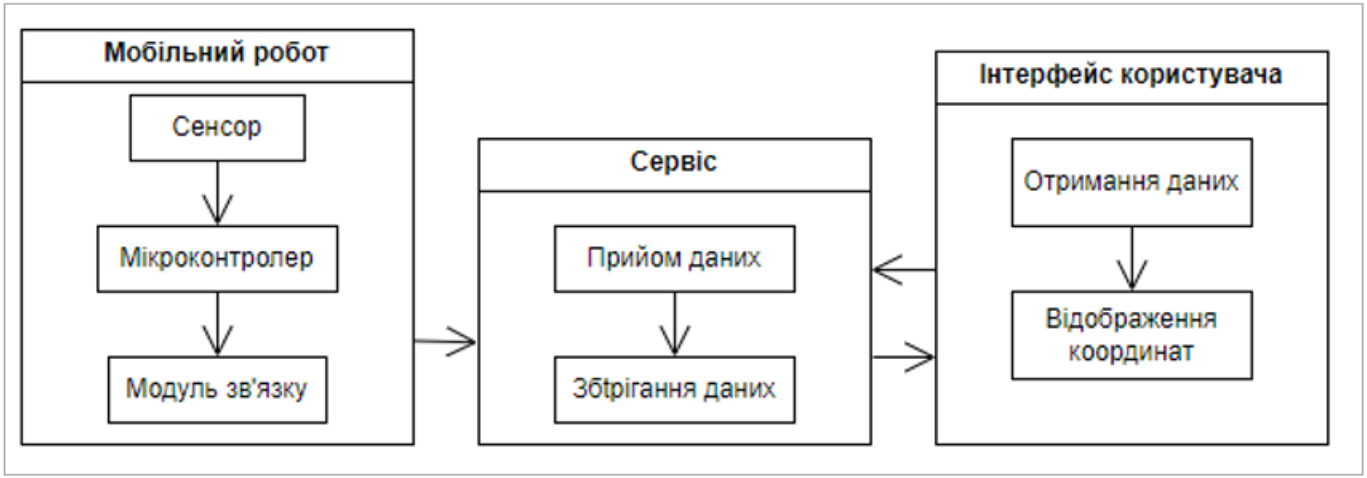

Рисунок 1 – Структурна схема системи відстеження та відображення

Вибір апаратних компонентів системи

Рисунок 2 – GPS-модуль NEO-6M

Використовується для отримання даних про місцезнаходження мобільного на карті місцевості

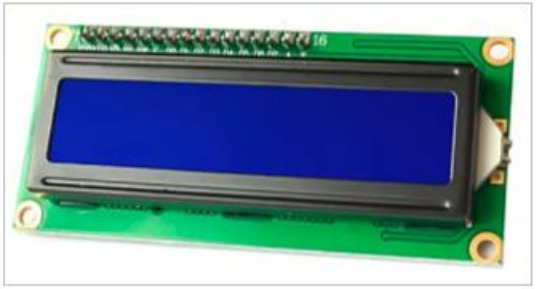

Рисунок 3 – LCD дисплей 1602

Використовується для відображення даних про місцезнаходження мобільного на карті місцевості та корегування його роботи

Рисунок 4 – Шина I2C

Використовується для значного скорочення кількості кабелів, необхідних для підключення дисплея до мікроконтролера

Вибір апаратних компонентів системи

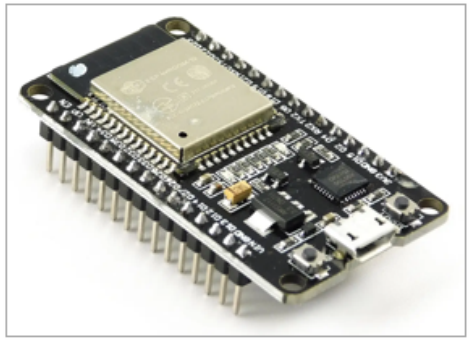

Рисунок 5 – Модуль ESP-WROOM-32

Використовується для обробки інформації місцерозташування мобільного робота та передачі відповідних даних на сервіс

Рисунок 6 – Модуль ESP32-CAM

Використовується для транслювання відеопотоку з мобільного робота на карті місцевості

Підключення апаратних компонентів системи

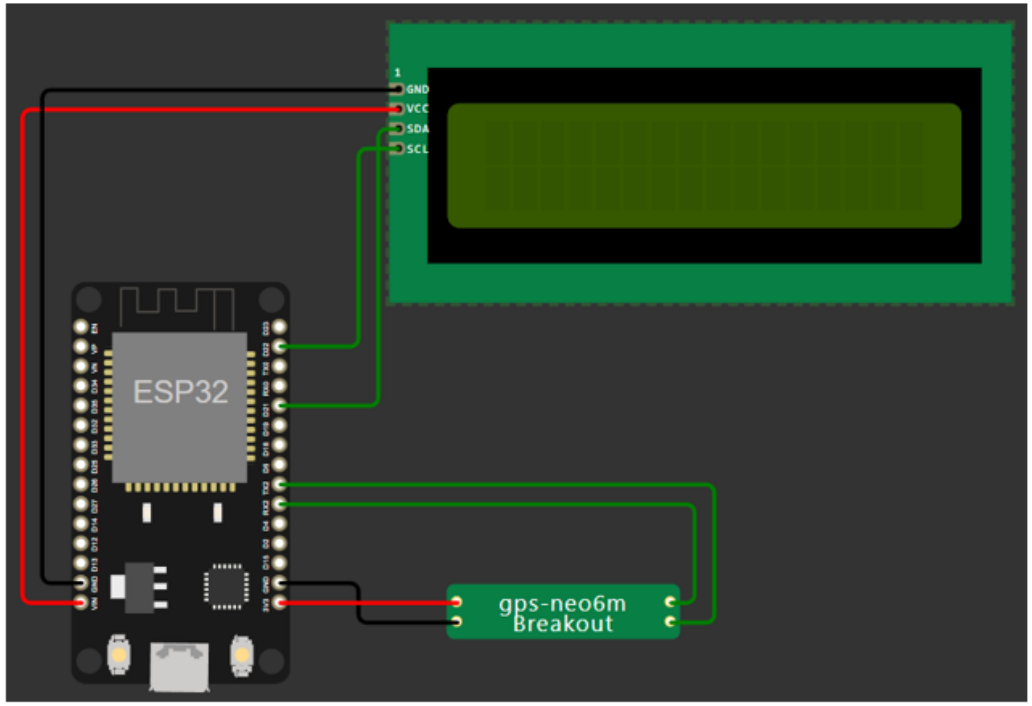

Рисунок 7 – Схема підключення апаратних компонентів системи

Зібраний фізичний макет

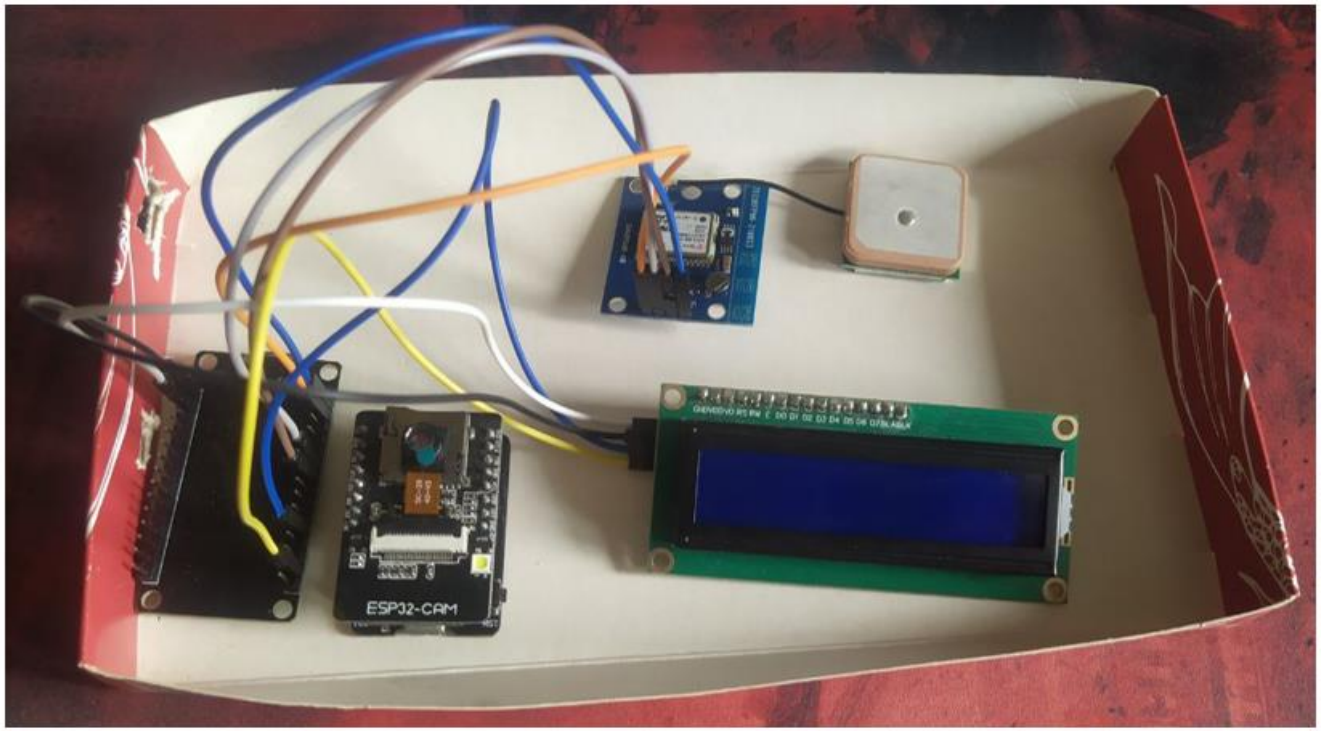

Рисунок 8 – Зібраний макет

Виконання 3D-моделі коробка для апаратних компонентів

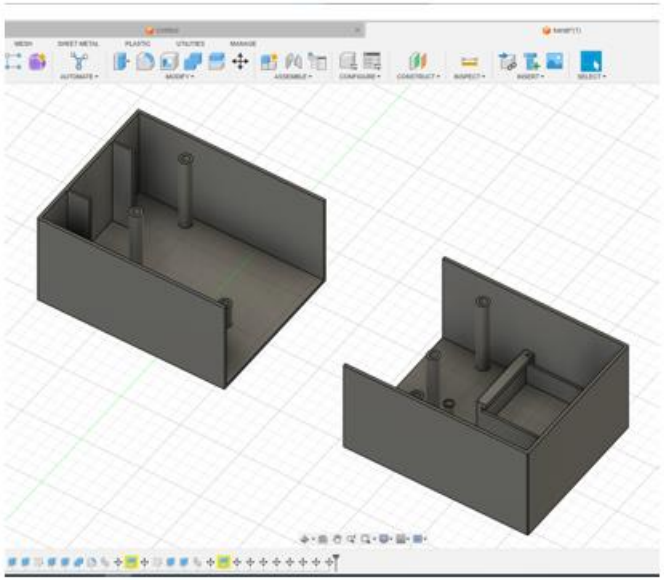

Рисунок 9 – 3D-модель коробка

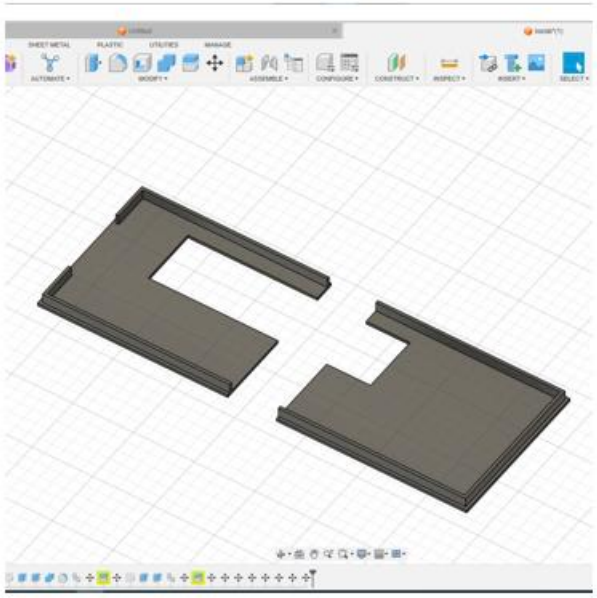

Рисунок 10 – 3D-модель кришки

Слайсування та друк 3D-моделі

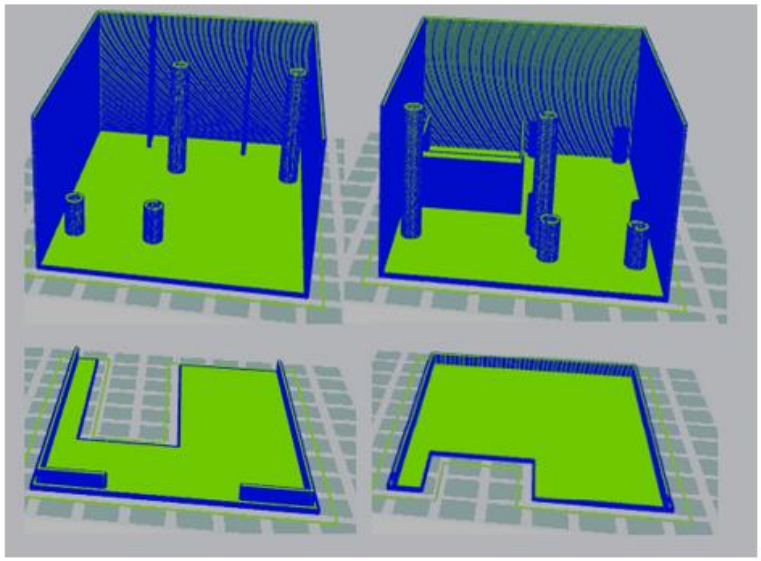

Рисунок 11 – Слайсування моделей

Рисунок 12 – Надруковані моделі

Комплектування апаратних компонентів системи

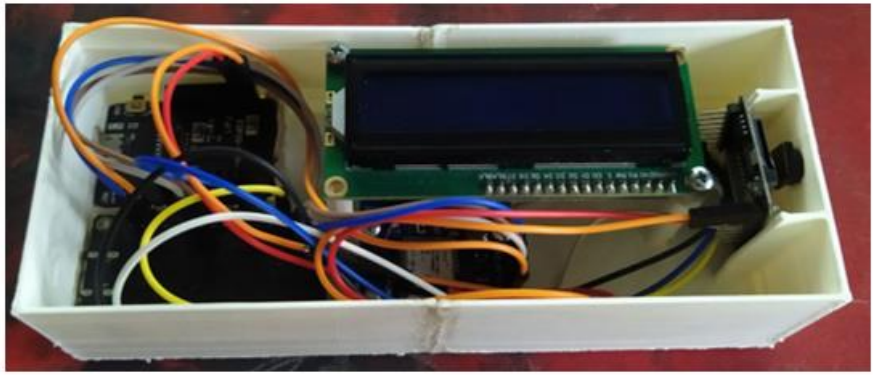

Рисунок 13 – Розташування електронних компонентів

Рисунок 14 – Повністю зібраний макет

Використання сервісу

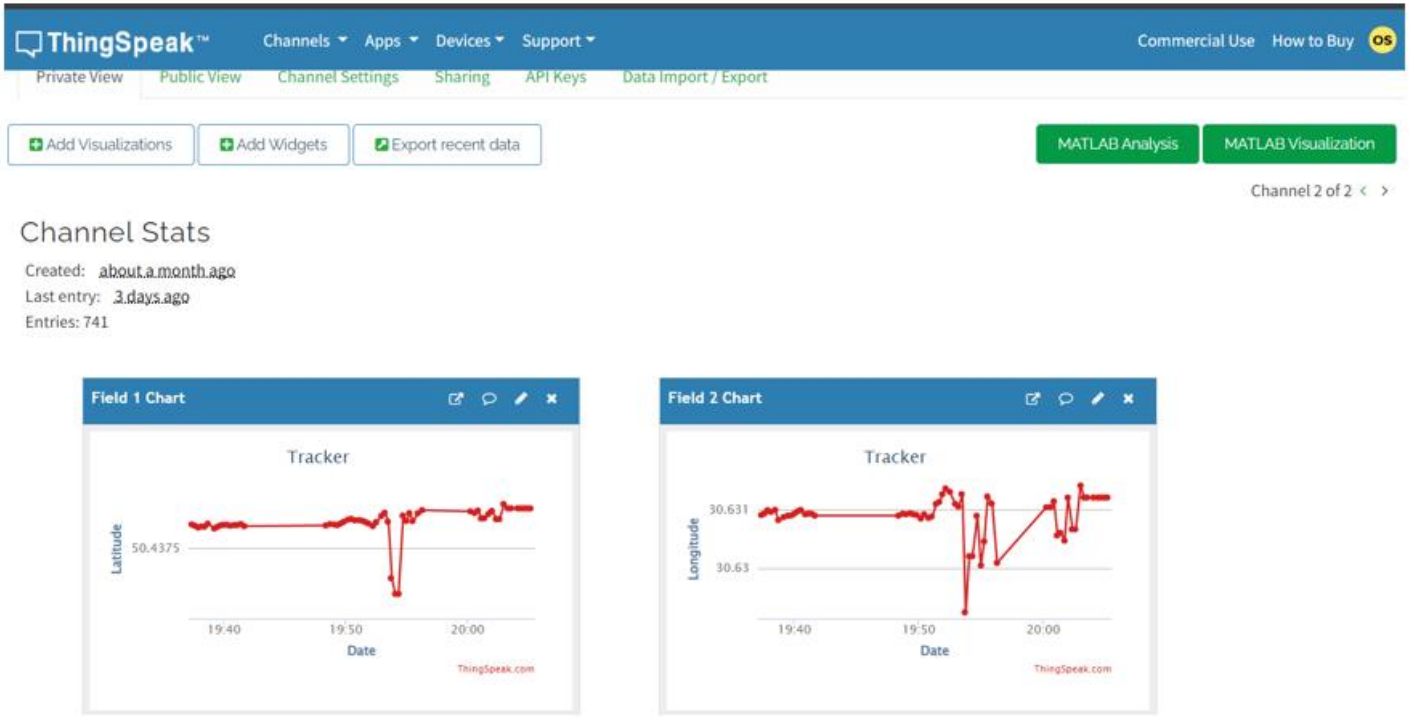

Рисунок 15 – Вигляд сервісу ThingSpeak

Використання інтерфейсу користувача

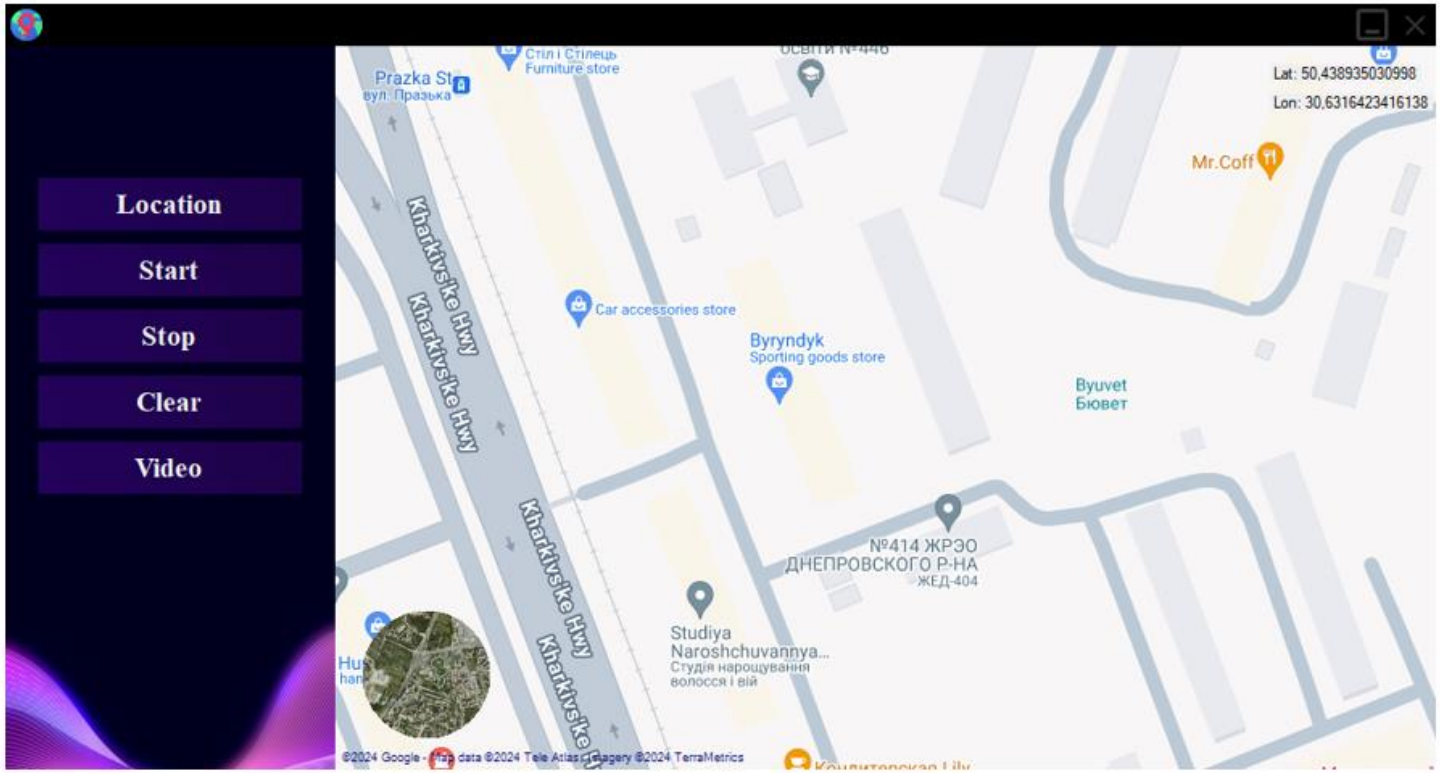

Рисунок 16 – Вигляд інтерфейсу користувача

Використання інтерфейсу користувача

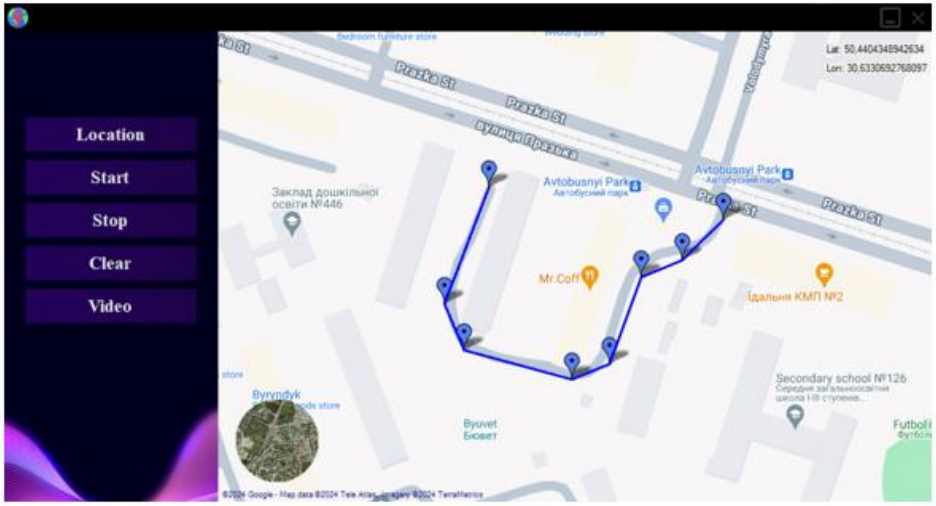

Рисунок 17 – Побудова руху мобільного робота

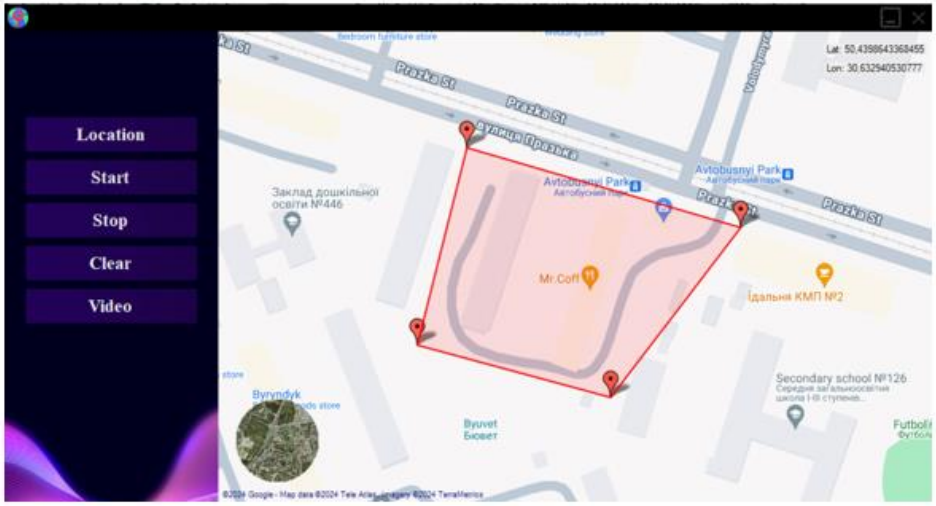

Рисунок 18 – Побудова поля

Використання інтерфейсу користувача

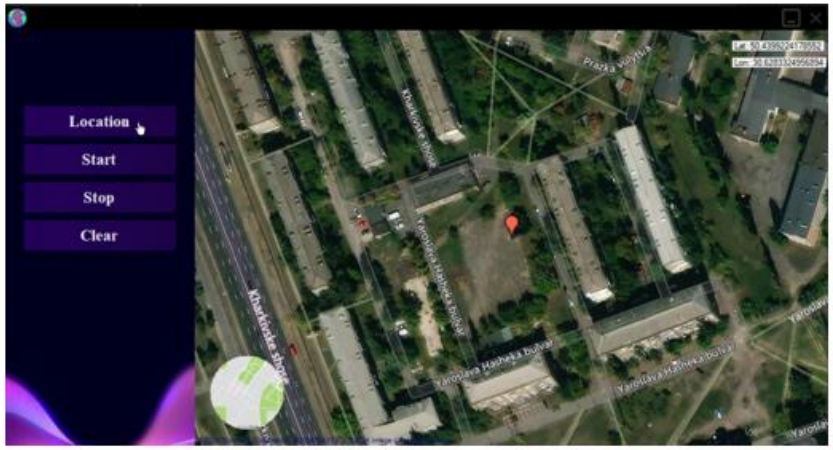

Рисунок 19 – Відображення місцезнаходження мобільного робота на карті місцевості

Рисунок 20 – Назначене поле

Використання інтерфейсу користувача

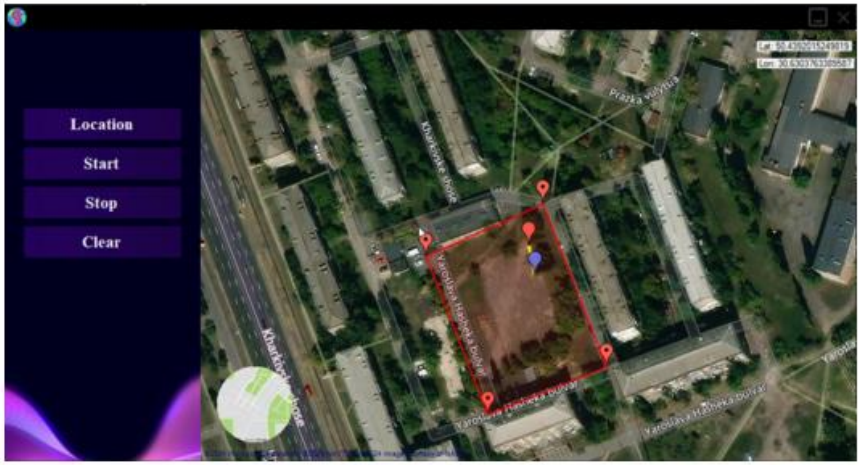

Рисунок 21 – Початок слідкування за рухом мобільного робота

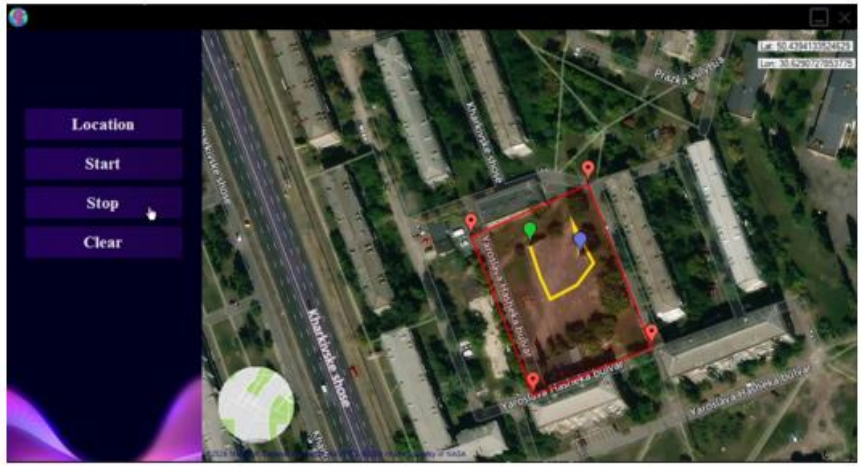

Рисунок 22 – Припинення слідкування за рухом мобільного робота

Відображення відеопотоку з мобільного роботу

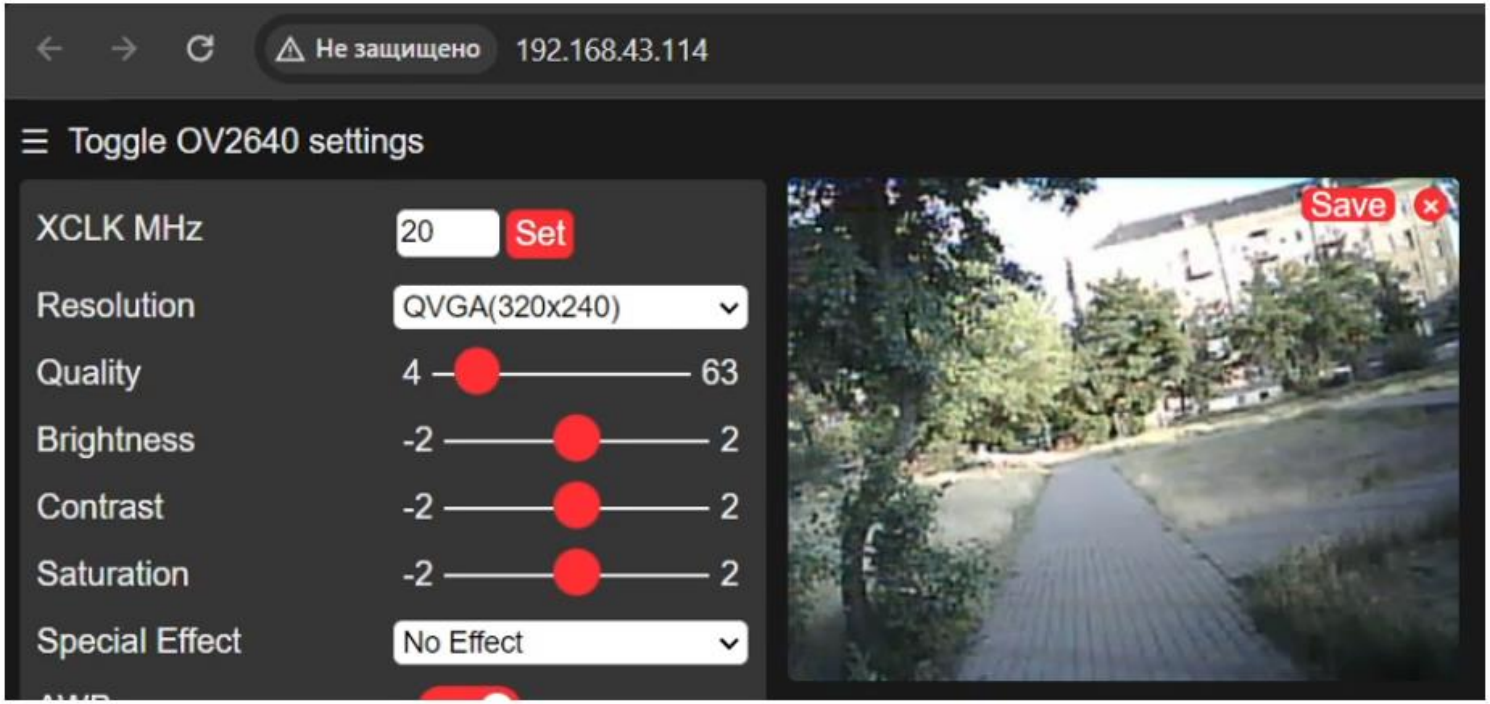

Рисунок 23 – Слідкування за рухом мобільного робота

Висновки

У ході виконання кваліфікаційної роботи було успішно реалізовано автоматизований модуль відстеження та відображення місцезнаходження мобільного робота на карті місцевості. Після аналізу технічного завдання були визначені вимоги до системи, які включали точне визначення місцезнаходження робота, підключення до картографічних сервісів та створення користувацького інтерфейсу.

На етапі розробки архітектури модуля було описано структурну схему та алгоритми його роботи, а також протоколи взаємодії з мобільним роботом. Ці рішення створили основу для побудови ефективної системи відстеження, яка може бути легко інтегрована в різні автоматизовані платформи.

Для реалізації апаратної частини було обрано такі компоненти, як GPS-модуль NEO-6M, LCD дисплей 1602 з шиною I2C, модуль ESP-WROOM-32 та модуль ESP32-CAM. Проведено симуляцію підключення у середовищі Wokwi, після чого придбано і підключено фізичні елементи відповідно до віртуального макету. Було розроблено та надруковано 3D-модель корпусу для компактного розташування електронних компонентів, що забезпечило зручність у зборці і експлуатації модуля.

Після завершення всіх етапів було проведено комплексне тестування апаратного та програмного модулів, що підтвердило високу точність і надійність відстеження та відображення місцезнаходження мобільного робота в реальному часі.

Розроблена система відстеження та відображення місцезнаходження мобільного робота повністю відповідає всім поставленим вимогам і може бути ефективно використаний у різних автоматизованих платформах. Реалізовані рішення забезпечують високу точність, надійність та інтеграцію з сучасними технологіями картографування і користувацькими інтерфейсами.

№ докум-ента	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. 467863.016 ПЗ	Пояснювальна записка	А4, 77 с.
		Додаткові матеріали	
2		Демонстраційний матеріал	А4, 17с.

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК. 467863.016 ВД			
Розроб.		Шевченко О. О.		13.06.2024	Розробка автоматизованого модуля відстеження та відображення мобільного робота на карті місцевості	Літера	Аркуш	Аркушів
Перевір.		Сичова О. В.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н. контр.		Стародубцев М. Г.						
Затв.		Нувлюдов І. Ш.						