Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра Обчислювальної Техніки

**Лабораторна робота №1**

з дисципліни "Розробка ігрових застосувань. Unity рішення"

Тема: "Дослідження базового патерну ігрового рушія Unity на прикладі двовимірної технології"

Виконав: Перевірив:

студент групи ІП-93 Катін Павло Юрійович

Домінський Валентин

Олексійович

Київ 2022

Зміст

[Мета: 3](#_Toc114348671)

[Завдання до роботи: 3](#_Toc114348672)

[Хід роботи: 4](#_Toc114348673)

[«Боротьба» з брязкотом кнопки: 4](#_Toc114348674)

[Робота з циклом: 5](#_Toc114348675)

[Широко імпульсна модуляція: 6](#_Toc114348676)

[Робота з функціями користувача, глобальними та локальними змінними: 7](#_Toc114348677)

[Контрольні запитання: 7](#_Toc114348678)

[Висновки: 9](#_Toc114348679)

[Додатки: 10](#_Toc114348680)

[Вихідний код: 10](#_Toc114348681)

[Посилання: 13](#_Toc114348682)

# Мета:

1. полягає у набутті знань, умінь та навичок з технології розроблення основ проекту з використанням обраної мови програмування у обраній парадигмі. Надається досвід створення репозиторію у системі контролю версій

# Завдання до роботи:

1. Репозиторій у системі контролю версій. Створити проект 2D. Загальні вимоги. Акаунт на GitHab, на даному етапі за бажанням. Репозиторій на GitHab з проектом. Назва GameProgLab1GroupNum, де зафарбовано номер групи.
2. Установка ігрового рушія. Створений проект IDE (2D) на основі рушія, що містить 1 сцену, ігровий персонаж. Можуть бути включені інші елементи. Розроблений і налагоджений скрипт для управління ігровим персонажем. Достатньо продемонструвати рух ліворуч, праворуч, стрибки, коректну фізику, зупинку перед перешкодою. Проект розташовано у репозиторій на GitHab

# Хід роботи:

Варіант – 8 -> 2D Stone Age World Platformer Starter Kit та коло

## «Боротьба» з брязкотом кнопки:

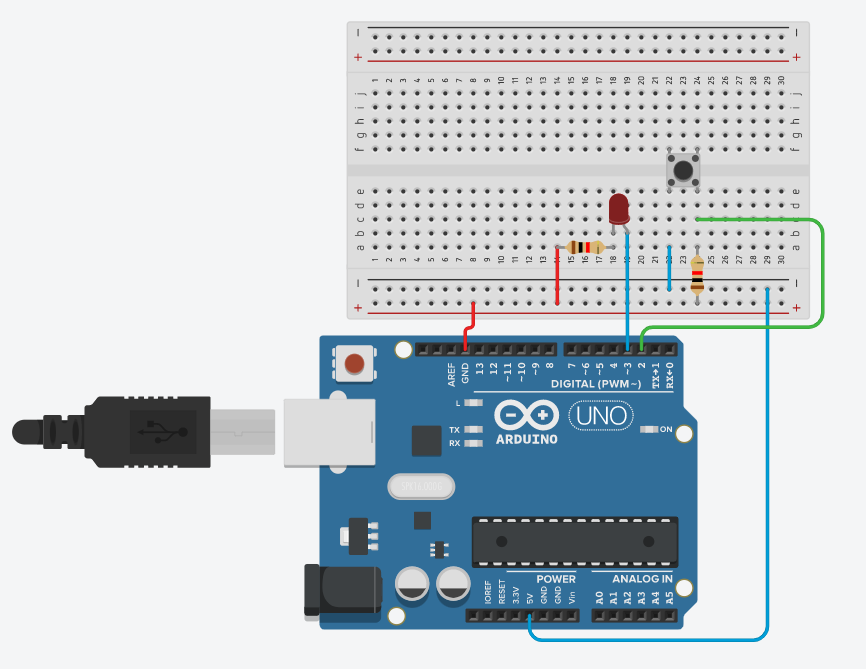
Для даного завдання Ми використаємо такий алгоритм:

1. При натисканні кнопки Ми запам’ятовуємо значення та чекаємо певний час. Якщо значення не змінилося – користувач дійсно натиснув на кнопку та хоче, аби щось спрацювало
2. Перевіряємо, якщо минула та теперішня кнопки відрізняються. Якщо так, то інвертуємо значення світлодіоду

Умови:

t -> 91 mod 5 = 1 -> 6 cекунд

Вхідний порт – 3



## Робота з циклом:

Тут усе доволі просто:

Пишемо цикл від і = n, доки i <= 100\*(n mod 5)+1000 та додаємо кожну ітерацію 100 до і

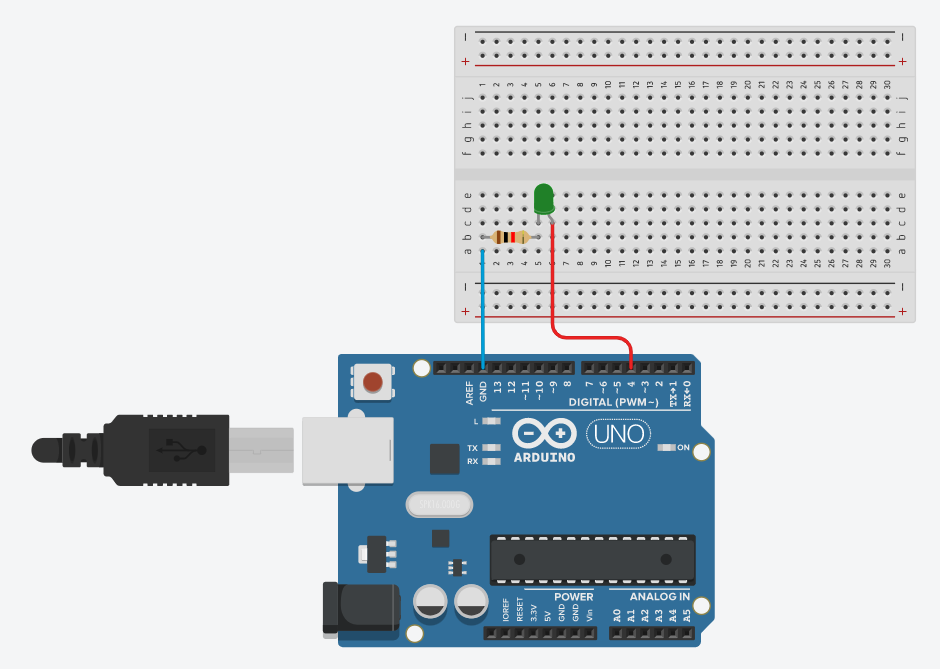
Умови:

Вхідний порт –> 91 mod 14 = 7 -> 7

Початкове значення n = 91

Крок = 91+100

Кінцеве значення параметру циклу 100\*(91 mod 5)+1000 -> 100 \* 1 + 1000 = 1100



## Широко імпульсна модуляція:

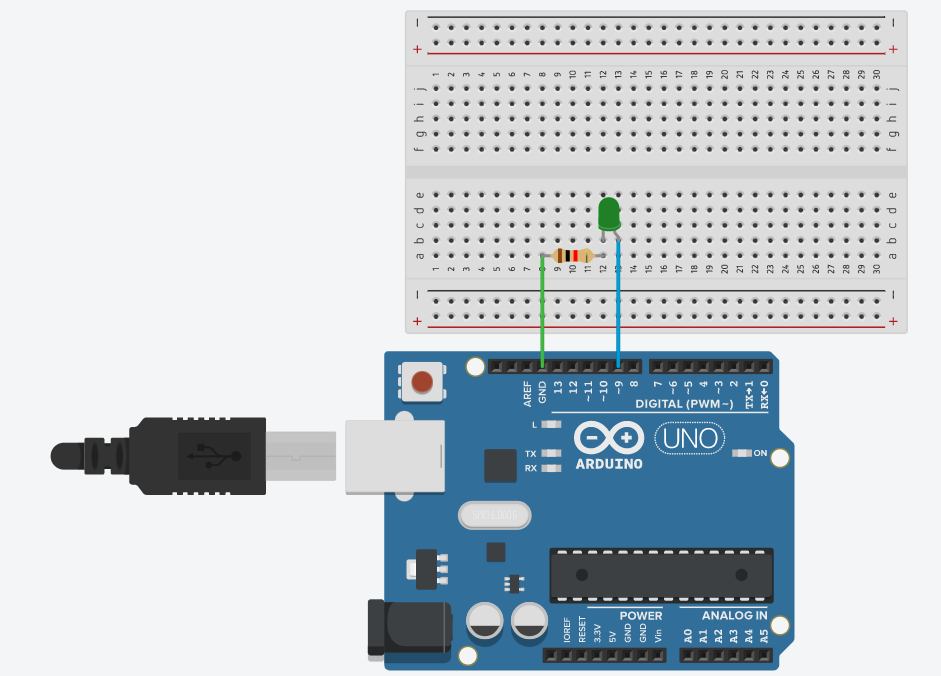
Для цього завдання треба 2 цикли:

Один починається з min та йде до max, а другий - навпаки

Умови:

Min = 91

Max = 255 – 91 = 164

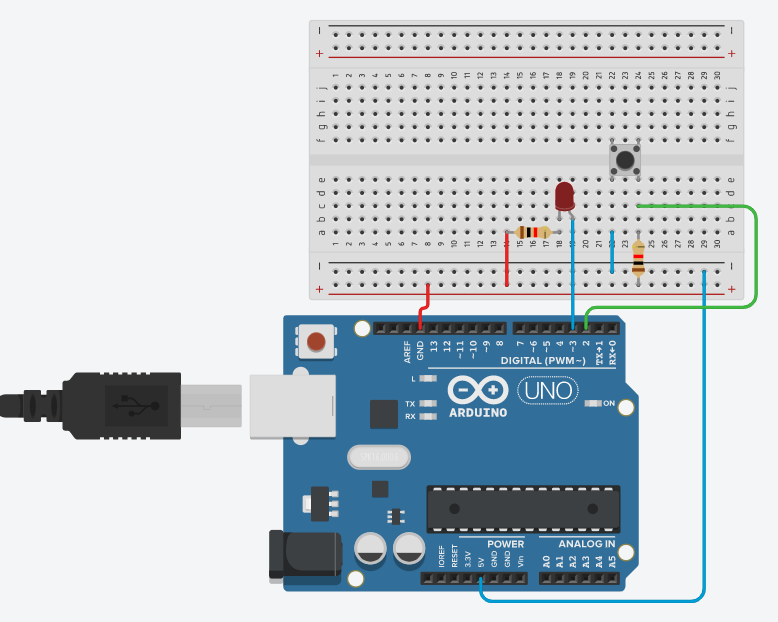


## Робота з функціями користувача, глобальними та локальними змінними:

Тут у Нас є 4 стани, в яких перебуває система, які міняються почергово після натискання кнопки. Для цього Я зроблю перерахування можливих варіантів і при взаємодії користувача буду міняти стан на наступний

Умови:

Відповідні значення для min та max та кроку взяті з задачі №3



# Контрольні запитання:

1. Що таке макетна плата? Як вона влаштована?
   1. Макетна плата -зручний інструмент для експериментів, дозволяє легко збирати прості схеми без виготовлення друкованих плат і пайки
   2. З двох сторін по всій довжині макетної плати розташовані червоні і сині отвори . Всі червоні отвори з'єднані між собою і служать, як правило, для подачі живлення. Для більшості проектів з цієї книги це +5 В. Всі сині отвори теж електрично з'єднані один з одним і грають роль шини заземлення . Кожні п'ять отворів, розташованих вертикальними рядами , також з’єднані один з одним . Посередині є вільне місце для зручності установки компонентів на макетної платі
2. Що таке закон Ома?
   1. Закон Ома визначає співвідношення між напругою (вимірюється в вольтах), струмом (вимірюється в амперах) і опором (вимірюється в Омах) в ланцюзі
3. Як потрібно підключати світлодіоди?
   1. Як виходи
4. Чим відрізняється константа від змінної?
   1. Константа не змінюється
5. Що таке цикл?
   1. Набір операцій, які відбуваються протягом Н-ої к-сті ітерацій
6. Яка структура циклу for і як він працює?
   1. Цикл for складається з 3 частин:
      1. Задання початкового значення
      2. Умова закінчення
      3. Крок
   2. Усе, що знаходиться в циклі буде повторено стільки разів, доки не буде задоволений цикл
7. Що таке ШІМ?
   1. Широко-імпульсна модуляція. Використовується для емуляції аналогового виходу; зміна скважності
8. Що таке скважність?
   1. Відношення періоду до тривалості імпульсу
9. Навіщо потрібна функція analogwrite()? Які у неї аргументи?
   1. Встановлює скважність послідовності прямокутних імпульсів в залежності від значення
   2. Номер контакту, 8-розрядний значення в діапазоні від 0 до 255, яке встановлюється на цьому контакті
10. Як за допомогою ШІМ і циклу забезпечити плавне зростання яскравості світлодіоду?
    1. Ми будемо поступово збільшувати скважність і таким чином обманювати Наш мозок
11. Навіщо потрібний «стягуючий» резистор?
    1. Він гарантує, що кнопка не створить коротке замикання між 5 В і землею при натисканні і що вхідний контакт не залишиться в "підвішеному" стані
12. Що таке «брязкіт» кнопки? Чому він виникає? Як його усунути?
    1. При натисканні на кнопку сигнал не просто змінюється від низького до високого, він протягом декількох мілісекунд неодноразово змінює своє значення, перш ніж установиться рівень HIGH
    2. Після натискання кнопки треба зачекати деякий час
13. Основні типи даних в Arduino
    1. Boolean, byte, int, word, unsigned int, float, double, string, String
14. Арифметичні оператори в Arduino.
    1. =, +, -, \*, /, %
15. Оператори порівняння в Arduino
    1. ==, !=, <, >, <=, >=
16. Логічні оператори в Arduino
    1. &&, ||, !
17. Унарні оператори в Arduino
    1. ++, --
18. Основні оператори керування в Arduino
    1. If, else if, else
19. Як зробити функцію користувача в Arduino?
    1. Тип значення, що повертається; назва функції; параметри; фігурні дужки; оператор, який повертає ззначення; тіло функції
20. Глобальні та локальні змінні в Arduino. Область видимості змінних
    1. Глобальні видимі всьому коду. Локальні – лише певній області видимості

# Висновки:

Я вперше спробував попрацювати з ШІМ та кнопкою. Зумів розібратися з проблемою «брязкоту». Пописав цикли (як звичайні, так і зворотні). Розібрався з умовними операторами, функціями користувача. Зрозумів різницю між глобальними змінними та локальними.

# Додатки:

# Вихідний код:

Задача №1:

const int led\_pin = 3;

const int button\_pin = 2;

const int time\_delay = 6;

boolean lastButton = LOW;

boolean currentButton = LOW;

boolean ledOn = false;

void setup()

{

pinMode(led\_pin, OUTPUT);

pinMode(button\_pin, INPUT);

}

boolean debounce(boolean last)

{

boolean current = digitalRead(button\_pin);

if (last != current)

{

delay(time\_delay);

}

return digitalRead(button\_pin);

}

void loop()

{

currentButton = debounce(lastButton);

if(lastButton == LOW && currentButton == HIGH)

{

ledOn = !ledOn;

}

lastButton = currentButton;

digitalWrite(led\_pin, ledOn);

}

Задача №2:

const int led\_pin = 7;

const int start\_n = 91;

const int final\_i = 1100;

void setup()

{

pinMode(led\_pin, OUTPUT);

}

void loop()

{

for (int i = start\_n; i <= final\_i; i += 100)

{

digitalWrite(led\_pin, HIGH);

delay(i);

digitalWrite(led\_pin, LOW);

delay(i);

}

}

Задача №3:

const int led\_pin = 9;

const int min = 91;

const int max = 164;

const int time\_delay = 10;

void setup()

{

pinMode(led\_pin, OUTPUT);

Serial.begin(9600);

}

void loop()

{

for (int i = min; i <= max; i ++)

{

analogWrite(led\_pin, i);

Serial.println(i);

delay(time\_delay);

}

for (int i = max; i >= min; i --)

{

analogWrite(led\_pin, i);

Serial.println(i);

delay(time\_delay);

}

}

Задача №4:

enum class ButtonType { MinToMax = 1, MaxToMin, ledOn, ledOff };

const int led\_pin = 3;

const int button\_pin = 2;

boolean lastButton = LOW;

boolean currentButton = LOW;

const int min = 91;

const int max = 164;

const int time\_delay = 10;

ButtonType currentButtonType = ButtonType::ledOff;

void setup()

{

pinMode(led\_pin, OUTPUT);

pinMode(button\_pin, INPUT);

Serial.begin(9600);

}

boolean debounce(boolean last)

{

boolean current = digitalRead(button\_pin);

if (last != current)

{

delay(time\_delay);

}

return digitalRead(button\_pin);

}

void handleButtonExecution()

{

int nextButtonType = static\_cast<int>(currentButtonType) + 1;

if (nextButtonType > 4)

{

nextButtonType = 1;

}

currentButtonType = static\_cast<ButtonType>

(nextButtonType);

Serial.println(nextButtonType);

switch(currentButtonType)

{

case ButtonType::MinToMax:

minToMax();

break;

case ButtonType::MaxToMin:

maxToMin();

break;

case ButtonType::ledOn:

maxLight();

break;

case ButtonType::ledOff:

minLight();

break;

}

}

void minToMax()

{

for (int i = min; i <= max; i ++)

{

analogWrite(led\_pin, i);

delay(time\_delay);

}

}

void maxToMin()

{

for (int i = max; i >= min; i --)

{

analogWrite(led\_pin, i);

delay(time\_delay);

}

}

void maxLight()

{

digitalWrite(led\_pin, HIGH);

}

void minLight()

{

digitalWrite(led\_pin, LOW);

}

void loop()

{

currentButton = debounce(lastButton);

if(lastButton == LOW && currentButton == HIGH)

{

handleButtonExecution();

}

lastButton = currentButton;

}

# Посилання:

Проект - [посилання](https://www.tinkercad.com/things/grFxksPP3Km?sharecode=woNZdGHDwQRou3MyxT2OnZ_m13FfXy96OK5-lpMN-oU)