ЛАБОРАТОРНА РОБОТА № 7

Організація роботи з аналоговим датчиком температури LM35

Мета роботи: ознайомитися з принципами роботи датчика LM35. Отримати практичні навички по підключенню датчика до мікроконтролера, ознайомитись з роботою датчика та організації отримання значень з датчика.

Завдання 1.1: Реалізувати вимірювач опору потенціометру (використати лаб. роб. 2) з відображенням інформації на семисегментних індикаторах (реалізувати всі 3 варіанти, якщоможливо).

Посилання на проєкт: https://www.tinkercad.com/things/cLMfzwCRQeZ-lab-7-task-1-static

Код програми:

```
#define LATCH_PIN A4  // Пін для ST_CP (latch) зсувного регістру
#define CLOCK_PIN A5 // Пін для SH_CP (clock) зсувного регістру
#define DATA_PIN A3  // Пін для DS (data) зсувного регістру
#define POT_PIN A0 // Пін потенціометра


const unsigned char segdisp[12] = {
    0b00111111,
    0b00000110,
    0b01011011,
    0b01001111,
    0b01100110,
    0b01101101,
    0b01111101,
    0b00100111,
    0b01111111,
    0b01101111,
    0b01000000,
    0b00000000,
};

int NumDigitsInNumber(int number)
{
    int digitCount = 0;
    while (number > 0) {
        number /= 10;
```

```
        digitCount++;
    }
    return digitCount;
}

void ReverseArray(int arr[], int size) {
    int start = 0;
    int end = size - 1;

    while (start < end) {
        // Обмін значень між початковим та кінцевим індексами
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;

        // Пересуваємо початковий та кінцевий індекси
        start++;
        end--;
    }
}

void SplitInteger(int number, int* digits, int& digitsSize)
{

  // Визначаємо кількість цифр у числі
  int digitCount = NumDigitsInNumber(number);

  // Якщо число дорівнює 0
  if (number == 0) {
    digits[0] = 0;
    digitsSize = 1;
    return;
  }

  // Ініціалізуємо масив та його розмір
  int* tempArray = new int[digitCount];
  digitsSize = digitCount;

  // Розбиваємо число на цифри та зберігаємо у масиві
  for (int i = digitCount - 1; i >= 0; i--) {
    tempArray[i] = number % 10;
    number /= 10;
  }
```

```
  // Копіюємо дані у вихідний масив
  for (int i = 0; i < digitCount; i++) {
    digits[i] = tempArray[i];
  }

  // Звільняємо виділену пам'ять
  delete[] tempArray;
}

void LcdOutSegment(int number, int sizeLCD)
{
  // Перевірка на відємність числа
  bool isNegative = (number < 0);

  // Пертворення в додатнє число
  number = abs(number);

  // Отримуємо масив цифр розбитого цілого числа та його розмір
  int digitsCount = NumDigitsInNumber(number);

  int digitsSize;
  int digits[digitsCount];

  SplitInteger(number, digits, digitsSize);
  ReverseArray(digits, digitsSize);

  // Формуємо масив цифр та символів для дисплею
  int segmentDigits[sizeLCD];

  for (int i = 0; i < sizeLCD; i++) {
    if(i > digitsSize - 1){
      if(isNegative){
        segmentDigits[i] = segdisp[10];
        isNegative = false;
      }else{
        segmentDigits[i] = segdisp[11];
      }
    }else{
       segmentDigits[i] = segdisp[digits[i]];
    }
  }

  // Відображаємо на дисплеї
  digitalWrite(LATCH_PIN, LOW);
```

```
 for(int i = 0; i < sizeLCD; i++)
 {
  shiftOut(DATA_PIN, CLOCK_PIN, MSBFIRST, segmentDigits[i]);
 }

 digitalWrite(LATCH_PIN, HIGH);
}



// Напруга Ардуіно (вольти)
const int VOLTAGE_REFERENCE = 5.0;

// Роздільна здатність АЦП
const int ADC_RESOLUTION = 1023;

void setup()
{
  Serial.begin(9600);
  pinMode(LATCH_PIN, OUTPUT);
  pinMode(CLOCK_PIN, OUTPUT);
  pinMode(DATA_PIN, OUTPUT);
}

void loop()
{
  // Зчитуємо значення потенціометра
  int potVal = map(analogRead(POT_PIN), 0, 1023, 1023, 0);

   // Конвертація значень АЦП у напругу
  float voltage = (float)potVal / ADC_RESOLUTION *
VOLTAGE_REFERENCE;

  // Вихідний опір змінного резистора дорівнює 10 кОм
  float variableResistorResistance = 10000;

  // Розрахунок опору на виході змінного резистора
  float outputResistance = (voltage * variableResistorResistance) /
VOLTAGE_REFERENCE;

  // Відображаємо на дисплеї
  LcdOutSegment(outputResistance, 5);
  delay(1000);
}
```

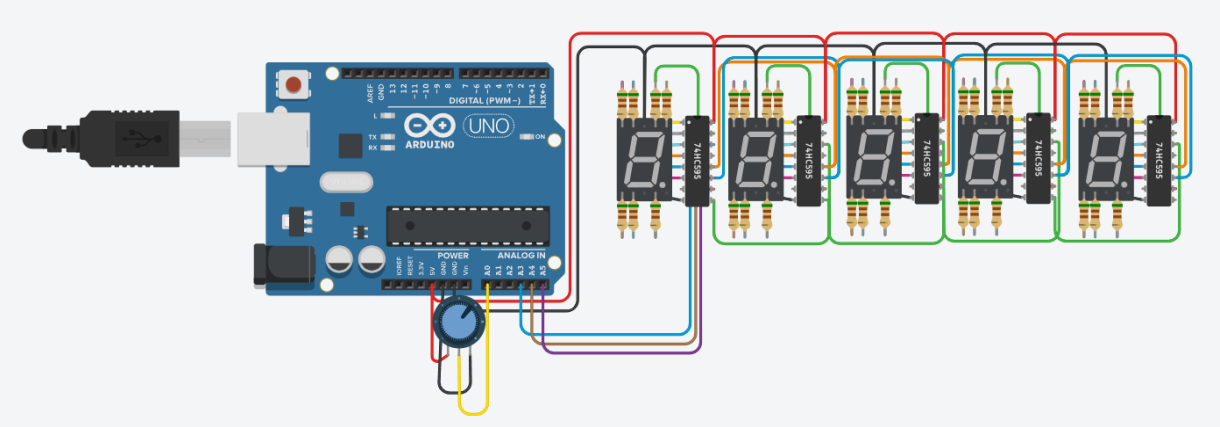

Рис. 1.1.1 Схема

Завдання 1.2

Посилання на проєкт: https://www.tinkercad.com/things/fOplNQtLaNo-lab-7-task-1-dynamic

Код програми:

```
#define POT_PIN A0 // Пін потенціометра


const unsigned char segdisp[12] = {
    0b00111111,
    0b00000110,
    0b01011011,
    0b01001111,
    0b01100110,
    0b01101101,
    0b01111101,
    0b00100111,
    0b01111111,
    0b01101111,
    0b01000000,
    0b00000000,
};

int NumDigitsInNumber(int number) {
   int digitCount = 0;
   while (number > 0) {
     number /= 10;
     digitCount++;
   }
   return digitCount;
}
```

```
void ReverseArray(int arr[], int size) {
    int start = 0;
    int end = size - 1;

    while (start < end) {
        // Обмін значень між початковим та кінцевим індексами
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;

        // Пересуваємо початковий та кінцевий індекси
        start++;
        end--;
    }
}

void SplitInteger(int number, int* digits, int& digitsSize)
{

  // Визначаємо кількість цифр у числі
  int digitCount = NumDigitsInNumber(number);

  // Якщо число дорівнює 0
  if (number == 0) {
    digits[0] = 0;
    digitsSize = 1;
    return;
  }

  // Ініціалізуємо масив та його розмір
  int* tempArray = new int[digitCount];
  digitsSize = digitCount;

  // Розбиваємо число на цифри та зберігаємо у масиві
  for (int i = digitCount - 1; i >= 0; i--) {
    tempArray[i] = number % 10;
    number /= 10;
  }

  // Копіюємо дані у вихідний масив
  for (int i = 0; i < digitCount; i++) {
    digits[i] = tempArray[i];
  }
```

```
  // Звільняємо виділену пам'ять
  delete[] tempArray;
}

void displaySegment(const int* segmentPins, int numSeg, byte num){
  // Створення масиву для зберігання стану кожного сегмента
    bool segments[7];

    // Заповнення масиву значеннями бітів з байта
    for (int i = 0; i < numSeg; i++) {
      segments[i] = !((num >> i) & 1);
    }

    for (int i = 0; i < numSeg; i++) {
      digitalWrite(segmentPins[i], segments[i]);
    }
}

void displayInteger(int number, const int* segmentPins, const int* analogPins, int
numSeg, int numA) {
    // Перевірка на від'ємність числа
    bool isNegative = (number < 0);

    // Перетворення в додатнє число
    number = abs(number);

    // Отримуємо масив цифр розбитого цілого числа та його розмір
    int digitsCount = NumDigitsInNumber(number);

    int digitsSize;
    int digits[digitsCount];

    SplitInteger(number, digits, digitsSize);

    ReverseArray(digits, digitsSize);


    // Формуємо масив цифр та символів для дисплею
    int segmentDigits[numA];

    for (int i = 0; i < numA; i++) {
      if(i > digitsSize - 1){
        if(isNegative){
```

```
          segmentDigits[i] = segdisp[10];
          isNegative = false;
        }else{
          segmentDigits[i] = segdisp[11];
        }
      }else{
        segmentDigits[i] = segdisp[digits[i]];
      }
    }
    ReverseArray(segmentDigits, numA);


    // Відображаємо на дисплеї

    for (int i = 0; i < numA; i++) {
      delay(3.5);
      digitalWrite(analogPins[i], HIGH);
      displaySegment(segmentPins, numSeg, segmentDigits[i]);
      delay(3.5);
      digitalWrite(analogPins[i], LOW);
    }

}



// Напруга Ардуіно (вольти)
const int VOLTAGE_REFERENCE = 5.0;

// Роздільна здатність АЦП
const int ADC_RESOLUTION = 1023;

// Піни сегмента
const int segmentPins[] = {2, 3, 4, 5, 6, 7, 8};

// Аналогові піни
const int analogPins[] = {A1, A2, A3, A4, A5};

void disableSegment(const int* analogPins, int numA) {
    for (int i = 0; i < numA; i++) {
      digitalWrite(analogPins[i], HIGH);
    }
}
```

```
void setup()
{
  Serial.begin(9600);
  // Ініціалізація пінів сегментів як виводів OUTPUT
  for (int i = 0; i < 7; i++) {
     pinMode(segmentPins[i], OUTPUT);
  }

  // Ініціалізація аналогових пінів як виводів OUTPUT
  for (int i = 0; i < 5; i++) {
     pinMode(analogPins[i], OUTPUT);
  }

  disableSegment(analogPins, 5);
}

void loop()
{
  // Зчитуємо значення потенціометра + інверсія
  int potVal = map(analogRead(POT_PIN), 0, 1023, 1023, 0);

   // Конвертація значень АЦП у напругу
  float voltage = (float)potVal / ADC_RESOLUTION *
VOLTAGE_REFERENCE;

  // Вихідний опір змінного резистора дорівнює 10 кОм
  float variableResistorResistance = 10000;

  // Розрахунок опору на виході змінного резистора
  float outputResistance = (voltage * variableResistorResistance) /
VOLTAGE_REFERENCE;

  displayInteger(outputResistance, segmentPins, analogPins, 7, 5);
}
```

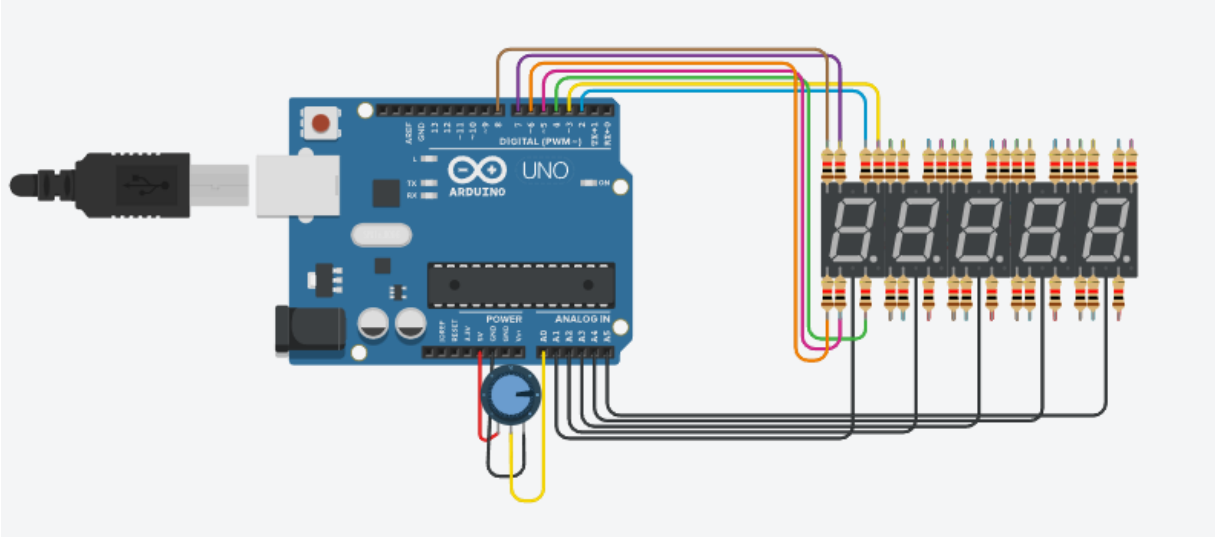

Рис. 1.2.1 Схема

Завдання 2.1: Реалізувати вимірювач температури (використати лаб. роб. 2) з відображенням інформації на семисегментних індикаторах (реалізувати всі 3 варіанти, якщо можливо).

Посилання на проєкт: https://www.tinkercad.com/things/3DfmSXvPMzw-lab-7-task-2-static

Код програми:

```
#define LATCH_PIN A4  // Пін для ST_CP (latch) зсувного регістру
#define CLOCK_PIN A5 // Пін для SH_CP (clock) зсувного регістру
#define DATA_PIN A3  // Пін для DS (data) зсувного регістру
#define TEMP_PIN A0 // Пін датчика температури


const unsigned char segdisp[12] = {
    0b00111111,
    0b00000110,
    0b01011011,
    0b01001111,
    0b01100110,
    0b01101101,
    0b01111101,
    0b00100111,
    0b01111111,
    0b01101111,
    0b01000000,
    0b00000000,
};
```

```
int NumDigitsInNumber(int number)
{
  int digitCount = 0;
    while (number > 0) {
      number /= 10;
      digitCount++;
    }
    return digitCount;
}

void ReverseArray(int arr[], int size) {
    int start = 0;
    int end = size - 1;

    while (start < end) {
        // Обмін значень між початковим та кінцевим індексами
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;

        // Пересуваємо початковий та кінцевий індекси
        start++;
        end--;
    }
}

void SplitInteger(int number, int* digits, int& digitsSize)
{

  // Визначаємо кількість цифр у числі
  int digitCount = NumDigitsInNumber(number);

  // Якщо число дорівнює 0
  if (number == 0) {
    digits[0] = 0;
    digitsSize = 1;
    return;
  }

  // Ініціалізуємо масив та його розмір
  int* tempArray = new int[digitCount];
  digitsSize = digitCount;

  // Розбиваємо число на цифри та зберігаємо у масиві
```

```
  for (int i = digitCount - 1; i >= 0; i--) {
    tempArray[i] = number % 10;
    number /= 10;
  }

  // Копіюємо дані у вихідний масив
  for (int i = 0; i < digitCount; i++) {
    digits[i] = tempArray[i];
  }

  // Звільняємо виділену пам'ять
  delete[] tempArray;
}


void LcdOutSegmentFloat(float number, int sizeLCD)
{
  // Перевірка на від'ємність числа
    bool isNegative = (number < 0);

    // Перетворення в додатнє число
    number = abs(number);

    // Перетворюємо дробове число на ціле
    int integer = round(number * 100);

  // Шукаємо позицію крапки
    int indexDot = String(number).substring(String(number).indexOf('.')).length() -
1;

    // Отримуємо масив цифр розбитого цілого числа та його розмір
    int digitsCount = NumDigitsInNumber(integer);

    int digitsSize;
    int digits[digitsCount];

    SplitInteger(integer, digits, digitsSize);
    ReverseArray(digits, digitsSize);

    // Формуємо масив цифр та символів для дисплею
    int segmentDigits[sizeLCD];

    for (int i = 0; i < sizeLCD; i++) {
      if(i > digitsSize - 1){
```

```
        if(isNegative){
          segmentDigits[i] = segdisp[10];
          isNegative = false;
        }else{
          segmentDigits[i] = segdisp[11];
        }
      }else{
        if(indexDot == i){
          segmentDigits[i] = segdisp[digits[i]] ^ (1 << 7);
          continue;
        }
        segmentDigits[i] = segdisp[digits[i]];
      }
    }

    // Відображаємо на дисплеї
    digitalWrite(LATCH_PIN, LOW);
    for(int i = 0; i < sizeLCD; i++)
    {
      shiftOut(DATA_PIN, CLOCK_PIN, MSBFIRST, segmentDigits[i]);
    }

    digitalWrite(LATCH_PIN, HIGH);
}

// Напруга в резисторі (вольти)
const float VOLTAGE_REFERENCE = 5.0;

// Роздільна здатність АЦП
const int ADC_RESOLUTION = 1023;

// Масштаб для отримання температури в градусах Цельсія
const float TEMP_SCALE = 100.0;

void setup()
{
  Serial.begin(9600);
  pinMode(LATCH_PIN, OUTPUT);
  pinMode(CLOCK_PIN, OUTPUT);
  pinMode(DATA_PIN, OUTPUT);
}

void loop()
{
```

```
        // Зчитуємо значення з датчика температури
        int tempVal = analogRead(TEMP_PIN);

        // Конвертація значень АЦП у напругу
        float voltage = (float)tempVal / ADC_RESOLUTION *
VOLTAGE_REFERENCE;

        // Розрахунок температури датчика TMP36
        float temperature = (voltage - 0.5) * TEMP_SCALE;

        // Відображаємо на дисплеї
        LcdOutSegmentFloat(temperature, 5);
        delay(1000);
      }
```

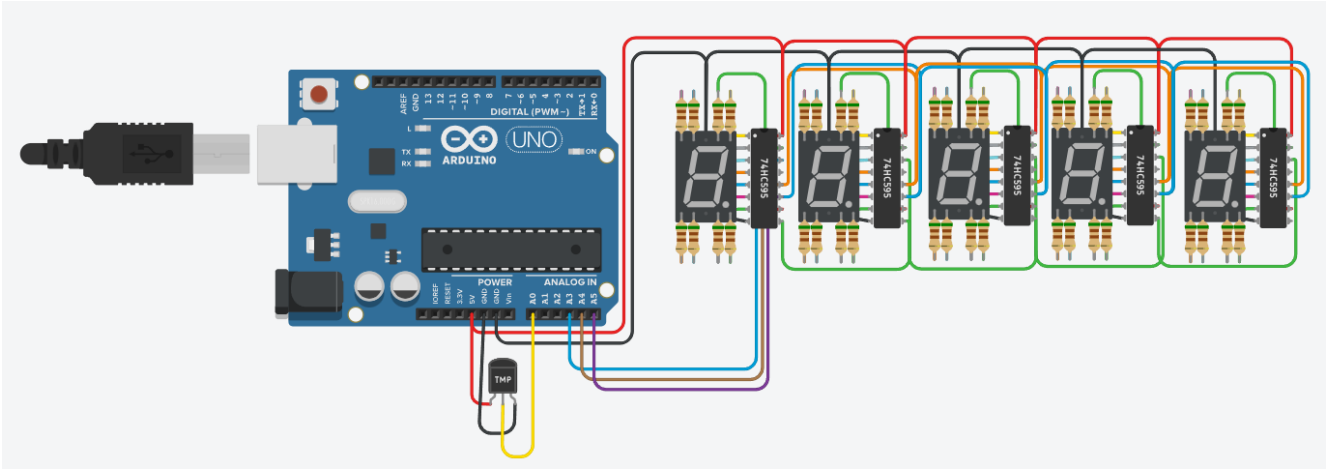

Рис. 2.1 Схема

Завдання 2.2

Посилання на проєкт: https://www.tinkercad.com/things/cOUQ5C0so84-lab-7-task-2-dynamic-2

Код програми:

```
#define TEMP_PIN A0 // Пін датчика температури


const unsigned char segdisp[12] = {
    0b00111111,
    0b00000110,
    0b01011011,
    0b01001111,
    0b01100110,
    0b01101101,
```

```
    0b01111101,
    0b00100111,
    0b01111111,
    0b01101111,
    0b01000000,
    0b00000000,
};

int NumDigitsInNumber(int number) {
  int digitCount = 0;
  while (number > 0) {
    number /= 10;
    digitCount++;
  }
  return digitCount;
}

void ReverseArray(int arr[], int size) {
  int start = 0;
  int end = size - 1;

  while (start < end) {
    // Обмін значень між початковим та кінцевим індексами
    int temp = arr[start];
    arr[start] = arr[end];
    arr[end] = temp;

    // Пересуваємо початковий та кінцевий індекси
    start++;
    end--;
  }
}

void SplitInteger(int number, int* digits, int& digitsSize)
{

  // Визначаємо кількість цифр у числі
  int digitCount = NumDigitsInNumber(number);

  // Якщо число дорівнює 0
  if (number == 0) {
    digits[0] = 0;
    digitsSize = 1;
    return;
```

```
    }

    // Ініціалізуємо масив та його розмір
    int* tempArray = new int[digitCount];
    digitsSize = digitCount;

    // Розбиваємо число на цифри та зберігаємо у масиві
    for (int i = digitCount - 1; i >= 0; i--) {
      tempArray[i] = number % 10;
      number /= 10;
    }

    // Копіюємо дані у вихідний масив
    for (int i = 0; i < digitCount; i++) {
      digits[i] = tempArray[i];
    }

    // Звільняємо виділену пам'ять
    delete[] tempArray;
  }

  void displaySegment(const int* segmentPins, int numSeg, byte num){
    // Створення масиву для зберігання стану кожного сегмента
      bool segments[7];

      // Заповнення масиву значеннями бітів з байта
      for (int i = 0; i < numSeg; i++) {
        segments[i] = !((num >> i) & 1);
      }

      // Тепер ви можете використовувати масив segments для виведення на семисегментний індикатор
      for (int i = 0; i < numSeg; i++) {
        digitalWrite(segmentPins[i], segments[i]);
      }
  }


  void displayFloat(float number, const int* segmentPins, const int* analogPins, int numSeg, int numA) {
    // Перевірка на від'ємність числа
      bool isNegative = (number < 0);
```

```
  // Перетворення в додатнє число
  number = abs(number);

  // Перетворюємо дробове число на ціле
  int integer = round(number * 100);

 // Шукаємо позицію крапки
  int indexDot = String(number).substring(String(number).indexOf('.')).length() -
1;

  // Отримуємо масив цифр розбитого цілого числа та його розмір
  int digitsCount = NumDigitsInNumber(integer);

  int digitsSize;
  int digits[digitsCount];

  SplitInteger(integer, digits, digitsSize);

  ReverseArray(digits, digitsSize);


  // Формуємо масив цифр та символів для дисплею
  int segmentDigits[numA];

  for (int i = 0; i < numA; i++) {
    if(i > digitsSize - 1){
      if(isNegative){
        segmentDigits[i] = segdisp[10];
        isNegative = false;
      }else{
        segmentDigits[i] = segdisp[11];
      }
    }else{
       segmentDigits[i] = segdisp[digits[i]];
    }
  }
  ReverseArray(segmentDigits, numA);

  // Відображаємо на дисплеї

  for (int i = 0; i < numA; i++) {
    delay(3.5);
    digitalWrite(analogPins[i], HIGH);
    if (i == indexDot) {
```

```
            digitalWrite(9, LOW);
        }else{
                digitalWrite(9, HIGH);
        }
        displaySegment(segmentPins, numSeg, segmentDigits[i]);
        delay(3.5);
        digitalWrite(analogPins[i], LOW);
      }

  }


  // Напруга в резисторі (вольти)
  const float VOLTAGE_REFERENCE = 5.0;

  // Роздільна здатність АЦП
  const int ADC_RESOLUTION = 1023;

  // Масштаб для отримання температури в градусах Цельсія
  const float TEMP_SCALE = 100.0;


  // Піни сегмента
  const int segmentPins[] = {2, 3, 4, 5, 6, 7, 8};

  // Аналогові піни
  const int analogPins[] = {A1, A2, A3, A4, A5};

  void disableSegment(const int* analogPins, const int* segmentPins, int numSeg,
int numA) {
      for (int i = 0; i < numA; i++) {
        digitalWrite(analogPins[i], HIGH);
      }
      for (int i = 0; i < numSeg; i++) {
        digitalWrite(segmentPins[i], HIGH);
      }
  }


  void setup()
  {
    Serial.begin(9600);
    // Ініціалізація пінів сегментів як виводів OUTPUT
    for (int i = 0; i < 7; i++) {
```

```
    pinMode(segmentPins[i], OUTPUT);
  }

  // Ініціалізація аналогових пінів як виводів OUTPUT
  for (int i = 0; i < 5; i++) {
    pinMode(analogPins[i], OUTPUT);
  }

  disableSegment(analogPins, segmentPins, 7, 5);

  // Dot
  pinMode(9, OUTPUT);
  digitalWrite(9, HIGH);
}

void loop()
{
  // Зчитуємо значення з датчика температури
  int tempVal = analogRead(TEMP_PIN);

  // Конвертація значень АЦП у напругу
  float voltage = (float)tempVal / ADC_RESOLUTION *
VOLTAGE_REFERENCE;

  // Розрахунок температури датчика TMP36
  float temperature = (voltage - 0.5) * TEMP_SCALE;

  displayFloat(temperature, segmentPins, analogPins, 7, 5);
}
```

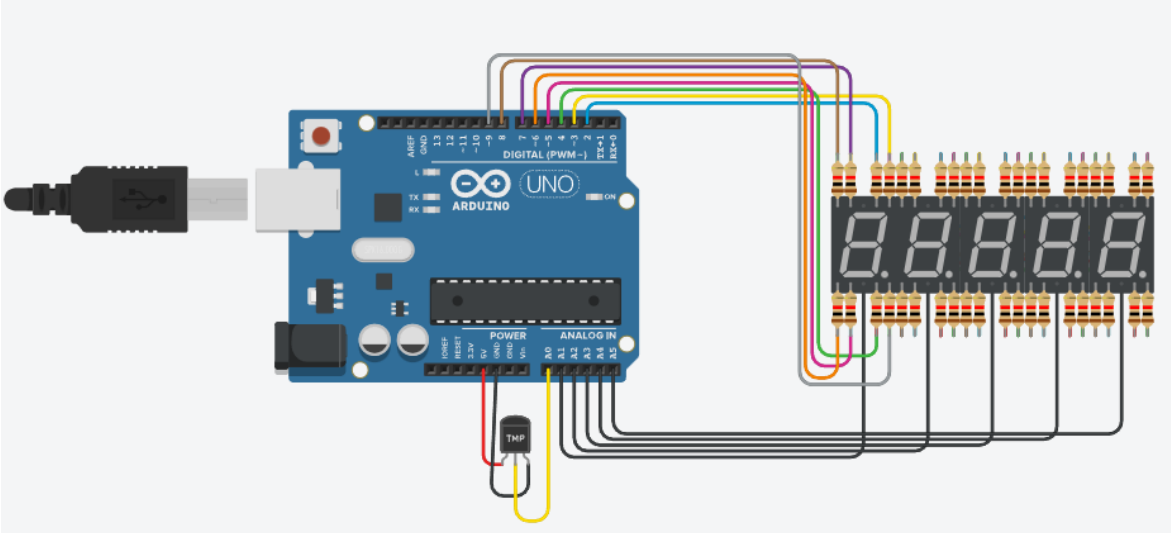

Рис. 2.2 Схема

Завдання 3: Реалізувати систему керування світлом (використати світлодіод) з відображенням порогу спрацювання на семисегментних індикаторах (використати любий вариант з 3х). Тільки для тих, хто хоче оцінку відмінно.

Посилання на проєкт: https://www.tinkercad.com/things/1gunzGAcY9u-lab-7-task-3-static

Код програми:

```
#define LATCH_PIN A4  // Пін для ST_CP (latch) зсувного регістру
#define CLOCK_PIN A5 // Пін для SH_CP (clock) зсувного регістру
#define DATA_PIN A3  // Пін для DS (data) зсувного регістру
#define PT_PIN A0 // Пін фоторезистора
#define LED_PIN 13 // Пін світлодіода

const unsigned char segdisp[12] = {
    0b00111111,
    0b00000110,
    0b01011011,
    0b01001111,
    0b01100110,
    0b01101101,
    0b01111101,
    0b00100111,
    0b01111111,
    0b01101111,
    0b01000000,
    0b00000000,
};

// Поріг спрацювання
const int threshold = 500;

int NumDigitsInNumber(int number)
{
  int digitCount = 0;
    while (number > 0) {
      number /= 10;
      digitCount++;
    }
    return digitCount;
}

void ReverseArray(int arr[], int size) {
```

```
    int start = 0;
    int end = size - 1;

    while (start < end) {
        // Обмін значень між початковим та кінцевим індексами
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;

        // Пересуваємо початковий та кінцевий індекси
        start++;
        end--;
    }
}

void SplitInteger(int number, int* digits, int& digitsSize)
{

  // Визначаємо кількість цифр у числі
  int digitCount = NumDigitsInNumber(number);

  // Якщо число дорівнює 0
  if (number == 0) {
    digits[0] = 0;
    digitsSize = 1;
    return;
  }

  // Ініціалізуємо масив та його розмір
  int* tempArray = new int[digitCount];
  digitsSize = digitCount;

  // Розбиваємо число на цифри та зберігаємо у масиві
  for (int i = digitCount - 1; i >= 0; i--) {
    tempArray[i] = number % 10;
    number /= 10;
  }

  // Копіюємо дані у вихідний масив
  for (int i = 0; i < digitCount; i++) {
    digits[i] = tempArray[i];
  }

  // Звільняємо виділену пам'ять
```

```
  delete[] tempArray;
}

void LcdOutSegment(int number, int sizeLCD)
{
 // Перевірка на відємність числа
 bool isNegative = (number < 0);
 // Пертворення в додатнє число
 number = abs(number);

 // Отримуємо масив цифр розбитого цілого числа та його розмір

 int digitsCount = NumDigitsInNumber(number);

 int digitsSize;
 int digits[digitsCount];

 SplitInteger(number, digits, digitsSize);
 ReverseArray(digits, digitsSize);

 // Формуємо масив цифр та символів для дисплею
 int segmentDigits[sizeLCD];

 for (int i = 0; i < sizeLCD; i++) {
   if(i > digitsSize - 1){
     if(isNegative){
       segmentDigits[i] = segdisp[10];
       isNegative = false;
     }else{
       segmentDigits[i] = segdisp[11];
     }
   }else{
      segmentDigits[i] = segdisp[digits[i]];
   }
 }

 // Відображаємо на дисплеї
 digitalWrite(LATCH_PIN, LOW);
 for(int i = 0; i < sizeLCD; i++)
 {
   shiftOut(DATA_PIN, CLOCK_PIN, MSBFIRST, segmentDigits[i]);
 }

 digitalWrite(LATCH_PIN, HIGH);
```

```
}

void setup()
{
  Serial.begin(9600);
  pinMode(LATCH_PIN, OUTPUT);
  pinMode(CLOCK_PIN, OUTPUT);
  pinMode(DATA_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
}

void loop()
{
  // Зчитуємо значення з фоторезистора
  int ptVal = analogRead(PT_PIN);

  if (ptVal >= threshold){
  digitalWrite(LED_PIN, HIGH);
  }else{
  digitalWrite(LED_PIN, LOW);
  }

  // Відображаємо на дисплеї поріг спацювання
  LcdOutSegment(threshold, 3);
  delay(1000);

}
```

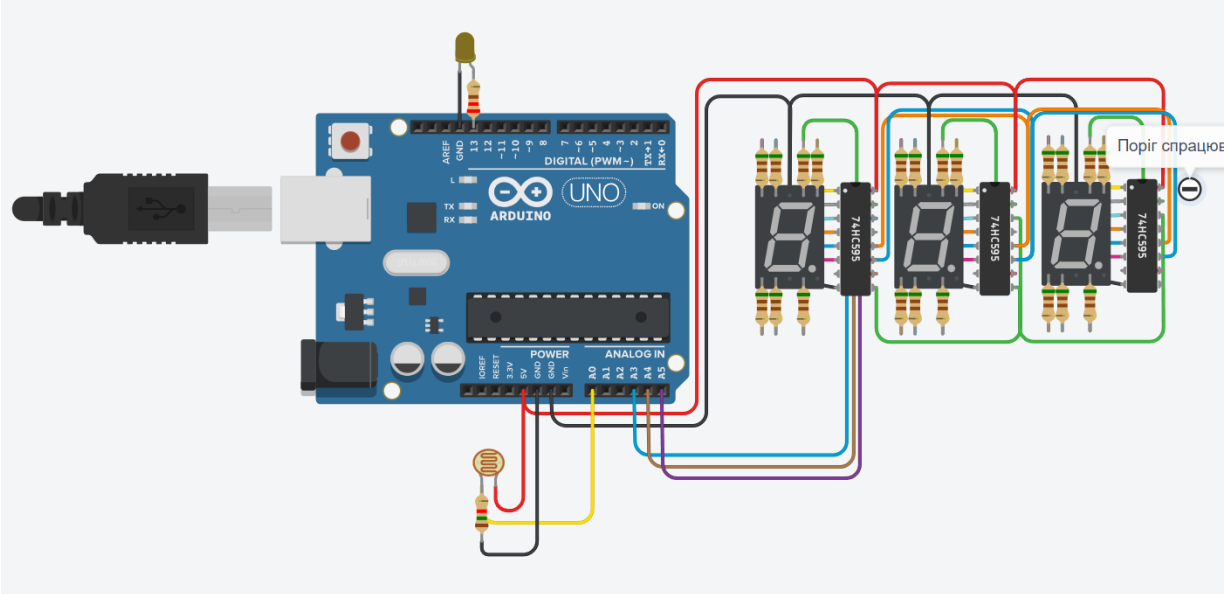

Рис. 3.1 Схема

Висновок: В результаті виконання лабораторної роботи було ознайомлено з принципами роботи семисегментних світлодіодних індикаторів та отримано практичні навички по підключенню їх до мікроконтролера; ознайомлено з нарощуванням кількості індикаторів