Тестове завдання

На довільному мікроконтролері написати програму вольтметра, яка вимірює напругу в межах від 0 до 10 вольт з похибкою в 1% або менше. Вивід можна зробити на 7-сегментний дисплей або через порт на комп'ютері.

У пристрої має бути роз'єм живлення і роз'єм для вимірювання напруги. Також потрібно зробити регульоване джерело напруги регульоване, щоб можна було потестити пристрій.

Буде +, якщо зробити наступне:

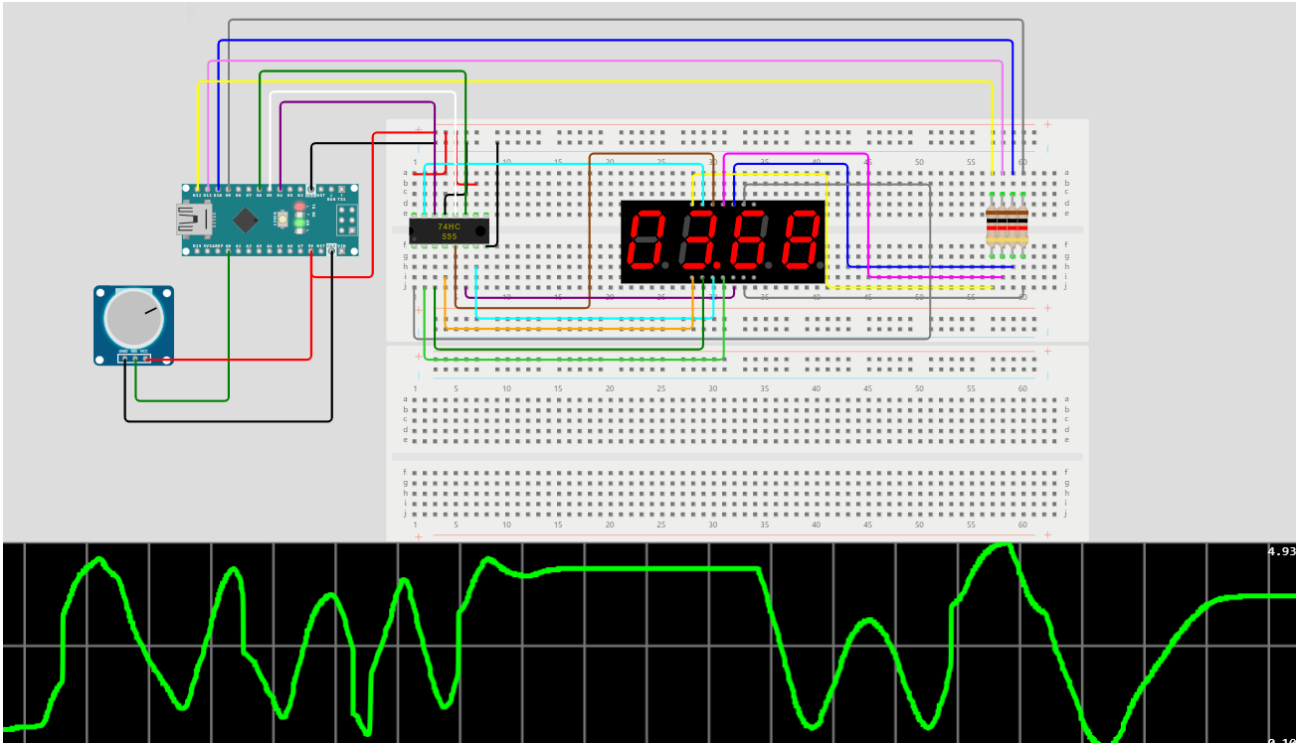
1) Корпус пристрою

2) Вимірювання зворотної напруги

3) Якщо реалізовано виведення на ПК, то зробити GUI (в ідеалі на Python), у якому можна буде бачити це значення.

Реалізація:

Приклад ARDUINO NANO з 1 зсувним регістром і 4-розрядним 7-сегментним індикатором



Приклад модуляції, для динамічного опору використовувався потенціаметр.

Онлайн-модель:

https://wokwi.com/projects/414119755329588225

```
// Схема розроблена для моделювання роботи пристрою
// При моделюванні використано лише живлення 5В


int digits[4]; // Кількість Індикаторів
int CAS[4] = {12, 11, 10, 9}; // Піни dig індекаторів
int count = 0; // Пін A0 для зчитування
int clk = 6;       //
int latch = 5;     // Піни підключення до 74hc595
int data = 4;      //

// байти цифр
byte numbers[10] = {
  B00000011,
  B10011111,
  B00100101,
  B00001101,
  B10011001,
  B01001001,
  B01000001,
  B00011111,
  B00000001,
```

```
  B00001001
};

// байти цифр з крапкою
byte fnumbers[10] = {
  B00000010,
  B10011110,
  B00100100,
  B00001100,
  B10011000,
  B01001000,
  B01000000,
  B00011110,
  B00000000,
  B00001000
};


#define VREF 5.0
//#define DIV_R1 1000  Значення макс. робочої напруги і резисторів дільника напруги
//#define DIV_R2 1000
//float voltageKoff = VREF * ((DIV_R1 + DIV_R2) / DIV_R2) / 1024;
float voltageKoff = VREF/1024;
float voltage;
float filterVoltage;


void setup() {
  Serial.begin(9600);
  for (int i = 0; i < 4; i++) {
    pinMode(CAS[i], OUTPUT);
    digitalWrite(CAS[i], LOW);
  }
  pinMode(clk, OUTPUT);
  pinMode(latch, OUTPUT);
  pinMode(data, OUTPUT);

}

void loop() {
  int analogValue = analogRead(A0);

  voltage = voltageKoff*(float)analogValue; // Значення напруги
  filterVoltage = expRunningAverageAdaptive(voltage); // Значення напуги з фільтром
  Serial.println(filterVoltage);


  break_number(filterVoltage);
    display_number();
}

// Розбиття числа на розряди 2 цілих і 2 дробових
void break_number(float num) {

  int integer_part = (int)num;
  int decimal_part = (int)((num - integer_part) * 100);
```

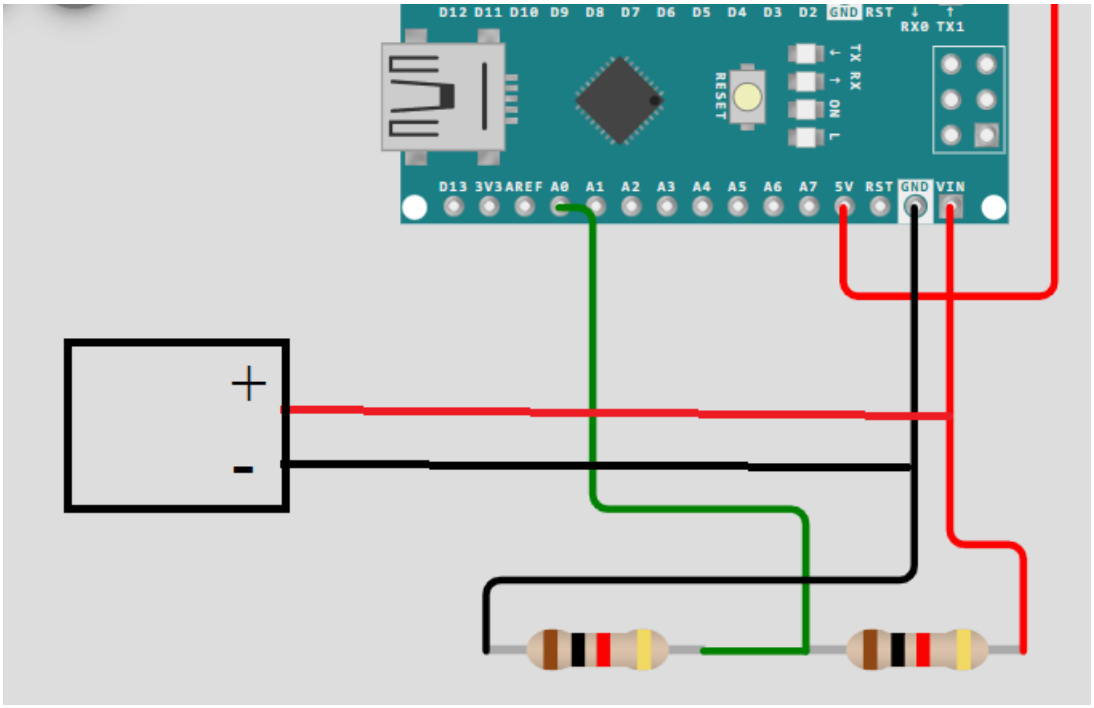
```
  // 2 цифри цілої частини:
  digits[0] = (integer_part / 10) % 10;
  digits[1] = integer_part % 10;

  // 2 цифри дробової частини:
  digits[2] = (decimal_part / 10) % 10;
  digits[3] = decimal_part % 10;
}

// Вивід числа на сегмент
void display_number() {
  digitalWrite(latch, LOW);
  if(count == 1) shiftOut(data, clk, LSBFIRST, fnumbers[digits[count]]);
  else shiftOut(data, clk, LSBFIRST, numbers[digits[count]]);

  digitalWrite(latch, HIGH);
  digitalWrite(CAS[count], HIGH);
  delay(5);
  digitalWrite(CAS[count], LOW);
  count = (count + 1) % 4;
}

// фільтр біжуче середнє з адаптивним коефіцієнтом
// Не впевнений, чи потрібно було додавати, залежить від умов використання
float expRunningAverageAdaptive(float newVal) {
  static float filVal = 0;
  float k;
  // резкость фильтра зависит от модуля разности значений
  if (abs(newVal - filVal) > 1.5) k = 0.9;
  else k = 0.03;

  filVal += (newVal - filVal) * k;
  return filVal;
}
```

Для збільшення максимальної вимірювальної напруги, додаємо просто дільник напруги (мікроконтролер має заздалегіть знати значення регістрів, які використовуємо).

```
int digits[4]; // Кількість Індикаторів
int CAS[4] = {12, 11, 10, 9}; // Піни dig індекаторів
int count = 0; // Пін A0 для зчитування
int clk = 6;       //
int latch = 5;     // Піни підключення до 74hc595
int data = 4;      //

// байти цифр
byte numbers[10] = {
  B00000011,
  B10011111,
  B00100101,
  B00001101,
  B10011001,
```

```
  B01001001,
  B01000001,
  B00011111,
  B00000001,
  B00001001
};

// байти цифр з крапкою
byte fnumbers[10] = {
  B00000010,
  B10011110,
  B00100100,
  B00001100,
  B10011000,
  B01001000,
  B01000000,
  B00011110,
  B00000000,
  B00001000
};


#define VREF 5.0
#define DIV_R1 1000  //Значення макс. робочої напруги і резисторів дільника
напруги
#define DIV_R2 1000
float voltageKoff = VREF * ((DIV_R1 + DIV_R2) / DIV_R2) / 1024;
float voltage;
float filterVoltage;


void setup() {
  Serial.begin(9600);
  for (int i = 0; i < 4; i++) {
    pinMode(CAS[i], OUTPUT);
    digitalWrite(CAS[i], LOW);
  }
  pinMode(clk, OUTPUT);
  pinMode(latch, OUTPUT);
  pinMode(data, OUTPUT);

}

void loop() {
  int analogValue = analogRead(A0);

  voltage = voltageKoff*(float)analogValue; // Значення напруги
  filterVoltage = expRunningAverageAdaptive(voltage); // Значення напуги з
фільтром
  Serial.println(filterVoltage);


  break_number(filterVoltage);
    display_number();
}

// Розбиття числа на розряди 2 цілих і 2 дробових
void break_number(float num) {
```

```
  int integer_part = (int)num;
  int decimal_part = (int)((num - integer_part) * 100);

  // 2 цифри цілої частини:
  digits[0] = (integer_part / 10) % 10;
  digits[1] = integer_part % 10;

  // 2 цифри дробової частини:
  digits[2] = (decimal_part / 10) % 10;
  digits[3] = decimal_part % 10;
}

// Вивід числа на сегмент
void display_number() {
  digitalWrite(latch, LOW);
  if(count == 1) shiftOut(data, clk, LSBFIRST, fnumbers[digits[count]]);
  else shiftOut(data, clk, LSBFIRST, numbers[digits[count]]);

  digitalWrite(latch, HIGH);
  digitalWrite(CAS[count], HIGH);
  delay(5);
  digitalWrite(CAS[count], LOW);
  count = (count + 1) % 4;
}

// фільтр біжуче середнє з адаптивним коефіцієнтом
// Не впевнений, чи потрібно було додавати, залежить від умов використання
float expRunningAverageAdaptive(float newVal) {
  static float filVal = 0;
  float k;
  // резкость фильтра зависит от модуля разности значений
  if (abs(newVal - filVal) > 1.5) k = 0.9;
  else k = 0.03;

  filVal += (newVal - filVal) * k;
  return filVal;
}
```



Приклад з Raspberry Pi Pico H і виводом на пк GUI для того, щоб показати свої можливості

Паралельно працюють 2 програми:

На Pico зчитується GP26 і передається через COM

```
import sys
import utime
from machine import ADC, Pin

adc = ADC(Pin(26))

VREF = 3.33
DIV_R1 = 2000
DIV_R2 = 1000

voltKoef = VREF * ((DIV_R1 + DIV_R2) / DIV_R2) / 655536

while True:
    # Відправка даних через Ком-порт

    adc_value = adc.read_u16()
    voltage = str(voltKoef*adc_value) + '\n'
    sys.stdout.write(voltage)
    utime.sleep(0.01)
```

У цей час комп'ютер слухає порт і будує графік

```
import serial
import matplotlib.pyplot as plt
```

```
import matplotlib.animation as animation

# Обираємо наш порт
port = "COM9"
baudrate = 9600
ser = serial.Serial(port, baudrate=baudrate)

# Параметры графика
max_voltage = 10.0
sample_rate = 100
time_window = 1
data_points = sample_rate * time_window  # Кількість точок на графіку

# Ініціалізація даних графіку
voltages = [0] * data_points  # Буфер значень
times = [i / sample_rate for i in range(-data_points, 0)]  # Часові мітки

# Настройка графика
fig, ax = plt.subplots()
line, = ax.plot(times, voltages)
ax.set_ylim(0, max_voltage)
ax.set_xlim(-time_window, 0)
ax.set_xlabel("Time (s)")
ax.set_ylabel("Voltage (V)")
ax.set_title("Real time graphic")


# Функція оновлення графіку
def update(frame):
    if ser.in_waiting > 0:
        try:
            ser.reset_input_buffer()
            voltage = float(ser.readline().decode().strip())*max_voltage

            voltage = max(0, min(voltage, max_voltage))

            voltages.pop(0)
            voltages.append(voltage)

            line.set_ydata(voltages)
        except ValueError:
            pass

    return line,


# Налаштування анімації з використанням blit для оптимізації
ani = animation.FuncAnimation(fig, update, interval=10, blit=True)

# Запуск графіку
plt.show()

ser.close()
```

Відео роботи test2

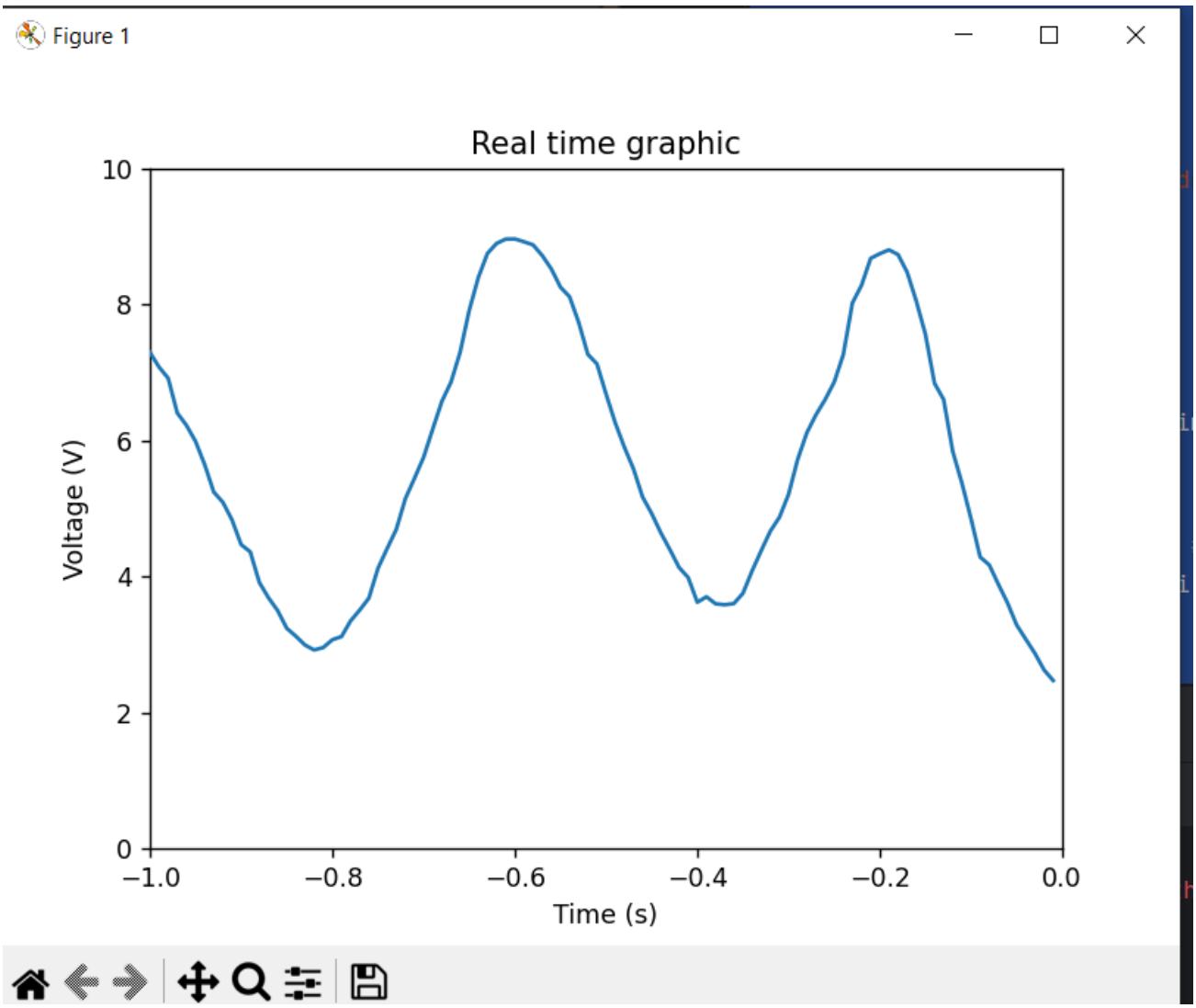


Figure 1
Real time graphic
Voltage (V)
Time (s)
10
8
6
4
2
0
−1.0
−0.8
−0.6
−0.4
−0.2
0.0