ДОДАТОК А

Текст програми

```
#define BLYNK_TEMPLATE_ID "TMPL43JnlmywK"
#define BLYNK_TEMPLATE_NAME "Home Automation"
#define BLYNK_AUTH_TOKEN "dOsi_Q6rjo4v3SPXXxY4eCSto-l-gese"

#include <AccelStepper.h>
#include <Wire.h>
#include <WiFi.h>
#include <Arduino.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include "DHTesp.h"

#define BLYNK_PRINT Serial

#define TEMPERATURE_VPIN V0
#define HUMIDITY_VPIN V1
#define perform_manual_action_switch V2
#define CDIOX_VPIN V3
#define MENU_VPIN V4
#define WSPEED_VPIN V5
#define WDIRECTION_VPIN V6
#define Energy_consumption V13

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Wokwi-GUEST";
char pass[] = ""
```

```
BlynkTimer timer;

DHTesp       dhtSensor;
const int DHT_PIN = 14;

const int WD_PIN = 32;
const int WS_PIN = 34;

const int MQ2_ANA = 17;
const int MQ2_DIG = 15;

const int CO2_PIN = 35;

const int LUX_PIN = 23;
const int GREEN = 13;
const int BURNED = 12;
const int BLUE = 27;
const int WHITE = 26;

const int DIR_PIN = 4;
const int STEP_PIN = 25;
AccelStepper stepper(AccelStepper::FULL4WIRE, STEP_PIN, DIR_PIN);

bool manualMode = false;
bool manualControlMessageSent = false;

bool greenState = false;
bool blueState = false;
bool burnedState = false;
```

```
bool whiteState = false;

float   m      =      1.2041      *      750;
float To = 17;
float R =  0.12;
float C = 1.005;

int selectedMode = 0;

#define LED_COUNT 4

const int LED_PINS[LED_COUNT] = {GREEN, BURNED, BLUE, WHITE};
const char LED_VPINS[LED_COUNT] = {V11, V8, V9, V10};

enum FanSpeed
{
  TEMPUnknown,
  TEMPNeutral,
  TEMPReversse,
  TEMPDrive,
  TEMPSport
};
const int TEMPNeutralSpeed = 0;
const int TEMPReversseSpeed = 500;
const int TEMPDriveSpeed = -500;
const int TEMPSportSpeed = -10000;

class SensorData
{
```

```
public:
        struct Info
  {
    float temperature;
    float humidity;
    int wspeed;
    int wdirection;
    int tgl;
    int cdiox;
    int lux;
    int fire;
    float Ti;
  };

  Info getSensorData()
  {
    Info data;

    data.temperature = dhtSensor.getTemperature();
    data.humidity = dhtSensor.getHumidity();

    int16_t wspeedValue = analogRead(WS_PIN);
    int16_t wdirectionValue = analogRead(WD_PIN);
    int16_t tglValue = digitalRead(MQ2_DIG);
    int16_t cdioxValue = analogRead(CO2_PIN);
    int16_t luxValue = !digitalRead(LUX_PIN);
    data.wspeed = wspeedValue;
    data.wdirection = wdirectionValue;
    data.tgl = tglValue;
```

```
        data.cdiox    =       cdioxValue  *       2;
    data.lux = luxValue;
    data.fire = data.lux == HIGH && data.tgl == HIGH;
    data.Ti = data.temperature;

    return data;
  }
};

SensorData sensorData;
SensorData::Info data;

unsigned long previousSensorMillis = 0;
unsigned long previousConsoleMillis = 0;
unsigned long previousLedMillis = 0;
const long sensorInterval = 0.1;
const long consoleInterval = 3500;
const long LedInterval = 3500;

void sendSensorData();
void controlLEDs();

bool isInRange(int value, int lower, int upper)
{
  return (value >= lower && value <= upper);
}
BLYNK_WRITE(perform_manual_action_switch)
{
 manualMode        =       param.asInt();
```

```
  if (manualMode)
  {
    stepper.setSpeed(TEMPNeutralSpeed);
    stepper.runSpeed();
  }

}
BLYNK_WRITE_DEFAULT() {
  int pin = request.pin;
  if (pin == MENU_VPIN) {
    int selectedMenuItem = param.asInt();
    switch (selectedMenuItem) {
      case 1:
        selectedMode = TEMPNeutral;
        break;
      case 2:
        selectedMode = TEMPReversse;
        break;
      case 3:
        selectedMode = TEMPDrive;
        break;
      case 4:
        selectedMode = TEMPSport;
        break;
      default:
        selectedMode = TEMPNeutral;
        break;
  }
    if (!manualMode) {
```

```
      stepper.setSpeed(getSpeedForMode(selectedMode));
      stepper.runSpeed();
    }
  }
}
int getSpeedForMode(int mode)
{
  switch (mode)
  {
  case TEMPNeutral:
    return TEMPNeutralSpeed;
  case TEMPReversse:
    return TEMPReversseSpeed;
  case TEMPDrive:
    return TEMPDriveSpeed;
  case TEMPSport:
    return TEMPSportSpeed;
  default:
    return 0;
  }
}

FanSpeed determineFanSpeed()
{
  if (manualMode)
  {
 return          TEMPNeutral;
  }
  if (data.fire == HIGH || data.wspeed >= 8 && isInRange(data.wdirection, 70, 110))
```

```
  {
    return TEMPNeutral;
  }

  if (isInRange(data.temperature, 10, 17) && isInRange(data.humidity, 15, 30) &&
isInRange(data.cdiox, 600, 800))
  {
    return TEMPNeutral;
  }

  if (data.tgl == HIGH || data.cdiox >= 1400)
  {
    return TEMPReversse;
  }

  if (isInRange(data.temperature, 17, 30) && isInRange(data.humidity, 30, 50) &&
isInRange(data.cdiox, 800, 1000))
  {
    return TEMPDrive;
  }

  if (isInRange(data.temperature, 30, INT_MAX) && isInRange(data.humidity, 50,
INT_MAX) && isInRange(data.cdiox, 1000, 1400))
  {
    return TEMPSport;
  }

  return TEMPUnknown;
}
```

```
void adjustFanSpeed() {
  if (manualMode) {
    stepper.setSpeed(getSpeedForMode(selectedMode));
    stepper.runSpeed();
  } else {
    int fanSpeed = determineFanSpeed();
    switch (fanSpeed) {
      case TEMPUnknown:
        stepper.setSpeed(0);
        break;
      case TEMPNeutral:
        stepper.setSpeed(TEMPNeutralSpeed);
        break;
      case TEMPReversse:
        stepper.setSpeed(TEMPReversseSpeed);
        break;
      case TEMPDrive:
        stepper.setSpeed(TEMPDriveSpeed);
        break;
      case TEMPSport:
        stepper.setSpeed(TEMPSportSpeed);
        break;
    }
    stepper.runSpeed();
  }
}
void controlLEDs() {
  for (int i = 0; i < LED_COUNT; i++) {
    bool ledState = determineLEDState(i);
```

```
    digitalWrite(LED_PINS[i], ledState);
    Blynk.virtualWrite(LED_VPINS[i], ledState ? 255 : 0);
  }
}

bool determineLEDState(int index) {
  switch (index) {
    case 0:  // GREEN
      return isInRange(data.temperature, 10, 17) && isInRange(data.humidity, 15, 30)
&& isInRange(data.cdiox, 600, 800);
    case 1:  // BURNED
      return data.fire == HIGH;
    case 2:  // BLUE
      return data.wspeed >= 8 && isInRange(data.wdirection, 70, 110);
    case 3:  // WHITE
      return determineFanSpeed() == TEMPUnknown;
    default:
      return false;
  }
}

void setup()
{
  Serial.begin(115200);
 Blynk.begin(auth, ssid,   pass);

  sensorData = SensorData();

  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
```

```
  pinMode(DIR_PIN, OUTPUT);
  pinMode(STEP_PIN, OUTPUT);
  digitalWrite(DIR_PIN, HIGH);

  pinMode(MQ2_ANA, INPUT);
  pinMode(MQ2_DIG, INPUT);

  pinMode(GREEN, OUTPUT);
  pinMode(LUX_PIN, INPUT);
  pinMode(BURNED, OUTPUT);
  pinMode(BLUE, OUTPUT);
  pinMode(WHITE, OUTPUT);

  stepper.setMaxSpeed(15000);
  stepper.setAcceleration(500);
}

void loop()
{
  Blynk.run();
  unsigned long currentMillis = millis();

 if (currentMillis - previousSensorMillis >= sensorInterval)

  {
    previousSensorMillis = currentMillis;
    data = sensorData.getSensorData();

    adjustFanSpeed()
```

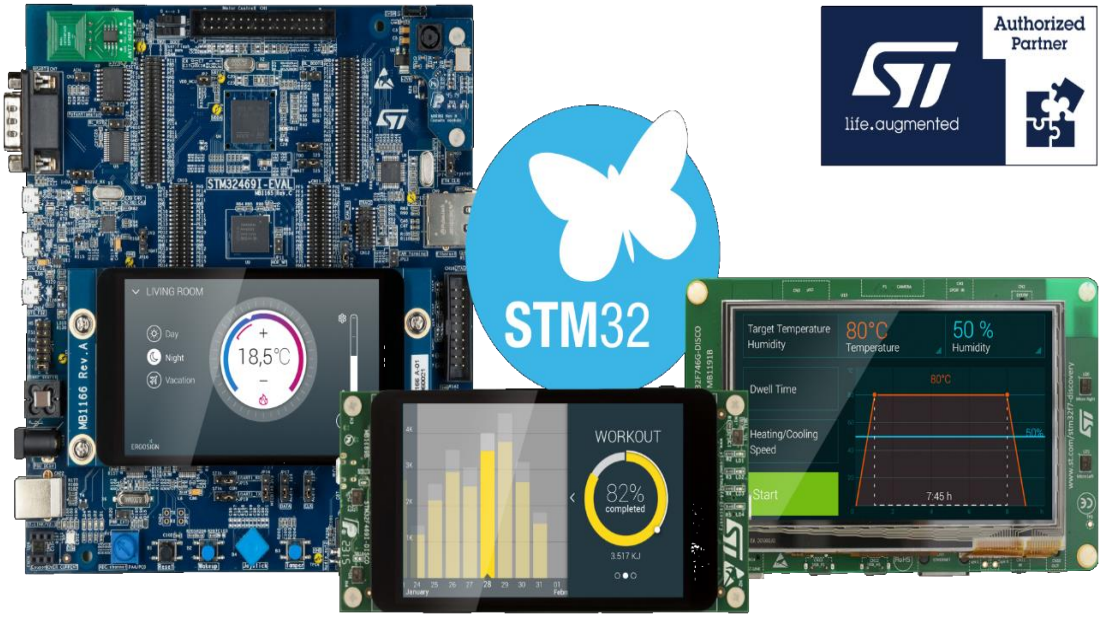
```
    if (currentMillis - previousConsoleMillis >= consoleInterval)
    {
      previousConsoleMillis = currentMillis;
      sendSensorData();
    }
      if (currentMillis - previousLedMillis >= LedInterval)
      {
        previousLedMillis = currentMillis;
        controlLEDs();
      }
  }
}

String fanSpeedToString(int fanSpeed)
{
  switch (fanSpeed)
  {
  case TEMPNeutral:
    return "TempNeutral";
  case TEMPReversse:
    return "TempReversse";
  case TEMPDrive:
    return "TempDrive";

  case TEMPSport:
    return "TempSport";
  case TEMPUnknown:
    return "TempUnknown";
  defaul
```

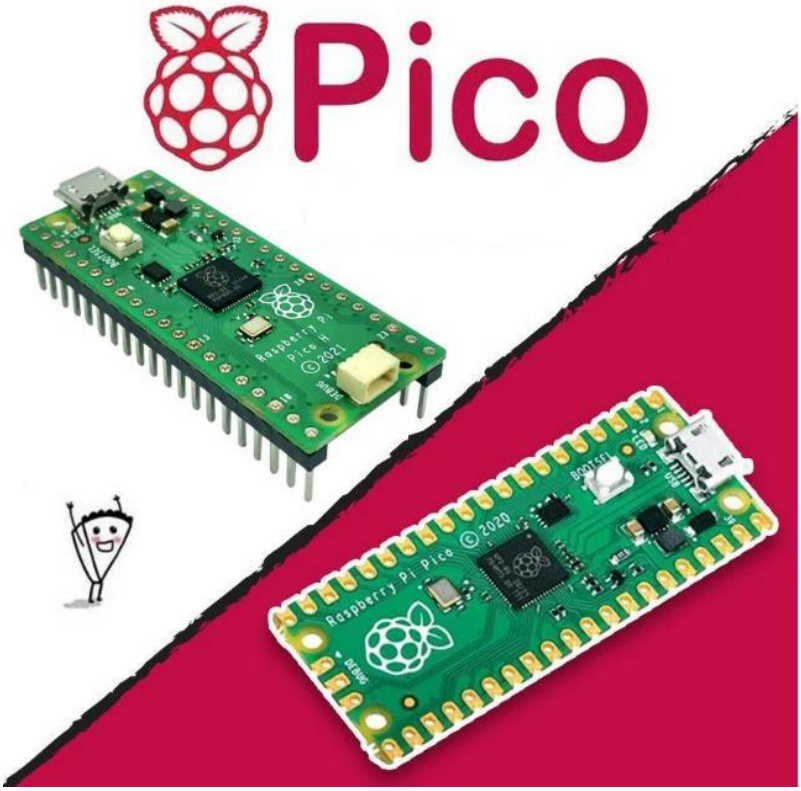
```
    return "Unknown";
  }
}

void sendSensorData()
{
  SensorData::Info data = sensorData.getSensorData();

  float E = m * (data.Ti - To) / (R * C);
  Serial.println("Energy consumption for air conditioning: " + String(E / 100000) + " kW·h");
   Blynk.virtualWrite(7, "Energy consumption for air conditioning: " + String(E / 100000) + " kW·h");

  if (manualMode) {
    Serial.println("DEVICE IN MANUAL CONTROL MODE");
    Serial.print("Fan State (Manual Control): ");
    Serial.println(fanSpeedToString(selectedMode));
    Blynk.virtualWrite(7, "DEVICE IN MANUAL CONTROL MODE");
          Blynk.virtualWrite(7, "Fan State (Manual Control): " + fanSpeedToString(selectedMode));
  } else

  {

    Serial.println("DEVICE IN AUTOMATIC CONTROL MODE");
    FanSpeed fanSpeed = determineFanSpeed();
    Serial.print("Fan State: ");
    Serial.println(fanSpeedToString(fanSpeed))
```

```
  Blynk.virtualWrite(7, "DEVICE IN AUTOMATIC CONTROL MODE");
  Blynk.virtualWrite(7, "Fan State: " + fanSpeedToString(fanSpeed));
}

Blynk.virtualWrite(TEMPERATURE_VPIN, data.temperature);
Blynk.virtualWrite(HUMIDITY_VPIN, data.humidity);
Blynk.virtualWrite(CDIOX_VPIN, data.cdiox);
Blynk.virtualWrite(WSPEED_VPIN, data.wspeed);
Blynk.virtualWrite(WDIRECTION_VPIN, data.wdirection);
Blynk.virtualWrite(Energy_consumption, E / 100000);
```

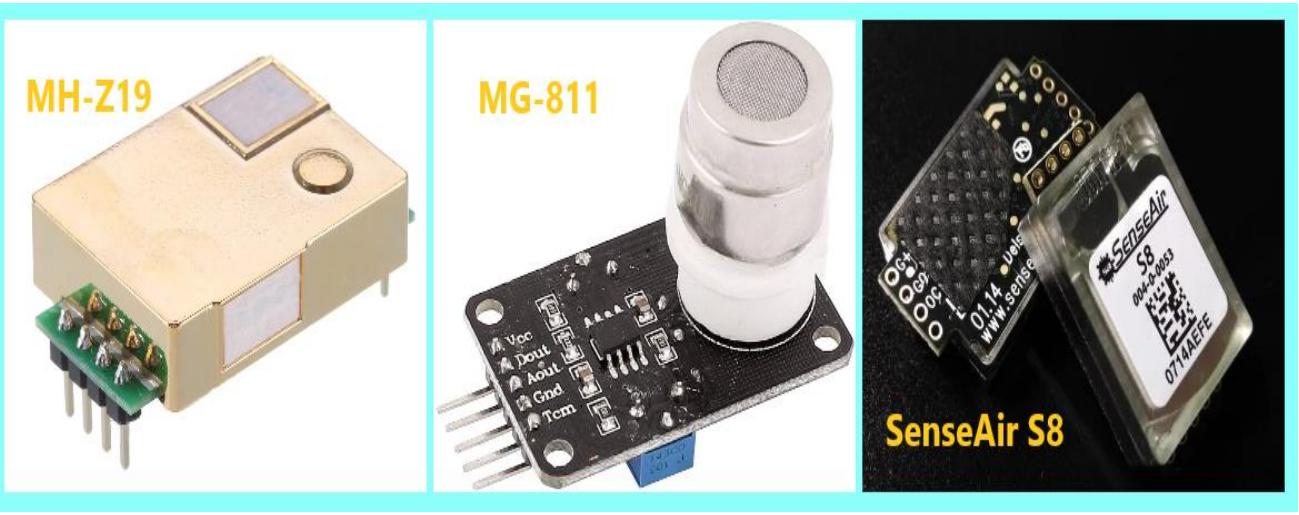
ДОДАТОК Б
ДЕМОНСТРАЦІЙНИЙ МАТЕРІАЛ

МІКРОКОНТРОЛЕРИ СЕРІЇ ARDUINO, ESP32, STM32, RASPBERRY PI

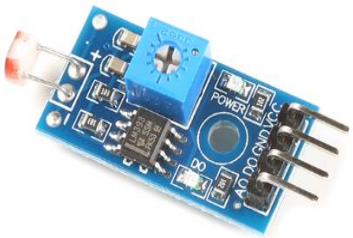

STM32469I-EVAL
LIVING ROOM
Day
Night
Vacation
18,5°C
STM32
WORKOUT
82%
completed
Target Temperature
Humidity
80°C
Temperature
50 %
Humidity
Dwell Time
Heating/Cooling
Speed
Start
7:45 h


Pico
Raspberry Pi Pico
© 2020
BOOTSEL
DEBUG

ДАВАЧІ ВОЛОГОСТІ ТА ТЕМПЕРАТУРИ, CO2, ДИМУ І ГАЗУ, ШВИДКОСТІ ВІТРУ, ІНТЕНСИВНОСТІ СВІТЛА ТА КРОКОВІ ДВИГУНИ

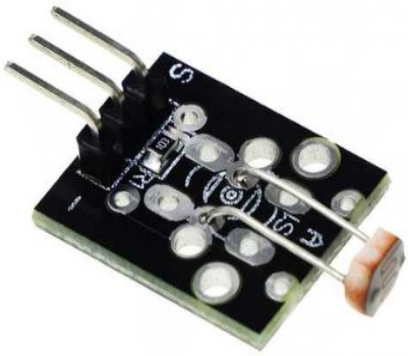


SparkFun Photocell Sensor



DFRobot SEN0014

KY-018

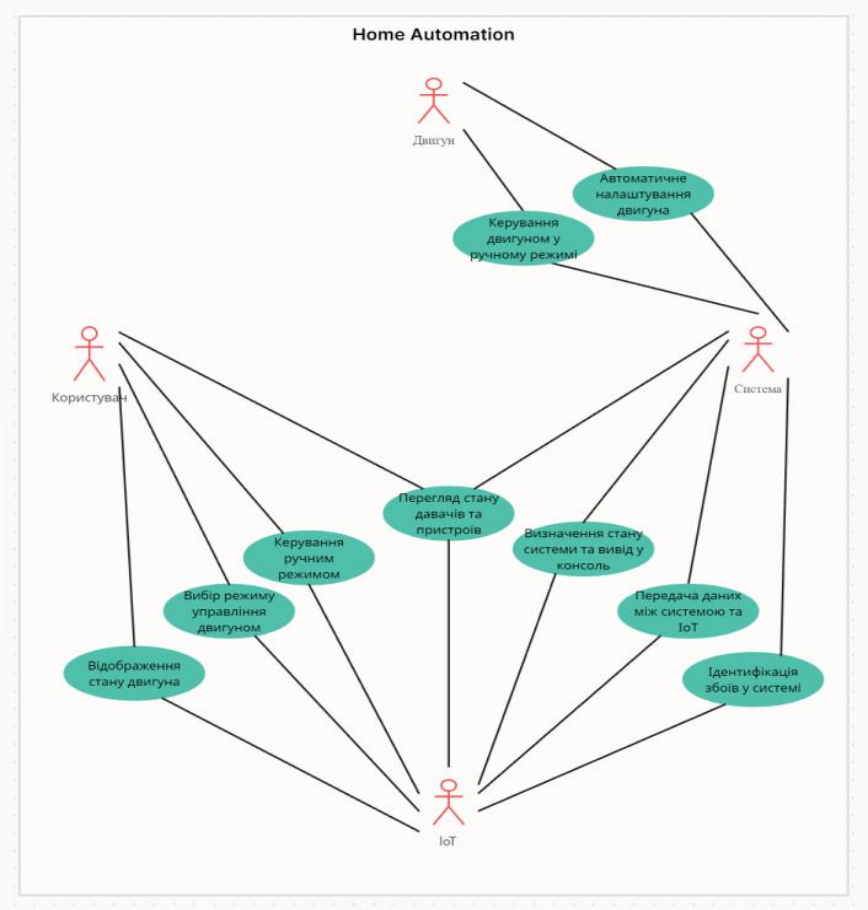


NEMA 17

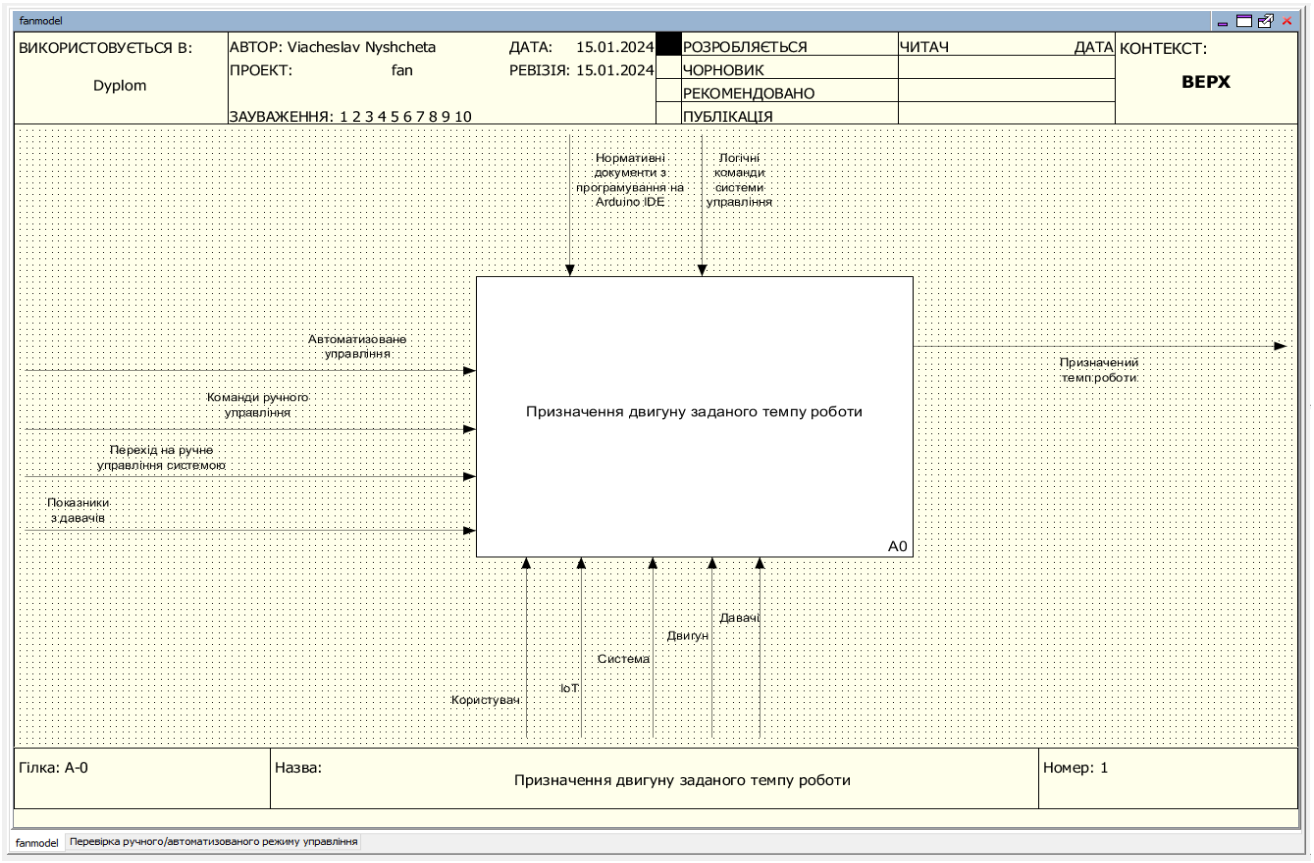
NEMA 23

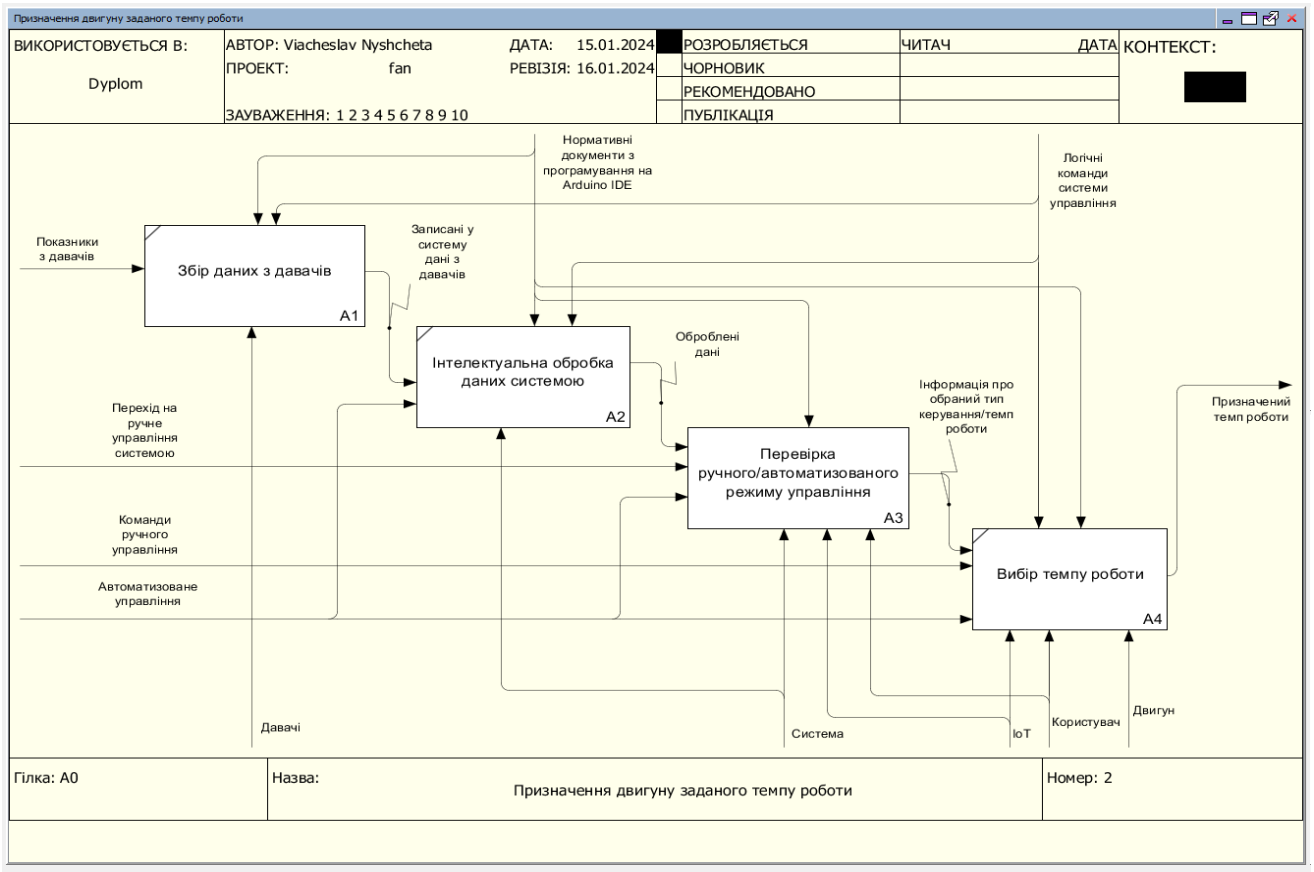
NEMA 14

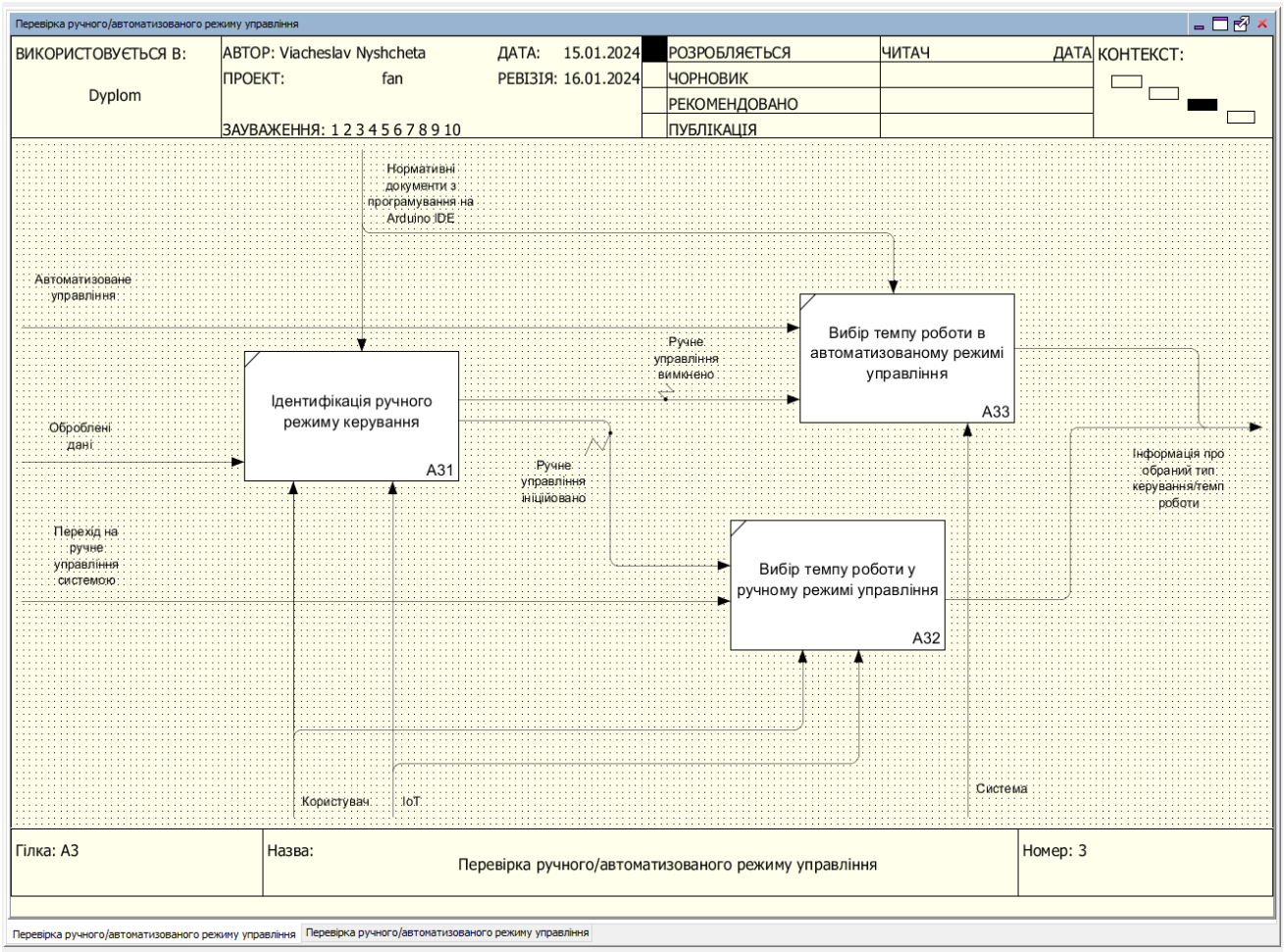
ДІАГРАМА ПРЕЦЕДЕНТІВ



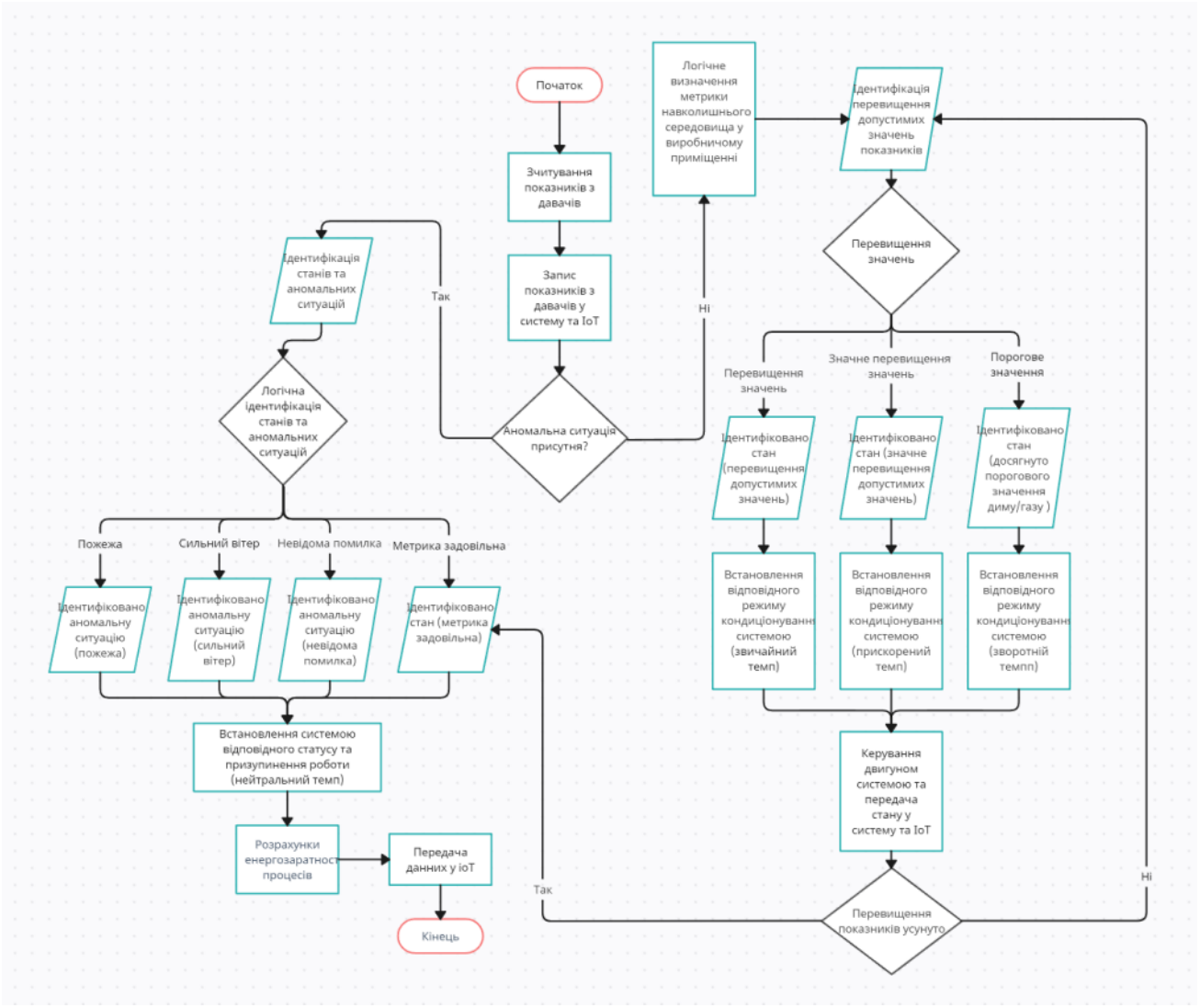
КОНТЕКСТНА ДІАГРАМА ПРОЦЕСУ ПРИЗНАЧЕННЯ ДВИГУНУ ЗАДАНОГО ТЕМПУ РОБОТИ ТА РІВНІ ДЕКОМПОЗИЦІЇ

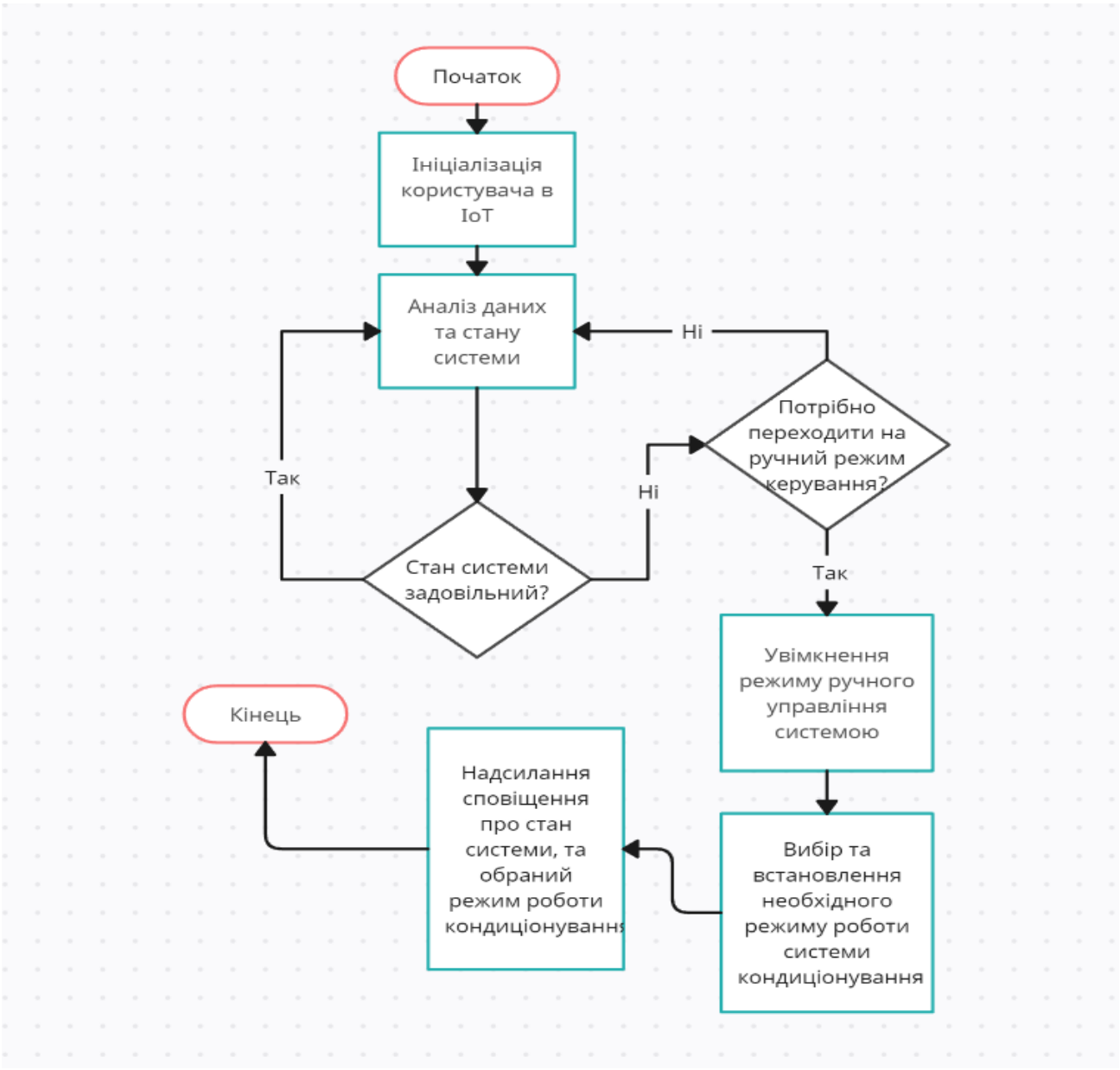




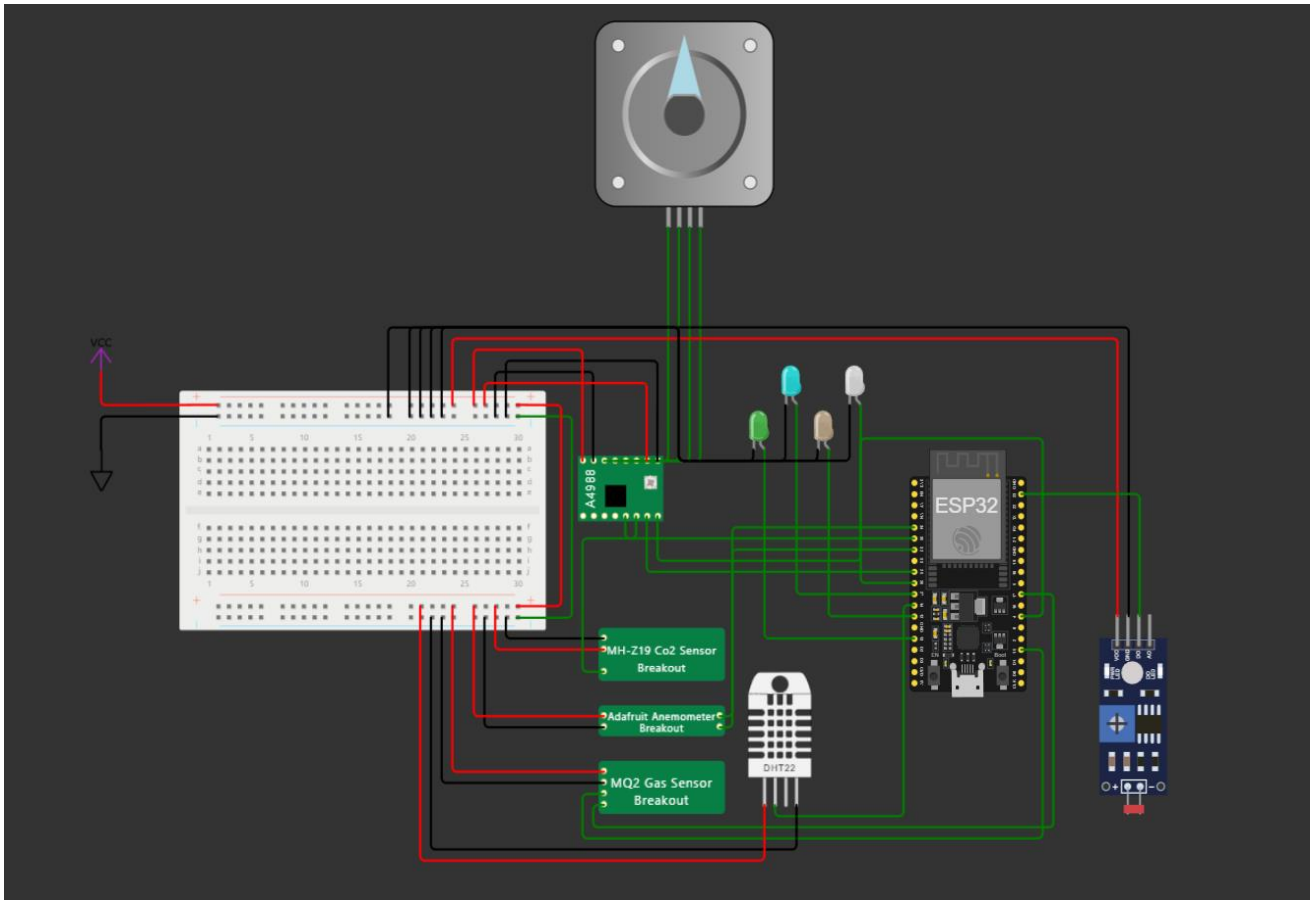

Перевірка ручного/автоматизованого режиму управління
ВИКОРИСТОВУЄТЬСЯ В:
Dyplom
АВТОР: Viacheslav Nyshcheta
ДАТА: 15.01.2024
ПРОЕКТ: fan
РЕВІЗІЯ: 16.01.2024
ЗАУВАЖЕННЯ: 1 2 3 4 5 6 7 8 9 10
РОЗРОБЛЯЄТЬСЯ
ЧОРНОВИК
РЕКОМЕНДОВАНО
ПУБЛІКАЦІЯ
ЧИТАЧ
ДАТА
КОНТЕКСТ:
Нормативні документи з програмування на Arduino IDE
Автоматизоване управління
Ручне управління вимкнено
Вибір темпу роботи в автоматизованому режимі управління
A33
Ідентифікація ручного режиму керування
A31
Оброблені дані
Ручне управління ініційовано
Інформація про обраний тип керування/темп роботи
Перехід на ручне управління системою
Вибір темпу роботи у ручному режимі управління
A32
Користувач
IoT
Система
Гілка: A3
Назва:
Перевірка ручного/автоматизованого режиму управління
Номер: 3
Перевірка ручного/автоматизованого режиму управління
Перевірка ручного/автоматизованого режиму управління

АЛГОРИТМИ АВТОМАТИЗОВАНОЇ РОБОТИ ТА РУЧНОГО УПРАВЛІННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ КЕРУВАННЯ КОНДИЦІОНУВАННЯМ

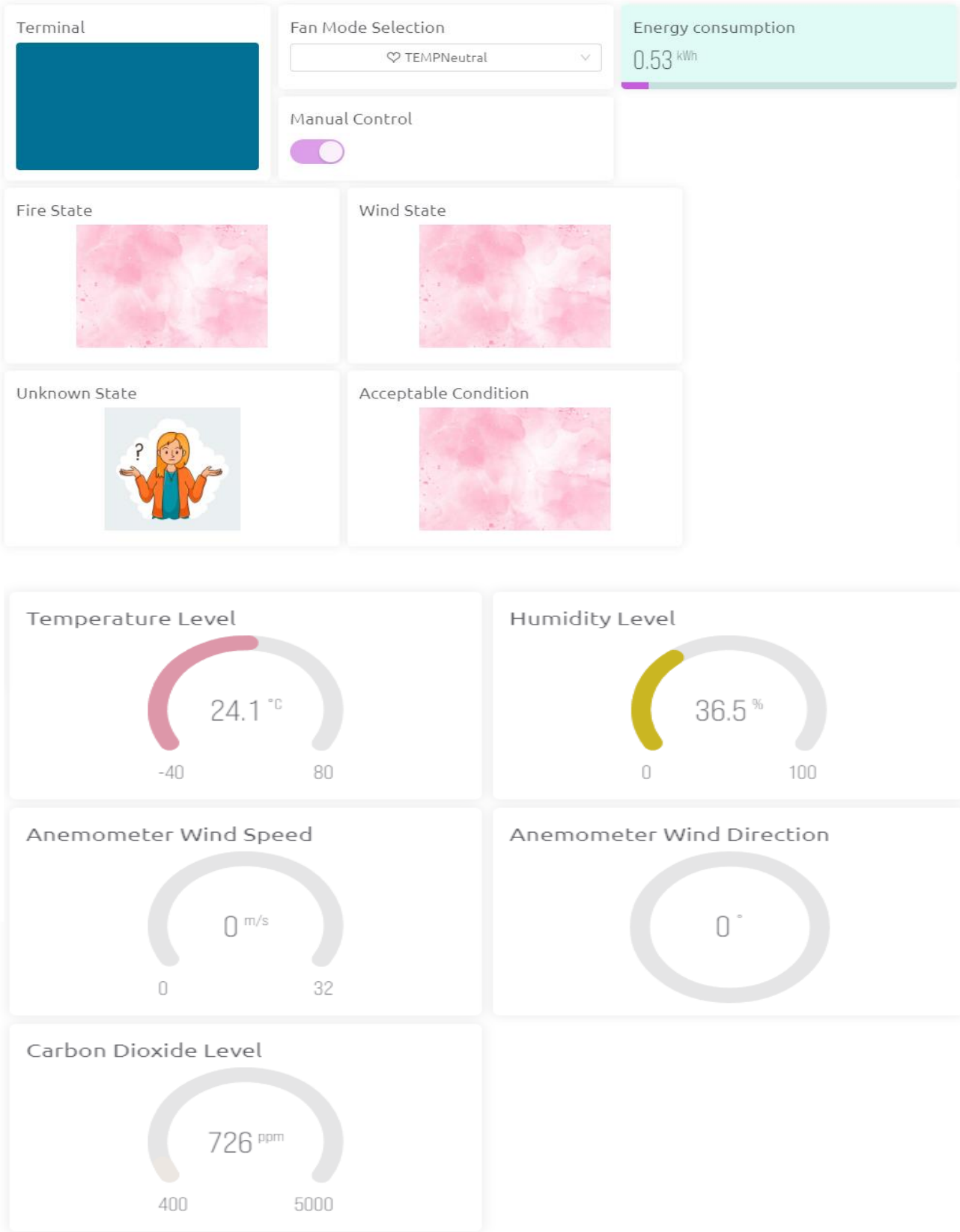



Початок
Ініціалізація користувача в IoT
Аналіз даних та стану системи
Ні
Потрібно переходити на ручний режим керування?
Так
Ні
Так
Стан системи задовільний?
Увімкнення режиму ручного управління системою
Кінець
Надсилання сповіщення про стан системи, та обраний режим роботи кондиціонування
Вибір та встановлення необхідного режиму роботи системи кондиціонування

МОДЕЛЬ СИСТЕМИ У СЕРЕДОВИЩІ МОДЕЛЮВАННЯ WOKWI



ВІДОБРАЖЕННЯ ІНТЕРФЕЙСУ СИСТЕМИ У СЕРЕДОВИЩІ BLYNK НА ПК ТА У ДОДАТКУ ДЛЯ СМАРТФОНІВ



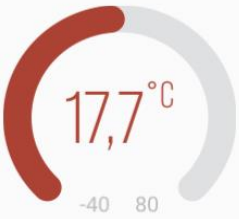
00:39

Home Automation

Terminal

```
DEVICE IN MANUAL CONTROL MODE
Fan State (Manual Control):
TempNeutral
Energy consumption for air
conditioning: 0.05 kW·h
DEVICE IN AUTOMATIC CONTROL MODE
Fan State: TempReversse
Energy consumption for air
conditioning: 0.05 kW·h
DEVICE IN MANUAL CONTROL MODE
Fan State (Manual Control):
TempNeutral
```

Temperature Level



Humidity Level

Manual Control

Fan Mode Selection

TEMPNeutral

Anemometer Wind Spe

Anemometer Wind Dire

Carbon Dioxide Level

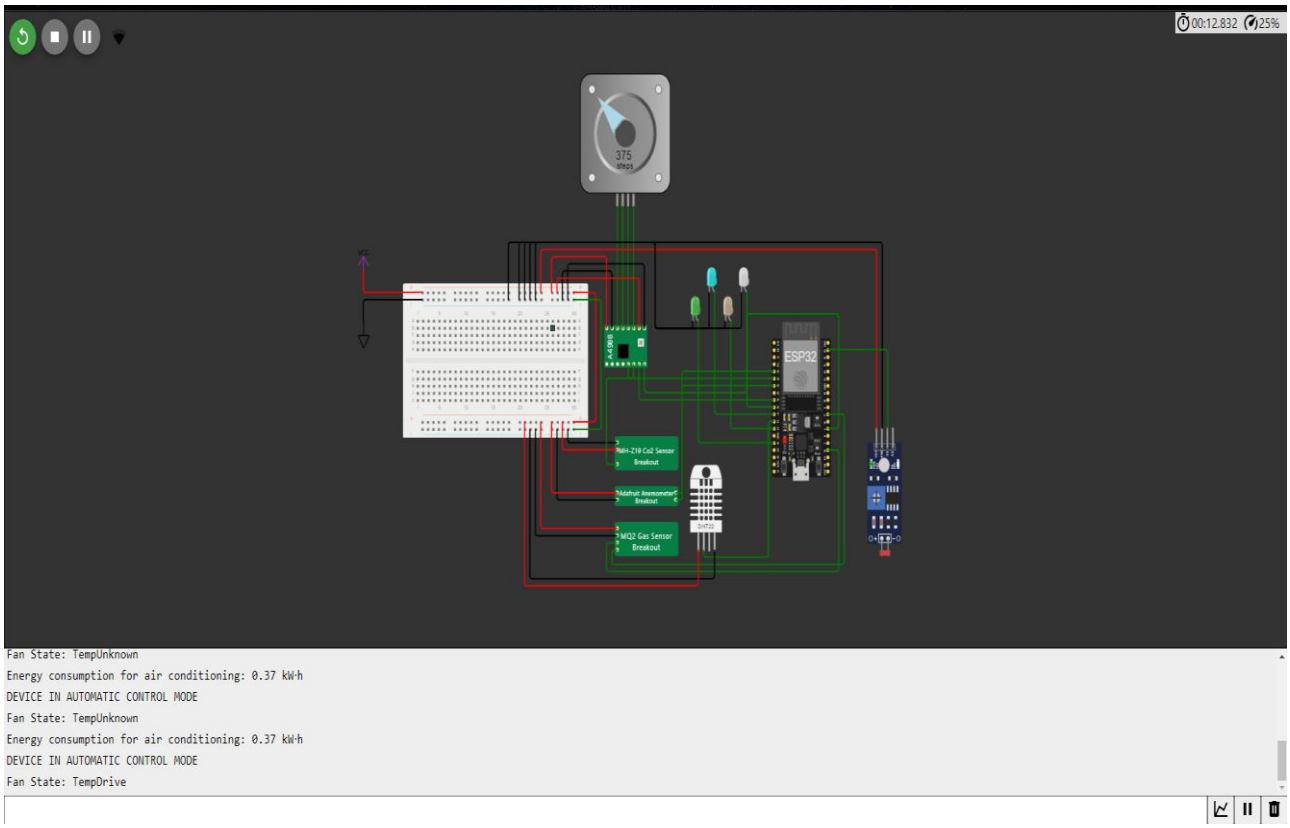
Fire State

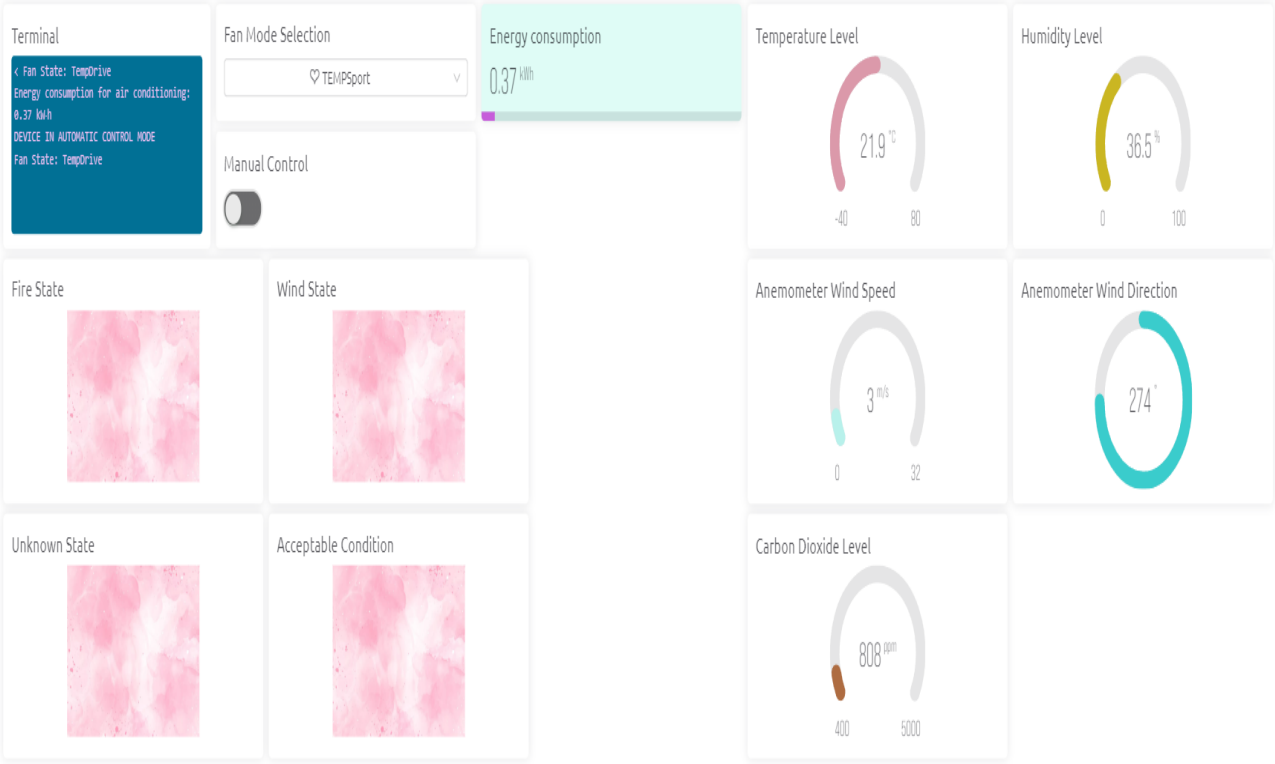
Wind State

Unknown State

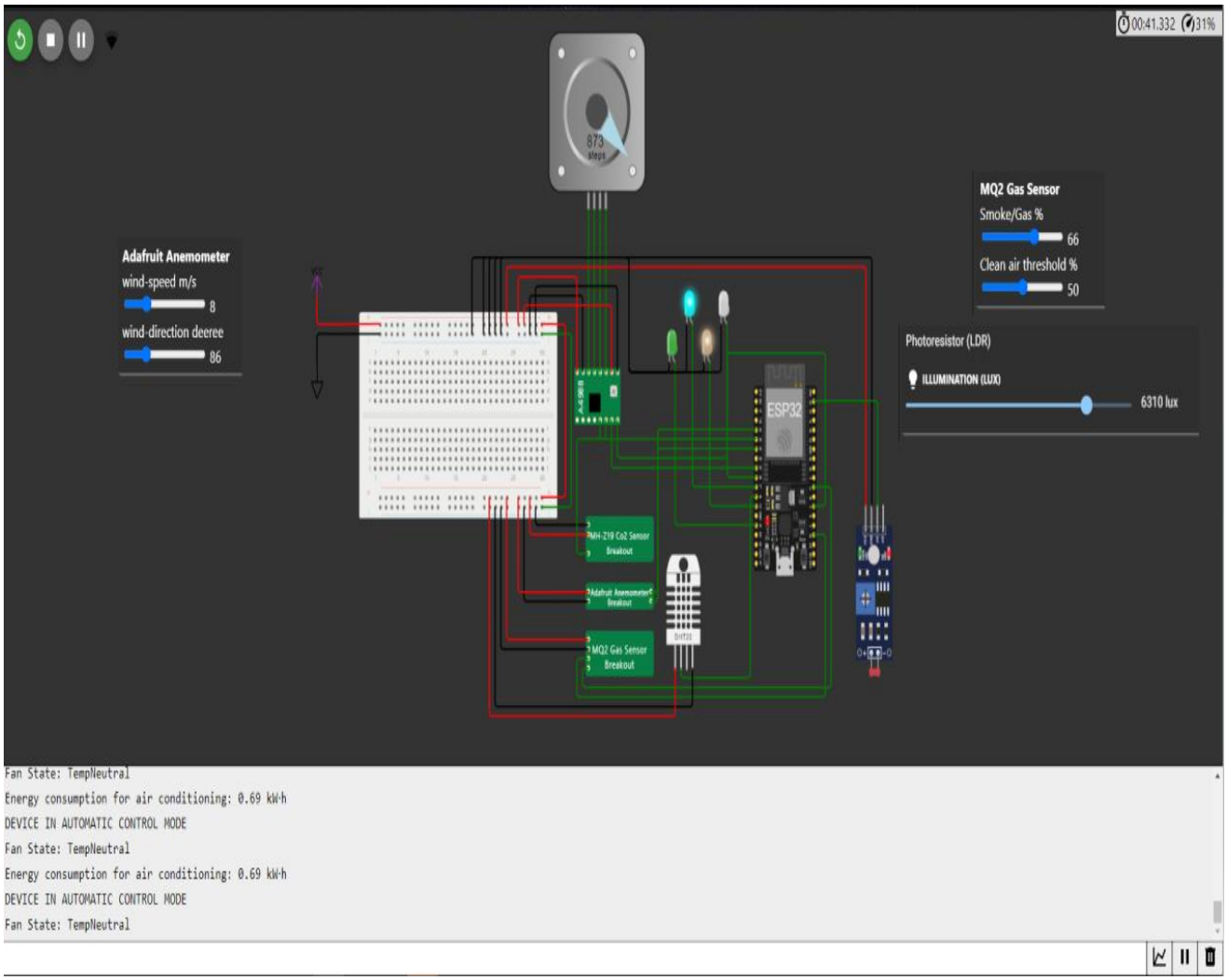
Acceptable Condition

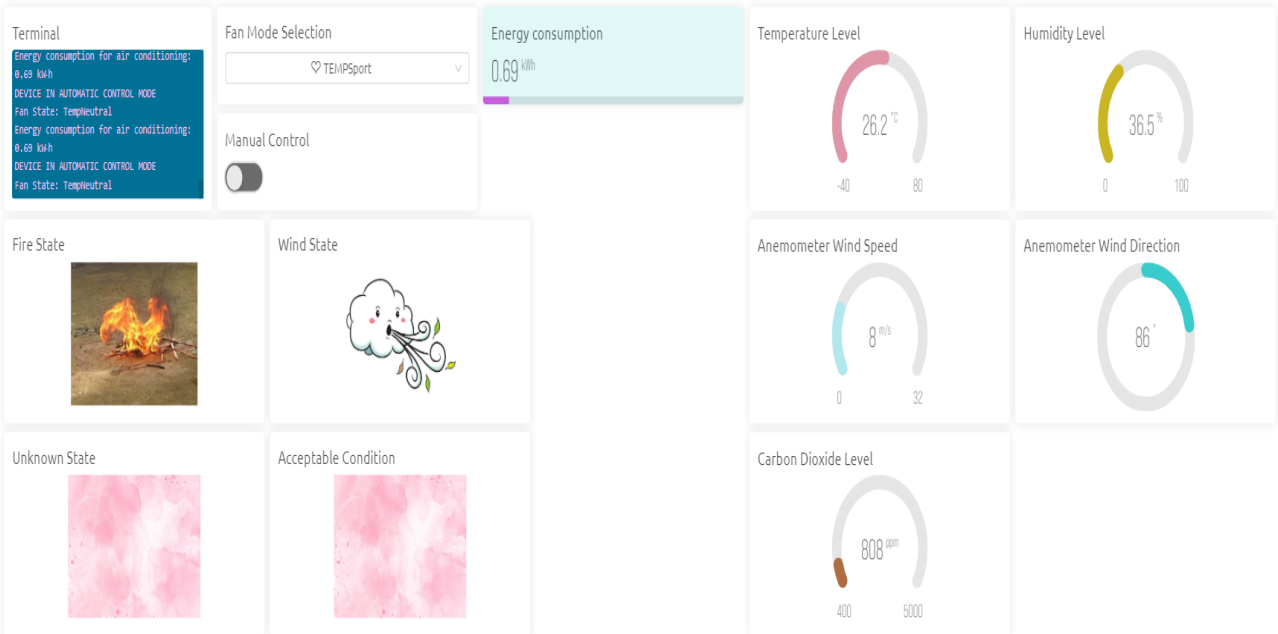
МОДЕЛЬ СИСТЕМИ. ДВИГУН У РЕЖИМІ TEMP DRIVE ТА ВІДПОВІДНІ ВІДОБРАЖЕННЯ ДАНИХ У СИСТЕМІ BLYNK



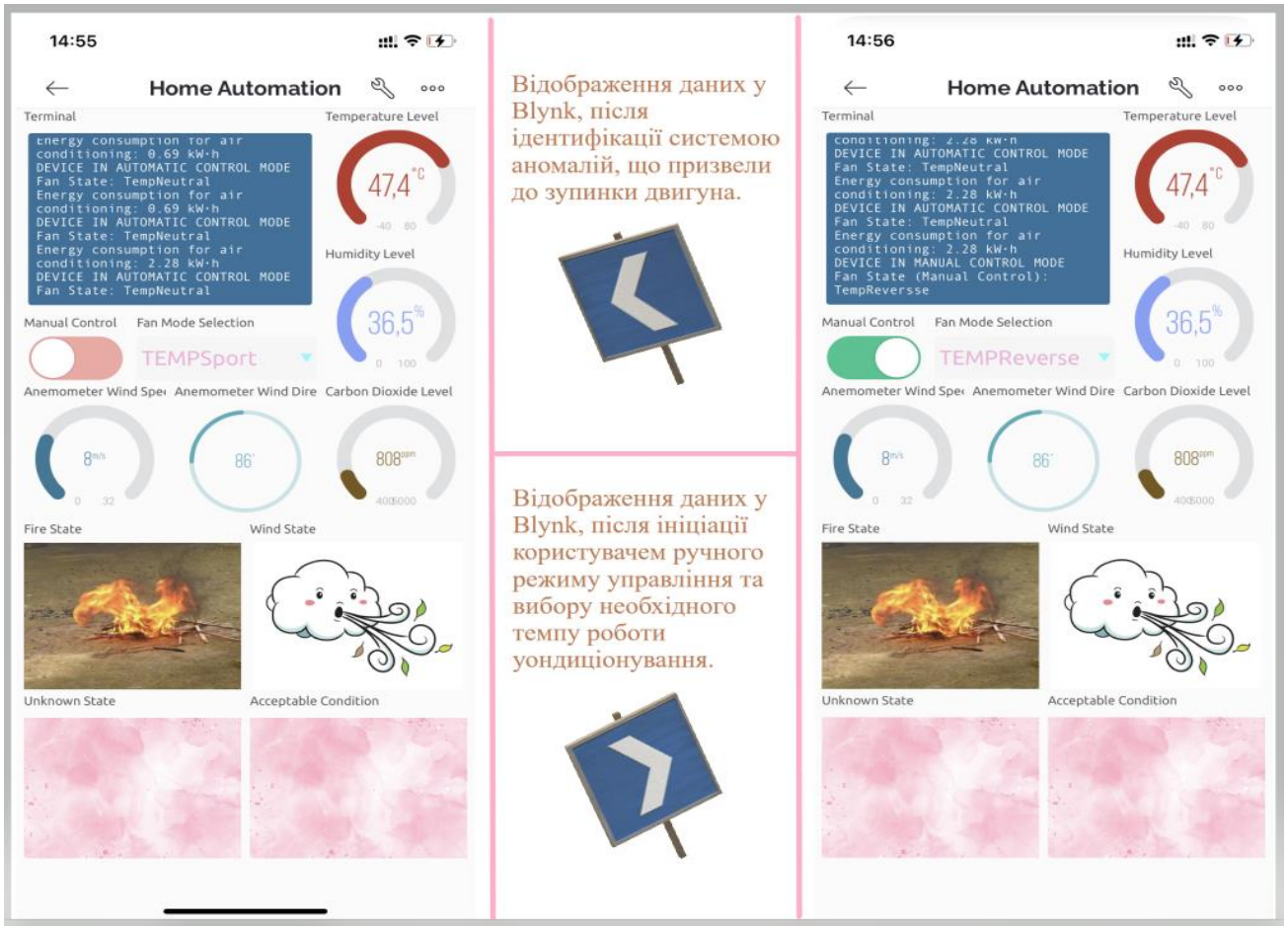


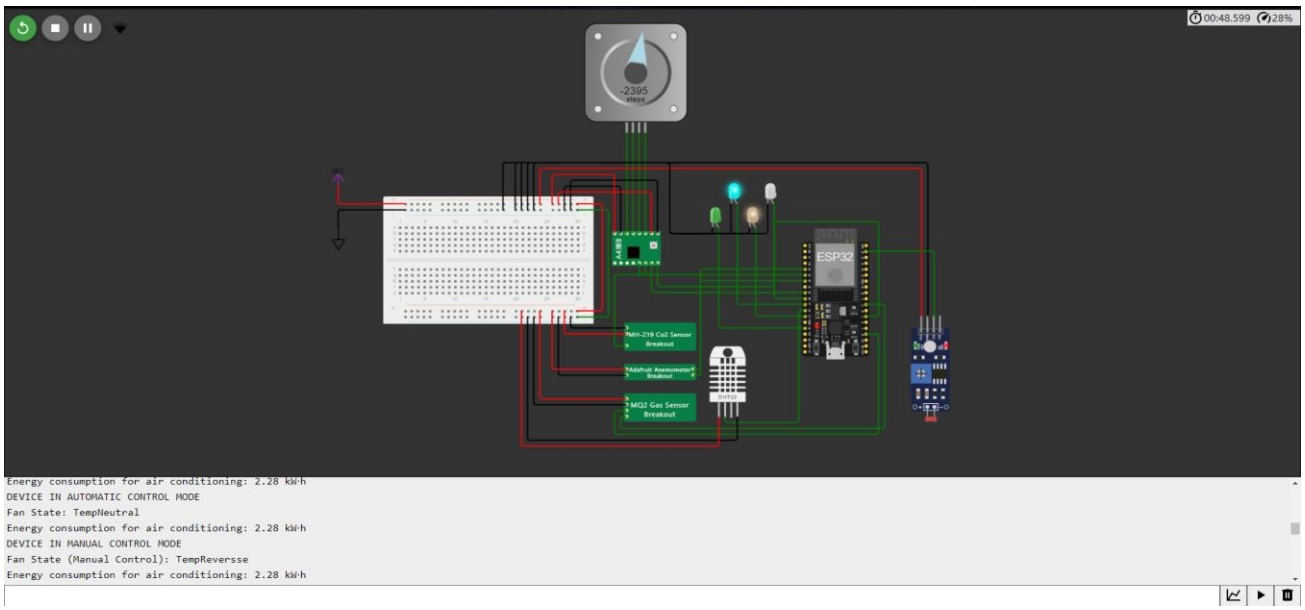
МОДЕЛЬ СИСТЕМИ. ІДЕНТИФІКАЦІЯ НАДЗВИЧАЙНИХ ПОДІЙ ТА ВИВЕДЕННЯ ВІДПОВІДНИХ ПОМИЛОК У BLYNK





ІНТЕРФЕЙС BLYNK У МОБІЛЬНОМУ ЗАСТОСУНКУ ТА МОДЕЛЬ СИСТЕМИ. РОБОТА В РЕЖИМІ РУЧНОГО УПРАВЛІННЯ





ДОДАТОК В
АПРОБАЦІЯ РЕЗУЛЬТАТІВ

Розроблення системи інтелектуального керування кондиціонуванням виробничого приміщення з використанням технології IoT

Бронніков А.І[1]., Нищета В.Є[2].
1. Кафедра КІТАМ, Харківський національний університет радіоелектроніки, УКРАЇНА, Харків, пр. Науки. 14., email: artem.bronnikov@nure.ua
2. Кафедра КІТАМ, Харківський національний університет радіоелектроніки, УКРАЇНА, Харків, пр. Науки. 14., email: viacheslav.nyshcheta@nure.ua

Анотація: Представлено інформацію з розроблення системи інтелектуального керування кондиціонуванням виробничого приміщення з використанням технології IoT. В якості техологій для розроблення системи було обрано технологію Internet of Things,а також плату керування Arduino для збору та обробки даних з сенсорів у реальному часі для прийняття найкращих управлінських рішень, забезпечуючи комфорт та ефективність роботи системи.

Ключові слова: розробка, системи інтелектуального керування, кондиціонуванням виробничого приміщення, Arduino, технологія Internet of Things(IoT).

I. ВСТУП

Сучасні виробничі приміщення стикаються з ростом вимог до оптимальних умов для виробничих процесів та зменшенням витрат на кондиціонування. У цьому контексті розроблення системи інтелектуального керування кондиціонуванням виробничого приміщення є актуальною та стратегічно важливою задачею [1].

Метою даного дослідження є розроблення системи, яка б забезпечувала оптимальні умови для виробничих процесів у виробничому приміщенні. Для досягнення цієї мети, ми використовуємо передову технологію Інтернету речей (IoT) та інтегруємо платформу Arduino в систему керування (рис. 1).

Рисунок 1 – Загальна концепція інтернет речей

Актуальність цієї задачі полягає в тому, що промисловість нині переживає період переосмислення стандартів і вимог до робочого середовища. Наявність комфортних умов у виробничому приміщенні впливає на ефективність праці, а також на здоров'я робітників. З іншого боку, ефективне управління кондиціонуванням може суттєво знизити витрати електроенергії, що стає все важливішою проблемою в умовах посиленого регулювання та вартості енергоресурсів [2].

Використання технології IoT в цій задачі дозволяє зібрати величезну кількість даних про стан приміщення, зовнішнього середовища та робочих процесів. Ця інформація використовується для аналізу та оптимізації роботи системи кондиціонування в реальному часі.

Однак інтеграція платформи Arduino має величезний потенціал для забезпечення надійності та ефективності системи. Arduino дозволяє розробити спеціалізовані датчики та актуатори, які відповідають за збір і обробку даних. Ми також розглянемо математичну модель, що допомагає визначити оптимальні параметри кондиціонування в залежності від зовнішніх умов та потреб виробничого процесу.

II. ТЕХНОЛОГІЯ IOT В ІНДУСТРІЇ

Технологія Internet of Things (IoT) стала ключовим фактором трансформації промисловості та індустріального сектора. Вона відкриває нові можливості для покращення продуктивності, ефективності та конкурентоспроможності виробництва.

Технологія Internet of Things – це концепція, за якої фізичні об'єкти та пристрої здатні з'єднуватися до Інтернету та обмінюватися даними, щоб забезпечувати збільшену автоматизацію, моніторинг та контроль. Вона включає в себе використання сенсорів, збору даних, аналізу та взаємодії з об'єктами та системами через мережу.

IoT дозволяє віддалено керувати та моніторити обладнання та машини в реальному часі. Це сприяє оптимізації виробничих процесів, підвищенню продуктивності та зменшенню витрат.

Сенсори IoT дозволяють стежити за станом обладнання та передбачати можливі відмови чи поломки. Це сприяє попередженню невідкладних ремонтів та підвищує

інших ресурсів для зменшення витрат і підвищення ефективності.

Збір та аналіз даних в реальному часі надає можливість приймати інформовані рішення та розробляти стратегії виробництва.

Переваги використання IoT в системах управління:

1. підвищення продуктивності: завдяки автоматизації та оптимізації процесів, IoT допомагає підвищити продуктивність виробництва;
2. зниження витрат: ефективне використання ресурсів, уникнення непередбачених збоїв та ремонтів дозволяють зменшити витрати;
3. покращення якості продукції: IoT допомагає виявляти і усувати проблеми на ранніх стадіях, що веде до покращення якості продукції;
4. збільшення безпеки: IoT включає системи моніторингу безпеки, що забезпечують безпеку працівників та обладнання;
5. вдосконалення аналізу та прийняття рішень: Збір та аналіз даних допомагає управлінцям [3].

III. ІНТЕГРАЦІЯ ARDUINO В СИСТЕМУ

Платформа Arduino грає важливу роль у системі керування кондиціонуванням завдяки своїм можливостям програмування та гнучкості [4].

Arduino може використовувати різноманітні сенсори, такі як температурні сенсори або сенсори вологості, для збору інформації про об'єктивні умови, а також виконує обробку зібраних даних для визначення поточних умов в приміщенні, таких як температура та вологість.

На основі аналізу даних приймається рішення щодо управління кондиціонуванням, наприклад, включає або вимикає кондиціонер для досягнення заданих параметрів.

Arduino може взаємодіяти з кондиціонером та іншими пристроями за допомогою реле, інфрачервоних передавачів або інших засобів комунікації.

Arduino може передавати інформацію про стан системи на інші пристрої або зберігати її для подальшого аналізу і контролю.

Опис сенсорів та пристроїв, що використовуються з Arduino:

1. температурні сенсори: наприклад, сенсори температури LM35 або DHT11 призначені для вимірювання температури в приміщенні;
2. сенсори вологості: до них відносяться сенсори вологості, такі як DHT11 або DHT22, які вимірюють вологість повітря;
3. сенсори руху: сенсори руху, такі як PIR-датчики, виявляють присутність людей у приміщенні;
4. датчики CO2: датчики CO2 вимірюють рівень вуглекислого газу в повітрі, що дає змогу визначати якість повітря;
5. інфрачервоні передавачі: їх можна використовувати для керування кондиціонером та іншими побутовими пристроями;
6. дисплеї: Arduino може бути підключена до LCD-дисплею або інших типів екранів для відображення інформації;
7. модулі комунікації: Arduino може взаємодіяти з іншими пристроями через Wi-Fi, Bluetooth або Інтернет.

Ці сенсори та пристрої спільно з Arduino допомагають збирати дані та керувати системою кондиціонування для забезпечення комфортних умов у приміщенні та раціонального споживання енергії [5].

IV. МАТЕМАТИЧНА МОДЕЛЬ СИСТЕМИ

Для прикладу розглянемо спрощену математичну модель, системи керування кондиціонуванням виробничого приміщення, яка враховує температурні умови в приміщенні [6].

Модель розрахунку витрат енергії на кондиціонування може бути виражена наступним рівнянням 1:

$$E = \frac{m*(Ti-To)}{R*C} \quad (1)$$

де: E – витрати енергії на кондиціонування (у ватах або кіловатах).

m – маса повітря, яке потрібно охолодити (у кількості кілограмів або літрах).

Ti – початкова температура повітря в приміщенні (у градусах Цельсія).

To – бажана температура в приміщенні (у градусах Цельсія).

R – коефіцієнт теплопровідності матеріалів ізоляції та стін (у ваттах на метр на градус Цельсія).

C – теплоємність повітря (у джоулях на кількість теплоносія).

Ця модель дозволяє розрахувати, скільки енергії буде витрачено на охолодження повітря виробничого приміщення з початковою температурою Ti до бажаної температури To, враховуючи теплоізоляцію і теплоємність повітря.

Розглянемо деякі можливості розширення математичної моделі, додавши до неї додаткові фактори та параметри, які впливають на роботу системи кондиціонування:

1. врахування кількості працівників: додавання параметру, який враховує кількість працівників в приміщенні, може бути корисним. чим більше людей працює в приміщенні, тим більше тепла вони виробляють, що може впливати на навантаження системи кондиціонування;
2. зовнішні умови: врахування зовнішніх температурних та вологості умов може покращити точність моделі. Зміни в погоді можуть впливати на теплообмін через вікна і стіни;
3. інші джерела навантаження: при використанні виробничого приміщення можуть бути інші джерела тепла, такі як обладнання, освітлення та інше. Ці джерела також можуть бути враховані в моделі;
4. динамічні зміни в часі: модель може бути розширена для врахування динамічних змін у витратах енергії протягом дня. Наприклад, витрати можуть змінюватися в залежності від часу доби або робочого графіку;

інтеграція даних з IoT-датчиків: якщо в вашій

1. системі використовуються датчики IoT, вони можуть надавати реальний час інформацію про температуру, вологість та інші параметри, які можуть бути використані для точніших розрахунків;

2. енергоефективність системи кондиціонування: врахування коефіцієнта ефективності вашої системи кондиціонування дозволить оцінити, наскільки ефективно вона використовує енергію;

3. адаптація до режимів роботи: модель може враховувати різні режими роботи системи кондиціонування, такі як режим охолодження, обігріву, вентиляції тощо.

Розширена математична модель може бути корисною для більш точних розрахунків витрат енергії та оптимізації роботи системи кондиціонування в залежності від різноманітних умов [7].

V. ВИСНОВКИ

Метою роботи є розроблення системи інтелектуального керування кондиціонуванням виробничого приміщення з використанням технології IoT.

У даній роботі було представлено результати досліджень і розробки системи інтелектуального керування кондиціонуванням виробничого приміщення, в основі якої лежать технологія Internet of Things (IoT) та інтеграція платформи Arduino. Створено систему, яка дозволяє зменшити витрати енергії та сприяє екологічно чистому виробничому процесу.

У підсумку було отримано наступні ключові результати та висновки:

1. актуальність та важливість теми: зростаюча конкуренція та зміни в кліматичних умовах потребують нових підходів до управління кондиціонуванням виробничих приміщень. Ефективне та інтелектуальне керування стає обов'язковим для підтримання оптимальних умов роботи та збільшення продуктивності [8];

2. використання технології IoT: IoT виявляється потужним інструментом для збору та обробки даних з сенсорів у реальному часі. Вона дозволяє використовувати ці дані для прийняття найкращих управлінських рішень, забезпечуючи комфорт та ефективність;

3. інтеграція Arduino: платформа Arduino стала центральною фігурою в нашій системі, забезпечуючи зчитування даних з різних сенсорів та керування обладнанням. Це відкриває нові можливості для розробки інтелектуальних алгоритмів та оптимізації роботи кондиціонування;

4. математична модель: ми розробили математичну модель, яка дозволяє прогнозувати параметри кондиціонування на основі зібраних даних та динамічних характеристик системи. Ця модель спрощує прийняття рішень та підвищує точність управління.

За підсумками нашої роботи, ми впевнені, що використання технології IoT та платформи Arduino, разом із нашою математичною моделлю, може значно покращити управління кондиціонуванням у виробничих приміщеннях. Ця система може бути важливим кроком до створення більш ефективного та екологічно чистого виробничого середовища.

Розроблена система інтелектуального керування кондиціонуванням з використанням технології IoT відкриває широкі можливості для покращення умов у виробництві, зменшення енерговитрат і підвищення продуктивності. Практична направленість пов’язана з можливістю впровадження для створення більш стійких та ефективних виробничих середовищ.

ПЕРЕЛІК ПОСИЛАНЬ

[1] Інтелектуальні системи управління: Експертні системи – основи проектування та застосування в системах автоматизації [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» / КПІ ім. Ігоря Сікорського; уклад.: Л. Д. Ярощук. – Електронні текстові дані (1 файл: 2,56 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2019. – 136с.

[2] Економіка галузевих ринків: Навчально-методичний комплекс для студентів спеціальності «051 Економіки», освітніх програм «Економіка та економічна політика», «Економіка та право», «Економіка підприємства» / Упорядники Ігнатюк А.І., Колоша В.В., Коваленко О.Я.– К., 2018. – 44 с.

[3] Технології інтернету речей. Навчальний посібник [Електронний ресурс]: навч. посіб. для студ. спеціальності 126 «Інформаційні системи та технології», спеціалізація «Інформаційне забезпечення робототехнічних систем» / Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 12,5 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 271 с.

[4] I. Perenc, T. Jaworski, and P. Duch, ‘‘Teaching programming using dedicated Arduino educational board,’’ Comput. Appl. Eng. Educ., vol. 27, no. 4, pp. 943–954, Jul. 2019.

[5] P. Duch and T. Jaworski, ‘‘Enriching computer science programming classes with Arduino game development,’’ in Proc. 11th Int. Conf. Hum. Syst. Interact. (HSI), Jul. 2018, pp. 148–154.

[6] Математичні моделі та новітні технології управління Конспект лекцій дисципліни «Математичне моделювання» для студентів спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології»/ Упоряд. В.В. Безкоровайний. – Харків: ХНУРЕ, 2018. – 120 с.

[7] В. В. Лимаренко, "Інформаційна система підтримки рішень для автоматизації створення технологічних процесів механообробки деталей високоточного обладнання", дис.. канд. техн. наук, Нац. техн. ун-т "ХПІ", Харків, 2019.

[8] Зміна клімату: наслідки та заходи адаптації: аналіт. доповідь / [С.П. Іванюта, О. О. Коломієць, О. А. Малиновська, Л. М. Якушенко]; за ред. С. П. Іванюти. – К. : НІСД, 2020. – 110 с.

№	Формат	Позначення	Найменування	Кіл.	Прим.
			Документація текстова		
1	А4	ГЮІК. 505400.007 ПЗ	Пояснювальна записка	102 с.	
			Інші документи		
2	А4		Додаток А. Текст програми	14 с.	
3	А4		Додаток Б. Демонстраційний матеріал кваліфікаційної роботи	16 с.	
			Додаток В. Апробація результатів	4 с.	

Змін.	Арк.	Номер докум.	Підп.	Дата	ГЮИК.505400.007 ВД			
Розроб.		Нищета В. Є.			Розроблення системи інтелектуального керування кондиціонуванням виробничого приміщення з використанням технології IoT.	Літера	Аркуш	Аркушів
Перевір.		Косенко В.В.						1
						ХНУРЕ кафедра КІТАР		
Н.контр.		Демська Н. П.						
Затв.		Невлюдов І.Ш.						