

Міністерство освіти і науки України

Національний технічний університет України „КПІ

імені Ігоря Сікорського ”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

**ЗВІТ**

лабораторної роботи №2

курсу «Проектування мікропроцесорних систем на базі Arduino та Raspberry»

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

Перевірив: Виконав:

Доц. Голубєв Л. П. Гр. ІТ-04

Чапча Святослав

Київ 2023

ЗМІСТ

[1. Постановка задачі 3](#_Toc146628348)

[2. Виконання 4](#_Toc146628349)

[3. Контрольні питання 8](#_Toc146628350)

[4. Висновок 11](#_Toc146628351)

[5. Додатки 11](#_Toc146628352)

# 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:**

**В кожній з робіт потрібно розробити схеми та заставити її працювати за правилами, що викладені в задачах 1-4. До кожної із задач у зошиті повинні бути намальовані відповідні схеми! Всюди 93 – номер студента за списком.**

Завдання 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 – ;

Завдання 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.

Завдання 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.

Завдання 4. Робота з функціями користувача, глобальними та локальними змінними.

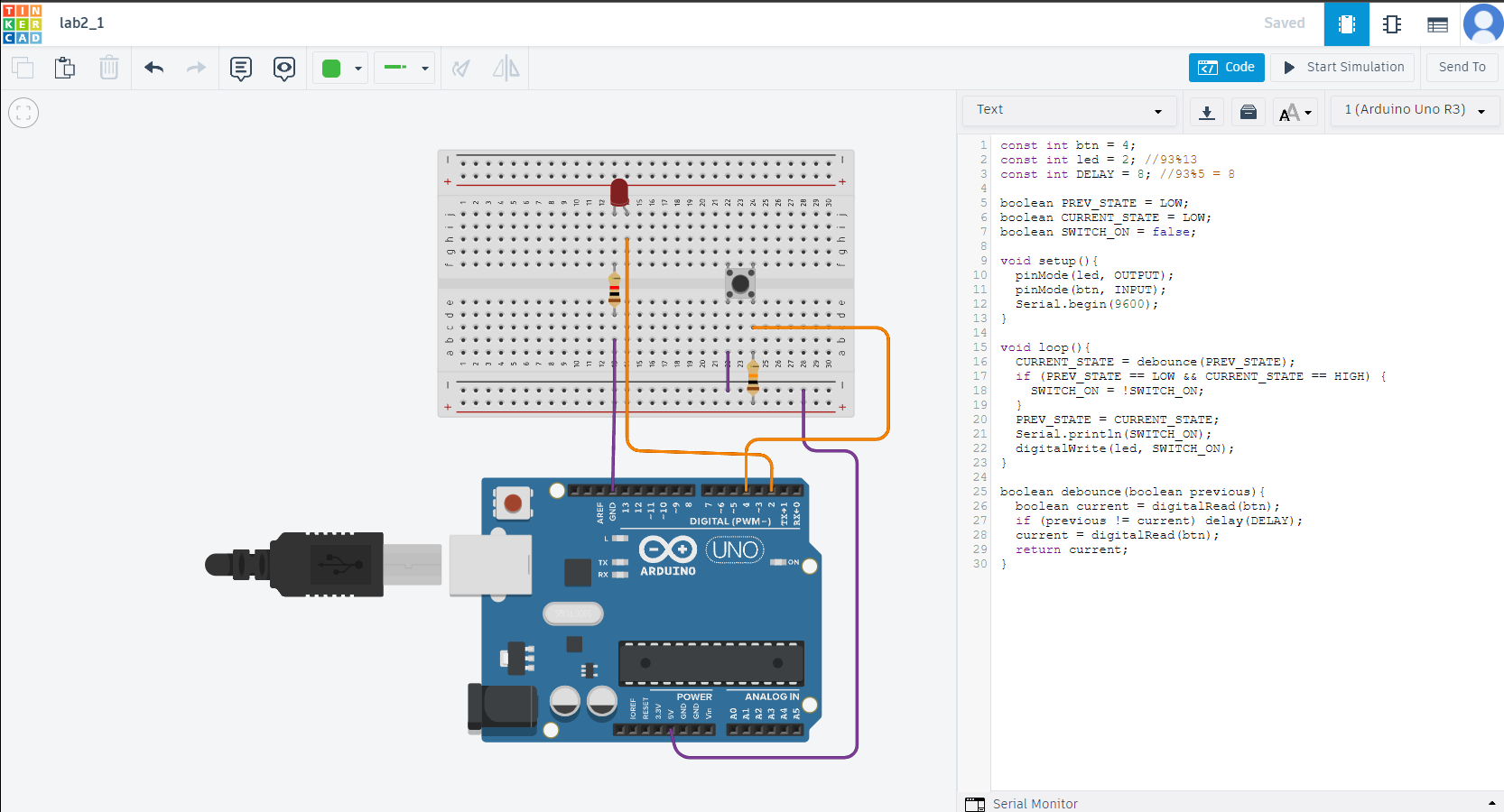
В схемі присутня кнопка та світлодіод. Початковий стан – кнопка не горить.

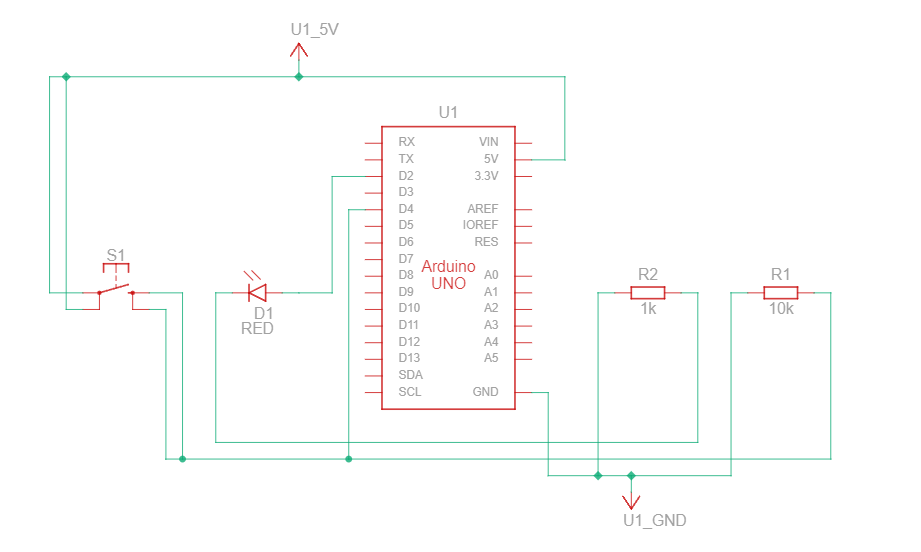
1. Натискування кнопки перший раз – світлодіод змінює яскравість від min до max;
2. Натискування кнопки другий раз – світлодіод змінює яскравість від max до min;
3. Натискування кнопки третій раз – світлодіод горить з максимальною яскравістю;
4. Натискування кнопки четвертий раз – світлодіод гасне і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

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

# Виконання

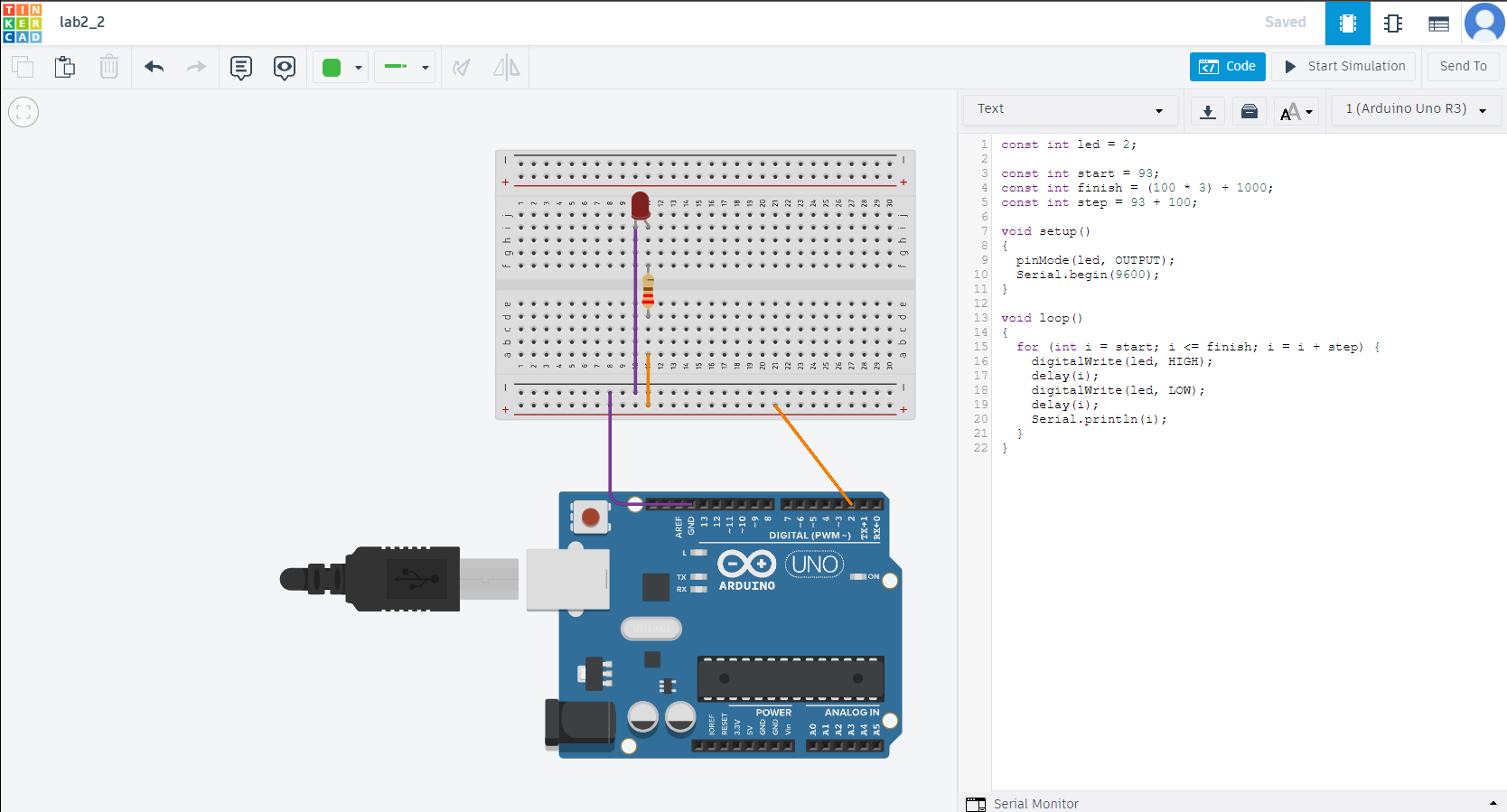
Створимо перший проект «Clang», суть якого полягає в розробці власної функції боротьби з брязкотом кнопки, код та знімки екрану наведені нижче:

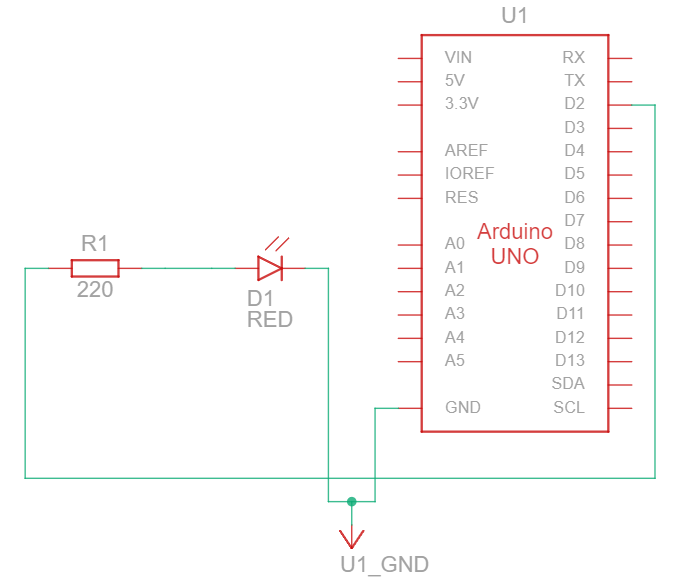




Як бачимо, усе працює, отже можна переходити до створення другого проекту «Loop».

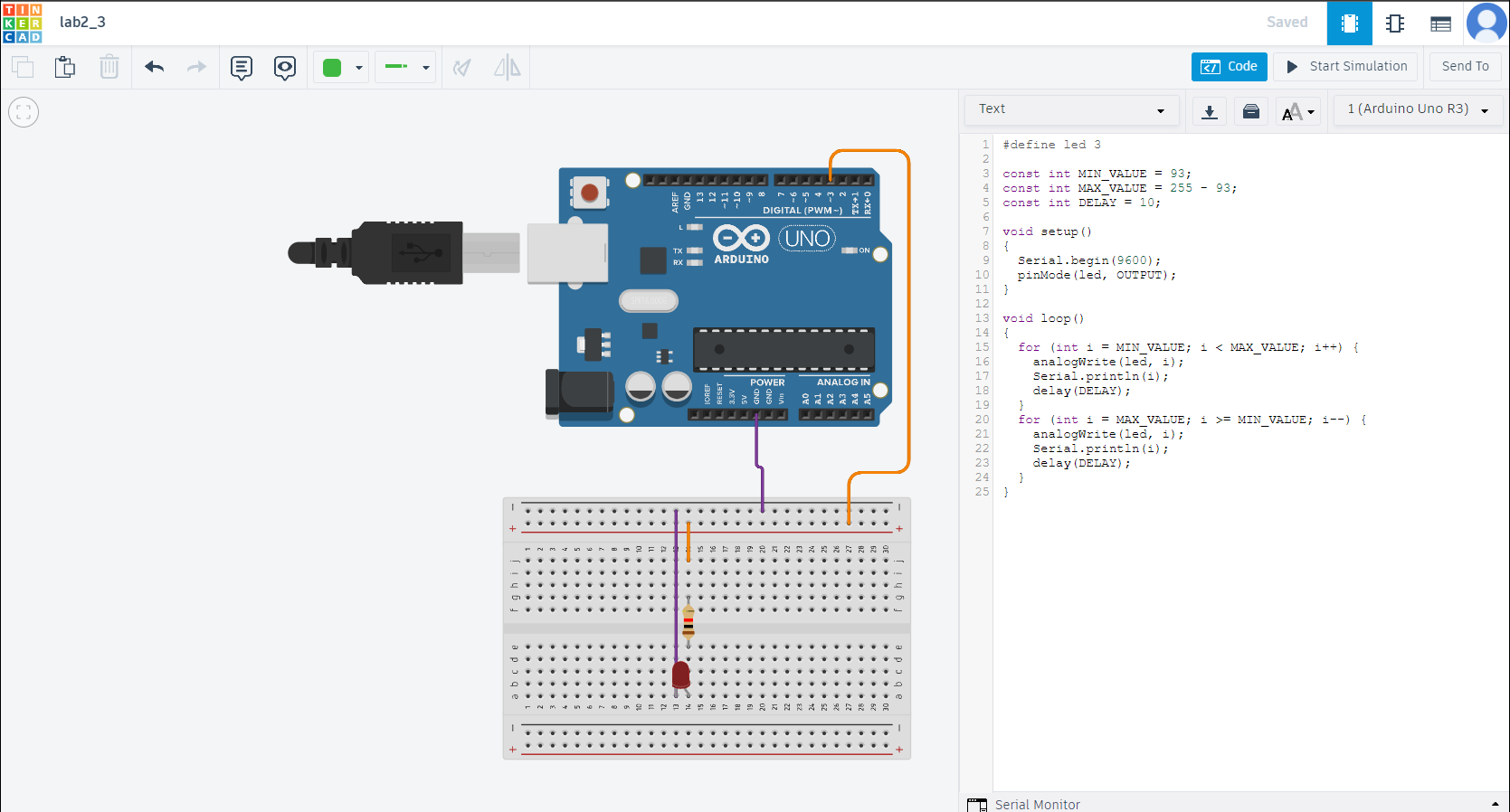
Для виконання даного проекту потрібно розробити код, в якому світлодіод буде блимати в циклі по зазначеним правилам. Код та знімки наведені нижче:

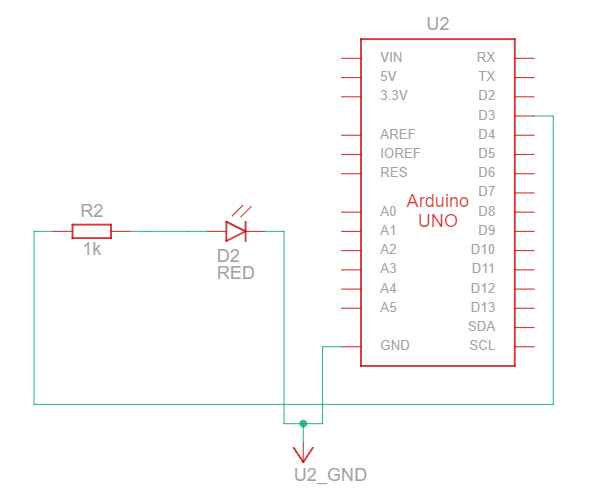




Як бачимо, усе працює, отже можна переходити до створення третього проекту «Modul».

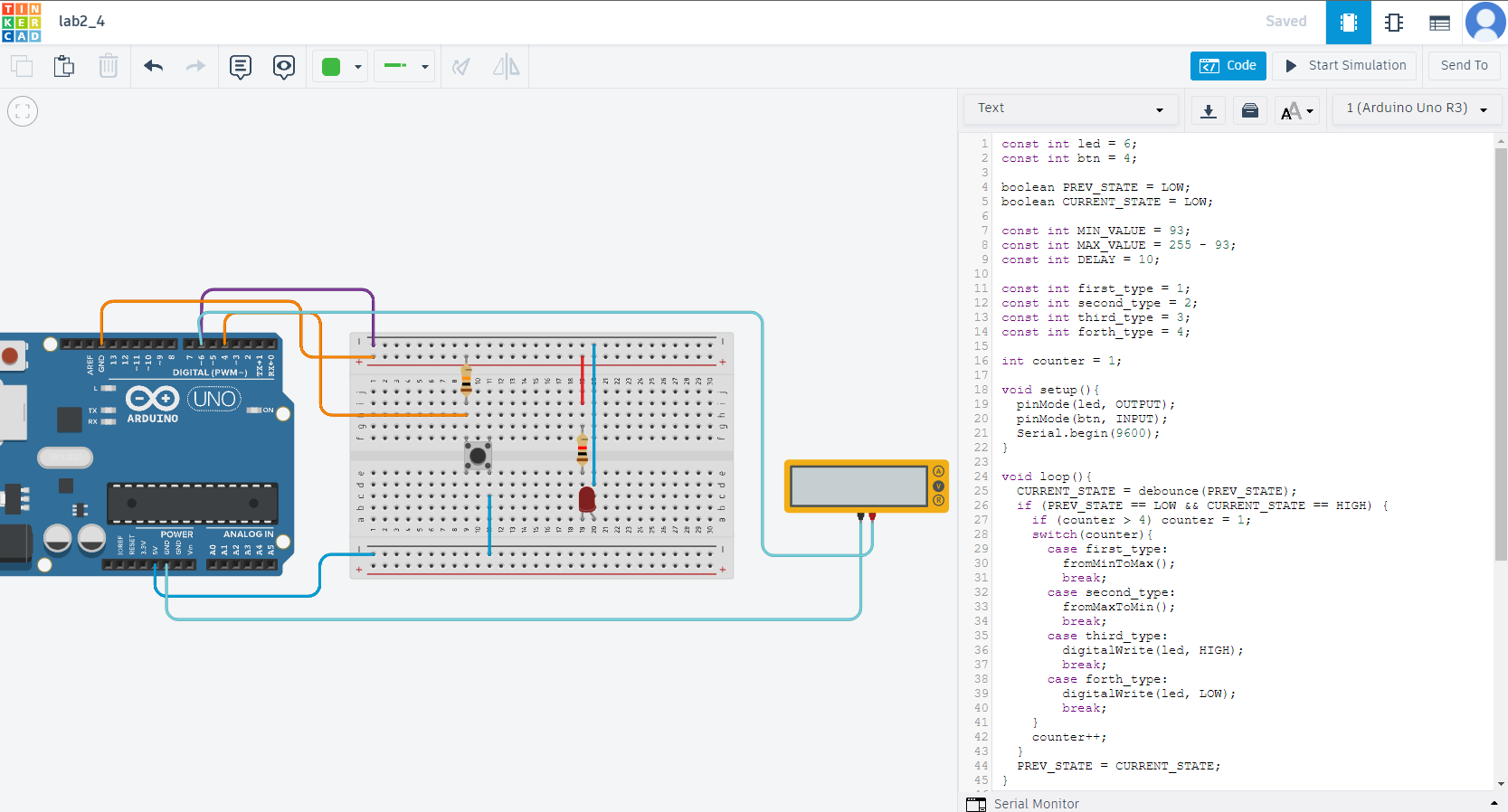
Суть полягає у створенні проекту по зміні яскравості світлодіоду від min до max, та від max до min з кроком 1. Код та знімки наведені нижче:

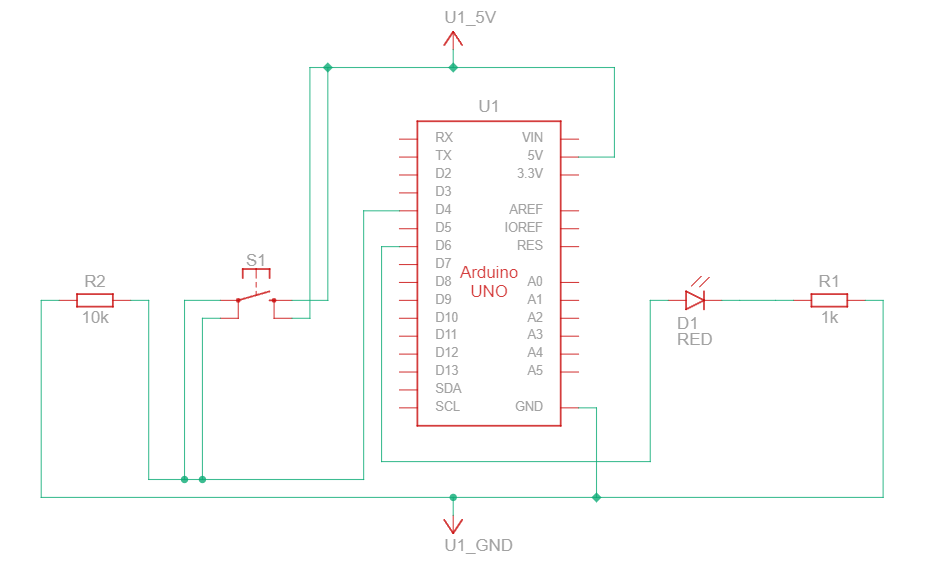




Як бачимо, усе працює, отже можна переходити до створення четвертого проекту «Multibtn».

Суть проекту полягає у вивченні функції користувача та знайомстві з глобальними та локальними змінними





# Контрольні питання

1. Що таке макетна плата? Як вона влаштована?

Макетна плата (або платка для прототипування) - це спеціальна плата з отворами і провідниками, призначена для збирання та тестування електронних прототипів. Вона дозволяє підключати електронні компоненти, такі як резистори, транзистори, мікроконтролери тощо, без необхідності паяння. Макетна плата має спеціальну сітку отворів і провідників, які дозволяють легко підключати компоненти і створювати з'єднання між ними.

1. Що таке закон Ома?

Закон Ома - це фундаментальний закон електротехніки, який встановлює зв'язок між напругою (V), опором (R) та силою струму (I) в електричному колі. Він формулюється як V = I \* R, де V - напруга в вольтах, I - сила струму в амперах, R - опір в омах. Закон Ома стверджує, що напруга пропорційна силі струму і опору в колі.

1. Як потрібно підключати світлодіоди?

Світлодіоди підключаються до Arduino шляхом підключення їх аноду (довшого виводу) до виходу піна і катоду (коротшого виводу) до землі (GND) або іншого джерела нульової напруги.

1. Чим відрізняється константа від змінної?

Константа - це значення, яке залишається незмінним протягом всього виконання програми. Змінна - це ім'я, яке використовується для зберігання та зміни даних під час виконання програми.

1. Що таке цикл?

Цикл - це повторюваний фрагмент коду, який виконується багато разів до виконання певної умови.

1. Яка структура циклу for і як він працює?

Цикл for в Arduino має таку структуру: for (ініціалізація; умова; крок). Він спочатку виконує ініціалізацію, потім перевіряє умову. Якщо умова виконується, цикл виконує свій тіло і зростає (або зменшує) змінну крок. Цикл повторюється до тих пір, поки умова не стане хибною.

1. Що таке ШІМ?

ШИМ (широтно-імпульсна модуляція) - це метод керування сигналами, який використовується для регулювання аналогових сигналів (наприклад, яскравості світлодіода або швидкості обертання мотора) за допомогою цифрових пінів Arduino.

1. Що таке скважність?

Скважність - це відношення часу, протягом якого сигнал у ШИМ встановлений в HIGH (включений), до загального періоду сигналу.

1. Навіщо потрібна функція analogwrite()? Які у неї аргументи?

Функція analogWrite() використовується для генерації ШИМ-сигналу на певному піні Arduino. Її аргументи включають: Перший параметр - номер піна, на якому буде генеруватися ШИМ-сигнал. Другий параметр - значення ШИМ (від 0 до 255), де 0 відповідає 0% яскравості, а 255 - 100%.

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

Для плавного зростання яскравості світлодіода за допомогою ШИМ і циклу, можна використовувати цикл for, який збільшує значення ШИМ від 0 до бажаного максимального значення з певною затримкою між ітераціями циклу.

1. Навіщо потрібний «стягуючий» резистор?

"Стягуючий" резистор (пуль-ап резистор) використовується для підтягування лінійного або іншого низькострумового сигналу до високого рівня (наприклад, до VCC). Він забезпечує надійне читання сигналів в умовах шуму та перешкод.

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

"Брязкіт" кнопки виникає внаслідок низької якості зв'язку між контактами кнопки. Щоб усунути цей брязкіт, можна використовувати програмне затримування після зчитування стану кнопки або використовувати "стабілізуючий" конденсатор.

1. Основні типи даних в Arduino.

Основні типи даних в Arduino включають цілі числа (int), беззнакові цілі числа (unsigned int), дійсні числа (float), символи (char), булеві значення (boolean) та інші.

1. Арифметичні оператори в Arduino.

Арифметичні оператори в Arduino включають + (додавання), - (віднімання), \* (множення), / (ділення) та % (остача від ділення).

1. Оператори порівняння в Arduino.

Оператори порівняння в Arduino включають == (рівність), != (не рівність), < (менше), > (більше), <= (менше або рівне) та >= (більше або рівне).

1. Логічні оператори в Arduino.

Логічні оператори в Arduino включають && (логічне "І"), || (логічне "АБО") та ! (логічне заперечення).

1. Унарні оператори в Arduino.

Унарні оператори в Arduino включають ++ (інкремент, збільшення на 1) та -- (декремент, зменшення на 1).

1. Основні оператори керування в Arduino.

Основні оператори керування в Arduino включають if, else, while, for, switch, case, break, та return.

1. Як зробити функцію користувача в Arduino?

Для створення функції користувача в Arduino, ви можете використовувати ключове слово void, за яким слідує ім'я функції та список параметрів у круглих дужках.

1. Глобальні та локальні змінні в Arduino. Область видимості змінних.

Глобальні змінні в Arduino оголошуються за межами будь-яких функцій і можуть бути доступні з будь-якого місця в програмі. Локальні змінні оголошуються всередині функцій і мають обмежену область видимості, що обмежує їх доступність на рівні функції, де вони оголошені.

# Висновок

У даній лабораторній роботі я познайомився з основами програмування на Arduino та створенням приладів, з використанням світлодіодів та кнопок керування. В процесі виконання я створив чотири проекти, в яких я розробив власну функцію боротьби з брязкотом кнопки, ознайомився з циклом, з широко імпульсною модуляцією, функціями користувача, глобальними та локальними змінними. Усі результати наведені на знімках екрану вище, код програми та посилання на сам проект

# Додатки

Проект Clang: <https://www.tinkercad.com/things/0KTnHEhv23Z?sharecode=EvZ1UtnoI7PYGcTMsV2fGnSrnE0TMka8imlIMTRWbnQ>

Проект Loop: <https://www.tinkercad.com/things/7Frk1g8syBT?sharecode=DTiIx7Jkc0DMfmCoTwBCLtFHAAHOW6c0NvgExVwIVc0>

Проект Modul:

<https://www.tinkercad.com/things/2Glzu4rcmuF?sharecode=Gp4LWEVgkPvsWrAeem2rQMALC8ycBSDRB_7KRBNt2Bw>

Проект Multibtn:

<https://www.tinkercad.com/things/b2Jk86n44oh?sharecode=NrzzP0wuorf_yMgVL2nvWOQOUVSA2lgDPTdlfU-k7hs>

Clang.cpp

const int btn = 4;

const int led = 2; //93%13

const int DELAY = 8; //93%5 = 8

boolean PREV\_STATE = LOW;

boolean CURRENT\_STATE = LOW;

boolean SWITCH\_ON = false;

void setup(){

  pinMode(led, OUTPUT);

  pinMode(btn, INPUT);

  Serial.begin(9600);

}

void loop(){

  CURRENT\_STATE = debounce(PREV\_STATE);

  if (PREV\_STATE == LOW && CURRENT\_STATE == HIGH) {

    SWITCH\_ON = !SWITCH\_ON;

  }

  PREV\_STATE = CURRENT\_STATE;

  Serial.println(SWITCH\_ON);

  digitalWrite(led, SWITCH\_ON);

}

boolean debounce(boolean previous){

  boolean current = digitalRead(btn);

  if (previous != current) delay(DELAY);

  current = digitalRead(btn);

  return current;

}

Loop.cpp

const int led = 2;

const int start = 93;

const int finish = (100 \* 3) + 1000;

const int step = 93 + 100;

void setup()

{

  pinMode(led, OUTPUT);

  Serial.begin(9600);

}

void loop()

{

  for (int i = start; i <= finish; i = i + step) {

    digitalWrite(led, HIGH);

    delay(i);

    digitalWrite(led, LOW);

    delay(i);

    Serial.println(i);

  }

}

Modul.cpp

#define led 3

const int MIN\_VALUE = 93;

const int MAX\_VALUE = 255 - 93;

const int DELAY = 10;

void setup()

{

  Serial.begin(9600);

  pinMode(led, OUTPUT);

}

void loop()

{

  for (int i = MIN\_VALUE; i < MAX\_VALUE; i++) {

    analogWrite(led, i);

    Serial.println(i);

    delay(DELAY);

  }

  for (int i = MAX\_VALUE; i >= MIN\_VALUE; i--) {

    analogWrite(led, i);

    Serial.println(i);

    delay(DELAY);

  }

}

Multibtn.cpp

const int led = 6;

const int btn = 4;

boolean PREV\_STATE = LOW;

boolean CURRENT\_STATE = LOW;

const int MIN\_VALUE = 93;

const int MAX\_VALUE = 255 - 93;

const int DELAY = 10;

const int first\_type = 1;

const int second\_type = 2;

const int third\_type = 3;

const int forth\_type = 4;

int counter = 1;

void setup(){

  pinMode(led, OUTPUT);

  pinMode(btn, INPUT);

  Serial.begin(9600);

}

void loop(){

  CURRENT\_STATE = debounce(PREV\_STATE);

  if (PREV\_STATE == LOW && CURRENT\_STATE == HIGH) {

    if (counter > 4) counter = 1;

    switch(counter){

      case first\_type:

        fromMinToMax();

        break;

      case second\_type:

        fromMaxToMin();

        break;

      case third\_type:

        digitalWrite(led, HIGH);

        break;

      case forth\_type:

        digitalWrite(led, LOW);

        break;

    }

    counter++;

  }

  PREV\_STATE = CURRENT\_STATE;

}

void fromMinToMax(){

  for (int i = MIN\_VALUE; i < MAX\_VALUE; i++) {

    analogWrite(led, i);

    Serial.println(i);

    delay(DELAY);

  }

}

void fromMaxToMin(){

  for (int i = MAX\_VALUE; i >= MIN\_VALUE; i--) {

    analogWrite(led, i);

    Serial.println(i);

    delay(DELAY);

  }

}

boolean debounce(boolean last){

  boolean current = digitalRead(btn);

  if (last != current) delay(5);

  current = digitalRead(btn);

  return current;

}