

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Кафедра ПЗ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни
«Фізичні основи сучасних інформаційних технологій»
для студентів за спеціальністю 122 Комп'ютерні науки
усіх форм навчання

2019

Методичні вказівки до виконання лабораторних робіт з дисципліни «Фізичні основи сучасних інформаційних технологій» для студентів за спеціальністю 122 Комп'ютерні науки усіх форм навчання. /Укл.: А.В. Пархоменко, О.М. Гладкова – Запоріжжя: НУ «Запорізька політехніка», 2019. – 55 с.

Укладачі: А.В. Пархоменко, к.т.н., доцент кафедри ПЗ,
О.М. Гладкова, к.т.н., доцент кафедри ПЗ.

Рецензент: Г.М. Шило, д.т.н., зав. кафедри ІТЕЗ.

Відповідальний
за випуск: С.О. Субботін, д.т.н., професор, зав. кафедри ПЗ.

Затверджено
на засіданні кафедри
програмних засобів
Протокол № 9 від “13” червня 2019

ЗМІСТ

ВСТУП	5
1 Лабораторна робота 1. Розробка інтерактивного графічного інтерфейсу для взаємодії з платформою ARDUINO	6
1.1 Теоретичні відомості	6
1.1.1 <i>Взаємодія Arduino і Processing</i>	<i>6</i>
1.1.2 <i>Середовище Processing IDE</i>	<i>7</i>
1.1.3 <i>Базові функції</i>	<i>8</i>
1.1.4 <i>Змінні за замовчанням</i>	<i>9</i>
1.2 Приклади виконання лабораторної роботи	9
1.2.1 <i>Увімкнення / вимкнення вбудованого LED'а</i>	<i>9</i>
1.2.2 <i>Регуляція яскравості LED'а</i>	<i>12</i>
1.2.3 <i>Робота з сервоприводом та потенціометром</i>	<i>15</i>
1.2.4 <i>Обробка помилок</i>	<i>22</i>
1.3 Завдання до виконання лабораторної роботи	22
1.4 Контрольні запитання	23
1.5 Зміст звіту	23
2 Лабораторна робота 2. Підключення датчиків до ARDUINO	24
2.1 Теоретичні відомості	24
2.1.1 <i>Датчики та їх використання</i>	<i>24</i>
2.1.2 <i>Підключення датчиків до Arduino</i>	<i>30</i>
2.1.3 <i>Текст програми</i>	<i>31</i>
2.1.4 <i>Основні функції</i>	<i>33</i>
2.2. Завдання до виконання лабораторної роботи	34
2.3 Контрольні запитання	34
2.4 Зміст звіту	34

3 Лабораторна робота 3. Налаштування Raspberry Pi та підключення до Arduino	35
3.1 Теоретичні відомості	35
3.1.1 Встановлення ОС Raspbian	36
3.1.2 Приклад встановлення ОС Raspbian за допомогою пакета NOOBS.....	36
3.1.3 Приклад встановлення ОС Raspbian за допомогою монтування файлу–образу.....	37
3.1.4 Перший запуск ОС Raspbian	39
3.1.5 Налаштування мережі Інтернет	43
3.1.6 Підключення Arduino із завантаженим скетчем.....	44
3.2 Завдання до виконання лабораторної роботи	45
3.3 Контрольні запитання	45
3.4 Зміст звіту	46
4 Лабораторна робота 4. Встановлення та налаштування OpenHAB на Raspberry Pi	47
4.1 Теоретичні відомості	47
4.1.1 Встановлення та налаштування OpenHAB на Raspberry Pi	48
4.2 Завдання до виконання лабораторної роботи	54
4.3 Контрольні запитання	54
4.4 Зміст звіту	54
РЕКОМЕНДОВАНА ЛІТЕРАТУРА	54

ВСТУП

Розумний будинок - єдина система управління в будинку або квартирі, що включає в себе датчики, керуючі елементи і виконавчі пристрої. Керуючі елементи приймають сигнали з датчиків і контролюють роботу виконавчих пристроїв, діючи відповідно до заданих алгоритмів та об'єднуючи такі підсистеми:

- генерації електроенергії (сонячна або вітрогенерація);
- контролю енергоспоживання, обмеження пікових навантажень і розподілу навантажень по фазах мережі живлення;
- управління джерелами резервного електроживлення (акумуляторними і дизель-генераторами);
- опалення будинку та постачання гарячої води (за допомогою котлів на різних видах палива або електроенергії, теплових насосів, геліосистеми);
- вентиляції, кондиціонування, рекуперації, пасивного охолодження;
- охоронної та пожежної сигналізації;
- контролю доступу;
- контролю аварійних ситуацій (витоку води, газу, аварії в електромережі);
- відеоспостереження (локальне та віддалене);
- управління внутрішнім і вуличним освітленням;
- мультирум та розподілу відео й аудіо;
- управління обігрівом зливової каналізації, сходів і доріжок;
- управління каналізаційною насосною станцією та системою автополиву зелених територій;
- управління воротами і шлагбаумами;
- управління шторами, ролетами і жалюзі;
- віддаленого моніторингу і управління всіма підсистемами будинку через Інтернет.

Технології створення Розумного будинку цікаві і корисні, оскільки вони дозволяють зробити наше життя більш комфортним, безпечним і забезпечити економію ресурсів.

1 ЛАБОРАТОРНА РОБОТА 1. РОЗРОБКА ІНТЕРАКТИВНОГО ГРАФІЧНОГО ІНТЕРФЕЙСУ ДЛЯ ВЗАЄМОДІЇ З ПЛАТФОРМОЮ ARDUINO

Мета роботи: виконати розробку графічного інтерфейсу для взаємодії з платформою Arduino з використанням середовища Processing IDE.

1.1 Теоретичні відомості

Інтегроване середовище розробки Arduino IDE є крос-платформним додатком, який написаний на мові програмування Java. Воно було створено з IDE для мов Processing [1] і Wiring [2]. Як і Wiring, Arduino була створена з метою спрощення програмування мікроконтролерів. Тобто, щоб користувачі, які не мають навичок роботи з електронікою могли легко розробити власний проект на базі мікроконтролера та інших датчиків.

Середовище Processing IDE навпаки було створене з метою дати можливість студентам технічних спеціальностей більш простий спосіб роботи з графікою.

Таким чином, оскільки Arduino і Processing IDE тісно пов'язані між собою, а саме, один є нащадком іншого, вони можуть взаємодіяти, що дозволяє створити простий графічний інтерфейс для зручного керування Arduino платформою або навпаки.

1.1.1 Взаємодія Arduino і Processing

Дані можуть передаватися між IDE Processing і платою Arduino за допомогою порту послідовної взаємодії. Для роботи з послідовним портом у Arduino і Processing існує бібліотека Serial, в якій містяться функції для спрощення реалізації послідовного обміну даними. Serial - це формат даних, який надсилає один байт за один раз. Для Arduino, байти є тип даних, який може зберігати значення від 0 до 255; він працює як цілочисловий формат даних int, але з набагато меншим діапазоном. Великі номери відправляються з розбиттям в список байтів та наступним їх повторним збиранням.

В наведеному нижче прикладі реалізовано просту програму для демонстрації двостороннього зв'язку між Arduino та Processing.

1.1.2 Середовище Processing IDE

Якщо відкрити файл `processing.exe`, то з'явиться вікно, зображене на рис. 1.1. Як бачимо, воно абсолютно схоже з середовищем Arduino IDE. Це так зване вікно створення скетчу, тобто вихідного тексту програми. Воно складається з наступних елементів: Меню, Панель інструментів, Вкладки, Текстовий редактор, Вікно повідомлень та Консоль.

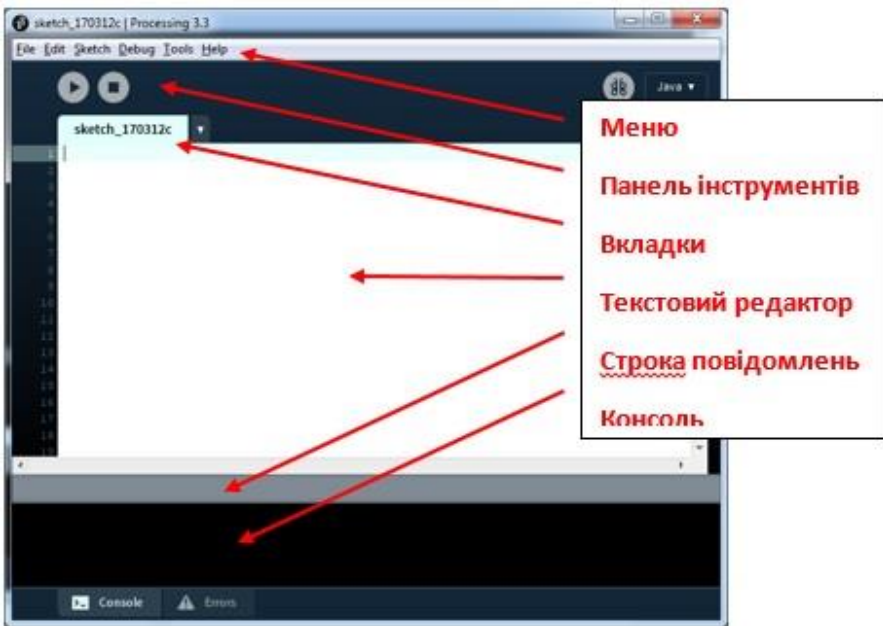


Рисунок 1.1 – Інтерфейс Processing

Мова програмування Processing є діалектом Java, але з деякими особливостями для роботи з графікою та інтерактивним наповненням. Processing багато що взяв з PostScript (що послужило основою для формату PDF) та OpenGL (специфікація 3D графіки).

Для того, щоб розпочати роботу з Processing, необхідно володіти знаннями щодо основ утворення RGB кольорів та розуміння пікселів. Також, необхідно вміти працювати з Arduino.

Щоб побудувати першу програму треба оволодіти базовими функціями та змінними за замовчанням (для більш детального вивчення синтаксису мови та принципів роботи з Processing дивіться [1,3]).

1.1.3 Базові функції

void setup() та *void draw()* – основні функції роботи програми, усі інші функції пишуться усередині цих двох циклів; *void setup()* виконується один раз на початку роботи програми; *void draw()* – це функція нескінченного циклу, вона виконується постійно.

size(X, Y) – створює робочу область (або вікно) розміром X на Y пікселів, за замовчуванням вона 100 на 100 пікселів.

background(a) – задає колір фону; *background(a)* еквівалентно запису *background(a,a,a)*, де три змінні відповідають основним кольорам колірної моделі RGB: червоному, зеленому та синьому.

fill(R, G, B) – вказує колір заповнення фігури, можна додати четвертий параметр до цієї функції T – це прозорість, за замовчуванням, якщо не вказувати цей параметр то він дорівнює 255, тобто 100% не прозоре.

line(x1, y1, x2, y2) – малює лінію з точки (x1, y1) до точки (x2, y2).

rect(x, y, w, h) – малює прямокутник, починаючи з лівої верхньої точки (x, y), з параметрами w – довжина, h – висота, можна також додати п'ятий параметр t – радіус округлення кутів прямокутника.

mousePressed() – викликається кожного разу, коли натискається кнопка миші.

Serial.begin(швидкість_передачі) – встановлює швидкість передачі інформації COM порту в «біт в секунду» для послідовної передачі даних. Для того щоб підтримувати зв'язок з комп'ютером, використовується одна з нормованих швидкостей: 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, або 115200 (наприклад, *Serial.begin(4800);*). За замовчанням в моніторі послідовного порту Arduino встановлена швидкість 9600 біт/сек.

Serial.available() – байти які надходять з послідовного порту потрапляють до буферу мікроконтролера, звідки програма може їх зчитати. Функція повертає кількість накопичених в буфері байт. Послідовний буфер може зберігати до 128 байт.

Повертає значення типу `uint8_t` (`typedef uint8_t byte;`) – кількість байт, які знаходяться у доступі для читання у послідовному буфері, або 0, якщо нічого не доступне.

Serial.read() – зчитує наступний байт з буферу послідовного порту. Повертає перший доступний байт вхідних даних з послідовного порту, або -1 якщо нема вхідних даних.

1.1.4 Змінні за замовчанням

mouseX, mouseY // використовує положення курсора в робочому екрані по осі X та Y;

height, width // відповідають значенням висоти и ширини робочого екрану, які задаються функцією `size()`.

1.2 Приклади виконання лабораторної роботи

Розглянемо декілька прикладів.

1.2.1 Увімкнення / вимкнення вбудованого LED'а

Відкрийте середовище Arduino IDE. Скопіюйте код, наведений у прикладі нижче, або відкрийте файл **Arduino_led.ino**, відкомпілюйте та завантажте програму до плати Arduino.

Приклад для Arduino:

```
char val; // дані, які отримуються з послідовного
порту
int ledPin = 13; // ініціалізація 13 вбудованого
піна плати

void setup() {
  pinMode(ledPin, OUTPUT); // режим роботи піна на
вихід
```

```

    Serial.begin(9600); // розпочати обмін даними з
    9600 біт/сек.
    }

    void loop() {
        while (Serial.available()) { // якщо порт
доступний
            val = Serial.read(); // читати дані та записувати
в val
        }
        if (val == 'H') { // якщо було отримано H
            digitalWrite(ledPin, HIGH); // LED загоряється
        } else {
            digitalWrite(ledPin, LOW); // в іншому разі
вимкнений
        }
    }
}

```

Після цього, відкрийте середовище Processing, скопіюйте програму, яка наведена нижче у прикладі та збережіть програму на диску комп'ютера (**File -> Save**) або відкрийте файл **Processing_led.pde**.

Приклад для Processing:

```

import processing.serial.*;

Serial arduino; // створення об'єкту класу Serial

void setup()
{
    size(390, 200);
    String portName = Serial.list()[0];
    arduino = new Serial(this, portName, 9600);
    delay(1000);
}

void draw() {
    // Turn ON/OFF LED
    textAlign(CENTER);
    textSize(32);
}

```

```

fill(153, 100);
text("On/Off LED", 197, 42);
fill(0, 102, 153);
text("On/Off LED", 195, 40); // створення тіні
тексту
stroke(102);

    if(mouseOverRect() == true && mousePressed) {
//перевірка чи курсор миші над кнопкою
        fill(0, 117, 13, 240);
        rect(147, 77, 96, 46, 7); // відображення
кнопки
        arduino.write('H'); // відправлення H для
визначення, що курсор над квадратом
    }
    else {
        fill(0, 200, 51); // зміна кольору
        rect(145, 75, 100, 50, 7); //
малювання меншого квадрата для створення ефекту
натискання кнопки
        arduino.write('L') // у іншому випадку
надсилаємо L
    }
}

boolean mouseOverRect() { // перевірка чи курсор
над кнопкою
    return ((mouseX >= 145) && (mouseX <= 245) &&
(mouseY >= 75) && (mouseY <= 125));
}

```

Клікніть на панелі інструментів кнопку **Run (Запуск)**, це запустить компіляцію програми.

У результаті ви побачите вікно у якому зображена кнопка (див. рис. 1.2). При натисненні на цю кнопку, вона змінить колір, а також вбудований світлодіод на 13 порту платформи Arduino загориться.



Рисунок 1.2 – Увімкнення / вимкнення вбудованого LED’а

1.2.2 Регуляція яскравості LED’а

Наступний приклад дозволяє регулювати яскравість LED’а за допомогою змінення положення повзунка.

Відкрийте середовище Arduino IDE. Скопіюйте код, наведений у прикладі нижче, або відкрийте файл **Arduino_Brightness.ino**, відкомпілюйте та завантажте програму до плати Arduino.

Виконайте підключення світлодіоду та резистора до плати Arduino, як зображено на рис. 1.3.

Приклад для Arduino:

```
int val;
int ledPin = 4;

void setup() {
  pinMode(ledPin, OUTPUT); // Встановлення 4 піну
  як OUTPUT (працює на відправлення даних з плати)
  Serial.begin(9600); // початок послідовного
  зв’язку зі швидкістю 9600 біт/сек.
}

void loop() {
```

```

while (Serial.available()) { // данні доступні
для читання
    val = Serial.read(); // читання даних и
зберігання їх у змінну val
}

if (val>=0 && val<=255){
    analogWrite(ledPin, val); // встановлення
яскравості LED4
}
Serial.println(val);
}

```

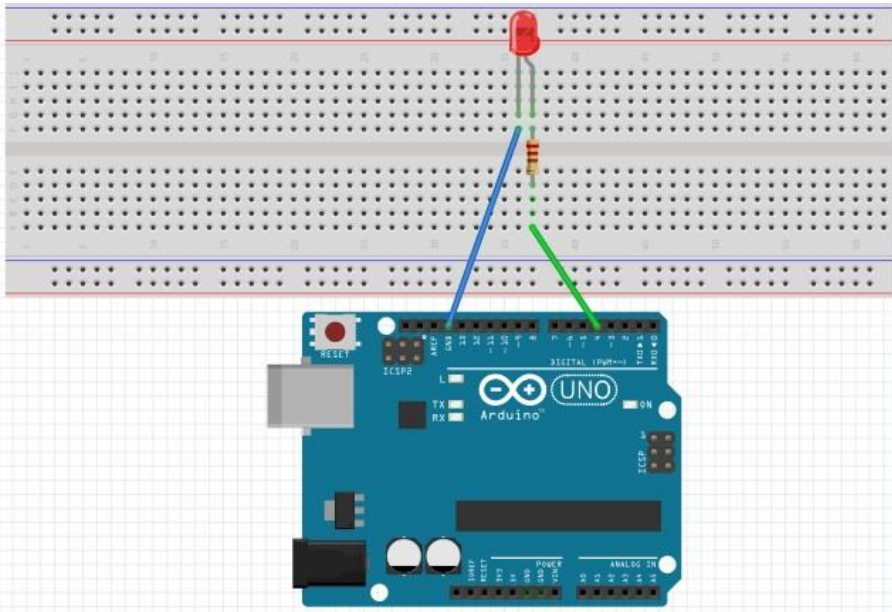


Рисунок 1.3 – Схема підключення світлодіоду та резистора до Arduino

Після цього, відкрийте середовище Processing, скопіюйте програму, яка наведена нижче у прикладі та збережіть програму на

дискун комп'ютера (**File -> Save**) або відкрийте файл **Processing_brightness.pde**.

Приклад для Processing:

```
import processing.serial.*; // імпорт бібліотеки
serial
Serial arduino; // створення об'єкту

int x = 10;

void setup(){
  size(300, 200);
  arduino = new Serial(this, Serial.list()[0], 9600);
  delay(1000);
}

void draw(){
  background(255);

  // Яскравість LED
  textAlign(CENTER);
  textSize(32);
  fill(153, 100);
  text("LED Brighthness", 147, 42);
  fill(0, 102, 153);
  text("LED Brighthness", 145, 40); // створення
  тіні для тексту

  // вимальовування керуючого повзунка
  fill(240);
  triangle(20, 115, 270, 95, 270, 115);
  line(20, 115, 270, 115);
  for(int i = 20; i<271; i=i+125){
    stroke(78);
    line(i, 95, i, 115);
  }
  fill(0, 0, 255);
  ellipse(x, 110, 20, 30);
  stroke(156);
  noFill();
```

```

    ellipse(x, 110, 18, 28);

    if(mouseOver() == true && mousePressed){ //
перевірка чи змінилась позиція повзунка
        x=mouseX;
    }
    else if(x>270) {
        x=270;
    }else if(x<20){
        x=20;
    }
    int pos = int (map(x, 20, 270, 0, 255)); //
присвоєння положення курсору значенню від 0 до 255
    arduino.write(pos);
}

boolean mouseOver(){ // перевірка чи курсор над
повзунком
    return((mouseX >= 20) && (mouseX <= 275) &&
(mouseY >= 75) && (mouseY <= 130));
}

```

Клікніть на панелі інструментів кнопку **Run (Запуск)**, це запустить компіляцію програми.

У результаті запуску цієї програми, з'явиться вікно, у якому зображений повзунок (див. рис. 1.4). Натисніть клавішу миші та перетягніть повзунок вправо. Після цього яскравість світлодіоду збільшиться. Якщо перетягнути повзунок вліво, то яскравість світлодіоду зменшиться.

1.2.3 Робота з сервоприводом та потенціометром

У наступному прикладі ми розглянемо взаємодію Processing з сервоприводом та потенціометром.

Відкрийте середовище Arduino IDE. Скопіюйте код, наведений у прикладі нижче або відкрийте файл **Arduino.ino**, відкомпілюйте та завантажте програму до плати Arduino.

Виконайте підключення сервоприводу та потенціометру до плати Arduino, як зображено на рис. 1.5.

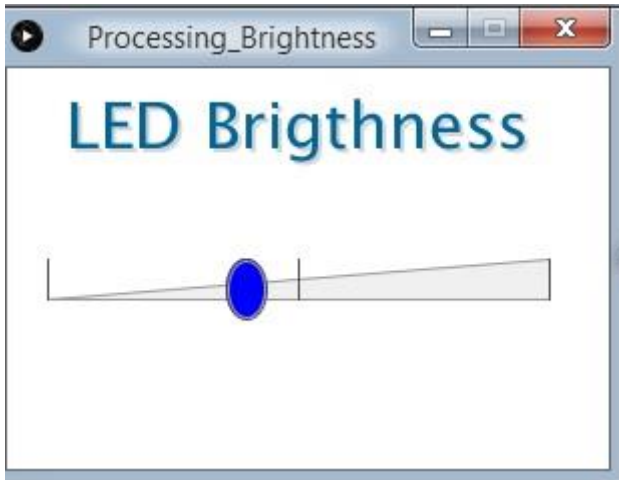


Рисунок 1.4 – Регулювання яскравості LED'а

Приклад для Arduino:

```
#include <Servo.h>
int val=0; // змінна куди будемо записувати значення
яскравості від положення ручки потенціометра
int vall; // зміна в яку будемо записувати значення
куту повороту для Servo
Servo myServo; // Зміна для роботи з сервоприводом

void setup() {
  myServo.attach(5); // Підключаємо серво до 5 порту
  Serial.begin(57600); // Розпочинаємо послідовний
зв'язок зі швидкістю 57600
}

void loop() {
  while (Serial.available()) { // якщо данні доступні
до читання,
    vall = Serial.read(); // прочитати та зберегти їх в
змінну vall
  }
  myServo.write(vall); // Записуємо значення vall до
серво, він починає обертатися
```

```

    val=analogRead(A0)/4;    // зчитуємо значення з
    аналогового входу і записуємо в змінну val
    Serial.write(val); // відправляємо записані значення
    в до послідовного порту
    delay(500); // Чекаємо 500 мілісекунд до наступного
    читання
}

```

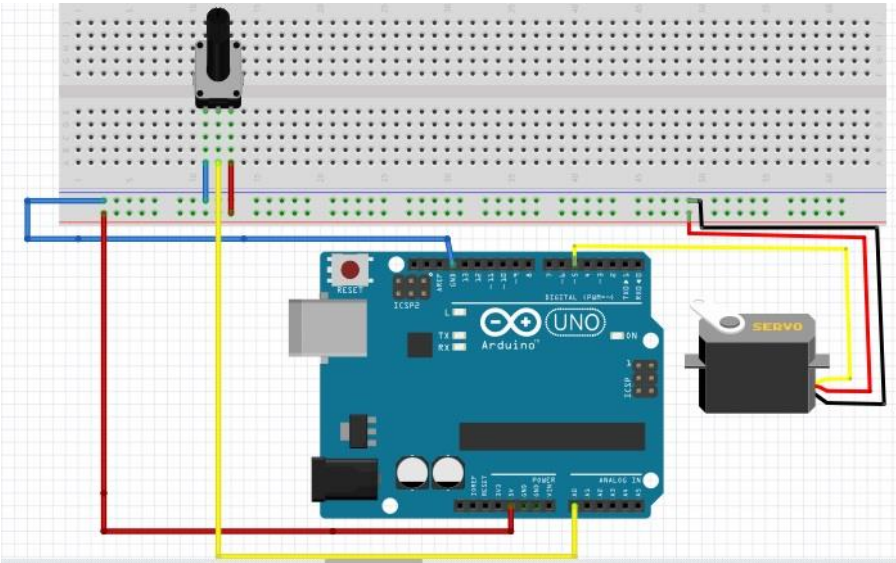


Рисунок 1.5 – Схема підключення серво та потенціометра до Arduino

Після цього, відкрийте середовище Processing, скопіюйте програму, яка наведена нижче у прикладі та збережіть програму на диску комп'ютера (**File -> Save**) або відкрийте файл **processing_interface.pde**.

Приклад для Processing:

```

import processing.serial.*;    // підключається
бібліотека serial
Serial myPort; // створення об'єкту класу Serial

```

```
int val;          // дані які відправляються/зчитуються
до/з Arduino
```

```
PFont f;
color rectHighlight;
color rectColor;
// Створюємо змінні для перевірки положення курсора
boolean rectOver1 = false;
boolean rectOver2 = false;
boolean rectOver3 = false;
```

```
void setup()
{
    size(650, 250);
    rectColor = color(255, 0, 0);
    rectHighlight = color(255,100,100);
    // Створюємо шрифт
    printArray(PFont.list());
    f = createFont("Arial-Black.vlw", 12);
    textFont(f);
    String portName = Serial.list()[1];
    myPort = new Serial(this, portName, 57600); //
Створюємо послідовний порт з ім'ям portName та
швидкістю послідовного зв'язку 57600
}
```

```
void draw()
{
    update(mouseX, mouseY);
    background(255);
    smooth();
    stroke(0);
    fill(255);
    arc(150, 162, 149, 124, -PI, 0); // малюємо дугу
    // Малюємо риски
    line(75,162,80,162);
    line(225,162, 220,162);
    line(150, 100, 150, 105);
```

```
// Зміна кольору при наведення курсора миші над
прямокутником «0 degree»
    if (rectOver1) {
```

```

        fill(rectHighlight);
    } else {
        fill(rectColor);
    }
    //створення прямокутника
    rect(25, 150, 50, 25); // малюємо прямокутники
«0 degree»

```

```

    // Зміна кольору при наведення курсора миші над
    прямокутником «180 degree»
    if (rectOver2) {
        fill(rectHighlight);
    } else {
        fill(rectColor);
    }
    rect(225, 150, 50, 25); // малюємо прямокутники
«180 degree»

```

```

    // Зміна кольору при наведення курсора миші над
    прямокутником «90 degree»
    if (rectOver3) {
        fill(rectHighlight);
    } else {
        fill(rectColor);
    }
    rect(125, 75, 50, 25); // малюємо прямокутники
«90 degree»

```

```

    //Створюємо текст за відповідними координатами
    textAlign(CENTER);
    fill(0);
    textSize(12);
    text("0 deg", 50, 167);
    text("180 deg", 250, 167);
    text("90 deg", 150, 92);
    textSize(16);
    text("From Processing To Arduino", 150, 20);
    text("From Arduino To Processing ", 480, 20);
    text("Experiment with Servo", 150, 220);
    text("Experiment with Potentiometer", 480, 220);
    if ( myPort.available() > 0) { // якщо порт
відкритий

```

```

        val = myPort.read();           // читати дані з
нього і зберегти у змінну val
    }
    textSize(12);
    text("Value", 365, 115); // напис Value
    stroke(0); // колір лінії
    line (350, 120, 605, 120); // лінія для розмітки
    for (int i = 350; i < 605; i = i+10) { // риски
через 10 пікселів
        stroke(0); // колір рисок
        line(i, 120, i, 123); // рисуємо риски
    }
    fill(0); // колір повзунка
    noStroke(); // без контуру прямокутник
    rect(350, 125, val, 10); // рисуємо прямокутник
(повзунок), який змінює свою ширину в залежності від
отриманого значення у змінну val
    delay(100);
}

// Перевіряємо чи знаходиться курсор над одним з
прямокутників
void update(int x, int y) {
    // «0 degree»
    if ( overRect(25, 150, 50, 25) ) {
        rectOver1 = true;
    } else {
        rectOver1 = false;
    }

    // «180 degree»
    if ( overRect(225, 150, 50, 25) ) {
        rectOver2 = true;
    } else {
        rectOver2 = false;
    }

    // «90 degree»
    if ( overRect(125, 75, 50, 25) ) {
        rectOver3 = true;
    } else {
        rectOver3 = false;
    }
}

```

```

    }
}

void mousePressed() {
    if (rectOver1) { // Якщо кнопка миші
        натиснута на прямокутнику «0 degree»,
        myPort.write(0); // відправляємо до
        послідовного порту '0'
    }
    if (rectOver2) { // Якщо кнопка
        миші натиснута на прямокутнику «180 degree»,
        myPort.write(180); // відправляємо
        значення '180' до послідовного порту
    }
    if (rectOver3) { // Якщо кнопка миші
        натиснута на прямокутнику «90 degree»,
        myPort.write(90); // відправляємо
        значення '90' до послідовного порту
    }
}

boolean overRect(int x, int y, int width, int
height) {
    if (mouseX >= x && mouseX <= x+width &&
        mouseY >= y && mouseY <= y+height) {
        return true;
    } else {
        return false;
    }
}

```

Клікніть на панелі інструментів кнопку **Run (Запуск)**, це запустить компіляцію програми.

У результаті ви побачите вікно, що відображає роботу Processing, у якому зліва зображені 3 прямокутні кнопки для роботи із серво, з права шкала зі значенням положення ручки потенціометру (рис. 1.6). Після натиснення на кнопку Processing відправляє значення кута повороту, до Arduino. Після цього сервопривід, підключений до Arduino здійснює обертання за отриманим значенням із Processing. У свою чергу, при обертанні ручки потенціометра підключеного до Arduino, буде змінюватися довжина повзунка, що відображає нові отримані значення.

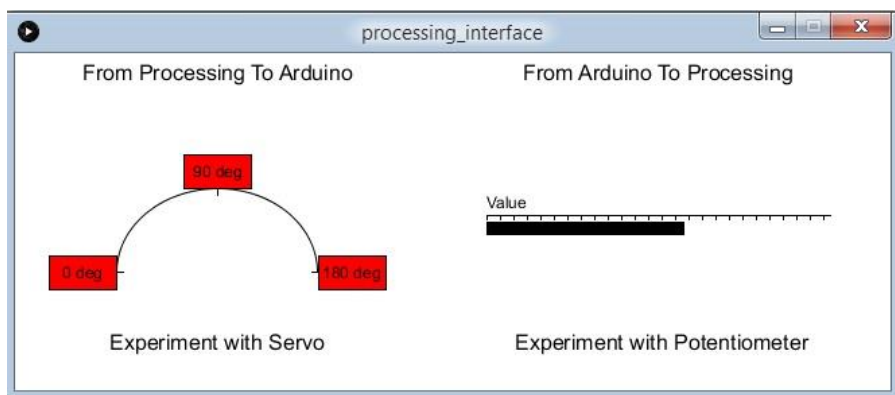


Рисунок 1.6 – Результат роботи в Processing

1.2.4 Обробка помилок

Якщо у вас декілька пристроїв, підключених до послідовного порту ПК, то можливо при компіляції вам буде висвічуватись помилка стосовно послідовного порту. У цьому випадку спробуйте змінити значення у `String portName = Serial.list()[0];` з `[0]` на `[1]`, або `[2]`, це залежить від того, скільки пристроїв підключено по ПК.

Перелік усіх доступних послідовних портів можна побачити, після натискання кнопки запуску програми **Run** у консолі **Console** (див. рис. 1.7).

1.3 Завдання до виконання лабораторної роботи

1.3.1 Ознайомитися з теоретичними відомостями.

1.3.2 Виконати приклади роботи з світлодіодом, потенціометром, сервоприводом.

1.3.3 Оформити звіт до лабораторної роботи.

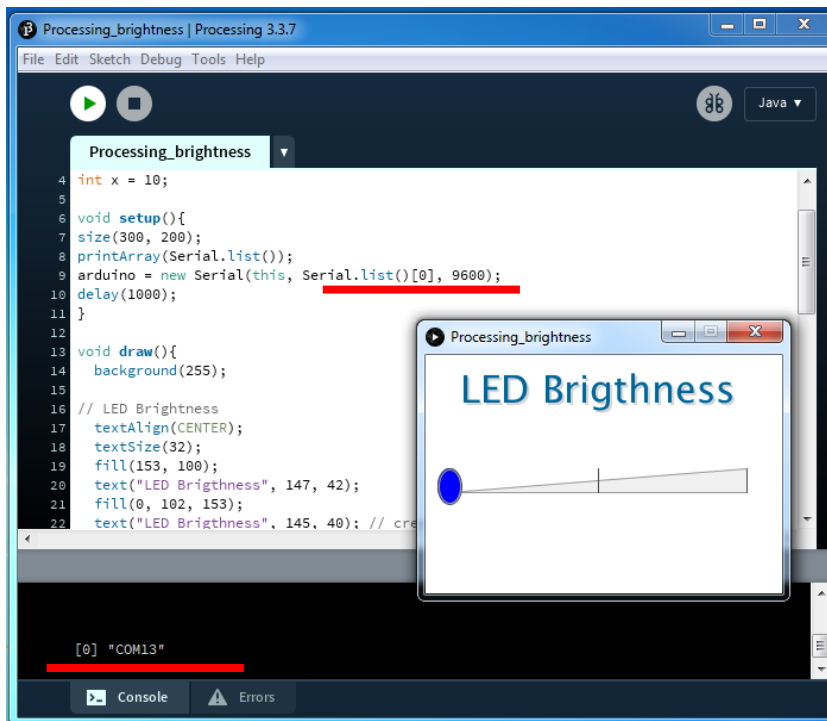


Рисунок 1.7 – Перелік доступних послідовних портів

1.4 Контрольні запитання

1.4.1 Чим подібні та чим відрізняються середовища Processing та Arduino IDE.

1.4.2 Як організувати взаємодію платформи Arduino та середовища Processing?

1.4.3 Які базові функції та змінні надає середовище Processing?

1.5 Зміст звіту

1.5.1 Мета лабораторної роботи.

1.5.2 Короткі теоретичні відомості.

1.5.3 Знімки екрану, які відображають виконані завдання.

1.5.4 Висновки до роботи.

2 ЛАБОРАТОРНА РОБОТА 2. ПІДКЛЮЧЕННЯ ДАТЧИКІВ ДО ARDUINO

Мета роботи: оволодіти основними принципами роботи датчиків та підключення датчиків до Arduino.

2.1 Теоретичні відомості

2.1.1 Датчики та їх використання

Датчик води

У якості датчика води можливо використовувати аналоговий датчик рівня рідини (рис. 2.1). Робоча напруга аналогового сенсора – 5v. Вихідна напруга (показання датчика) залежить від ступеня занурення датчика в рідину і від параметрів, що впливають на коефіцієнт передачі напруги, наприклад, провідність рідини.

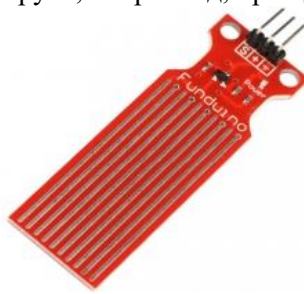


Рисунок 2.1 - Аналоговий датчик рівня рідини

Датчик освітлення

У якості датчика освітлення може бути обрано модуль датчика світла (рис. 2.2) з пороговим компаратором. Поріг спрацьовування компаратора регулюється змінним резистором. Нижче перелічені деякі його характеристики:

- чутливий елемент - фоторезистор;
- вихід компаратора більш ніж 15 мА;
- регулювання порога спрацьовування змінним резистором;
- робоча напруга: від 3.3В до 5В;
- цифровий вихід компаратора (0 і 1).



Рисунок 2.2- Модуль датчика світла

Датчик руху

PIR (пасивні інфрачервоні датчики) сенсори дозволяють вловлювати рух (рис. 2.3). Дуже часто використовуються в системах сигналізації. Ці датчики малі за габаритами, недорогі, споживають мало енергії, легкі в експлуатації, практично не схильні до зносу.

Основні технічні характеристики:

- зона роботи датчика: до 6 метрів;
- робоча напруга: 5 - 9В.



Рисунок 2.3 - Датчик руху

Датчик газу

Датчик газу, побудований на базі газоаналізатора MQ-2 (рис. 2.4) дозволяє виявляти наявність в навколишньому повітрі вуглеводневих газів (пропан, метан, н-бутан), диму (зважені частинки, які є результатом горіння), водню.

Датчик має 4 контакти (живлення, земля, цифровий висновок, аналоговий вивід), але також датчик може мати 3 контакти (живлення, земля, аналоговий вивід).



Рисунок 2.4 - Датчик газу

Датчик температури та вологості

Для вимірювання температури та вологості може бути обрано датчик DHT11 (рис. 2.5). Датчик DHT11 - це цифровий датчик температури і вологості, що дозволяє калібрувати цифровий сигнал на виході. Складається з ємнісного датчика вологості і термістора. Також, датчик містить в собі АЦП для перетворення аналогових значень вологості і температури. Характеристики:

- робоча напруга: від 3.3В до 5В;
- визначення вологості 20-95% з 5% точністю;
- визначення температури 0-50 град. з точністю 2 град;
- частота опитування не більше 1 Гц (не більше одного разу за 1 сек.).



Рисунок 2.5 - Датчик температури та вологості DHT11

Датчик звуку

Складається датчик звуку (рис. 2.6) з плати на якому змонтовані виходи, підсилювач звуку, підлаштовний резистор і електронний мікрофон, чутливий до звуку, що приходить у всіх напрямках.

Регулятором чутливості (змінним резистором) можна вибирати від якого звуку буде спрацьовувати датчик.

Плата розширення для Arduino дозволяє перевести звукові коливання в цифровий сигнал. При коливанні мембрани в мікрофоні від звукових хвиль, змінюється ємність його конденсатора, внаслідок чого виявляється зміна напруги на виходах датчика звуку, відповідне звуковим сигналом.



Рисунок 2.6 - Датчик звуку

Датчик відтінку кольору

Датчик відтінку кольору TCS230 (рис. 2.7) для Ардуіно здатний розпізнавати 4 кольори і перетворює інтенсивність колірного спектра в вихідний сигнал різної частоти.

У колірній палітрі RGB, будь-який колір можна представити у вигляді комбінації трьох основних кольорів: червоний (R), зелений (G) і синій (B). Тому для визначення кольору необхідно вимірювати червоний, синій і зелений спектр. Як чутливий елемент в датчику використовується мікросхема TCS230, що складається з масиву фотодіодів 8×8 - по 16 фотодіодів на три кольори і 16 фотодіодів без фільтра.

Технічні характеристики:

- напруга живлення від 2,7 до 5,5 В;
- програмовані кольору і вихідна частота сигналу;
- функція автоматичного відключення живлення;
- мала похибка вихідної частоти - 0,2%.

Датчик використовується для визначення відтінку кольору об'єкта на відстані до 10 мм, для підсвічування місця вимірювання на датчику є чотири світлодіоди. Зі зворотного боку датчика є дві

колодки з чотирма контактами. Через ці контакти датчик TCS230 підключається до мікроконтролера Arduino. Контакти «S0» і «S1» служать для масштабування частоти імпульсів на виході «OUT» датчика відтінку кольору.



Рисунок 2.7 - Датчик відтінку кольору

Ультразвуковий датчик

Ультразвуковий датчик визначає відстань до об'єкта так само, як це роблять кажани або дельфіни. Датчик HC-SR04 (рис. 2.8) генерує вузько спрямований сигнал на частоті 40 кГц і ловить відбитий сигнал (відлуння). За часом поширення звуку до об'єкта і назад можна досить точно визначити відстань до нього.

Ультразвукової далекомір HC-SR04 має діапазон виміру від 2 см до 400 см, працює при температурах від 0 ° до 60 ° С. Точність вимірювання складає ± 1 см, робоча напруга датчика до 5,5 В.



Рисунок 2.8 - Ультразвуковий датчик

Датчик вібрації

Датчик вібрації Arduino (рис. 2.9) використовується для визначення наявності зовнішніх вібраційних впливів на модуль і застосовується при створенні домашньої сигналізації.

Даний сенсор може випускатися у вигляді модуля, наприклад, SW-420, logo sensors v1.5 або сенсора вібрації 140с001 для Ардуіно. Основа датчика SW-420 - металева пружина всередині пластикової трубки, яка починає коливатися при вібрації. На модулі є підсилювач сигналу, підлаштовний резистор, для регулювання чутливості датчика і три висновки для підключення до мікроконтролеру.

Усередині сенсора 801S знаходиться металева кулька на пружинці, який реагує на найменші коливання. Принцип роботи даних датчиків можна порівняти з тактовою кнопкою - якщо контакти на кнопці замикаються, коли ми на неї натискаємо, то контакти в розглянутому нами датчику замикаються при виникненні вібрації.



Рисунок 2.9 - Датчик вібрації

Датчик вологості ґрунту

Датчик вологості ґрунту (рис. 2.10) з дискретним і аналоговим виходом. Складається з чутливого елемента і плати перетворювача. При пересиханні ґрунту (рівень пересихання задається підлаштовним резистором на платі перетворювача) на дискретний вихід подається логічна одиниця. З аналогового виходу можна отримувати точну величину, що характеризує вологість ґрунту, в межах від нуля до величини напруги живлення.

Технічні характеристики:

–розміри чутливого елемента 60 x 30 мм;

- розміри перетворювача 30 x 15 мм;
- напруга живлення 3,3 ... 5 В;
- довжина проводу 210 мм.



Рисунок 2.10 - Датчик вологості ґрунту

2.1.2 Підключення датчиків до Arduino

Для того щоб підключити датчики до Arduino потрібно зібрати їх за схемою, що представлена на рисунку 2.11.

Зазвичай на датчиках надруковано, які контакти потрібно з'єднувати це:

- VCC – живлення, зазвичай 5V.
- OUT (або щось інше) – аналоговий вивід (підключення до порта).
- GND – земля.

Якщо є 4 контакти підключайте 3 (живлення, земля, аналоговий вивід).

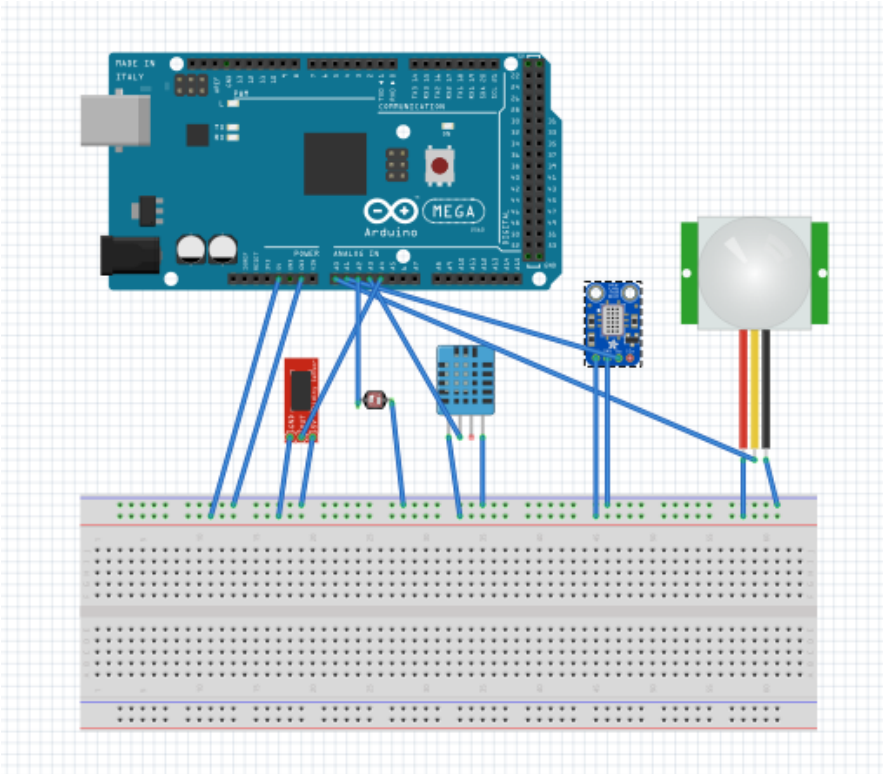


Рисунок 2.11 - Загальна схема підключення датчиків

Задля коректної роботи програми та отримання даних про температуру та вологість із датчику DHT11 потрібно підключити бібліотеку DHT.h та Adafruit_Sensor.h. Для цього перейдіть за посиланнями (https://github.com/adafruit/Adafruit_Sensor та <https://github.com/adafruit/DHT-sensor-library>), завантажте бібліотеки та скопіюйте разархівовані файли у директорію libraries свого Arduino IDE.

2.1.3 Текст програми

Для коректної роботи програми потрібно аналоговий виводи датчиків підключити до тих самих портів:

- Датчик світла до 5.
- Датчик рівня рідини до A1.

- Датчик газу до A0.
- Датчик руху до 1.
- Датчик вологості до 2.

```
#include <DHT.h>
#include <DHT_U.h>
#include <Adafruit_Sensor.h>
#define DHTPIN 3 // зберігає номер порту датчику
температури та вологості
DHT dht(DHTPIN, DHT11); // об'єкт для роботи із
датчиком DHT11
const int MQ2 = A0; // зберігає номер порту датчику
газу
const int ledPin = 13; // зберігає номер порту діоду
int sensorValue = 0; //зберігає значення зчитане із
датчику газу
int x; // змінна для датчика потопу
int inputPin = 2; // зберігає номер порту датчику
руху
int pirState = LOW; // змінна для датчика руху
int val = 0; // змінна для датчика руху
int buttonState = 0; // змінна для датчика світла

void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT); // Задає режим для 13 пина
  dht.begin();
  pinMode(A1, INPUT); // water_level
  pinMode(inputPin, INPUT); // declare sensor as
input
  pinMode(buttonPin, INPUT);
}

void loop() {
  Serial.print(dht.readHumidity());
  Serial.print(",");
  Serial.print(dht.readTemperature());
  Serial.print(",");
  sensorValue = analogRead(MQ2);
  Serial.print(sensorValue);
  Serial.print(",");
  x = analogRead(A1); if (x > 100) {
```

```

digitalWrite(13, HIGH); }
if (x < 100) { digitalWrite(13, LOW); }
Serial.print(x);
Serial.print(",");
val = digitalRead(inputPin);
    if (val == HIGH)
    {
        Serial.print("M_detected,");
        pirState = HIGH;}
    else {
        Serial.print("No_motion,");
        pirState = LOW;}
buttonState = digitalRead(buttonPin);
if (buttonState == HIGH)
{digitalWrite(ledPin, HIGH);
Serial.print("Light_off,");}
else
{digitalWrite(ledPin, LOW);
Serial.print("Light_on");}
Serial.println("");
}

```

2.1.4 Основні функції

Після запуску програми починається ініціалізація змінних та портів вводу-виводу. Потім програма переходить в головний цикл у якому вона проводить відповідних дій. У таблиці 2.1 наведено опис функцій програми.

Таблиця 2.1 – Опис функцій програми

Назва функції	Опис функції
Serial.begin(9600)	Відкриває послідовний порт, встановлює швидкість 9600 біт/с
pinMode(ledPin, OUTPUT)	Встановлює режим роботи - вихід
dht.begin()	Запускаємо датчик вологості та температури за допомогою бібліотеки DHT.h
pinMode(A1, INPUT)	Встановлює режим роботи – вхід
Serial.print(",")	Виводить на монітор порту кому

Продовження таблиці 2.1

dht.readHumidity()	Функція зчитує вологість
dht.readTemperature()	Функція зчитує температуру
analogRead(MQ2)	Функція зчитує значення з датчика газу
digitalWrite(13, HIGH)	Включає світлодіод
digitalWrite(13, LOW)	Вимикає світлодіод

2.2. Завдання до виконання лабораторної роботи

2.2.1. Ознайомитися з теоретичними відомостями.

2.2.2 Завантажити та встановити бібліотеки.

2.2.3 Зібрати датчики за схемою.

2.2.4 Написати програму для роботи з датчиками, що виконує такі функції:

- отримання даних про температуру та вологість із датчику DHT11;
- отримання та відправка показань із датчику рівня рідини;
- отримання та відправка значень про рівень освітлення;
- отримання та відправка значень про рівень газу;
- отримання та відправка значень про появу руху.

1.2.5 Оформити звіт до лабораторної роботи.

2.3 Контрольні запитання

2.3.1 Назвіть основні датчики для Arduino?

2.3.2 Де використовується датчик руху?

2.3.4 Які бібліотеки потрібно завантажити для роботи датчику DHT11?

2.3.3 Для чого використовується функція Serial.begin(9600)?

2.3.4 Для чого використовується функція dht.begin()?

2.4 Зміст звіту

2.4.1 Мета лабораторної роботи.

2.4.2 Короткі теоретичні відомості.

2.4.3 Код програми.

2.4.4 Знімки екрану, які відображають виконане завдання.

2.4.5 Висновки до роботи.

3 ЛАБОРАТОРНА РОБОТА 3. НАЛАШТУВАННЯ RASPBERRY PI ТА ПІДКЛЮЧЕННЯ ДО ARDUINO

Мета роботи: навчитись встановлювати операційну систему Raspbian на Raspberry Pi. Та підключати Raspberry Pi до Arduino.

3.1 Теоретичні відомості

Raspberry Pi - одноплатний комп'ютер розміром з банківську карту, спочатку розроблений як бюджетна система для навчання інформатиці, що згодом одержав набагато більш широке застосування і популярність. Перші версії були створені в 2011 році. Побудований на основі процесора архітектури ARM.

Raspberry Pi працює в основному на операційних системах, заснованих на Linux ядрі, таких як:

- Raspbian (модифікація Debian);
- Pidora (модифікація Fedora);
- Arch Linux ARM;
- Kali Linux.

На рисунку 3.1 зображено компоненти типові для моделі Raspberry Pi 2B та їх розташування на платі.

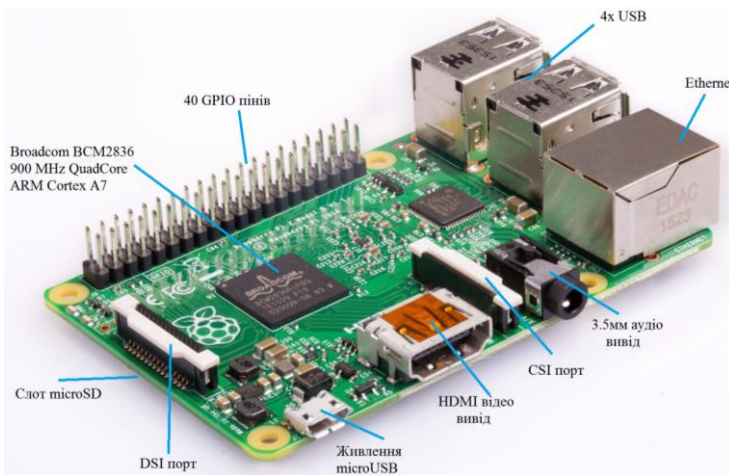


Рисунок 3.1 – Компоненти Raspberry Pi 2B

3.1.1 Встановлення ОС Raspbian

Існує два способи встановлення ОС Raspbian на Raspberry Pi:

- 1) завантаження пакета NOOBS з офіційного сайту на карту пам'яті MicroSD та подальше встановлення;
- 2) монтування файлу-образу ОС Raspbian на карту пам'яті MicroSD. В цьому випадку можна буде приступати до роботи одразу після включення Raspberry Pi.

3.1.2 Приклад встановлення ОС Raspbian за допомогою пакета NOOBS

Необхідно завантажити пакет з офіційного сайту <https://www.raspberrypi.org/downloads/noobs/>. Слід обрати NOOBS «Offline and network install» для встановлення без мережі Інтернет чи NOOBS LITE «Network install only» для встановлення з мережею Інтернет (рис.3.2).

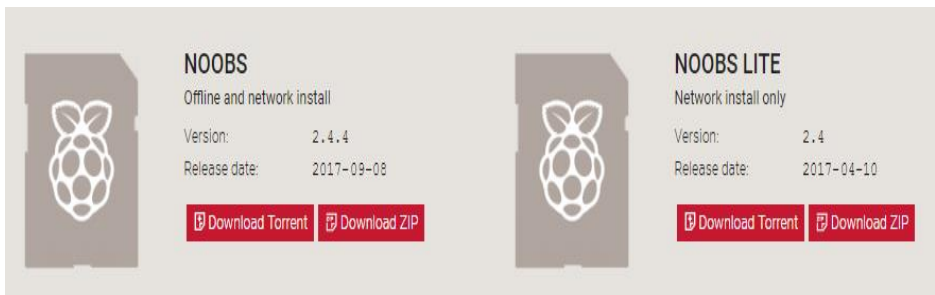


Рисунок 3.2 – Вибір пакета NOOBS

Перед тим як записувати файли NOOBS на карту пам'яті microSD, її слід відформатувати у файлову систему FAT32.

Далі слід розпакувати архів NOOBS на карту пам'яті microSD.

Наступний крок – підключення периферії до Raspberry Pi: USB-клавіатури, USB-миші, монітору (через HDMI), а також необхідно встановити карту пам'яті microSD у відповідний слот. В останню чергу підключається живлення MicroUSB.

В окні що відобразиться слід обрати ОС Raspbian (перша опція). Друга опція пропонує розбивку розділів карти пам'яті вручну,

третя опція – завантаження безпосередньо у пакет Scratch – програму для створення комп'ютерної графіки та анімацій.

Необхідно натиснути «Install» та підтвердити запис даних на карту пам'яті MicroSD. Слід дочекатися завершення процесу встановлення та перезавантаження. У вікні Configuration Tool (налаштування можна змінити пізніше) слід обрати опцію «Enable boot to Desktop» та підтвердити вибір, що дозволить зробити LXDE інтерфейсом за замовчуванням при завантаженні. Далі слід натиснути «Done» та погодитися на перезавантаження. Після чого все буде готово до роботи.

3.1.3 Приклад встановлення ОС Raspbian за допомогою монтування файлу-образу

Спочатку необхідно завантажити ОС з офіційного сайту: <https://www.raspberrypi.org/downloads/raspbian/>. Рекомендовано обрати «Raspbian Stretch With Desktop» (рисуюнок 3.3).



Рисуюнок 3.3 – Завантаження файлу-образу

Також необхідно встановити на комп'ютер утиліту Win32DiskImager (<https://sourceforge.net/projects/win32diskimager/>) та за її допомогою записати файл-образ ОС Raspbian на карту пам'яті microSD:

- необхідно розпакувати архів з файлом-образом на комп'ютері;
- слід відформатувати карту пам'яті стандартним засобом ОС Windows;

– далі необхідно у програмі Win32DiskImager обрати файл-образ ОС Raspbian (наприклад: «2017-09-07-raspbian-stretch.img»);
 – у розділі «Device» слід обрати ім'я карти пам'яті microSD. Слід бути уважним, якщо помилитись можна нанести шкоду жорсткому диску (рисунок 3.4);

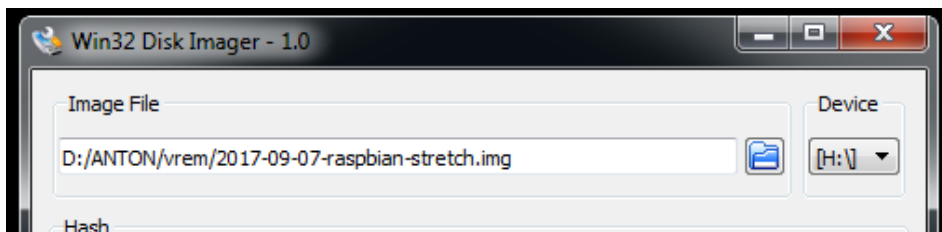


Рисунок 3.4 – Обрано файл-образу та пристрій для запису

– нарешті слід обрати «Write» та дочекатися завершення процесу запису. Приблизний час запису становить 5–10 хвилин.

Далі необхідно здійснити підключення периферії до Raspberry Pi: USB-клавіатури, USB-миші, монітору (через HDMI), а також необхідно встановити карту пам'яті microSD у відповідний слот. В останню чергу підключається живлення MicroUSB.

Відбудеться завантаження ОС Raspbian та все буде готово до роботи.

Якщо використовується Unix-подібні ОС:

- слід відформатувати карту пам'яті microSD у FAT 32/ext2;
- виконати команду: `sudo dd bs=4m if=/home/user/raspbian.img of=/dev/disk1;`

Далі необхідно здійснити підключення периферії до Raspberry Pi: USB-клавіатури, USB-миші, монітору (через HDMI), а також необхідно встановити карту пам'яті microSD у відповідний слот. В останню чергу підключається живлення MicroUSB.

Відбудеться завантаження ОС Raspbian та все буде готово до роботи.

Важливо запам'ятати, що запуск Raspberry Pi відбувається одразу після підключення MicroUSB, та що вимикати Raspberry Pi слід наступним чином:

- 1) завершити всі дії;
- 2) натиснути на лого Raspberry Pi у верхньому лівому куті;

3) обрати опцію «Shutdown» (рис. 3.5) та у новому окні підтвердити – «Shutdown» (рис. 3.6).

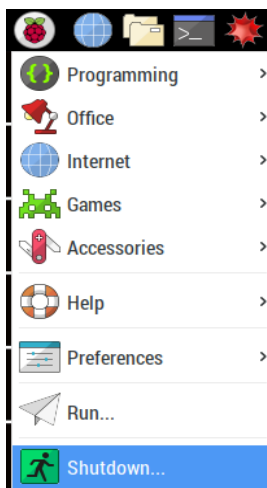


Рисунок 3.5 – Вимкнення Raspberry Pi

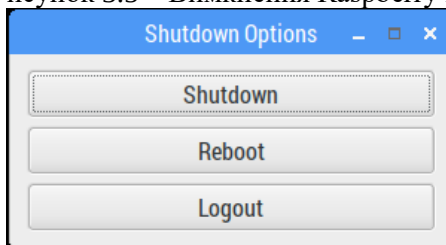


Рисунок 3.6 – Меню вимкнення Raspberry Pi

Тільки після того як Raspberry Pi буде виключено, можна від'єднати шнур MicroUSB та змінювати склад периферії, підключеної до Raspberry Pi.

3.1.4 Перший запуск ОС Raspbian

Якщо було ввімкнено завантаження у графічне середовище, введення паролю користувача не потрібне, але якщо пароль було змінено, може знадобитися його введення.

У випадку консольного режиму необхідно ввести логін та пароль («pi» та «raspberrypi», відповідно). Щоб запустити графічне середовище, необхідно виконати команду «startx».

У верхній частині екрану знаходиться панель задач, панель швидкого доступу та кнопка меню (рисунок 3.7).

На панелі швидкого запуску знаходяться:

- браузер;
- Провідник;
- утиліта Terminal;
- програми пакету Wolfram.

Натиснувши на кнопку меню (лого Raspberry Pi) можна переглянути програми встановлені в системі (рисунок 3.8).

В меню містяться такі розділи:

- Programming: містить програми та засоби для програмування (Python IDE, Geany, BlueJ, тощо);
- Office: містить «офісні» програми (пакет OpenOffice);
- Internet: містить програми та засоби для роботи в мережі Інтернет;
- Games: містить ігри (MineCraft);
- Accessories: містить додаткові супутні програми (Archiver, калькулятор);
- Sound & Video: містить відео- та аудіо-плеєри (VLC Player);
- System Tools: містить системні програми (Midnight Commander);
- Help: довідка;
- Preferences: налаштування системи;
- Run: виконати дію;
- Shutdown: перехід до вікна вимкнення плати.



Рисунок 3.7 – Панель швидкий запуску

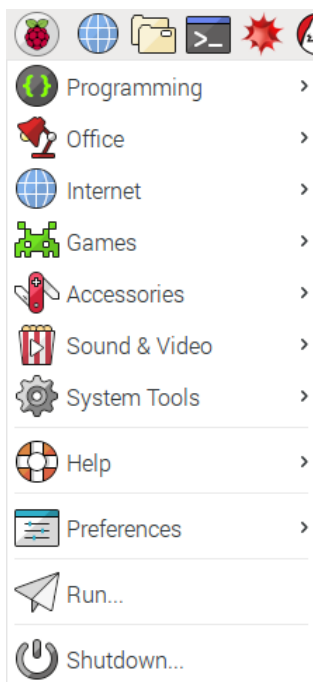


Рисунок 3.8 – Меню Raspberry Pi

В основному, для виконання більшості дій у Raspbian (як у будь-якій Unix-подібній ОС) використовується утиліта Terminal (рис. 3.9). Для її запуску слід натиснути відповідну піктограму (рис. 3.10).

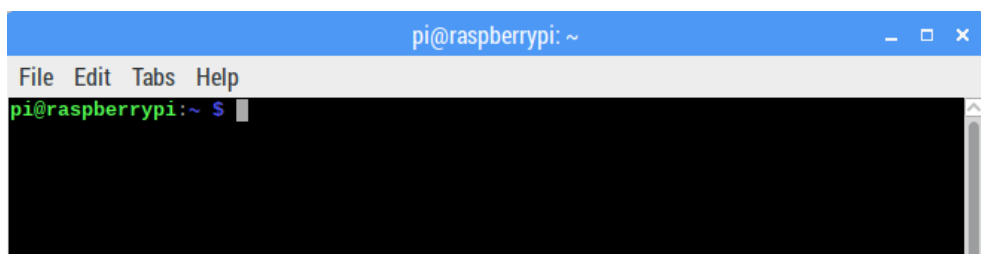


Рисунок 3.9 – Вікно Terminal



Рисунок 3.10 – Піктограма Terminal

В процесі роботи може знадобитися змінити конфігурацію системи, змінити пароль користувача, увімкнути камеру, тощо. Все це можна зробити за допомогою команди `sudo raspi-config` (рисунок 3.10). Приклад вікна налаштувань наведено на рисунку 3.11.

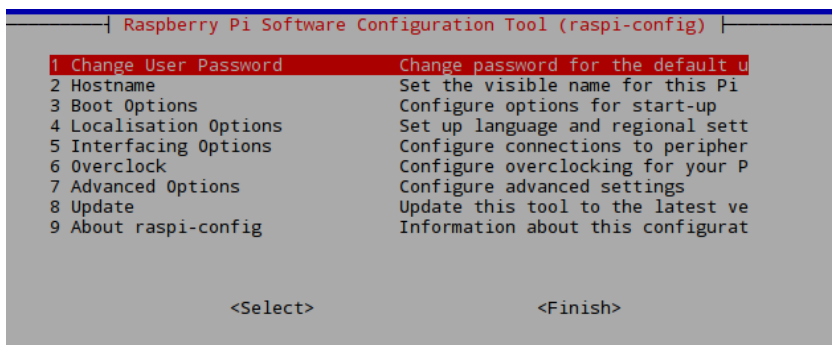
Рисунок 3.10 – Виконання команди `sudo raspi-config`

Рисунок 3.11 – Вікно налаштувань

Докладніше про пункти меню:

- 1) Change User Password – зміна пароля користувача;
- 2) Hostname – ім'я комп'ютера в мережі;
- 3) Boot options – налаштування завантаження системи;

4) Localisation Options – вибір мови та регіональних налаштувань;

5) Interfacing Options – налаштування інтерфейсів (камера, SSH, VNC, SPI, I2C, Serial, 1-Wire, GPIO);

6) Overclock – збільшення частоти процесору;

7) Advanced Options – додаткові параметри;

8) Update – оновлення програми `raspi-config`;

9) About `raspi-config` – інформація про програму.

Завершивши налаштування, слід обрати `Finish`. ОС запросить дозвіл на перезавантаження. Слід погодитися та дочекатися закінчення перезавантаження.

При зміні налаштувань системи слід бути уважним, помилка може призвести до небажаних наслідків.

3.1.5 Налаштування мережі Інтернет

У випадку підключення до мережі, в якій налаштовано DHCP-сервер та автоматичний розподіл адрес (наприклад, домашня мережа), додатково нічого не потрібно робити.

Якщо DHCP відсутній, параметри мережі можна задати за допомогою консолі. Для цього необхідно запустити термінал, де слід відкрити файл налаштувань мережевих інтерфейсів за допомогою команди: `sudo nano /etc/network/interfaces`. До файлу треба додати наступне:

```
iface <інтерфейс> inet static
address <ip-адреса>
netmask <маска підмережі>
gateway <адреса шлюзу>
dns-nameservers <адреси DNS-серверів, розділенні пробілом>
auto <інтерфейс>,
```

де інтерфейс – мережевий інтерфейс, підключений до Raspberry Pi (найчастіше це `eth0`);

ip-адреса – адреса, яку необхідно присвоїти;

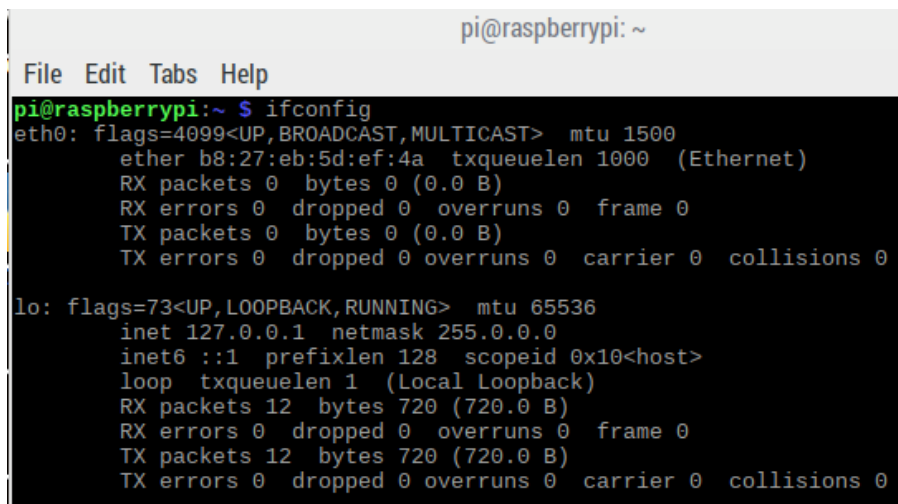
маска підмережі – маска;

адреса шлюзу – ip-адреса комп'ютеру, що служить в мережі шлюзом.

Далі необхідно вийти зі збереженням змін, для чого слід натиснути `Ctrl+X`, підтвердити збереження натисненням клавіші `Y`, а

потім Enter для підтвердження перезапису файлу. Після змін файлу треба перезавантажити Raspberry Pi, виконавши команду `sudo reboot`.

Для перевірки налаштувань мережі слід виконати команду `ifconfig` (рис. 3.12).



```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether b8:27:eb:5d:ef:4a txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Local Loopback)
    RX packets 12 bytes 720 (720.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 12 bytes 720 (720.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
  
```

Рисунок 3.12 – Приклад виконання команди `ifconfig`

Для роботи з проксі-сервером необхідно певні строки до файлів `/etc/environment` (загальносистемні налаштування проксі) та `/etc/apt/apt.conf` (налаштування пакетного менеджера `apt-get`).

До файлу `/etc/environment` слід дописати:

`export http_proxy=«http://host:port»`, а до файлу `/etc/apt/apt.conf`: `Acquire::http::proxy «http://host:port»; .`

Після виконання даних дій, можна буде користуватися мережевими функціями, в тому числі і менеджером `apt-get`.

3.1.6 Підключення Arduino із завантаженням скетчем

Для того щоб підключити Arduino до Raspberry Pi потрібно:

1. Взяти скетч (програму) з лабораторної роботи №1 та підключеними датчиками до Arduino. Запустити скетч та залишити працювати.
2. Витягнути Arduino USB-дріт з комп'ютера.
3. Вставити Arduino USB-дріт у працюючий Raspberry Pi (рис. 2.13)
4. Надати повний доступ до порту, що буде використовуватися Arduino. Для цього у терміналі написати:
`chmod 777 /dev/ttyACM0`

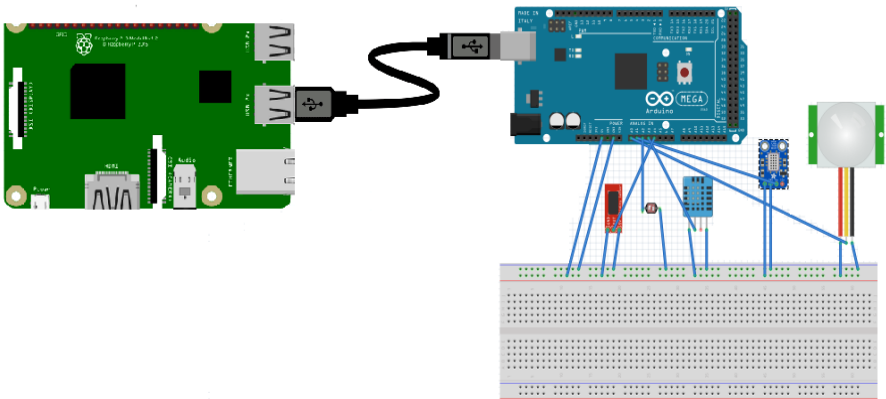


Рисунок 3.13 – Схема підключення Arduino до Raspberry Pi

3.2 Завдання до виконання лабораторної роботи

- 3.2.1 Ознайомитися з теоретичними відомостями.
- 3.2.2 Виконати установку і базове налаштування Raspbian на Raspberry Pi.
- 3.2.3 Налаштувати підключення до мережі Інтернет на Raspberry Pi.
- 3.2.4 Підключити Arduino до Raspberry Pi.
- 3.2.5 Оформити звіт до лабораторної роботи.

3.3 Контрольні запитання

- 3.3.1 Що таке Raspberry Pi?

3.3.2 На яких ОС працює Raspberry Pi? Які компоненти типові для Raspberry Pi 2B?

3.3.3 Які існують способи встановлення ОС Raspbian на Raspberry Pi?

3.3.4 Як підключити Arduino до Raspberry Pi?

3.4 Зміст звіту

3.4.1 Мета лабораторної роботи.

3.4.2 Короткі теоретичні відомості.

3.4.3 Знімки екрану, які відображають виконані завдання.

3.4.4 Висновки до роботи.

4 ЛАБОРАТОРНА РОБОТА 4. ВСТАНОВЛЕННЯ ТА НАЛАШТУВАННЯ OPENHAB НА RASPBERRY PI

Мета роботи: навчитися встановлювати та налаштувати OpenHAB на Raspberry Pi.

4.1 Теоретичні відомості

OpenHAB (Open Home Automation Bus) – створений на Java open source проект з реалізації технології «smart house», розповсюджується під ліцензією GPLv3, та для організації роботи web-інтерфейсу використовується Jetty.

Ця відкрита платформа здатна виконувати такі функції як вмикання та вимикання освітлення, керування розетками, тощо.

OpenHAB надає засоби для організації роботи різних систем, обладнання та інтерфейсів домашньої автоматизації. При цьому OpenHAB не залежить від протоколів і обладнання, надаючи окремий рівень абстракції, що дозволяє взаємодіяти з різними типами пристроїв і програмного забезпечення. Для визначення керуючої логіки використовуються скрипти, написані на спеціальній предметно-орієнтованій мові програмування, розробленій за допомогою Eclipse –Xtext.

Основним сервісом OpenHAB є шина подій. Модулі, які не вимагають відстеження стану, використовують цю шину для обміну інформацією про події з іншими модулями.

Існує два основних типи подій:

1. Команди, які ініціюють будь-яку дію чи зміна стану певного елемента або пристрою.
2. Оновлення статусу, які повідомляють про зміну стану певного елемента або пристрою.

Прив'язки протоколів, що забезпечують зв'язок з реальними пристроями, повинні спілкуватися між собою саме через шину подій. Це гарантує існування зв'язку між модулями.

В openHAB використовується дуже потужна мова виразів, за допомогою якої створюються визначення сценаріїв. Сценарій або скрипт – це блок коду, який визначається користувачем і може викликатися і використовуватися в різних місцях. Скрипти

розміщуються в папці `openhabs/configurations/scripts`. В робочому середовищі є демонстраційний файл `demo.script`.

Скрипти також можуть знаходитися всередині файлу правил, в папці `openhabs/configurations/rules`: вони використовуються для визначення блоку виконання правила. Кожне правило складається з двох частин: в одній містяться перемикачі дій, в інших скриптах – для їх виконання.

Для ефективного використання скриптів OpenHAB надає доступ:

- до всіх елементів (отримати доступ можна звернувшись по імені);

- до всіх статусів та команд;

- до стандартних дій до виконання різних операцій.

Скрипт ідентифікується назвою (у файлі `demo.script` – ім'я скрипта `demo`). Кожен скрипт завжди повертає значення, що є результатом останнього виразу, який міститься в ньому.

Скрипти можуть викликатися:

- з правил;

- з XMPP-консолі;

- з записів в календарі Google.

4.1.1 Встановлення та налаштування OpenHAB на Raspberry Pi

Для встановлення OpenHAB на Raspberry Pi слід виконати послідовність дій та команд (за допомогою терміналу), що описані нижче.

1) Завантажити ключ:

```
wget -qO - 'https://bintray.com/user/downloadSubjectPublicKey?username=openhab' | sudo apt-key add -
```

2) Встановити репозиторій OpenHAB:

```
echo "deb http://dl.bintray.com/openhab/apt-repo stable main" | sudo tee /etc/apt/sources.list.d/openhab.list
```

3) Оновити системні файли:

```
sudo apt-get update
```

4) Встановити OpenHAB:

```
sudo apt-get install openhab-runtime
```

5) Додати користувача до групи:

sudo usermod -aG openhab pi

6) Запустити OpenHAB:

sudo systemctl enable openhab

7) Встановити доповнення для роботи з GPIO:

sudo apt-get install openhab-addon-io-gpio

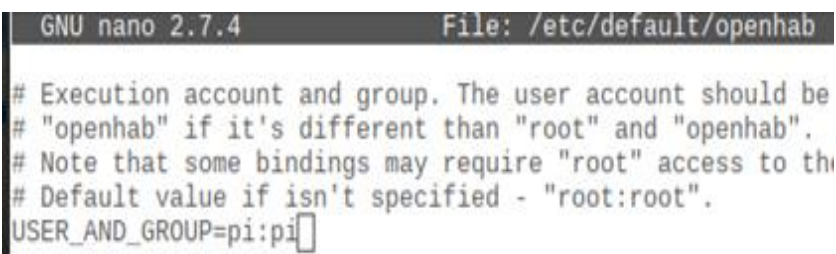
sudo apt-get install openhab-addon-binding-gpio

8) Встановити доповнення для роботи з портами:

sudo apt-get install openhab-addons-binding-serial

9) Задати ім'я користувача та назву групи:

sudo nano /etc/default/openhab



```
GNU nano 2.7.4 File: /etc/default/openhab
# Execution account and group. The user account should be
# "openhab" if it's different than "root" and "openhab".
# Note that some bindings may require "root" access to the
# Default value if isn't specified - "root:root".
USER_AND_GROUP=pi:pi
```

Рисунок 4.1 – Зміна імені користувача і групи

sudo nano /usr/lib/systemd/system/openhab.service



```
[Service]
EnvironmentFile=/etc/default/openhab
User=pi
Group=pi
```

Рисунок 4.2 - Зміна імені користувача і групи

10) Перезавантажити systemd:

sudo systemctl daemon-reload

11) Додати порт який використовуватиметься з Arduino:

sudo nano /usr/share/openhab/bin/openhab.sh

В розділі «JAVA_ARGS_DEFAULT» додати в кінець:

```
-Dgnu.io.rxtx.SerialPorts=/dev/ttyACM0"
```

Рисунок 4.3 – Додавання порту

12) Перевантажити OpenHAB:

```
sudo service openhab restart
```

13) Написати код для отримання даних від Arduino:

```
sudo nano /etc/openhab/configurations/items/home.items
```

```
String Arduino "Arduino [%s]" {serial="/dev/ttyACM0"}

Number Hum "hum: [%.2f]"
Number Temp "temp: [%.2f]"

Number Water "water: [%f]"

String Motion "motion: [%s]"
String Lights "lights: [%s]"
```

Рисунок 4.4 – Код отримання даних від Arduino

14) Створити інтерфейс для відображення даних:

```
sudo nano /etc/openhab/configurations/sitemaps/home.sitemap
```

```
sitemap home label="home"
{
    Frame label="RPi + Arduino"
    {
        Frame
        {
            Text item=Hum
            Text item=Temp

            Text item=Water
            Text item=Motion
            Text item=Lights
        }
    }
}
```

Рисунок 4.5 – Створення інтерфейсу для відображення даних

15) Перевантажити OpenHAB:

`sudo service openhab restart`

16) Знайти процес Java та запам'ятати його номер:

`top`

```
pi@raspberrypi:~$ top
```

top - 12:49:29 up 1:58, 2 users, load average: 1.09, 0.53, 0.29
 Tasks: 146 total, 1 running, 145 sleeping, 0 stopped, 0 zombie
 %Cpu(s): 0.9 us, 0.4 sy, 0.0 ni, 98.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
 KiB Mem : 949576 total, 37712 free, 263564 used, 648300 buff/cache
 KiB Swap: 102396 total, 102396 free, 0 used. 599936 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3349	pi	20	0	8100	3244	2780	R	2.0	0.3	0:00.24	top
3234	pi	20	0	365756	91912	12496	S	1.3	9.7	0:46.82	java

Рисунок 4.6 – Відображення процесів

17) Зупинити процес Java. Наприклад, номер Java процесу – 3234, тоді:

`kill 3234`

18) Перейменувати файл `openhab_default.cfg` у `openhab.cfg` в каталозі `/etc/openhab/configurations/`

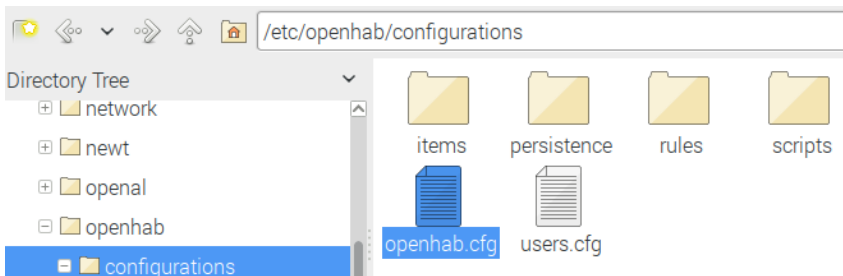


Рисунок 4.7 – Файл перейменовано

20) Створити правило OpenHAB для керування датчиками:

`sudo nano etc/openhab/configurations/rules/home.rules`

```

import java.io.*
rule "Arduino"
when
    Item Arduino received update
then
    var String ArdUpd=Arduino.state.toString().split("\n")

    for (String s:ArdUpd){
        var String[] ar=s.split(",")

        var Double h = new Double(ar.get(0))
        var Double t=new Double(ar.get(1))

        var Double w=new Integer(ar.get(2))

        var String m=new String(ar.get(3))
        var String l=new String(ar.get(4))

        Hum.postUpdate(h)
        Temp.postUpdate(t)

        Water.postUpdate(w)
        Motion.postUpdate(m)
        Lights.postUpdate(l)
    }
end

```

Рисунок 4.8 – Код правила

21) Підключити Arduino із завантаженим скетчем;

22) Надати повний доступ до порту, що буде використовуватися

Arduino:

chmod 777 /dev/ttyACM0

23) Перейти до каталогу openHab:

cd /usr/share/openhab/bin

24) Запустити файл .sh:

sudo ./openhab.sh

24) Перейти за адресою: RaspberryPi-

IP:8080/openhab.app?sitemap=home



Рисунок 4.9 – Сторінка OpenHAB

```

dated to 25.0
2017-11-25 16:12:48.658 [INFO ] [runtime.busevents      ] - Water state u
pdated to 270
2017-11-25 16:12:48.660 [INFO ] [runtime.busevents      ] - Motion state
updated to No_motion
2017-11-25 16:12:48.663 [INFO ] [runtime.busevents      ] - Lights state
updated to Light_on
2017-11-25 16:12:48.687 [INFO ] [runtime.busevents      ] - Hum state upd
ated to 34.0
2017-11-25 16:12:48.689 [INFO ] [runtime.busevents      ] - Temp state up
dated to 25.0
2017-11-25 16:12:48.691 [INFO ] [runtime.busevents      ] - Water state u
pdated to 266
2017-11-25 16:12:48.693 [INFO ] [runtime.busevents      ] - Motion state
updated to No_motion
2017-11-25 16:12:48.697 [INFO ] [runtime.busevents      ] - Lights state
updated to Light_on
2017-11-25 16:12:48.720 [INFO ] [runtime.busevents      ] - Hum state upd
ated to 34.0
2017-11-25 16:12:48.722 [INFO ] [runtime.busevents      ] - Temp state up
dated to 25.0
2017-11-25 16:12:48.724 [INFO ] [runtime.busevents      ] - Water state u
pdated to 0

```

Рисунок 4.10 – Вивід в консолі

4.2 Завдання до виконання лабораторної роботи

4.2.1 Ознайомитися з теоретичними відомостями.

4.2.2 Виконати встановлювання та налаштування OpenHAB на Raspberry Pi.

4.2.3 Оформити звіт до лабораторної роботи.

4.3 Контрольні запитання

4.3.1 Що таке OpenHAB?

4.3.2 Для чого використовують скрипти?

4.3.3 З чого можуть викликатися скрипти?

4.3.4 Якою командою можливо зупинити процес?

4.4 Зміст звіту

4.4.1 Мета лабораторної роботи.

4.4.2 Короткі теоретичні відомості.

4.4.3 Знімки екрану, які відображають виконані завдання.

4.4.4 Висновки до роботи.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна

1. Пархоменко А. В., Туленков А.В., Соколянський О.В., Залюбовський Я І., Пархоменко А.В. Програмно-апаратна платформа для навчання технологіям Інтернету речей: навчальний посібник. – Запоріжжя: Дике Поле, 2017. – 120 с.
2. Parkhomenko, A. Integrated Complex for IoT Technologies Study [Text] / A. Parkhomenko, A. Tulenkov, A. Sokolyanskii, Y. Zalyubovskiy, A. Parkhomenko // Online Engineering & Internet of Things. – Cham: Springer, 2017. – P. 322-330 (Lecture Notes in Network and Systems, vol. 22) https://doi.org/10.1007/978-3-319-64352-6_31 (ISI Proceedings, MetaPress, Springerlink)
3. Parkhomenko, A. The application of the remote lab for studying the issues of Smart House systems power efficiency, safety and cybersecurity [Text] / A. Parkhomenko, A. Tulenkov, A.

Sokolyanskii, Y. Zalyubovskiy, A. Parkhomenko, A. Stepanenko // Smart Industry & Smart Education. – Cham: Springer, 2018. – P.395-403 (Lecture Notes in Network and Systems, vol. 47)

Додаткова

До лабораторної роботи 1

4. Coburn J. “Getting Started With Arduino: A Beginner’s Guide”. [Electronic resource]. – Access mode: <https://www.makeuseof.com/tag/getting-started-with-arduino-a-beginners-guide/>
5. Ardino IDE. [Electronic resource]. – Access mode: <https://www.arduino.cc/en/main/software>
6. Arduino Lesson 3. RGB LEDs. [Electronic resource]. – Access mode: <https://learn.adafruit.com/adafruit-arduino-lesson-3-rgb-leds/overview>
7. Processing. [Electronic resource]. – Access mode: <https://processing.org/>
8. Wiring. [Electronic resource]. – Access mode: <http://wiring.org.co/>
9. Reas C., and Fry B. “Getting Started with Processing”/ Published by O’Reilly Media, Inc., June 2010. P.209

До лабораторної роботи 2

10. Tinkercad. [Electronic resource]. – Access mode: <https://www.tinkercad.com/>.
11. Arduino Tutorials. [Electronic resource]. – Access mode: <https://www.arduino.cc/en/Tutorial/HomePage?from=Main.Tutorials>

До лабораторної роботи 3

12. Raspberry Pi. Downloads. <https://www.raspberrypi.org/downloads/>
13. Raspberry Pi Azure IoT Online Simulator. URL: <https://azure-samples.github.io/raspberry-pi-web-simulator/>.

До лабораторної роботи 4

14. OpenHAB Documentation. [Electronic resource]. – Access mode: <https://www.openhab.org/docs/tutorial/>
15. Getting Started with OpenHAB Home Automation on Raspberry Pi [Electronic resource]. – Access mode: <https://www.makeuseof.com/tag/getting-started-openhab-home-automation-raspberry-pi/>