Національний технічний університет України «КПІ ім. Ігоря Сікорського» Факультет Інформатики та Обчислювальної Техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №2

з дисципліни «Програмування мікропроцесорних систем»

на тему

«Програмування мікропроцесорних систем. Цифрові порти введення-виведення, широко-імпульсна модуляція (ШІМ), робота з кнопкою»

Виконав:

студент групи ІП-11   
Лошак В.

Викладач:

доц. Голубєв Л. П.

Київ – 2024

**Зміст**

[**Зміст** 2](#_Toc174357344)

[**Постановка задачі** 3](#_Toc174357345)

[**Виконання** 4](#_Toc174357346)

[**Контрольні питання** 8](#_Toc174357347)

[**Висновок** 10](#_Toc174357348)

[**Додатки** 11](#_Toc174357349)

**Постановка задачі**

**Мета:** познайомити студентів з основами програмування на Arduino та використанням нової інформації для створення приладів, з використанням світлодіодів та кнопок керування.

**Завдання до роботи:**

**Варіант 14  
Задача №1. «Боротьба» з брязкотом кнопки.**

Розробити власну функцію боротьби з **брязкотом кнопки**, використовуючи відповідну затримку t в часі :

Перевірити роботу програми на одному світлодіоді. Натискання кнопки вона загоряється, наступне натискання – гасне. Вхідний порт для світлодіодів – ;

**Задача №2. Робота з циклом.**

Створити проект , в якому світлодіод буде блимати в циклі по правилу:  
1) вхідний порт для світлодіоду – номер студента за списком n%13;  
2) час «горіння» та паузи співпадають та дорівнюють у мс:  
початкове значення n;  
крок – n+100;  
кінцеве значення параметру циклу 100\*(n % 5)+1000.

**Задача №3. Широко імпульсна модуляція**.

Скласти проекти по зміні яскравості світлодіоду від min до max, та від max до min з кроком 1.

min = n,

max = 255-n.

**Задача №4. Робота з функціями користувача, глобальними та локальними змінними.**

В схемі присутня кнопка та світлодіод. Початковий стан – кнопка не горить.

1. Натискування кнопки перший раз – світлодіод змінює яскравість від min до max;
2. Натискування кнопки другий раз – світлодіод змінює яскравість від max до min;
3. Натискування кнопки третій раз – світлодіод горить з максимальною яскравістю;
4. Натискування кнопки четвертий раз – світлодіод гасне і ми переходимо до початкового стану.

Відповідні значення для min та max та кроку взяти із задачі №3.

**Виконання**

**Задача №1. «Боротьба» з брязкотом кнопки.**

Для уникнення ефекту брязкоту ми використали функцію debounce що перевіряє стан кнопки з певною затримкою щоб переконатися що вона все ж була натиснута. Ефект брязкоту виникає бо механічні кнопки складаються з металевих контактів. При натисканні ці контакти фізично стикаються. Через пружність металу, вони можуть відскакувати один від одного кілька разів до остаточного з'єднання.

Мікроконтролери працюють дуже швидко і можуть зчитувати стан кнопки тисячі разів на секунду.Через це замість одного чіткого переходу від HIGH до LOW (або навпаки), мікроконтролер може зафіксувати кілька швидких переходів. Це може призвести до неправильної роботи програми, коли одне фізичне натискання може бути зареєстровано як кілька натискань.

A computer screen shot of a computer

Description automatically generated

A diagram of a circuit board

Description automatically generated

**Задача №2. Робота з циклом.**

Для цього завдання потрібно створити проект, де світлодіод блимає за певним правилом. Цей код реалізує блимання світлодіода за заданим правилом. Час горіння та паузи збільшується з кожною ітерацією.

A computer screen shot of a computer

Description automatically generated

A diagram of a circuit board

Description automatically generated

**Задача №3. Широко імпульсна модуляція**.

Використовуючи ШІМ ми можемо плавно змінювати яскравість діоду, коригуючи довжину його спалахів ми впливаємо на яскравість яку сприймає наш мозок дивлячись на світлодіод.

A computer screen shot of a computer

Description automatically generated

A circuit board with wires and a red and green line

Description automatically generated with medium confidence

**Задача №4. Робота з функціями користувача, глобальними та локальними змінними.**

В цьому завданні ми використовували глобальну змінну для підрахунку кількості натискань на кнопку, та switch statement щоб визначити дію що має бути виконана на поточному кроці програми. За основу було взято код з 1го завдання, і модифіковано таким чином щоб ми могли надсилати аналогові сигнали світлодіоду та рахувати кількість натискань кнопки

A computer screen shot of a computer

Description automatically generated

A diagram of a circuit board

Description automatically generated

**Контрольні питання**

1. Що таке макетна плата? Як вона влаштована?

Макетна плата - це інструмент для експериментів, що дозволяє легко збирати прості схеми без пайки. Вона має червоні і сині отвори з боків для живлення та заземлення відповідно. Кожні п'ять отворів у вертикальних рядах з'єднані між собою.

2. Що таке закон Ома?

Закон Ома визначає співвідношення між напругою (U), струмом (I) і опором (R) в електричному колі: U = I \* R.

3. Як потрібно підключати світлодіоди?

Світлодіоди підключають з урахуванням полярності. Анод (довша ніжка) підключається до джерела струму, катод (коротша ніжка) - до землі. Обов'язково потрібно використовувати резистор для обмеження струму.

4. Чим відрізняється константа від змінної?

Константа - це значення, яке не змінюється під час виконання програми. Змінна може змінювати своє значення.

5. Що таке цикл?

Цикл - це конструкція в програмуванні, яка дозволяє повторювати виконання блоку коду кілька разів.

6. Яка структура циклу for і як він працює?

Цикл for має три вирази: ініціалізація, умова продовження, крок зміни. Він виконує блок коду, поки умова істинна, змінюючи змінну циклу на кожній ітерації.

7. Що таке ШІМ?

ШІМ (широтно-імпульсна модуляція) - це метод імітації аналогового сигналу за допомогою цифрового виходу шляхом швидкої зміни стану виходу між високим і низьким рівнем.

8. Що таке скважність?

Скважність - це відношення тривалості імпульсу до періоду сигналу в ШІМ.

9. Навіщо потрібна функція analogWrite()? Які у неї аргументи?

analogWrite() використовується для генерації ШІМ-сигналу. Вона приймає два аргументи: номер піна і значення від 0 до 255, яке визначає скважність.

10. Як за допомогою ШІМ і циклу забезпечити плавне зростання яскравості світлодіоду?

Використовуйте цикл for для поступового збільшення значення, що передається в analogWrite().

11. Навіщо потрібний «стягуючий» резистор?

Стягуючий резистор забезпечує стабільний низький рівень сигналу на вході, коли кнопка не натиснута, запобігаючи "плаваючому" стану.

12. Що таке «брязкіт» кнопки? Чому він виникає? Як його усунути?

Брязкіт кнопки - це багаторазове швидке замикання/розмикання контактів при натисканні. Виникає через механічні особливості кнопки. Усувається програмно (затримки, фільтрація) або апаратно (конденсатори).

13. Основні типи даних в Arduino:

int, long, float, double, char, boolean.

14. Арифметичні оператори в Arduino:

+, -, \*, /, % (модуль).

15. Оператори порівняння в Arduino:

==, !=, <, >, <=, >=.

16. Логічні оператори в Arduino:

&&, ||, !

17. Унарні оператори в Arduino:

++, --, ! (логічне заперечення).

18. Основні оператори керування в Arduino:

if, else, for, while, switch, case.

19. Як зробити функцію користувача в Arduino?

Функція користувача створюється шляхом визначення типу повернення, імені функції, параметрів і тіла функції.

20. Глобальні та локальні змінні в Arduino. Область видимості змінних:

Глобальні змінні доступні в усій програмі, оголошуються поза функціями. Локальні змінні доступні тільки всередині функції або блоку, де вони оголошені. Область видимості визначає, де змінна може бути використана в програмі.

**Висновок**

У даній лабораторній роботі я розробив власну функцію для боротьби з брязкотом кнопки, використовуючи затримку, що залежить від номера студента, що дозволило зрозуміти важливість обробки вхідних сигналів для стабільної роботи системи. Також я створив проект з використанням циклу для керування блиманням світлодіода за заданим правилом, що допомогло освоїти роботу з циклами та цифровими виходами Arduino. Крім того, я реалізував зміну яскравості світлодіода за допомогою широтно-імпульсної модуляції (ШІМ), навчившись використовувати функцію analogWrite() для емуляції аналогового виходу на цифрових пінах. Нарешті, я розробив складну програму з використанням функцій користувача, глобальних та локальних змінних для керування світлодіодом за допомогою кнопки, що дозволило застосувати отримані знання для створення більш складної логіки роботи пристрою.

Усі результати наведені на знімках екрану вище, код програми та посилання на сам проект наведені в додатках нижче.

**Додатки**

**Посилання на проект:** [**https://www.tinkercad.com/things/78upEJRY15q-lab2loshak/editel?sharecode=CtHOTMs-QuDIejQPVhZ8w9fK2hg0V-aA7VHDoNFYtig**](https://www.tinkercad.com/things/78upEJRY15q-lab2loshak/editel?sharecode=CtHOTMs-QuDIejQPVhZ8w9fK2hg0V-aA7VHDoNFYtig)

**Код Завдання 1:**

const int n = 14;

const int buttonPin = 12; *// Вхідний порт для кнопки*

const int ledPin = n % 13; *// Вихідний порт для світлодіода*

bool ledState = false; *// Стан світлодіода*

bool lastButtonState = LOW; *// Попередній стан кнопки*

int getDebounceDelay(int n) {

  switch (n % 5) {

    case 0: return 5;

    case 1: return 6;

    case 2: return 7;

    case 3: return 8;

    case 4: return 9;

    default: return 5; *// На всяк випадок*

  }

}

const int debounceDelay = getDebounceDelay(n); *// Затримка для усунення брязкоту*

bool debounce(bool last) {

  bool current = digitalRead(buttonPin);

  if (last != current) {

    delay(debounceDelay);

    current = digitalRead(buttonPin);

  }

  return current;

}

void setup() {

  pinMode(buttonPin, INPUT);

  pinMode(ledPin, OUTPUT);

}

void loop() {

  bool currentButtonState = debounce(lastButtonState);

*// if statement виконується лише якщо ми натиснули на кнопку*

*// якщо кнопка утримується чи була відпущена то if не виконується*

  if (lastButtonState == LOW and currentButtonState == HIGH) {

    ledState = !ledState;

  }

*// коли кнопка буде відпущена, lastButtonState оновиться до LOW*

*// що дасть можливість if знову виконатися коли кнопку натиснуть.*

  lastButtonState = currentButtonState;

  digitalWrite(ledPin, ledState);

}

**Код Завдання 2:**

const int n = 14;

const int ledPin = n % 13; *// Вхідний порт для світлодіода*

void setup() {

  pinMode(ledPin, OUTPUT);

}

void loop() {

  for (int i = n; i <= 100 \* (n % 5) + 1000; i += n + 100) {

    digitalWrite(ledPin, HIGH);

    delay(i);

    digitalWrite(ledPin, LOW);

    delay(i);

  }

}

**Код Завдання 3:**

const int n = 14;

const int ledPin = 3; *// Вхідний порт для світлодіода*

const int minBrightness = n;

const int maxBrightness = 255 - n;

void setup() {

  pinMode(ledPin, OUTPUT);

}

void loop() {

*// Зміна яскравості від min до max*

  for (int i = minBrightness; i <= maxBrightness; i++) {

    analogWrite(ledPin, i);

    delay(10);

  }

  delay(1000); *// Пауза на максимальній яскравості*

*// Зміна яскравості від max до min*

  for (int i = maxBrightness; i >= minBrightness; i--) {

    analogWrite(ledPin, i);

    delay(10);

  }

  delay(1000); *// Пауза на мінімальній яскравості*

}

**Код Завдання 4:**

const int n = 14;

const int buttonPin = 12; *// Вхідний порт для кнопки*

const int ledPin = 3; *// Вихідний порт для світлодіода*

bool lastButtonState = LOW;

const int minBrightness = n;

const int maxBrightness = 255 - n;

int ledBrightness = 0;

int buttonPressCount = 0;

int getDebounceDelay(int n) {

  switch (n % 5) {

    case 0: return 5;

    case 1: return 6;

    case 2: return 7;

    case 3: return 8;

    case 4: return 9;

    default: return 5; *// На всяк випадок*

  }

}

const int debounceDelay = getDebounceDelay(n); *// Затримка для усунення брязкоту*

bool debounce(bool last) {

  bool current = digitalRead(buttonPin);

  if (last != current) {

    delay(debounceDelay);

    current = digitalRead(buttonPin);

  }

  return current;

}

void changeBrightness(int start, int end, int step) {

  for (int i = start; i != end; i += step) {

    analogWrite(ledPin, i);

    delay(10);

  }

}

void setup() {

  pinMode(buttonPin, INPUT);

  pinMode(ledPin, OUTPUT);

}

void loop() {

  bool currentButtonState = debounce(lastButtonState);

  if (lastButtonState == LOW && currentButtonState == HIGH) {

    buttonPressCount++;

    switch (buttonPressCount % 4) {

      case 1:

        changeBrightness(minBrightness, maxBrightness + 1, 1);

        break;

      case 2:

        changeBrightness(maxBrightness, minBrightness - 1, -1);

        break;

      case 3:

        analogWrite(ledPin, maxBrightness);

        break;

      case 0:

        digitalWrite(ledPin, LOW);

        break;

    }

  }

  lastButtonState = currentButtonState;

}