Дрогобицький державний педагогічний університет
імені Івана Франка

кафедра фізики та інформаційних систем

Роман ЛЕШКО, Ольга ЛЕШКО, Юрій ПАВЛОВСЬКИЙ

ОСВІТНЯ РОБОТОТЕХНІКА

Методичні рекомендації до виконання індивідуальних проектних завдань

Дрогобич
2026

УДК 004.896:37.016(075.8)
ББК 74.263.2я73 + 32.816я73

Роман Лешко, Ольга Лешко, Юрій Павловський. Освітня робототехніка [Методичні рекомендації до виконання індивідуальних проектних завдань]. Дрогобич : Редакційно-видавничий відділ ДДПУ ім. І. Франка, 2026. 42с.

Методичні рекомендації **“Освітня робототехніка”** написані відповідно до робочої програми навчальної дисципліни “**Освітня робототехніка**” для підготовки фахівців магістерського рівня вищої освіти спеціальностей А4 Середня освіта (Технології та інформатика), затвердженою вченою радою Дрогобицького державного педагогічного університету імені Івана Франка.

Наведено базові рекомендації, що слугують вадливим інструментом для реалізації індивідуальних проєктів з робототехніки. Подано приклади з інструкціями та описом як реалізовувати ці проєкти.

Бібліографія 14 назв

Рецензенти:

- доцент кафедри фізики та інформаційних систем Дрогобицького державного педагогічного університету імені Івана Франка, кандидат фізико-математичних наук **Олег КУЗИК**.
- доцент кафедри технологічної та професійної освіти Дрогобицького державного педагогічного університету імені Івана Франка, кандидат фізико-математичних наук **Володимир ПОПОВИЧ**;

Відповідальний за випуск: **Віктор БРИТАН**, кандидат фізико-математичних наук, доцент кафедри фізики та інформаційних систем Дрогобицького державного педагогічного університету імені Івана Франка.

Рекомендовано до друку вченою радою Дрогобицького державного педагогічного університету імені Івана Франка як навчальний посібник
(протокол № _3_ від _26.02.2026_ р.)

З М І С Т

ПЕРЕДМОВА

Шановні студенти, викладачі та ентузіасти освітньої робототехніки!

У сучасному світі, де технології стрімко еволюціонують і проникають у всі сфери життя, освіта не може залишатися осторонь цих змін. Особливо актуальною стає інтеграція STEM-підходів (Science, Technology, Engineering, Mathematics), які не лише озброюють учнів фундаментальними знаннями з природничих наук, техніки, інженерії та математики, але й розвивають критичне мислення, креативність, навички командної роботи та вміння вирішувати реальні проблеми. Курс "Освітня робототехніка" є важливим елементом підготовки майбутніх педагогів спеціальності "Середня освіта (Технології та інформатика)", адже він поєднує теоретичні основи з практичними навичками створення робототехнічних систем, адаптованих для шкільної освіти. Завдяки цьому студенти не тільки опановують технічні аспекти, але й вчаться передавати ці знання учням, роблячи навчання інтерактивним і мотивуючим.

Ці методичні рекомендації до виконання індивідуальних проектних завдань є практичним доповненням до робочої програми дисципліни, розробленої в Дрогобицькому державному педагогічному університеті імені Івана Франка. Вони спрямовані на поглиблення розуміння принципів освітньої робототехніки, освоєння мікроконтролерів таких як Arduino та ESP32, а також на формування вмінь інтегрувати апаратні та програмні засоби для реалізації інноваційних проєктів. Метою є не лише технічна компетентність, але й педагогічна готовність: ви навчитеся адаптувати матеріал до вікових особливостей учнів, організовувати командну роботу, проводити гурткові заняття та впроваджувати робототехніку в міжпредметні STEM-уроки, поєднуючи її з фізикою, математикою,

інформатикою та технологіями. Це дозволить формувати у школярів ключові компетентності, необхідні для життя в цифрову епоху.

Індивідуальні проектні завдання, описані в програмі, – це серцевина курсу. Вони дозволяють студентам перейти від теорії до практики, реалізуючи конкретні проєкти, такі як "Розумна лампа з голосовим керуванням", "Автоматичний полив рослин" чи "Міні-метеостанція для школи". Кожен проєкт передбачає етапи планування, моделювання, програмування, тестування та презентації, що сприяє розвитку інженерного мислення, креативності та дослідницьких навичок. Ці рекомендації нададуть чіткі інструкції, приклади схем підключення, алгоритмів програмування, а також критеріїв оцінювання, аби ваша самостійна робота була ефективною, структурованою та мотивуючою. Крім того, вони враховують аспекти безпеки, етики та інтеграції з сучасними освітніми тенденціями, такими як використання відкритих бібліотек і хмарних сервісів.

Як розробники цього посібника, ми переконані, що освітня робототехніка не лише готує учнів до викликів майбутнього, але й робить навчання захопливим, практично орієнтованим і доступним для всіх. Вона відкриває двері до світу інновацій, де кожен може стати творцем, а не лише споживачем технологій. Бажаємо вам натхнення, успішних реалізацій проєктів, плідної співпраці та відкриття нових горизонтів у світі технологій!

1. Проєкт “Пульсометр” (вимірювання частоти серцевих скорочень і вивід на дисплей).”

Мета:

Метою проекту є створення пристрою для вимірювання частоти серцевих скорочень (BPM) за допомогою оптичного датчика пульсу та виведення середнього значення на LCD-дисплей. Проєкт демонструє інтеграцію сенсорів, обробку сигналів та візуалізацію даних у реальному часі, що сприяє розвитку навичок у біомедичній електроніці та програмуванні.

Обладнання:

1. Макетна плата.
2. Платформа Arduino Nano.
3. Датчик пульсу MAX30105 (або аналогічний MAX30102 з інфрачервоним та червоним світлодіодами для вимірювання пульсу через шкіру).
4. LCD-дисплей 16x2 з I2C-інтерфейсом (адреса 0x27).
5. З'єднувальні проводи.
6. Резистори (якщо потрібно для стабілізації сигналу, але в базовій схемі не обов'язково).

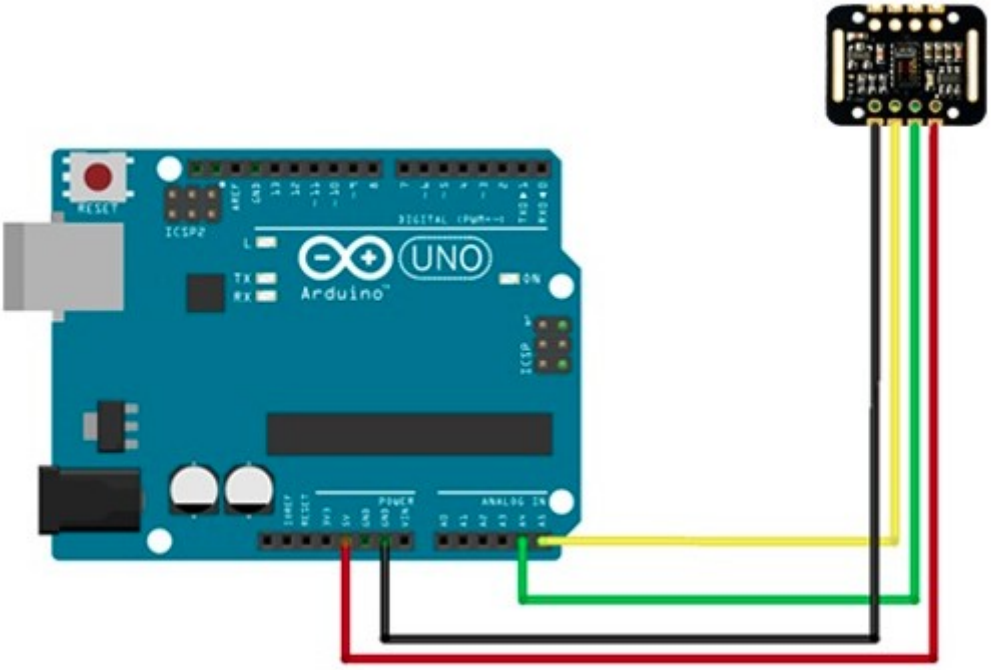
Реалізація проекту

Для реалізації розглянемо спочатку як працюють окремі елементи, що будуть використані у проєкті.

Для моделювання використовується макетна плата BREADBOARD MB-10, особливості функціонування якої можна знайти в [1, 2]. З сімейств

Arduino використовується платформа Arduino Nano, основні принципи роботи якої також описано в [1–4].

Датчик пульсу MAX30105 [4–7] є оптичним сенсором, який вимірює зміни в кровотоку за допомогою інфрачервоного (IR) та червоного світла. Він підключається через I2C-інтерфейс: VCC до 5V, GND до GND, SCL до A5 (SCL), SDA до A4 (SDA). Цей датчик виявляє наявність пальця (якщо IR-значення > 50000) та розраховує удари серця за допомогою бібліотеки heartRate.h. Схематично у стилі tinkercad [8] підключення має вигляд:



LCD-дисплей з I2C-інтерфейсом дозволяє виводити текст на екран. Він підключається аналогічно: VCC до 5V, GND до GND, SCL до A5, SDA до A4 (спільний I2C-bus з датчиком). Бібліотека LiquidCrystal_I2C керує дисплеєм.

Схема підключення має вигляд:

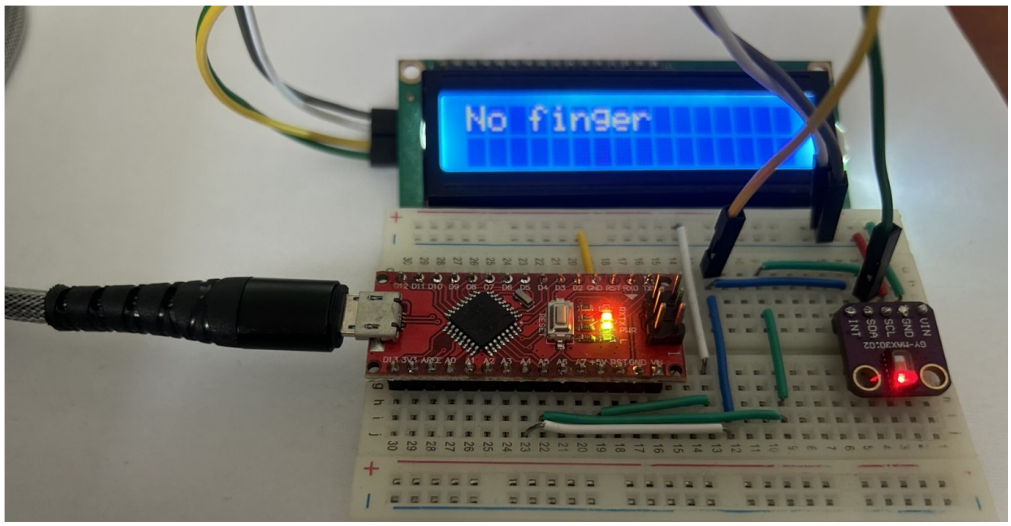
Arduino Nano: Живлення від USB або 5V.

MAX30105: VCC → 5V, GND → GND, SCL → A5, SDA → A4.

LCD I2C: VCC → 5V, GND → GND, SCL → A5, SDA → A4.

Аналоговий пін A0: Використовується для додаткового моніторингу (наприклад, кнопки), але в базовому коді виводить значення в Serial для налагодження.

Візуально це виглядає так:



Код, що дає змогу вимірювати пульс та виводити на дисплей, має вигляд:

```
#include <Wire.h>
#include "MAX30105.h"
#include "heartRate.h"
#include <LiquidCrystal_I2C.h>

int An = A0;

MAX30105 particleSensor;
LiquidCrystal_I2C lcd(0x27, 16, 2);

const unsigned long measurementTime = 20000; // 20 секунд
unsigned long startTime = 0;
int beatCount = 0;
float sumBPM = 0;

bool measuring = false;
bool showAverage = false;
float averageBPM = 0;
bool fingerDetected = false;

void setup() {
  pinMode(An, INPUT_PULLUP);
  Serial.begin(9600);

  lcd.init();
  lcd.backlight();
```

```
  lcd.setCursor(0, 0);
  lcd.print("Initializing...");

  if (!particleSensor.begin(Wire, I2C_SPEED_STANDARD)) {
    lcd.clear();
    lcd.print("MAX30102 ERROR!");
    Serial.println("MAX30102 not found. Check wiring!");
    while (1);
  }

  particleSensor.setup();
  particleSensor.setPulseAmplitudeRed(0x3F);
  particleSensor.setPulseAmplitudeIR(0x3F);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("No finger");
  lcd.setCursor(0, 1);
  lcd.print("                ");
  delay(1000);
}

void loop() {

  Serial.println(analogRead(An)); // просто десяткове число

  long irValue = particleSensor.getIR();

  if (irValue < 50000) {
    // Палець прибрали
    fingerDetected = false;
    measuring = false;
    showAverage = false;
    beatCount = 0;
    sumBPM = 0;
    lcd.setCursor(0, 0);
    lcd.print("No finger       ");
    lcd.setCursor(0, 1);
    lcd.print("                "); // очищаємо другий рядок
    return;
  } else {
    // Палець на датчику
    if (!fingerDetected) {
      fingerDetected = true;
      lcd.setCursor(0, 0);
      lcd.print("Measuring...    ");
      startTime = millis();
      beatCount = 0;
      sumBPM = 0;
      measuring = true;
    }
```

```
  }

  if (measuring) {
    if (checkForBeat(irValue)) {
      static unsigned long lastBeat = 0;
      unsigned long delta = millis() - lastBeat;
      lastBeat = millis();
      float bpm = 60.0 / (delta / 1000.0);

      if (bpm > 40 && bpm < 180) {
        beatCount++;
        sumBPM += bpm;
      }
    }

    // Перевірка часу вимірювання
    if (millis() - startTime >= measurementTime) {
      measuring = false;
      showAverage = true;

      lcd.clear();
      lcd.setCursor(0, 0);

      if (beatCount > 0) {
        averageBPM = sumBPM / beatCount;
        lcd.print("Average BPM:");
        lcd.setCursor(0, 1);
        lcd.print(averageBPM, 0);

        //Serial.print("Average BPM = ");
        //Serial.println(averageBPM, 0);
      } else {
        lcd.print("Pulse absent");
        lcd.setCursor(0, 1);
        lcd.print("               ");
        Serial.println("Pulse absent");
      }
    }
  }
}
```

Проаналізуємо код. У блоку setup ініціалізуємо піни, Serial для налагодження, LCD-дисплей та датчик MAX30105. Якщо датчик не знайдено, виводимо помилку та зупиняємо програму. Встановлюємо амплітуду світлодіодів для кращої чутливості.

У блоці loop зчитуємо IR-значення з датчика для виявлення пальця. Якщо палець відсутній (irValue < 50000), скидаємо вимірювання та виводимо "No finger" на дисплей. Якщо палець виявлено, починаємо 20-секундне вимірювання.

Функція checkForBeat(irValue) (з бібліотеки heartRate.h) виявляє удари серця. Розраховуємо BPM як 60 / (час між ударами в секундах). Фільтруємо значення (40-180 BPM) для уникнення помилок. Після 20 секунд виводимо середнє BPM на дисплей.

Аналоговий пін A0 використовується для додаткового моніторингу (вивід у Serial), але не є критичним для основної функції.

Цей проєкт інтегрує знання з фізики (оптика, сигналів), математики (обчислення середнього) та інформатики (програмування, обробка даних), роблячи його ідеальним для STEM-освіти.

Практично робочий прототип можна переглянути за посиланням: https://drive.google.com/file/d/1OWq6gzTpZMkHswFCkPe1hgKfpNniqF1E/view.

2. Проєкт “Радар для відстеження об’єктів з сервоприводом”

Мета:

Метою проекту є створення радарної системи, яка обертає ультразвуковий датчик за допомогою серводвигуна та візуалізує на екрані відстань до об’єкта в залежності від кута. Такий проект дозволяє ознайомитися з основами робототехніки, сенсорів та інтеграції апаратного та програмного забезпечення.

Обладнання:

1. Макетна плата.
2. Платформа Arduino (Arduino NANO або сумісна).
3. Серводвигун (Servo).
4. Ультразвуковий датчик відстані HC-SR04.
5. З’єднувальні проводи.
6. Комп’ютер з встановленим Processing для візуалізації даних.

Реалізація проекту

Для реалізації розглянемо спочатку як працюють окремі елементи, що будуть використані у проекті.

Для моделювання використовується макетна плата BREADBOARD MB-10, особливості функціонування якої можна знайти в [1, 2]. З сімейств Arduino використовується платформа Arduino-NANO, основні принципи роботи якої також описано в [1–4].

Розглянемо підключення компонентів. Серводвигун під’єднується до цифрового піна 9 Arduino і служить для повороту датчика від 0° до 180° і назад. Ультразвуковий датчик підключається за стандартною схемою: VCC до 5 В, GND до землі, TRIG до піна 2 (вихід), ECHO до піна 3 (вхід). При

цьому датчик випромінює ультразвуковий сигнал у момент, коли на TRIG подається короткий імпульс, і чекає на відбитий сигнал на піні ECHO.

Схема підключення має вигляд:

Arduino HC-SR04

5V → VCC

GND → GND

Pin 2 → TRIG

Pin 3 → ECHO

Pin 9 → Servo Signal

Таке підключення дозволяє Arduino керувати обертанням серво та вимірювати відстань до об’єктів на різних кутах.

Ультразвуковий датчик HC-SR04 працює за принципом ехолокації. Коли на контакт TRIG подається короткий сигнал тривалістю 10 мікросекунд, датчик випромінює серію ультразвукових хвиль у напрямку об’єкта. Коли хвиля відбивається від об’єкта і повертається назад, контакт ECHO отримує сигнал HIGH на час, пропорційний відстані до об’єкта. Arduino вимірює тривалість цього сигналу і, знаючи швидкість звуку (343 м/с), обчислює відстань у сантиметрах.

Код для Arduino виконує три основні функції:

1. Поворот серводвигуна від 0° до 180° і назад.

2. Вимірювання відстані до об’єктів на кожному куті.

3. Передача даних у комп’ютер через серійний порт у форматі «кут, відстань».

Програма організована так, що цикл руху серво плавний і забезпечує затримку для стабілізації руху, а функції calculateDistance() та printData() спрощують логіку коду, роблячи його зручним для розширення. Також використовується бібліотека Servo.h для керування сервоприводом. Код на мові c++ [9] для Arduino NANO подано нижче:

```
#include <Servo.h>

const int trigPin = 2;
const int echoPin = 3;
long duration;
int distance;
Servo myServo;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
  myServo.attach(9);
}

void loop() {
  for (int i = 0; i <= 180; i++) {
    distance = calculateDistance();
    printData(i, distance);
    myServo.write(i);
    delay(10);
  }
  for (int i = 180; i >= 0; i--) {
    distance = calculateDistance();
    printData(i, distance);
    myServo.write(i);
    delay(10);
  }
}

void printData(int angle, int dist) {
  Serial.print(angle);
  Serial.print(",");
  Serial.print(dist);
  Serial.print(".");
}

int calculateDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration / 58;
  return distance;
}
```

Проаналізуємо код. У блоку setup ініціалізуємо піни для датчика (trigPin як OUTPUT, echoPin як INPUT), серійний порт на 9600 бод та приєднуємо серводвигун до піну 9.

У блоці loop серво обертається від 0° до 180° і назад з кроком 1°, на кожному куті вимірюється відстань (calculateDistance()), дані виводяться в серійний порт (printData()), серво встановлюється в позицію (myServo.write(i)), з затримкою 10 мс для стабільності.

Функція calculateDistance надсилає імпульс на trigPin, зчитує тривалість на echoPin і розраховує відстань (duration / 58, оскільки швидкість звуку ≈ 29 мкс/см в один бік, /2 для туди-назад).

Функція printData виводить кут, кому, відстань, крапку для розмежування пакетів даних.

На стороні комп’ютера ми використовуємо Processing для отримання даних і візуалізації. Дані надходять через серійний порт і перетворюються у координати точок для малювання радарної діаграми. На екрані червона точка показує місце розташування об’єкта, а зелена лінія відображає поточний кут серво.

Особливістю візуалізації є плавне зникнення попередніх ліній, що створює ефект радарного “сліду” та робить систему більш наочною. Програмний код у середовищі Processing написаний на мові java має вигляд:

```
import processing.serial.*;

Serial myPort;
String angle="", distance="", data="";
float pixsDistance;
int iAngle, iDistance;

void setup() {
  size(1200, 700);
  smooth();
  myPort = new Serial(this, Serial.list()[3], 9600); // порт може 1,2,3. Тут 3 як приклад
  myPort.bufferUntil('.');
}
```

```
void draw() {
  fill(0, 10);
  rect(0, 0, width, height); // плавне затемнення попередніх слідів
  drawRadar();
  drawLine();
  drawObject();
}

void serialEvent(Serial myPort) {
  data = myPort.readStringUntil('.');
  data = data.substring(0, data.length()-1);
  int index1 = data.indexOf(",");
  angle = data.substring(0, index1);
  distance = data.substring(index1+1, data.length());
  iAngle = int(angle);
  iDistance = int(distance);
}

void drawRadar() {
  pushMatrix();
  translate(width/2, height-height*0.074);
  noFill();
  stroke(30, 15, 255);
  strokeWeight(2);
  for(int i=0; i<5; i++) {
    arc(0,0, width*0.5 + i*50, width*0.5 + i*50, PI, TWO_PI);
  }
  popMatrix();
}

void drawLine() {
  pushMatrix();
  translate(width/2, height-height*0.074);
  stroke(30, 250, 60);
  strokeWeight(3);
  line(0,0, 200*cos(radians(iAngle)), -200*sin(radians(iAngle)));
  popMatrix();
}

void drawObject() {
  pushMatrix();
  translate(width/2, height-height*0.074);
  stroke(255,0,0);
  strokeWeight(9);
  pixsDistance = iDistance*5;
  if (iDistance < 40) {
    point(pixsDistance*cos(radians(iAngle)), -pixsDistance*sin(radians(iAngle)));
  }
  popMatrix();
}
```

Принцип роботи системи:

1. Серводвигун обертає ультразвуковий датчик, вимірюючи відстань до об’єктів на різних кутах.

2. Дані надходять до комп’ютера через серійний порт у форматі «кут, відстань».

3. Processing перетворює кут і відстань у координати на екрані та малює радарну діаграму.

4. Попередні дані плавно зникають, створюючи ефект рухомого радарного променя.

Цей робототехнічний проект призначений для створення простої системи радара, яка імітує роботу реальних радіолокаційних систем. Він допомагає студентам та ентузіастам зрозуміти принципи ехолокації, роботи серводвигунів, ультразвукових датчиків та інтеграції апаратного забезпечення з програмним (Arduino + Processing). Проєкт розвиває навички програмування, електроніки та візуалізації даних, сприяючи формуванню інженерного мислення. Він може бути використаний у шкільній чи університетській освіті для демонстрації концепцій робототехніки, фізики (поширення хвиль) та математики (обчислення координат). Крім того, проєкт служить основою для подальших удосконалень, таких як автономні роботизовані системи або системи моніторингу середовища.

Практично робочий прототип можна переглянути за посиланням:
https://drive.google.com/file/d/1bDCYztHxyiqUeIYkDGtNSFW5GkrYsG48/view

3. Проєкт “Розумний смітник”

Мета:

Метою проєкту є створення автоматизованого смітника, який відкриває та закриває кришку без фізичного контакту користувача. Керування здійснюється за допомогою ультразвукового датчика відстані та серводвигуна. Проєкт демонструє принципи роботи сенсорів відстані, виконавчих механізмів і логіки станів у вбудованих системах, а також сприяє формуванню практичних навичок у галузі електроніки та програмування мікроконтролерів.

Обладнання:

1. Макетна плата (breadboard).
2. Платформа Arduino Uno / Nano.
3. Ультразвуковий датчик відстані HC-SR04.
4. Серводвигун SG90 (або аналогічний мікросерводвигун).
5. З’єднувальні проводи (male–male).
6. Джерело живлення (USB або 5 V).

Реалізація проекту

Для реалізації проєкту розглянемо принципи роботи окремих компонентів, що використовуються у системі.

Для моделювання використовується макетна плата breadboard, яка дозволяє швидко та без паяння з’єднувати електронні компоненти. Як керуючий пристрій застосовується плата Arduino, яка виконує вимірювання відстані, аналізує отримані дані та керує серводвигуном.

Ультразвуковий датчик HC-SR04 працює за принципом ехолокації. Він випромінює ультразвуковий імпульс і вимірює час, за який відбитий

сигнал повертається до приймача. Відстань до об’єкта обчислюється за формулою:

$$d = 1/2\ Vt,$$

де V — швидкість звуку в повітрі, t — тривалість імпульсу. Для підключення датчика використовують такі піни:

а) VCC → 5V;

б) GND → GND;

в) TRIG → цифровий пін D3;

г) ECHO → цифровий пін D2.

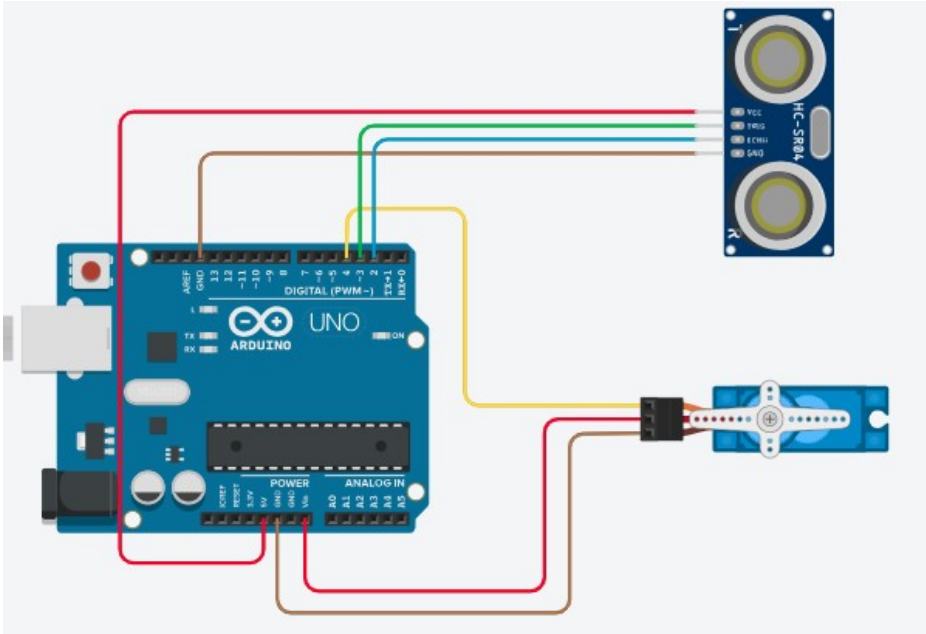
Серводвигун використовується для механічного відкривання та закривання кришки смітника. Він керується ШІМ-сигналом, що дозволяє точно задавати кут повороту в діапазоні 0–180°. У проєкті використовуються два основні положення: 0° — кришка закрита; 90° — кришка відкрита. Для підключення серводвигуна використовують такі піни:

а) червоний провід → 5V;

б) коричневий/чорний → GND;

в) жовтий/білий (керування) → цифровий пін D4.

Схематично у середовищі tinkercad схема має вигляд:



Код програми реалізовано мовою C++ з використанням стандартної бібліотеки Servo.h.

```
#include <Servo.h>

const byte echoPin = 2;
const byte trigPin = 3;
const byte servoPin = 4;

Servo myServo;
bool objectDetected = false;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
  myServo.attach(servoPin);
  myServo.write(0);
}

void loop() {
  float distance = measureDistance();

  if (distance < 15) {
    if (!objectDetected) {
      myServo.write(90);
      objectDetected = true;
      Serial.println("Object detected! Opening...");
    }
    delay(2000);
  } else {
    if (objectDetected) {
      myServo.write(0);
      objectDetected = false;
      Serial.println("Object lost! Closing...");
    }
    delay(100);
  }
}

float measureDistance() {
  long duration;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  return duration * 0.0343 / 2;
}
```

На початку роботи програми, у блоці setup(), Arduino готується до роботи. Тут ми вказуємо, які піни будуть використовуватись для ультразвукового датчика, а також підключаємо серводвигун. Серводвигун одразу встановлюється у початкове положення — кришка смітника залишається закритою. Також вмикається Serial Monitor, який допомагає спостерігати за роботою програми під час налагодження.

Основна логіка роботи знаходиться у блоці loop(), який постійно повторюється. У цьому блоці програма вимірює відстань до найближчого об’єкта за допомогою ультразвукового датчика. Якщо до датчика піднести руку на відстань менше ніж приблизно 15 см, Arduino «розуміє», що користувач хоче відкрити смітник.

Коли об’єкт з’являється вперше, серводвигун повертається на 90 градусів і відкриває кришку. Щоб кришка не відкривалась і не закривалась багато разів підряд, використовується спеціальна змінна, яка запам’ятовує, чи був уже виявлений об’єкт. Це робить роботу смітника плавною та стабільною. Коли рука забирається і об’єкт більше не знаходиться перед датчиком, програма повертає серводвигун у початкове положення, і кришка смітника закривається.

Окрема функція measureDistance() відповідає за вимірювання відстані. Вона надсилає короткий ультразвуковий сигнал, приймає відбитий сигнал та обчислює, на якій відстані знаходиться об’єкт. Отримане значення використовується для прийняття рішення — відкривати смітник чи ні.

Запропонований проєкт демонструє практичне застосування ультразвукових сенсорів та сервоприводів у побутових автоматизованих системах. Він поєднує знання з акустинки, математики та інформатики (алгоритмічне мислення й програмування), що робить його ефективним прикладом STEM-проєкту. Отриманий прототип є функціональним і може бути використаний як основа для подальшого розширення (додавання LCD, індикаторів, живлення від батареї тощо).

4. Проєкт “IoT-контролер: Керування освітленням через Web-інтерфейс”

Мета:

Метою проєкту є створення системи Internet of Things — IoT на базі мікроконтролера ESP8266, яка дозволяє дистанційно керувати групою світлодіодів через локальну Wi-Fi мережу. Керування здійснюється за допомогою графічного веб-інтерфейсу, доступного через браузер на смартфоні або комп’ютері. Проєкт демонструє принципи роботи веб-серверів на вбудованих системах, використання протоколу HTTP, роботу з GPIO-портами, а також основи створення динамічних веб-сторінок з використанням HTML, CSS та JavaScript (технологія AJAX). Це сприяє формуванню навичок у галузі мережевих технологій, веб-дизайну та програмування мікроконтролерів.

Обладнання:

1. Плата розробника NodeMCU v3 (на базі чипа ESP8266).
2. Макетна плата (breadboard).
3. Світлодіоди (червоний, жовтий, зелений).
4. Резистори 220–330 Ом (3 шт.).
5. З’єднувальні проводи (male–male).
6. Кабель Micro-USB для програмування та живлення.
7. Комп’ютер зі встановленим середовищем Arduino IDE.

Реалізація проекту

1. Теоретичні відомості про платформу

Основою проєкту є платформа NodeMCU, побудована на базі мікроконтролера ESP8266. На відміну від класичної Arduino Uno, цей

контролер має вбудований Wi-Fi модуль, працює на тактовій частоті 80 МГц (проти 16 МГц у Arduino) та має більший обсяг флеш-пам'яті (4 МБ). Це дозволяє не тільки керувати електронікою, а й розгортати повноцінний веб-сервер прямо на чипі. Особливістю плати є те, що вона працює з логічною напругою 3.3 В (на відміну від 5 В у Arduino), тому підключення датчиків вимагає уважності.

2. Підготовка програмного середовища (Arduino IDE)

Оскільки Arduino IDE за замовчуванням не підтримує ESP8266, необхідно виконати процедуру налаштування.

Етап А: Додавання підтримки плат

1. Відкрити Arduino IDE, перейти у меню Файл (File) → Налаштування (Preferences).

2. У полі «Додаткові посилання для Менеджера плат» вставити URL: http://arduino.esp8266.com/stable/package_esp8266com_index.json

3. Перейти в Інструменти (Tools) → Плата → Менеджер плат.

4. У пошуку ввести «esp8266», знайти пакет ESP8266 Community та натиснути Install (Встановити).

Етап Б: Вирішення проблем сумісності (Linux/Python). Якщо ви працюєте на Linux, то при компіляції можуть виникати помилки (наприклад, elf2bin error), це свідчить про відсутність необхідних бібліотек Python, який використовується для збірки прошивки. Необхідно оновити залежності через термінал:

```
sudo apt update
sudo apt install python3 python3-pip
pip3 install pyserial
```

Також слід переконатися, що користувач має права доступу до COM-порту (команда sudo usermod -a -G dialout $USER).

Етап В: Налаштування підключення.

1. Під’єднати плату через USB.

2. В меню Інструменти → Плата вибрати NodeMCU 1.0 (ESP-12E Module).

3. В меню Порт вибрати відповідний COM-порт.

4. Встановити швидкість завантаження (Upload Speed) на 115200 або 921600.

3. Схема підключення компонентів

У цьому проекті ми використаємо 3 свілодіоди і налаштуємо керування ними через WiFi та веб-інтерфейс. У NodeMCU маркування пінів на платі (D0, D1...) відрізняється від номерів GPIO (General Purpose Input/Output), які використовуються в коді. У цьому проєкті використано так конфігурацію:

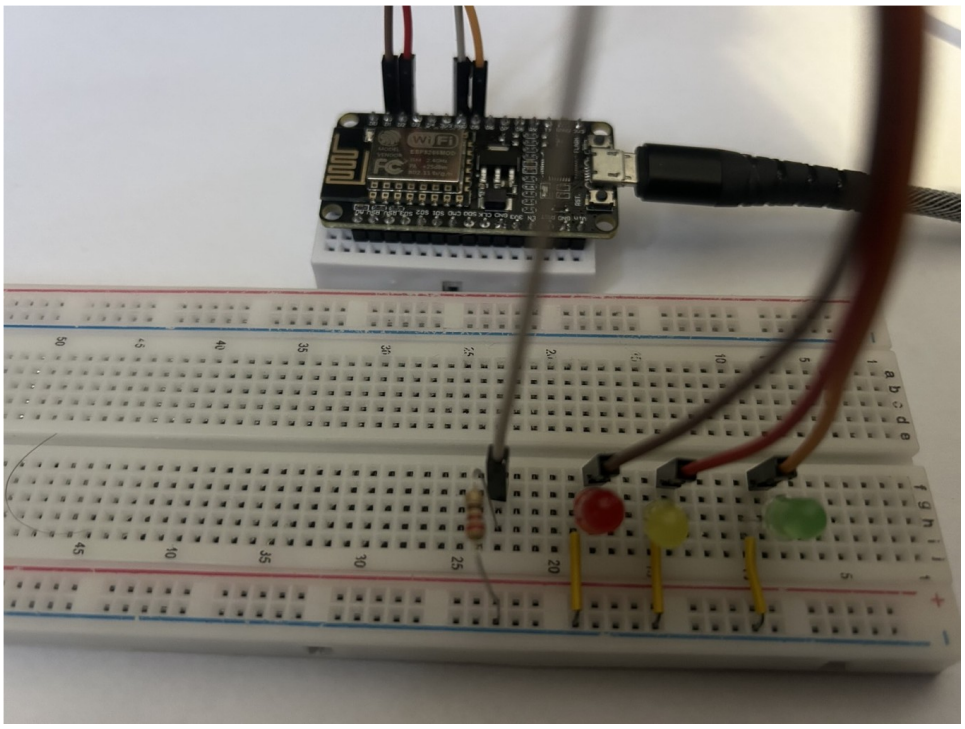
1. Червоний світлодіод: анод (+) до піна D1 (GPIO 5), катод (-) через резистор до GND.

2. Жовтий світлодіод: анод (+) до піна D2 (GPIO 4), катод (-) через резистор до GND.

3. Зелений світлодіод: анод (+) до піна D5 (GPIO 14), катод (-) через резистор до GND.

4. Вбудований синій світлодіод: розпаяний на платі, підключений до піна D4 (GPIO 2).

Резистор 220 Ом необхідний для обмеження струму, щоб запобігти виходу з ладу світлодіодів або портів мікроконтролера.



4. Код програми

Програма реалізована мовою C++ з використанням бібліотек ESP8266WiFi (для підключення до мережі) та ESP8266WebServer (для обробки HTTP-запитів).

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

// Налаштування Wi-Fi
const char* ssid = "oooo"; // НАЗВА WIFI мережі
const char* password = "password"; // ПАРОЛЬ ЦІЄЇ МЕРЕЖІ

ESP8266WebServer server(80); // Створення веб-сервера на порту 80

// Визначення пінів (відповідність D-пінів та GPIO)
const int ledBuiltin = 2;  // D4 (Вбудований LED)
const int ledRed     = 5;  // D1
const int ledYellow  = 4;  // D2
const int ledGreen   = 14; // D5
```

```
// Функція генерації головної сторінки
void handleRoot() {
  String html = "<!DOCTYPE html><html><head>";
  // Мета-теги для коректного відображення кирилиці та адаптації під мобільні
  html += "<meta charset='utf-8'><meta name='viewport' content='width=device-width,
initial-scale=1'>";

  // CSS-стилі для гарного оформлення інтерфейсу
  html += "<style>";
  html += "body { font-family: sans-serif; text-align: center; background-color: #f0f2f5;
margin: 0; padding: 20px; }";
  html += ".container { max-width: 400px; margin: auto; background: white; padding: 25px;
border-radius: 20px; box-shadow: 0 10px 25px rgba(0,0,0,0.1); }";
  html += "h1 { color: #1c1e21; font-size: 24px; margin-bottom: 30px; }";
  html += ".row { display: flex; align-items: center; justify-content: space-between; margin-
bottom: 20px; padding: 10px; border-radius: 12px; background: #fafafa; }";
  html += ".label { font-weight: bold; font-size: 16px; }";
  html += ".btn-group { display: flex; gap: 10px; }";
  // Стилі кнопок
  html += ".btn { padding: 12px 18px; border: none; border-radius: 8px; cursor: pointer; font-
weight: bold; transition: 0.2s; min-width: 70px; }";
  html += ".btn-on { background-color: #42b72a; color: white; }";
  html += ".btn-on:active { transform: scale(0.95); background-color: #36a420; }"; // Ефект
натискання
  html += ".btn-off { background-color: #eb4034; color: white; }";
  html += ".btn-off:active { transform: scale(0.95); background-color: #d1342a; }";
  // Кольори для тексту підписів
  html += ".red { color: #eb4034; } .yellow { color: #f1c40f; } .green { color:
#42b72a; } .blue { color: #1877f2; }";
  html += "</style>";

  // JavaScript для асинхронних запитів (AJAX) - керування без перезавантаження
  html += "<script>";
  html += "function toggle(path) {";
  html += "  fetch(path).then(response => console.log('Команда надіслана:', path));";
  html += "}";
  html += "</script>";

  html += "</head><body>";
  html += "<div class='container'>";
  html += "<h1>Smart Home Control</h1>";

  // Допоміжна лямбда-функція для генерації рядків HTML коду з кнопками
  auto makeRow = [](String name, String colorClass, String pathPrefix) {
    String row = "<div class='row'><span class='label " + colorClass + "'>" + name +
"</span>";
    row += "<div class='btn-group'>";
    // Кнопки викликають JS функцію toggle()
    row += "<button class='btn btn-on' onclick=\"toggle('/" + pathPrefix +
"/on')\">ВКЛ</button>";
```

```
    row += "<button class='btn btn-off' onclick=\"toggle('/" + pathPrefix +
"/off')\">ВИКЛ</button>";
    row += "</div></div>";
    return row;
  };

  // Формування блоків керування для кожного кольору
  html += makeRow("Червоний", "red", "red");
  html += makeRow("Жовтий", "yellow", "yellow");
  html += makeRow("Зелений", "green", "green");
  html += makeRow("Вбудований", "blue", "builtin");

  html += "</div></body></html>";

  server.send(200, "text/html", html);
}

// Універсальна функція для перемикання пінів
void handleAction(int pin, bool state, bool inverted = false) {
  // Для вбудованого світлодіода логіка інвертована (LOW = світиться)
  if (inverted) digitalWrite(pin, state ? LOW : HIGH);
  else digitalWrite(pin, state ? HIGH : LOW);

  // Відповідь сервера "200 OK", щоб браузер знав, що команда виконана
  server.send(200, "text/plain", "OK");
}

void setup() {
  Serial.begin(115200); // Ініціалізація Serial-порту

  // Налаштування режимів роботи пінів
  pinMode(ledBuiltin, OUTPUT);
  pinMode(ledRed, OUTPUT);
  pinMode(ledYellow, OUTPUT);
  pinMode(ledGreen, OUTPUT);

  // Початковий стан - все вимкнено
  digitalWrite(ledBuiltin, HIGH);
  digitalWrite(ledRed, LOW);
  digitalWrite(ledYellow, LOW);
  digitalWrite(ledGreen, LOW);

  // Підключення до Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  // Вивід IP-адреси в консоль
```

```
  Serial.println("\nWiFi Connected");
  Serial.println(WiFi.localIP());

  // Прив'язка URL-адрес до функцій
  server.on("/", handleRoot);

  // Використання лямбда-виразів для короткого запису обробників подій
  server.on("/red/on", []() { handleAction(ledRed, true); });
  server.on("/red/off", []() { handleAction(ledRed, false); });

  server.on("/yellow/on", []() { handleAction(ledYellow, true); });
  server.on("/yellow/off", []() { handleAction(ledYellow, false); });

  server.on("/green/on", []() { handleAction(ledGreen, true); });
  server.on("/green/off", []() { handleAction(ledGreen, false); });

  // Вбудований LED має прапорець inverted=true
  server.on("/builtin/on", []() { handleAction(ledBuiltin, true, true); });
  server.on("/builtin/off", []() { handleAction(ledBuiltin, false, true); });

  server.begin(); // Запуск сервера
}

void loop() {
  server.handleClient(); // Очікування клієнтів
}
```

Програма складається з двох основних частин: серверної логіки (C++) та клієнтського інтерфейсу (HTML/CSS/JS).

У блоці setup(): Відбувається ініціалізація пінів у режим OUTPUT. Важливо, що вбудований світлодіод на платі NodeMCU підключений за схемою зі "стягуванням до живлення", тому він вмикається сигналом LOW (0), а вимикається сигналом HIGH (1). Зовнішні світлодіоди працюють за класичною логікою (HIGH = увімкнено). Далі контролер підключається до заданої Wi-Fi мережі. Після успішного з'єднання в Монітор порту виводиться IP-адреса, яку привласнив роутер (наприклад, 192.168.0.102). Саме цю адресу користувач вводить у браузер. Функція server.on() налаштовує маршрутизацію: вона вказує контролеру, яку дію виконувати при зверненні за певною адресою (наприклад, при запиті /red/on запускається код вмикання червоного діода).

Веб-інтерфейс (функція handleRoot): Сервер генерує HTML-сторінку, яка містить не тільки структуру (кнопки), а й стилі (CSS) для естетичного вигляду. Ключовою особливістю є використання JavaScript та технології Fetch API. У звичайних веб-сторінках натискання посилання призводить до перезавантаження всієї сторінки. У цьому проєкті використано функцію toggle(path). Коли користувач натискає кнопку, браузер відправляє "тихий" запит на сервер ESP8266. Сервер виконує дію (вмикає світлодіод) і повертає коротку відповідь "OK". Сторінка при цьому не моргає і не перезавантажується, що забезпечує користувацький досвід, схожий на роботу мобільного додатка.

У блоці loop(): Викликається лише одна функція server.handleClient(). Вона постійно "слухає" вхідні з'єднання. Як тільки браузер надсилає команду, ця функція передає управління відповідному обробнику, який змінює стан напруги на пінах мікроконтролера.

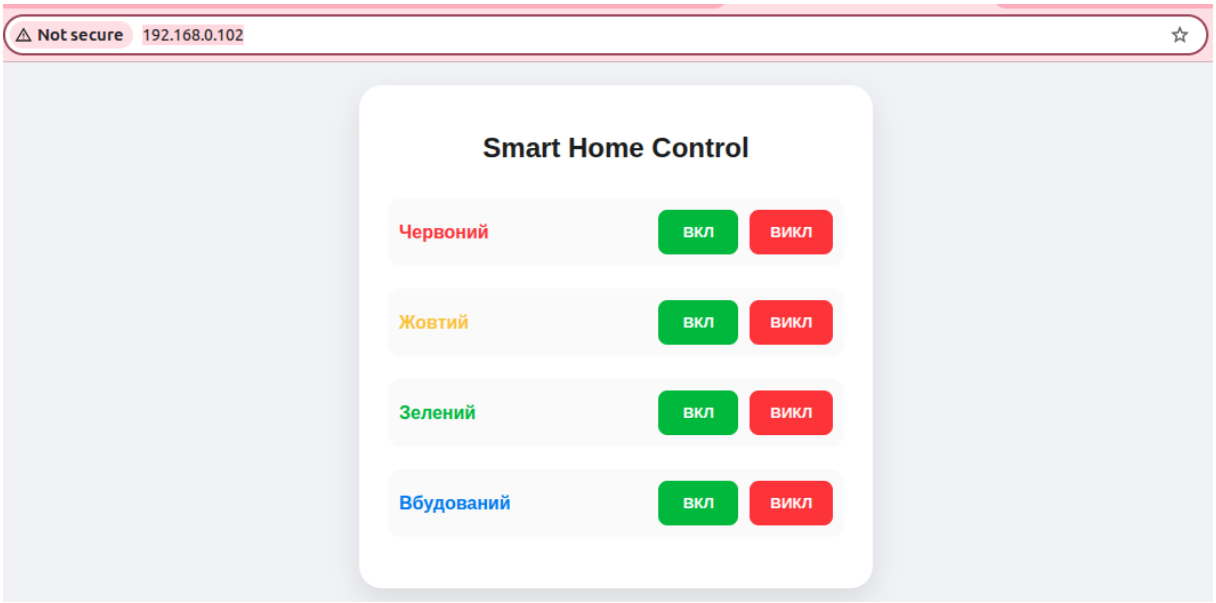
Після завантаження програми у мікроконтроллер у консолі Arduino будуть такі повідомлення, які свідчать про правильну загрузку і роботу WiFi.

```
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 307168 bytes to 222848...
Writing at 0x00000000... (7 %)
Writing at 0x00004000... (14 %)
Writing at 0x00008000... (21 %)
Writing at 0x0000c000... (28 %)
Writing at 0x00010000... (35 %)
Writing at 0x00014000... (42 %)
Writing at 0x00018000... (50 %)
Writing at 0x0001c000... (57 %)
Writing at 0x00020000... (64 %)
Writing at 0x00024000... (71 %)
Writing at 0x00028000... (78 %)
Writing at 0x0002c000... (85 %)
Writing at 0x00030000... (92 %)
Writing at 0x00034000... (100 %)
Wrote 307168 bytes (222848 compressed) at 0x00000000 in 19.8 seconds (effective 124.3 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Після успішного завантаження, ініціалізується сервер, на який можна зайти, за умови, що телефон чи комп'ютер підключені до того самого роутера.

Ввівши у браузері http://192.168.0.102/ (або іншу адресу, яку привласнив роутер), отримаємо веб-сторінку,



з якої можна керувати світлодіодами. Натискаючи потрібні кнопки, світлодіоди на платі вмикаються-вимикаються відповідно. Зазначимо, що програма буде працювати навіть без підключення USB кабелю.

Запропонований проєкт є повноцінним прототипом системи розумного будинку. Він демонструє, як можна поєднати фізичну електроніку з програмними веб-інтерфейсами. Проєкт навчає базовим принципам IoT: об'єкт (світлодіод) має IP-адресу і може керуватися з будь-якої точки локальної мережі. Отримана система є масштабованою: замість світлодіодів можна підключити реле для керування побутовими приладами (лампою, вентилятором) або додати датчики для виводу інформації на ту ж саму веб-сторінку.

Відео роботи програми подано за посиланням:
https://drive.google.com/file/d/1aleuJFncG8Cgn-69cWs1ju1RqgDA46E6/view

5. Проєкт "Вимірювач швидкості"

Мета:

Метою проєкту є створення автоматизованої системи вимірювання швидкості руху об'єктів (іграшкових автомобілів) за допомогою двох ультразвукових датчиків відстані. Система реєструє момент проходження об'єкта повз кожен датчик, обчислює швидкість руху та відображає результат на LCD-дисплеї. Проєкт демонструє принципи роботи з декількома сенсорами, обробку часових інтервалів, математичні обчислення та виведення інформації на дисплей, що робить його ефективним STEM-проєктом.

Обладнання:

1. Макетна плата (breadboard)
2. Платформа Arduino Uno / Nano
3. Два ультразвукові датчики відстані HC-SR04
4. LCD-дисплей 16×2 з I2C-інтерфейсом (адреса 0x27)
5. П'єзоелектричний випромінювач звуку (п'єзодинамік)
6. З'єднувальні проводи (male–male)
7. Джерело живлення (USB або 5V)

Реалізація проекту

1. Принцип роботи системи

Система працює за принципом вимірювання часу проходження об'єкта між двома контрольними точками на відомій відстані.

Ультразвукові датчики HC-SR04 розміщуються на фіксованій відстані один від одного (наприклад, S=20 см). Кожен датчик безперервно сканує простір перед собою. Коли іграшковий автомобіль проїжджає повз перший

датчик, система фіксує момент часу t_1. Коли автомобіль досягає другого датчика, фіксується момент t_2. Обчислення швидкості виконується за класичною фізичною формулою: $V = S/(t_2 - t_1)$.

LCD-дисплей 16×2 з I2C-інтерфейсом використовується для виведення результатів вимірювання. I2C-інтерфейс спрощує підключення, використовуючи лише два піни для передачі даних (SDA та SCL), що економить порти Arduino. Адреса дисплея зазвичай становить 0x27.

П'єзоелектричний випромінювач звуку (п'єзодинамік) подає звуковий сигнал при успішному вимірюванні швидкості, що робить систему більш інтерактивною.

2. Схема підключення

1. Перший датчик HC-SR04 (Датчик 1):
   - VCC → 5V
   - GND → GND
   - TRIG → цифровий пін D2
   - ECHO → цифровий пін D3
2. Другий датчик HC-SR04 (Датчик 2):
   - VCC → 5V
   - GND → GND
   - TRIG → цифровий пін D4
   - ECHO → цифровий пін D5
3. LCD-дисплей 16×2 (I2C):
   - VCC → 5V
   - GND → GND
   - SDA → A4 (на Arduino Uno)
   - SCL → A5 (на Arduino Uno)
4. П'єзодинамік:
   - Позитивний контакт → цифровий пін D8
   - Негативний контакт → GND

3. Алгоритм роботи програми

При запуску система підключає LCD-дисплей, налаштовує піни датчиків та п'єзодинаміка, виводить привітальне повідомлення. Далі система постійно «опитує» перший датчик, очікуючи появи об'єкта на відстані менше порогового значення (наприклад, 10 см). Коли об'єкт виявлено першим датчиком, фіксується час t_1 та встановлюється прапорець початку вимірювання. Система переходить до «опитування» другого датчика. Коли об'єкт досягає другого датчика, фіксується час t_2. Далі обчислюється швидкість за формулою, результат виводиться на LCD-дисплей, подається звуковий сигнал. Після цього система повертається до початкового стану очікування, очікування нового рухомого об'єкту.

Програмний код виглядає таким чином:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Піни для першого датчика
const byte trigPin1 = 2;
const byte echoPin1 = 3;

// Піни для другого датчика
const byte trigPin2 = 4;
const byte echoPin2 = 5;

// Пін для п'єзодинаміка
const byte buzzerPin = 8;

// Ініціалізація LCD (адреса 0x27, 16 символів, 2 рядки)
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Відстань між датчиками (у сантиметрах)
const float distanceBetweenSensors = 20.0;

// Порогова відстань для виявлення об'єкта (см)
const float detectionThreshold = 10.0;

// Змінні для вимірювання часу
unsigned long timeAtSensor1 = 0;
unsigned long timeAtSensor2 = 0;
bool sensor1Triggered = false;
bool measurementActive = false;

void setup() {
```

```
  // Налаштування пінів датчиків
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);

  // Налаштування п'єзодинаміка
  pinMode(buzzerPin, OUTPUT);

  // Ініціалізація Serial для налагодження
  Serial.begin(9600);

  // Ініціалізація LCD
  lcd.init();
  lcd.backlight();

  // Привітальне повідомлення
  lcd.setCursor(0, 0);
  lcd.print("Vymiruvach");
  lcd.setCursor(0, 1);
  lcd.print("shvydkosti");
  delay(2000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Ochikuvannya...");

  Serial.println("Sistema gotova do roboty");
}

void loop() {
  // Якщо вимірювання ще не почалося, опитуємо перший датчик
  if (!measurementActive) {
    float distance1 = measureDistance(trigPin1, echoPin1);

    if (distance1 > 0 && distance1 < detectionThreshold) {
      // Об'єкт виявлено першим датчиком
      timeAtSensor1 = millis();
      sensor1Triggered = true;
      measurementActive = true;

      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Ob'ekt vyjavleno!");
      lcd.setCursor(0, 1);
      lcd.print("Vymiruvanya...");

      Serial.println("Датчик 1 спрацював");
      Serial.print("Час t1: ");
      Serial.println(timeAtSensor1);
```

```
    // Короткий звуковий сигнал
    tone(buzzerPin, 1000, 100);

    delay(300); // Невелика затримка для стабільності
   }
  }

  // Якщо перший датчик спрацював, очікуємо на другий
  if (measurementActive && sensor1Triggered) {
   float distance2 = measureDistance(trigPin2, echoPin2);

   if (distance2 > 0 && distance2 < detectionThreshold) {
    // Об'єкт досяг другого датчика
    timeAtSensor2 = millis();

    Serial.println("Датчик 2 спрацював");
    Serial.print("Час t2: ");
    Serial.println(timeAtSensor2);

    // Обчислення швидкості
    calculateAndDisplaySpeed();

    // Скидання для нового вимірювання
    sensor1Triggered = false;
    measurementActive = false;

    delay(3000); // Показуємо результат 3 секунди

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Ochikuvannya...");
   }

   // Таймаут: якщо пройшло більше 5 секунд після першого датчика
   if (millis() - timeAtSensor1 > 5000) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Taymayt!");
    lcd.setCursor(0, 1);
    lcd.print("Sprobujte shche");

    Serial.println("Таймаут вимірювання");

    sensor1Triggered = false;
    measurementActive = false;

    delay(2000);

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Ochikuvannya...");
```

```
    }
  }

  delay(50); // Невелика затримка для стабільності
}

// Функція вимірювання відстані
float measureDistance(byte trigPin, byte echoPin) {
  long duration;

  // Генерація ультразвукового імпульсу
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Вимірювання тривалості відбитого сигналу
  duration = pulseIn(echoPin, HIGH, 30000); // Таймаут 30 мс

  // Обчислення відстані (см)
  if (duration == 0) {
    return -1; // Помилка вимірювання
  }

  return duration * 0.0343 / 2;
}

// Функція обчислення та відображення швидкості
void calculateAndDisplaySpeed() {
  // Обчислення різниці часу (мілісекунди)
  unsigned long timeDifference = timeAtSensor2 - timeAtSensor1;

  // Конвертація в секунди
  float timeInSeconds = timeDifference / 1000.0;

  Serial.print("Різниця часу: ");
  Serial.print(timeDifference);
  Serial.println(" мс");

  // Обчислення швидкості (см/с)
  float speedCmPerSec = distanceBetweenSensors / timeInSeconds;

  // Конвертація в м/с
  float speedMPerSec = speedCmPerSec / 100.0;

  // Конвертація в км/год (для наочності)
  float speedKmPerHour = speedMPerSec * 3.6;

  Serial.print("Швидкість: ");
  Serial.print(speedCmPerSec);
```

```
  Serial.print(" см/с = ");
  Serial.print(speedMPerSec);
  Serial.print(" м/с = ");
  Serial.print(speedKmPerHour);
  Serial.println(" км/год");

  // Відображення на LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("V: ");
  lcd.print(speedCmPerSec, 1);
  lcd.print(" sm/s");

  lcd.setCursor(0, 1);
  lcd.print("V: ");
  lcd.print(speedKmPerHour, 2);
  lcd.print(" km/h");

  // Звуковий сигнал про успішне вимірювання
  tone(buzzerPin, 2000, 200);
  delay(250);
  tone(buzzerPin, 2500, 200);
}
```

При запуску програми відбувається налаштування всіх компонентів. Піни ультразвукових датчиків конфігуруються як входи та виходи, п'єзодинамік налаштовується для виведення звуку. LCD-дисплей ініціалізується з адресою 0x27, вмикається підсвічування та виводиться привітальне повідомлення. Serial Monitor активується для можливості налагодження та спостереження за процесом вимірювання.

Програма працює у циклічному режимі з трьома основними станами. У стані очікування система постійно опитує перший датчик, шукаючи об'єкт на відстані менше 10 см. Коли об'єкт виявлено, фіксується час спрацювання та система переходить у стан активного вимірювання.

У стані активного вимірювання програма опитує другий датчик. При виявленні об'єкта другим датчиком фіксується другий часовий відлік. Система має захист від зависання через таймаут — якщо протягом 5 секунд об'єкт не досяг другого датчика, вимірювання скидається.

Функція measureDistance реалізує класичний алгоритм роботи з HC-SR04. Спочатку генерується короткий імпульс на піні TRIG, потім вимірюється тривалість високого рівня на піні ECHO. За часом проходження ультразвукової хвилі обчислюється відстань до об'єкта за формулою d = v·t/2, де швидкість звуку приймається 343 м/с або 0.0343 см/мкс. А функція calculateAndDisplaySpeed після отримання обох часових відміток функція обчислює різницю часу та швидкість руху об'єкта. Результат виводиться у трьох форматах: см/с (для точності), м/с (стандартна одиниця CI) та км/год (для наочності). На дисплей виводяться два значення: швидкість у см/с на першому рядку та у км/год на другому. П'єзодинамік подає подвійний звуковий сигнал для підтвердження успішного вимірювання.

Проєкт можна вдосконалити шляхом додавання запису результатів у пам'ять EEPROM для зберігання історії вимірювань, створення режиму калібрування для точного встановлення відстані між датчиками, додавання RGB-світлодіода для візуальної індикації різних станів системи, реалізації вимірювання напрямку руху (вперед чи назад) або інтеграції з модулем Bluetooth для передачі даних на смартфон.

Цей проєкт демонструє практичне застосування фізичних законів кінематики, роботу з датчиками реального часу та алгоритми обробки подій, що робить його цінним освітнім інструментом у галузі освіти.

ПІСЛЯМОВА

Шановні читачі!

Завершуючи роботу з цими методичними рекомендаціями, ви пройшли важливий шлях від теоретичних основ до практичної реалізації реальних робототехнічних систем. П'ять представлених проєктів — від пульсометра до вимірювача швидкості — демонструють лише невелику частину можливостей, які відкриває перед вами світ освітньої робототехніки та технологій Arduino.

Кожен із реалізованих проєктів є не просто технічним завданням, а комплексним освітнім інструментом, який об'єднує знання з фізики, математики, інформатики та інженерії. Пульсометр навчає основам біомедичної електроніки та обробки сигналів. Радар демонструє принципи ехолокації та візуалізації даних. Розумний смітник показує застосування автоматизації у повсякденному житті. IoT-контролер відкриває двері до концепції розумного дому та веб-технологій. Вимірювач швидкості інтегрує фізичні закони кінематики з практичною електронікою. Усі ці проєкти формують фундамент для розуміння сучасних технологічних рішень та розвивають інженерне мислення.

Особливо важливо, що ви опанували не лише технічні навички, але й педагогічні компетенції. Як майбутні педагоги, ви навчилися адаптувати складний матеріал для учнів різного віку, організовувати проєктну діяльність, пояснювати принципи роботи електронних компонентів доступною мовою. Ці вміння стануть основою для проведення гурткових занять, STEM-уроків та позакласної роботи, гуртків з робототехніки, де ви зможете надихати школярів на власні технічні відкриття.

Важливо розуміти, що представлені проєкти є лише відправною точкою. Кожен з них можна розширити, модифікувати та адаптувати під

конкретні освітні завдання чи реальні потреби. Пульсометр може стати основою для системи моніторингу здоров'я спортсменів. Радар можна використати в автономних роботах. IoT-контролер легко масштабується до повноцінної системи розумного класу. Вимірювач швидкості може еволюціонувати у складну систему відеофіксації для спортивних змагань. Ваша креативність та інженерна уява не мають меж.

Технології постійно розвиваються, з'являються нові мікроконтролери, датчики, способи комунікації між пристроями. Те, що сьогодні здається інноваційним, завтра стане стандартом. Тому найважливішою навичкою, яку ви розвинули, працюючи з цими матеріалами, є вміння навчатися самостійно, аналізувати документацію, експериментувати та не боятися помилок. Кожна несправна схема, кожна помилка в коді, кожен невдалий експеримент — це цінний досвід, який наближає вас до успіху.

Ми переконані, що отримані знання та навички стануть міцним фундаментом вашої професійної діяльності. Ви зможете не лише впроваджувати робототехніку в навчальний процес, але й надихати учнів на технічну творчість, формувати в них критичне мислення та інженерну культуру. Саме такі педагоги потрібні сучасній школі — ті, хто не боїться технологій, а вміло інтегрує їх у освітній процес, роблячи навчання захопливим та практично орієнтованим.

Завершуючи, хочемо побажати вам натхнення для нових проєктів, цікавих технічних ідей та задоволення від того, що ваша робота допомагає формувати майбутніх інженерів, програмістів та винахідників. Нехай кожен реалізований проєкт приносить радість відкриття, а кожен учень, якого ви надихнете на технічну творчість, стане вашою гордістю. Світ технологій величезний та захопливий — продовжуйте досліджувати його та ділитися своїми відкриттями з іншими!

Успіхів вам у педагогічній діяльності та реалізації нових робототехнічних проєктів!

Список використаних джерел

1. Arduino.ua [Електронний ресурс]. Режим доступу: https://arduino.ua/ .
2. Лешко Р. Інформаційно-керуючі системи та STEM-технології : методичні рекомендації до виконання лабораторних робіт. Дрогобич : ДДПУ ім. І. Франка, 2023. 40 с.
3. Роман Лешко, Ольга Лешко. Реалізація STEM-проектів на базі Arduino [Методичні рекомендації до виконання індивідуальних проектних завдань]. Дрогобич : Редакційно-видавничий відділ ДДПУ ім. І. Франка, 2024. 38с.
4. Arduino blog [Електронний ресурс]. Режим доступу: https://blog.arduino.cc/.
5. Arduino-DIY [Електронний ресурс]. Режим доступу: http://arduinodiy.com/.
6. Margolis M. Arduino Cookbook. O'Reilly Media, 2011. 662 p.
7. Perea F. Arduino Essentials, 2015. 206 p.
8. Tinkercad [Електронний ресурс]. Режим доступу: https://www.tinkercad.com/.
9. Microsoft C++. [Електронний ресурс]. Режим доступу: https://learn.microsoft.com/uk-ua/cpp/?view=msvc-170.
10. Грабко В. В., Розводюк М. П., Грабко Вал. В. Мікропроцесорні системи керування електроприводами. Вінниця: ВНТУ, 2012. 97 с.
11. Користувацькі посібники по роботі із платформою Arduino. [Електронний ресурс]. Режим доступу: https://www.arduino.cc/en/Tutorial/HomePage.
12. Програма технічного конструювання. Програми з позашкільної освіти науково-технічний напрям (інформаційно-технічний профіль). Київ, 2014. С. 15-32.

13. Vedat Ozan. Developing IoT Projects with ESP32: Automate Your Home Or Business with Inexpensive Wi-Fi Devices. Packt Publishing, 2021, 470 pages.
14. Roman Leshko, Olha Leshko. STEM-technologies based on the Arduino board [Guidelines for Performing Laboratory Works]. Drohobych : Drohobych Ivan Franko State Pedagogical University, 2024. 58 p.