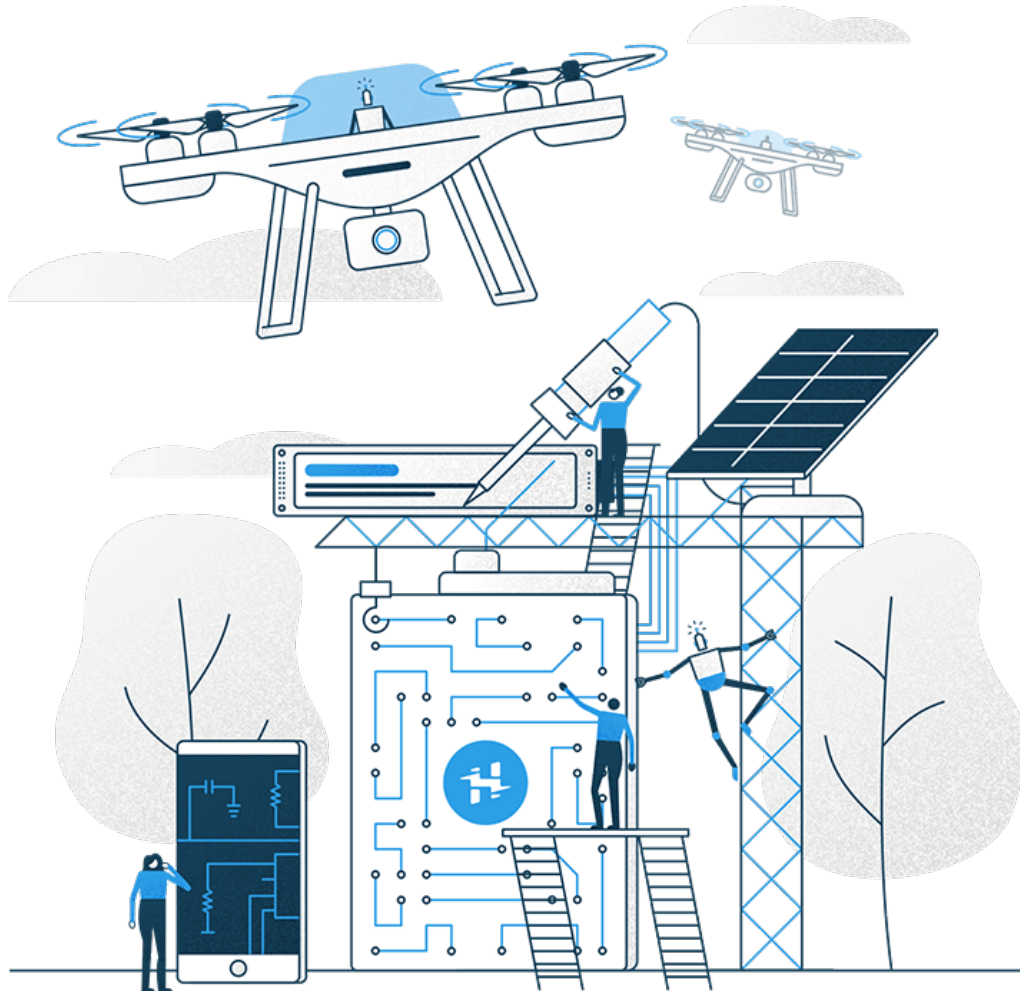


# МІКРОПРОЦЕСОРНІ СИСТЕМИ УПРАВЛІННЯ



Lesson 7

# Виконавчі пристрої

Практично жодна мікропроцесорна система управління не може обійтися без таких елементів, як виконавчі пристрої. Головне призначення будь-якої системи – це управління яким-небудь зовнішнім механізмом. Це можуть бути електродвигуни, нагрівачі, електромагнітні клапани. Тому, окрім датчиків, кнопок управління і елементів індикації до мікроконтролера обов'язково доведеться підключати і виконавчі пристрої. Для управління зовнішніми пристроями використовуються ті ж самі порти введення/виведення МК, які працюють на виведення. Сигнали з будь-якою з ліній будь-якого порту легко можуть бути використані для включення і виключення зовнішнього пристрою. Необхідно лише підсилити керуючий сигнал за потужністю до необхідного рівня. Для цього застосовуються різні схеми узгодження. Вибір схеми залежить від типу виконавчого пристрою.

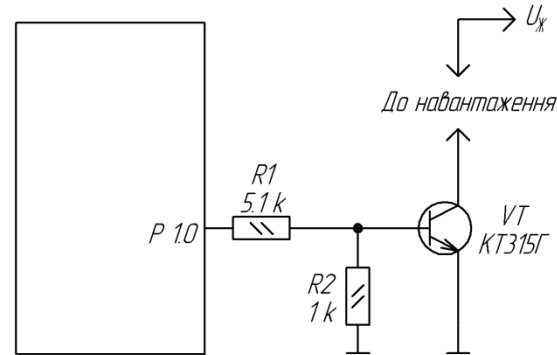
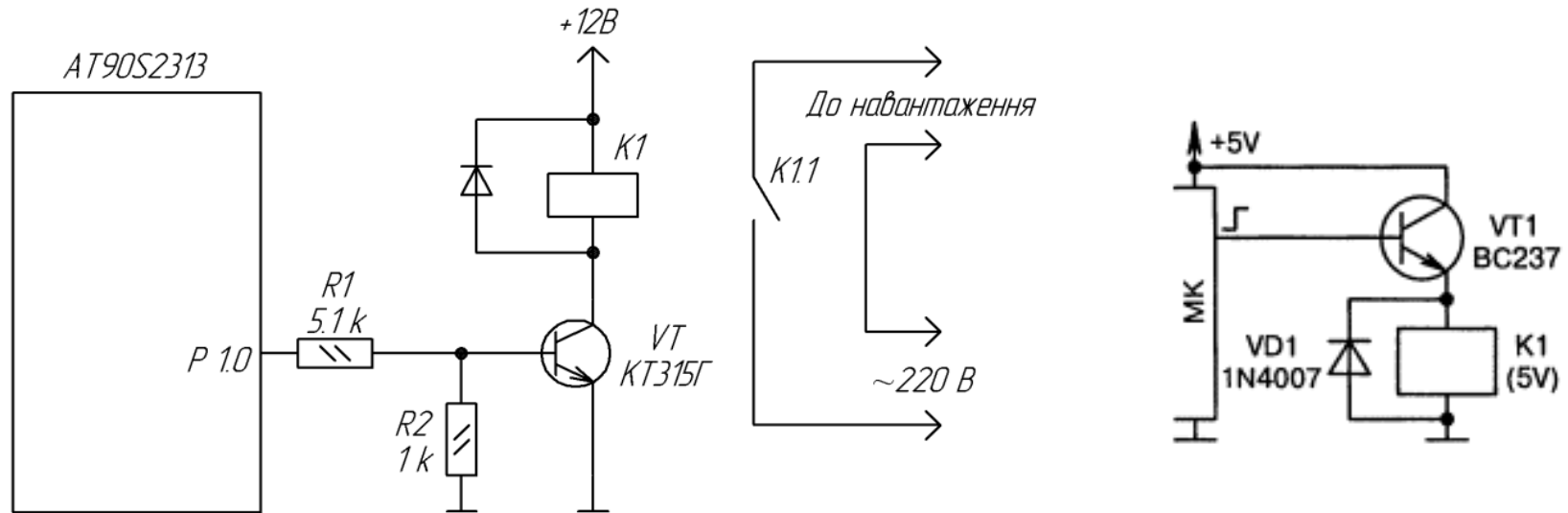


Рис. 1

У найпростішому випадку можна застосувати транзисторний ключ (рис 1). При використанні транзистора BC547 можна керувати зовнішніми колами із струмом споживання до 100 мА і напругою  $U_K$  до 15 В. Транзистор допускає також високу напругу, проте підвищення напруги можливе при зменшенні струму.

# Виконавчі пристрої

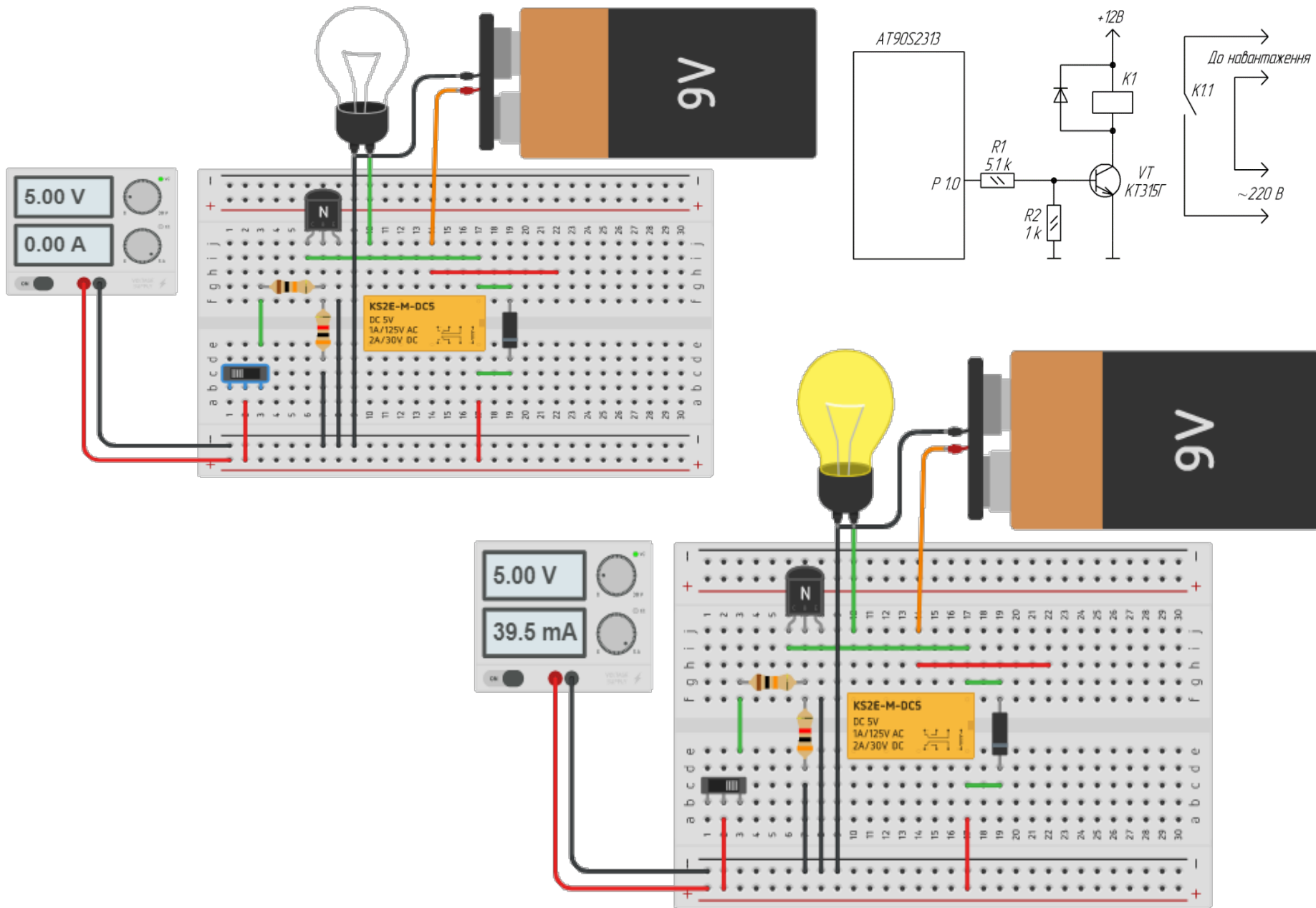
Для керування ланками з великим струмом потрібно застосувати потужніший транзистор або цілу транзисторну збірку. При виборі транзистора потрібно враховувати, що максимально допустимий струм навантаження для будь-якого з виходів МК не повинен перевищувати величини 20 мА. При написанні програми потрібно не забувати, що будь-який транзисторний ключ інвертує сигнал. Якщо на виході P1.0 (рис.1) встановити одиничний рівень, ключ відкривається і навантаження підключається до джерела живлення. При нульовому рівні на тому ж виході ключ закривається і навантаження відключається.



Виконавчий пристрій з використанням реле

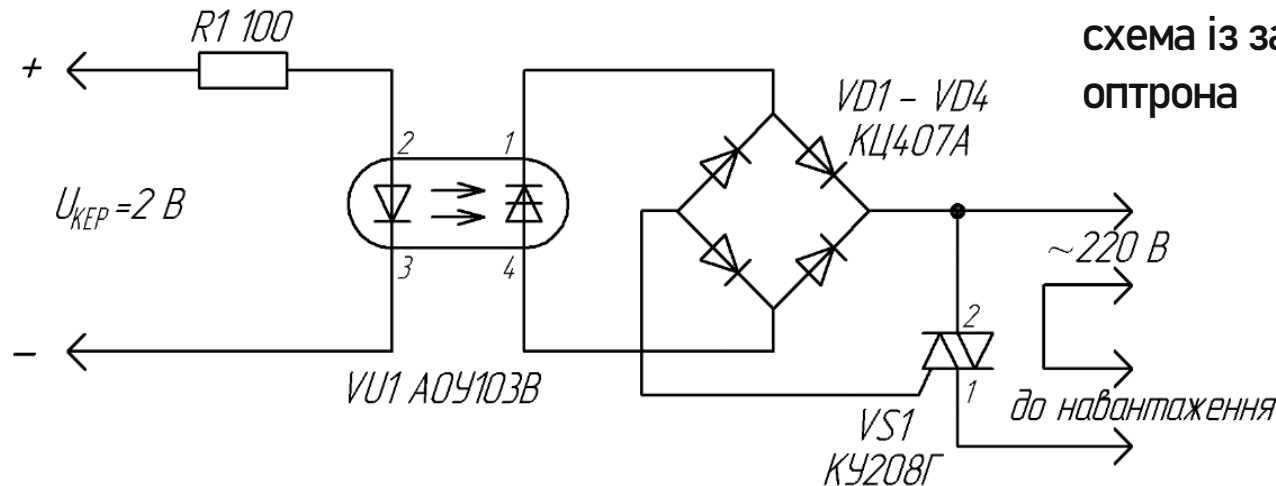
Якщо виконавчий механізм, яким повинна керувати МПС, живиться від мережі змінного струму 220 В, потрібно застосовувати схему управління з гальванічною розв'язкою. Один з можливих варіантів – релейна схема управління.

# Виконавчі пристрої



# Виконавчі пристрої

Гальванічна розв'язка між всіма колами МПС і силовою мережею 220 В забезпечує безпеку роботи з цією схемою. Діод VD1 призначений для захисту елементів схеми від напруги ЕРС самоіндукції, що виникає в котушці K1 у момент закривання ключа VT1. При виборі електромагнітного реле необхідно звертати увагу на такі параметри. По-перше, напруга спрацьовування реле. Для прикладу на рис. 2. вона має бути рівна 12 В. По-друге, максимально допустимий струм комутації і максимально допустима напруга для виконавчих контактів реле. Вони повинні відповідати реальним значенням струму і напруги в колі навантаження.



Досить часто реле замінюють оптоелектронними комутаційними вузлами, які мають малі струми і напруги керування, беззвучні і довговічні у роботі, можливість робити в середовищах постійного і змінного струму, комутації напруги (деяких приладів) до 400...600 В і струмів до 0,5 А.

# Виконавчі пристрої

На рис. 7.7 показаний ще один варіант включення - поєднання оптоелектронної розв'язки із застосуванням оптопари АОУ103В і симістора КУ208Г.

Управління пристроями навантаження ефективно здійснюється, якщо їх потужність не перевищує 600 Вт. Оптопара АОУ103В дозволяє самостійно комутувати високовольтне навантаження (з напругою до 350 В), проте струм комутації не повинен перевищувати 100 мА. Тому для управління потужним навантаженням в схему введений симістор КУ208Г

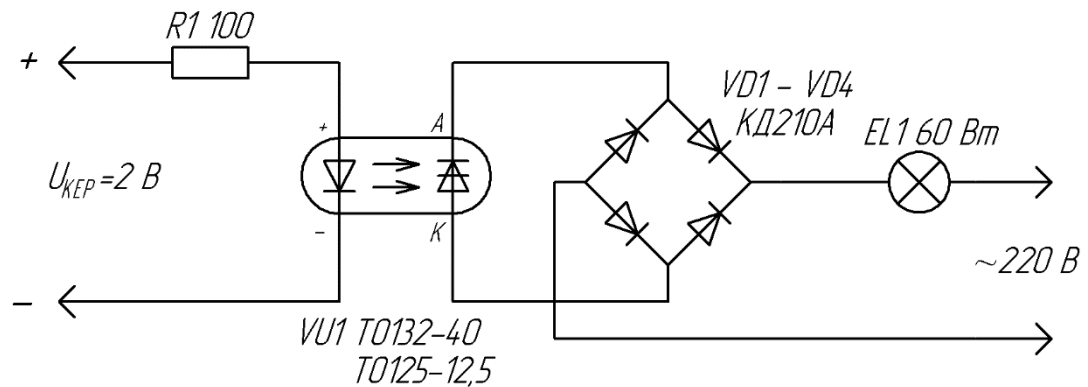


схема із застосуванням оптрона

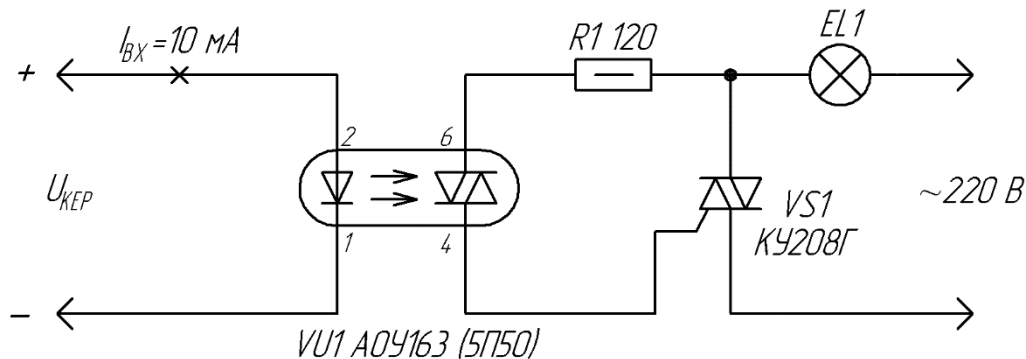
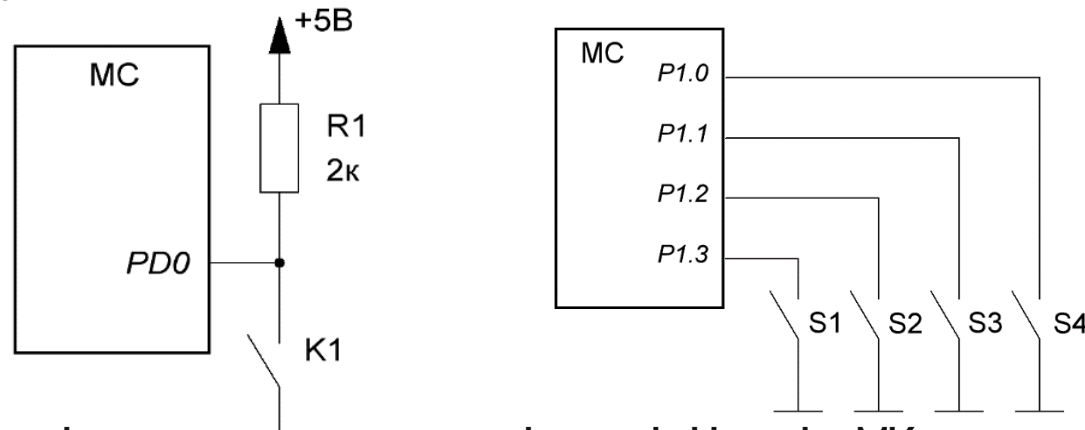


схема оптоелектронної розв'язки

# Кнопки та датчики

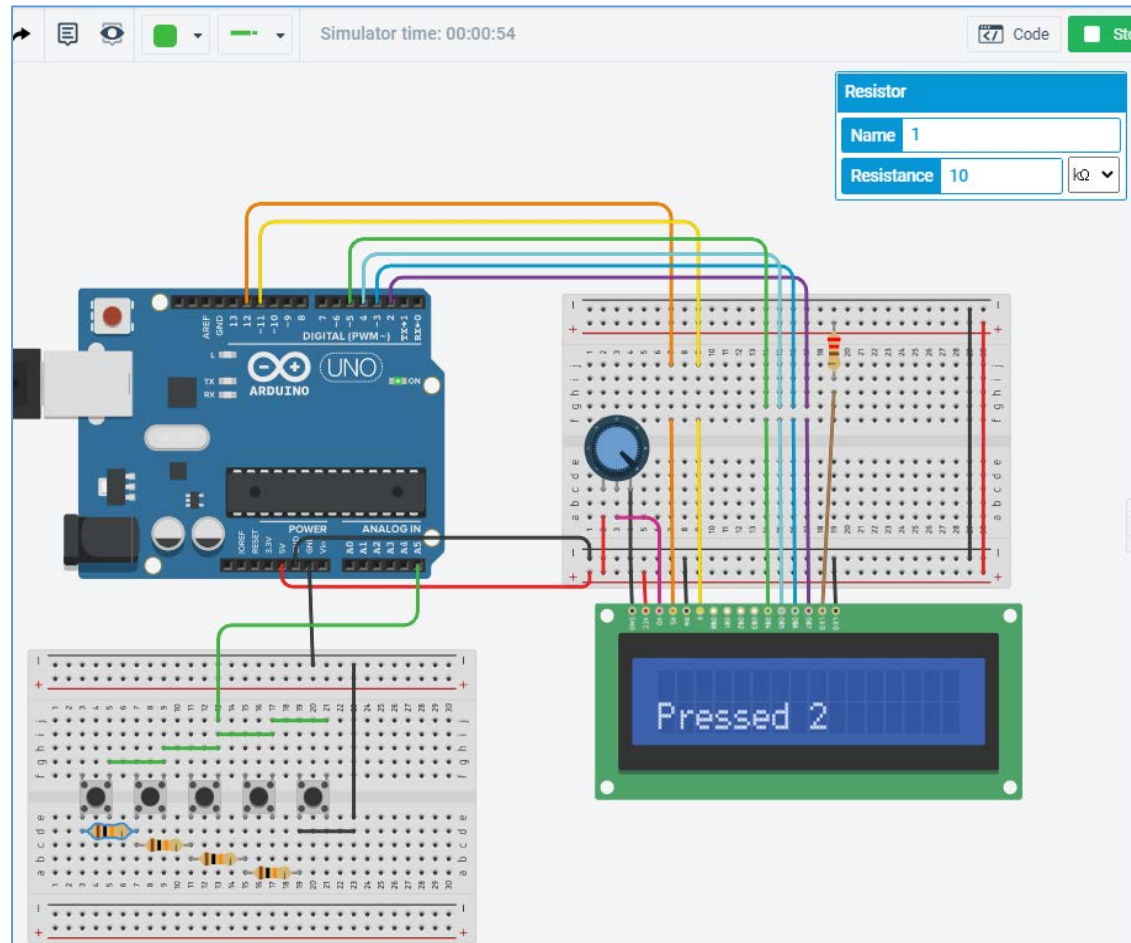
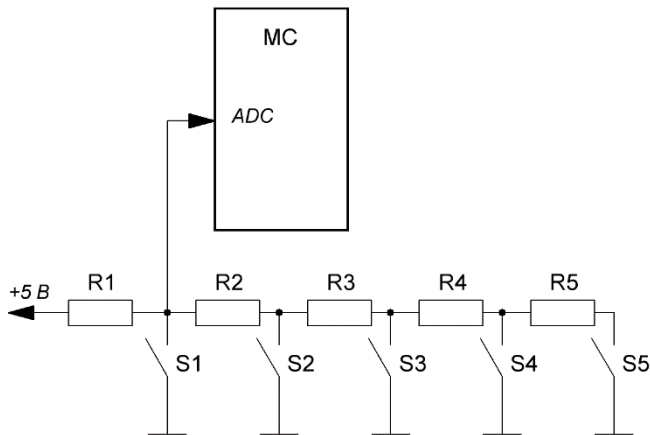
За допомогою кнопок і простих датчиків в мікропроцесорну систему управління надходить різна інформація, яка використовується для зміни алгоритму роботи програми. Схема підключення контактного датчика до МК наведена на рис. У наведеному прикладі датчик підключений до лінії PD0 порту D МК. Через цей вхід МК проводить зчитування стану датчика. Датчик можна підключити і до будь-якої іншої лінії будь-якого з портів МК.



У початковому стані контакти датчика розімкнені. На вхід МК через резистор R1 прикладається напруга від джерела живлення + 5 В. МК сприймає цю напругу як сигнал логічної одиниці. При спрацьовуванні датчика контакти замикаються і з'єднують вивід МК із загальним дротом. Тепер мікросхема сприймає вхідний рівень сигналу як логічний нуль. Резистор R1 при цьому служить струмообмежувальним елементом, запобігаючи короткому замиканню між шиною живлення і загальним дротом. Деякі МК мають свої внутрішні резистори навантаження, які можуть замінити зовнішній резистор. Схема підключення декількох датчиків або кнопок до МК зображена на рис..

# Аналогова клавіатура

У схемі, що зображена рис. при натисканні однієї з клавіш змінюється постійна напруга на відповідному вході процесора, яка розпізнається процесором і дешифрується в певну команду. Ця напруга максимальна (приблизно 5 В), коли кнопки не натиснуті, і мінімальна (0 В) при натиснутій клавіші S1.



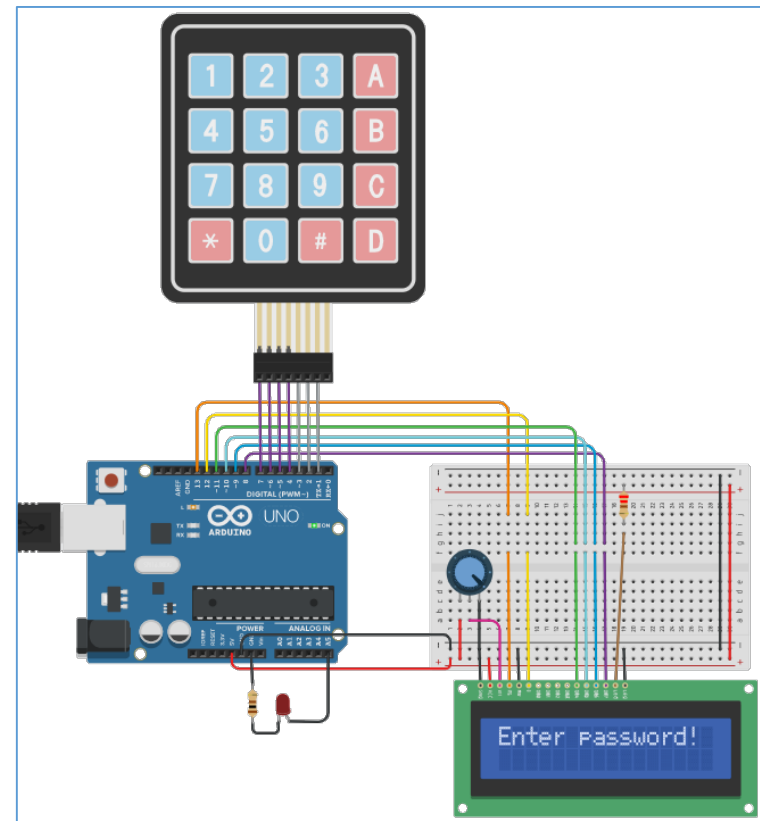
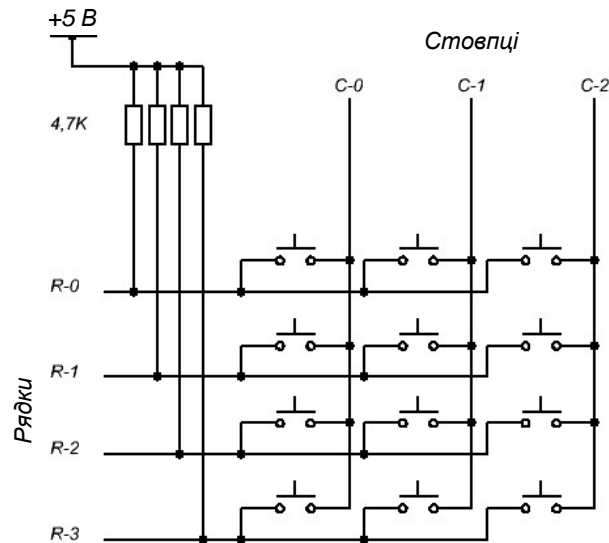
# Аналогова клавіатура

```
1 #include <LiquidCrystal.h>
2 LiquidCrystal lcd (12, 11, 5, 4, 2, 3);
3 int a=0;
4
5 void setup () {
6   lcd.begin (16, 2);
7   pinMode (A5, INPUT_PULLUP); // sets analog pin
8 }
9 // returns the button number pressed, or zero for
10 int readButtons (int pin) // int pin is the anal
11 {
12   int b,c = 0;
13   c=analogRead (pin); // get the analog value
14   if (c>1000) {
15     b=0; // buttons have not been pres
16   }
17   else if (c>440 && c<470) {
18     b=1; // button 1 pressed
19   }
20   else if (c<400 && c>370) {
21     b=2; // button 2 pressed
22   }
23   else if (c>280 && c<310) {
24     b=3; // button 3 pressed
25   }
26   else if (c>150 && c<180) {
27     b=4; // button 4 pressed
28   }
29   else if (c<20) {
30     b=5; // button 5 pressed
31   }
32   return b;
33 }
```

```
35 void loop () {
36   a=readButtons (19);
37   lcd.clear ();
38   if (a==0) // no buttons pressed
39   {
40     lcd.setCursor (0,1);
41     lcd.print ("Press a button");
42   }
43   else if (a>0) // someone pressed a button!
44   {
45     lcd.setCursor (0,1);
46     lcd.print ("Pressed button");
47     lcd.print (a);
48   }
49   delay (1000); // give the human time to read LCD
50 }
51
```

# Матрична клавіатура

Блок-схема 12-клавішної клавіатури зі скануванням показана на рис.. Клавіші розташовані у вузлах матриці, у якої чотири лінії рядків і три лінії стовпців. На лінії стовпців по черзі подається негативний імпульс (логічний «0»). У цей момент перевіряється стан чотирьох ліній рядків. Якщо натиснутих клавіш немає, всі лінії рядків мають високий рівень (вони підключені до напруги +5 В через резистори). Якщо ж клавіша натискається, і на лінії стовпця, відповідного натиснутій клавіші, все ще нуль, то лінія рядка також стає рівною нулю. Знаючи номери стовпця і рядка, можна отримати позицію натиснутої клавіші.



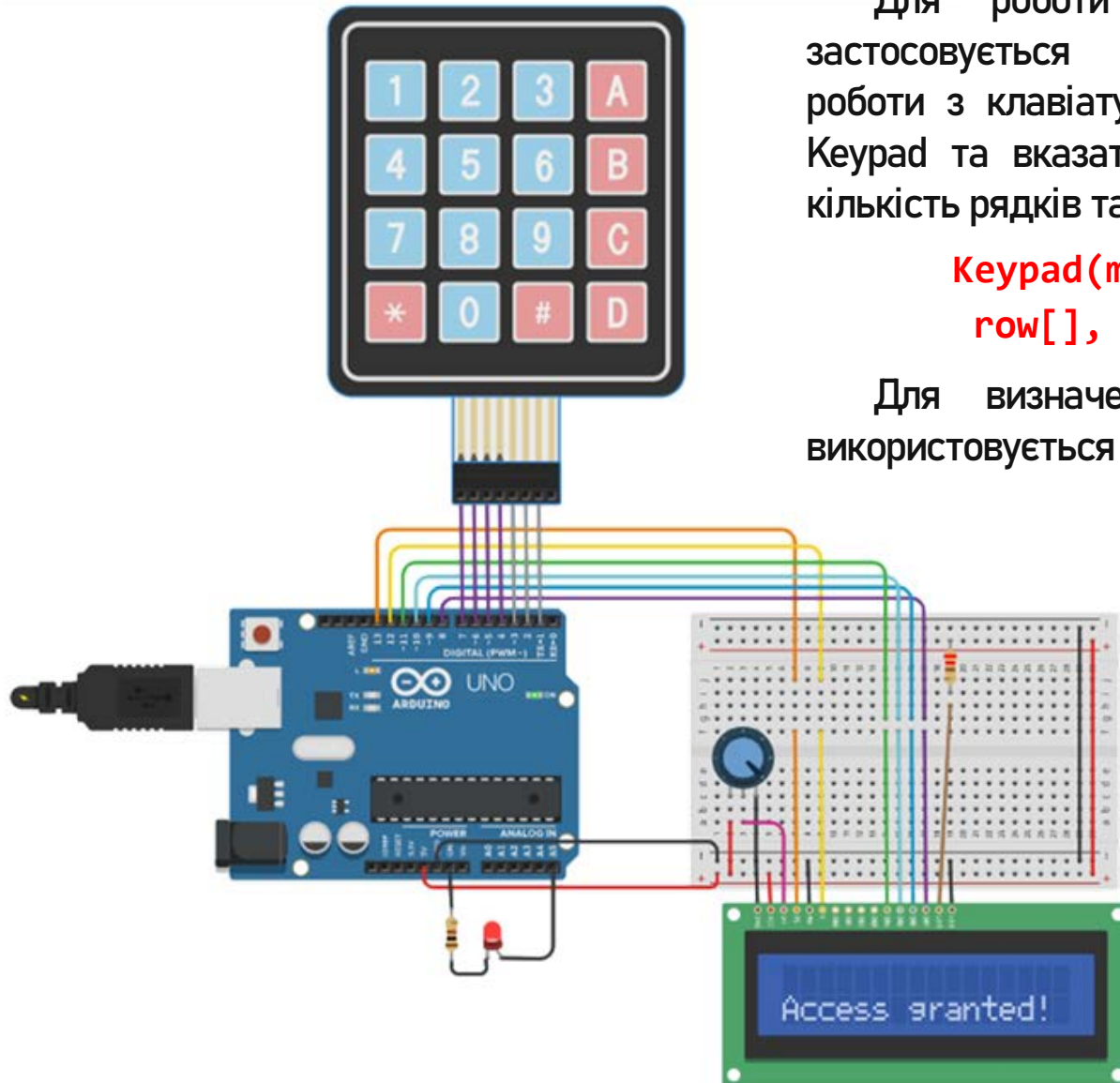
# Матрична клавіатура

Для роботи з матричною бібліотекою застосовується бібліотека <Keypad.h>. Для роботи з клавіатурою потрібно створити об'єкт Keypad та вказати виводи рядків та стовпців, кількість рядків та стовпців

```
Keypad(makeKeymap(userKeymap),  
        row[], col[], rows, cols)
```

Для визначення кнопки, що натиснута використовується функція

```
char getKey()
```



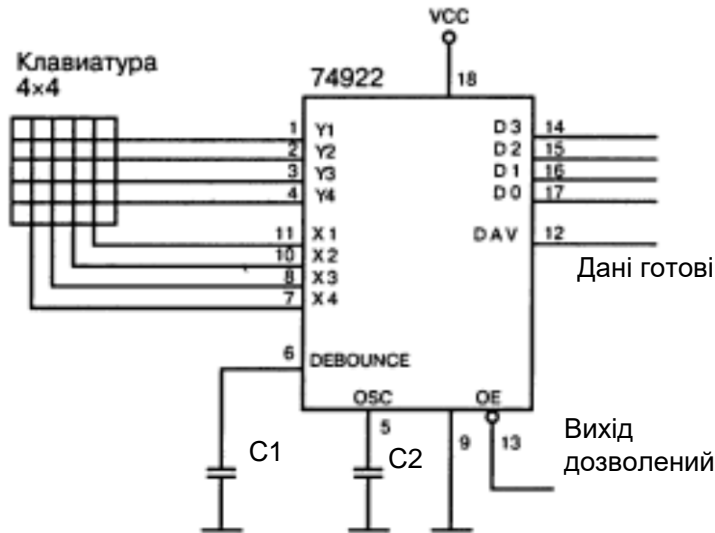
# Матрична клавіатура

```
1 #include <LiquidCrystal.h>
2 #include <Keypad.h>
3
4 // initialize the library with the numbers of the interface pins
5 LiquidCrystal lcd(13, 12, 11, 10, 9, 8);
6 const byte ROWS = 4; // число рядків клавіатури
7 const byte COLS = 3; // число стовпців клавіатури
8 char hexaKeys[ROWS][COLS] = {
9     {'1','2','3'},
10    {'4','5','6'},
11    {'7','8','9'},
12    {'*','0','#'}}
13 };
14 byte rowPins[ROWS] = {7, 6, 5, 4}; // виводи керування рядками
15 byte colPins[COLS] = {3, 2, 1}; // виводи керування стовпцями
16 char pass[4] = {'7', '3', '1', '5'}; // вірний пароль
17 char buttons[5]; // масив натиснутих кнопок
18 int k = 0; // лічильник натиснень
19
20 Keypad customKeypad=Keypad(makeKeymap(hexaKeys),rowPins, colPins,ROWS,COLS);
21
22 void setup() {
23     // set up the LCD's number of columns and rows:
24     lcd.begin(16, 2);
25     // Print a message to the LCD.
26     //lcd.print ("Press any key!");
27     pinMode (A5, OUTPUT);
28 }
```

# Матрична клавіатура

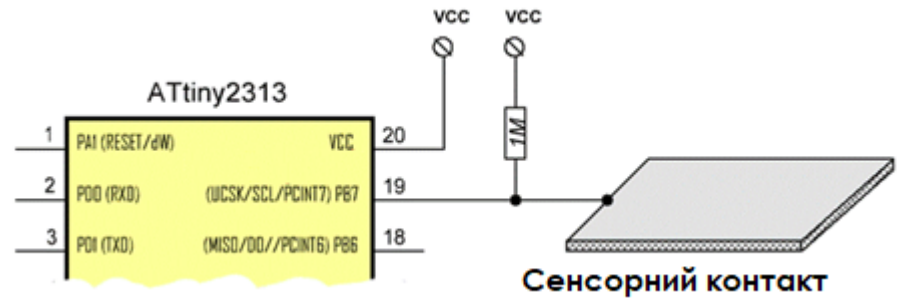
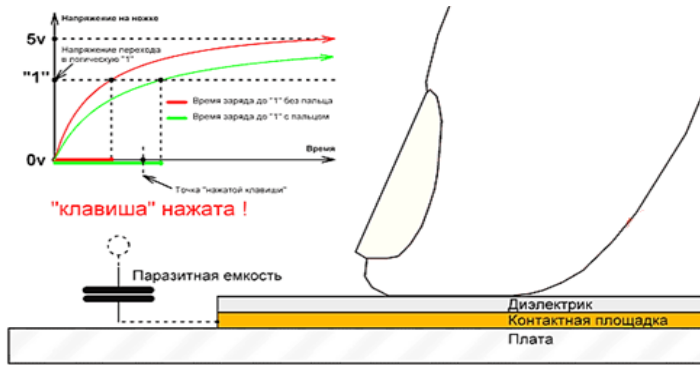
```
void loop() {  
  lcd.setCursor(0, 0); // курсор на початок першого рядку  
  lcd.print("Enter password!");  
  lcd.setCursor(0, 1); // курсор на початок другого рядка  
  char customKey = customKeypad.getKey();  
  if (customKey) {  
    buttons[k] = customKey; // зберігаємо значення кнопки у масиві  
    lcd.setCursor(k, 1);  
    lcd.print('*'); // виводимо символ '*' замість значення кнопки  
    k = k + 1; // збільшуємо лічильник натиснень на 1  
    if (k == 4) {  
      if(buttons[0]==pass[0]&&buttons[1]==pass[1]&&buttons[2]== pass[2]&&buttons[3]==pass[3]){  
        lcd.clear();  
        lcd.setCursor(0, 1);  
        lcd.print("Access granted!"); // якщо збіг паролю  
        digitalWrite (A5, HIGH);  
        delay (1000);  
        digitalWrite (A5, LOW);  
        lcd.clear();  
        k=0;  
      }  
      else {  
        lcd.clear();  
        lcd.setCursor(0, 1);  
        lcd.print("Access denied!"); // якщо пароль не вірний  
        lcd.clear();  
        k=0;  
      }  
    }  
  }  
}
```

# Клавіатура з кодуванням



Сенсорна кнопка реалізується досить просто. Від порту мікроконтролера на плюс живлення підключається резистор великого опору (pullup). До даного порту також підключається площадка, що проводить струм (сенсорна «кнопка»), яка ізолюється від прямого дотику діелектриком (скотч, лак, самоклеїтка)

# Сенсорна клавіатура



## Алгоритм визначення натиснення кнопки

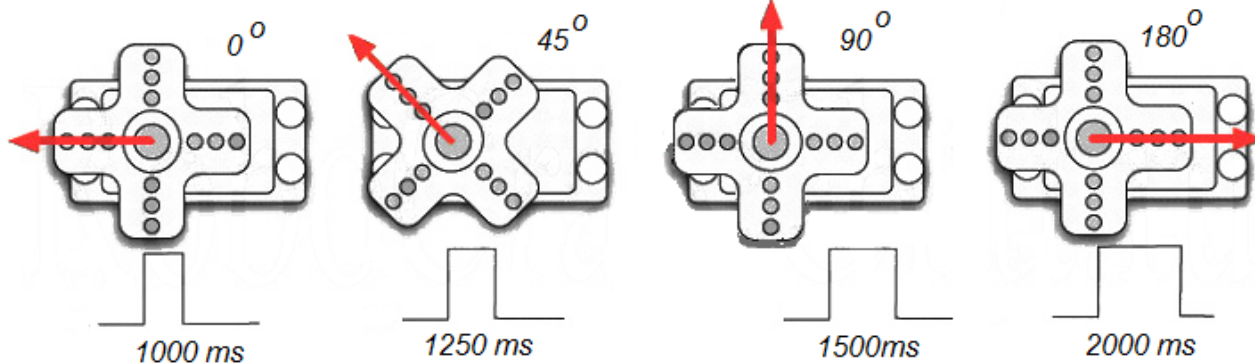
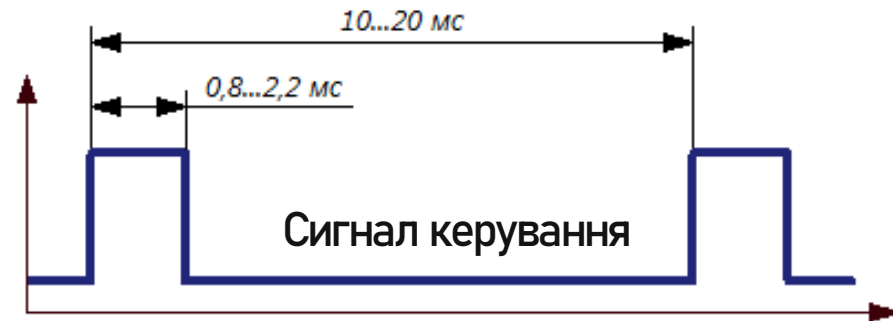
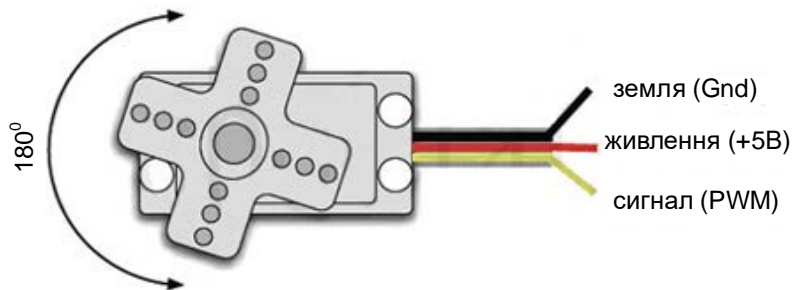
1. Порт переводиться на вивід (1  $\rightarrow$  DDRxn) та скидається в нуль (0  $\rightarrow$  PORTxn);
2. Порт переводиться на ввід, у Z-стан (0  $\rightarrow$  DDRxn). Так як на порту був 0, а елементи електричного кола порту мають певну ємність, то починається процес заряду цієї ємності через зовнішній резистор підтяжки;
3. З моменту переведення порту на ввід починається відлік часу з контролем стану порту (PINxn), як тільки на порту появиться рівень «1» (PINxn = 1) – зупиняється лічильник;
4. Значення лічильнику є ємністю ніжки у відносних одиницях. За величиною даного значення можна визначити чи є дотик до сенсору чи ні.

# Серводвигун

Сервопривід – це тип механічного приводу, до складу якого входить датчик (положення, швидкості, сили) та блок керування приводом, який автоматично підтримує необхідні параметри на датчику та на пристрої згідно заданому зовнішньому значенню.

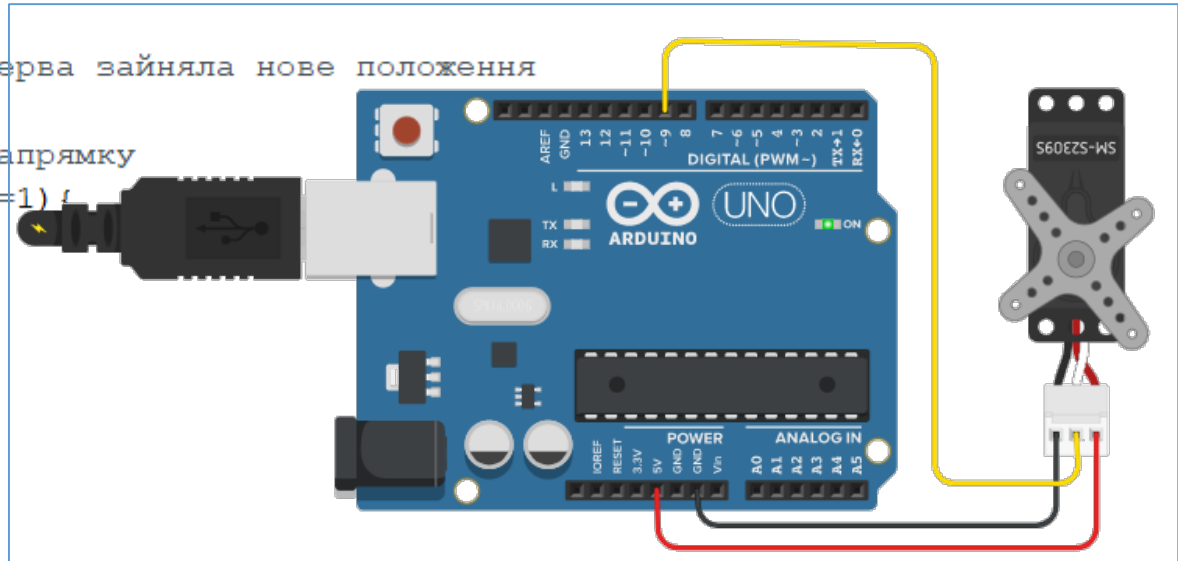
Сервопривід – мотор-редуктор дозволяє повертати вихідний вал строго у задане положення (на кут) та утримати його там.

Сервопривід живиться від постійної напруги в діапазоні від 4,8 В до 6В. Підключається сервопривід через універсальний з'єднувач (рис. 7.18) з трьома контактами: чорний – загальний провід або земля (GND), червоний – напруга живлення (+Vcc), жовтий – сигнал керування.



# Серводвигун

```
1 #include <Servo.h>
2 Servo myservo;
3 // створюємо об'єкт для контролю серводвигуна
4 // максимальна кількість таких об'єктів — 8
5 int pos = 0;
6 // змінна для зберігання позиції серви
7 void setup()
8 {
9   myservo.attach(9); // серводвигун підключений до D9
10 }
11 void loop()
12 {
13   for(pos = 0; pos < 180; pos += 1) {
14     // від 0 до 180 градусів
15     myservo.write(pos);
16     // встановлюємо положення
17     delay(15);
18     // чекаємо 15 мс, щоб серва зайняла нове положення
19   }
20   //обертаємо у зворотньому напрямку
21   for(pos = 180; pos>=1; pos-=1){
22     myservo.write(pos);
23     delay(15);
24   }
25 }
```



# Домашнє завдання

- 1) Модернізувати додані проєкти, щоб реалізувати кодовий замок, який при збігу коду обертає серводвигун протягом 5 с (оцінка 4).
- 2) Застосувати в схемі кодового замка виконавчий каскад з реле, який подає живлення на серво, який обертається 5с, при збігу коду (оцінка 5).