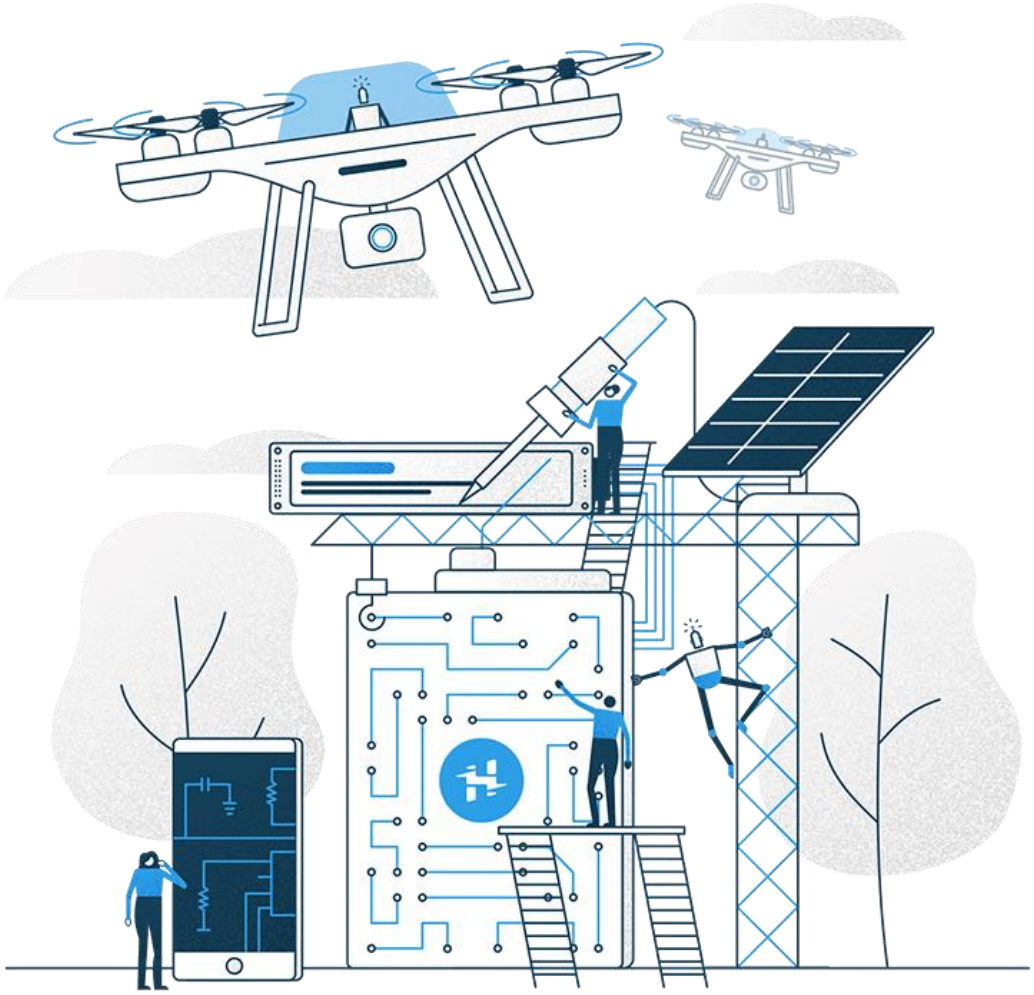
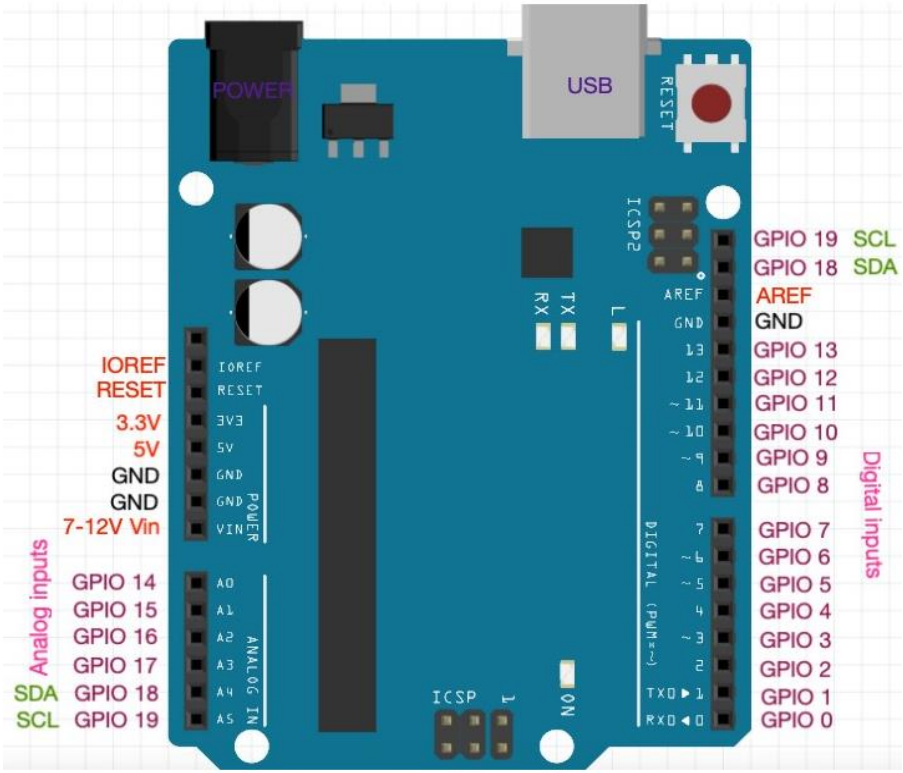
# МІКРОПРОЦЕСОРНІ СИСТЕМИ УПРАВЛІННЯ



Lesson 5

# Порти введення/виведення AVR




Мікроконтролер — це логічний пристрій, який створено для управління іншими пристроями за допомогою логічних (цифрових) сигналів. Це означає, що максимум можна зняти з порта 40 мА, а рекомендується не більше 20 мА. Що станеться, якщо зняти з порта більше, ніж він може віддати? Він зламається. Що буде, якщо зняти з декількох портів більше, ніж може віддати мікроконтролер в цілому? Згорить мікроконтролер. Цифрові порти служать для подачі команд іншим пристроям, наприклад реле / транзисторів для комутації навантажень.

# Порти введення/виведення AVR

За замовчуванням усі порти Arduino визначаються як входи, і немає потреби описувати це в коді. Порти зазвичай прописуються в функції ініціалізації змінних. Ініціалізація порту вводу-виводу Arduino:

pinMode (pin, mode). Параметри:

- pin: номер виводу, режим роботи якого задається;
- mode: приймає значення: INPUT — вхід, у цьому режимі відбувається зчитування даних з датчиків, стану кнопок, аналогового та цифрового сигналу. Порт знаходиться в так званому високоімпедансному стані, тобто на вході високий опір. OUTPUT — вихід, залежно від команди прописаної в коді, порт приймає значення 1 або 0. Вихід стає свого роду керованим джерелом живлення та видає максимальний струм (20 мА) у навантаження, що до нього підключене. INPUT_PULLUP — порт працює як вхід, але до нього підключається «PushUp» резистор з номіналом 20 – 50 кОм;
- значення, що повертаються – немає.

digitalWrite (pin, value). Параметри:

- pin: номер виводу;
- value: значення HIGH або LOW;
- значення, що повертаються — немає.

digitalRead (pin). Параметри:

- pin: номер цифрового виводу, з якого необхідно зчитати значення (int);
- значення, що повертаються HIGH або LOW.

# Порти введення/виведення AVR

Для зміни режимів роботи портів D2-D13, A0-A5 можна використати фрагмент програми:

```
for ( byte i = 2; i < = 19; i ++ ) {
    pinMode (i, OUTPUT ) ; // робимо виходами }
```

Приклад, в якому порти ініціалізуються як виходи, і на них подається сигнал:

```
void setup () {
  pinMode (10, OUTPUT )    ; // D10 як вихід
  pinMode (A3, OUTPUT )    ; // A3 як вихід
  pinMode (19, OUTPUT )    ; // A5 як вихід (Nano / UNO)
  digitalWrite (10, HIGH) ; // високий рівень на D10
  digitalWrite (A3, 1)     ; // високий рівень на A3
  digitalWrite (19, 1)     ; // високий рівень на A5
}
void loop () {}
```

Приклад читання стану порта

```
void setup () {
  Serial. begin ( 9600 ) ;
}
void loop () {
  Serial. println ( digitalRead ( 5 )) ;
}
```

# Порти введення/виведення AVR

Регістри портів дозволяють низькорівневі високошвидкісні маніпуляції з портами мікроконтролера. Мікроконтролери, що використовуються в Arduino мають три порти : B (D8-D13), C(A0-A7), D(D0-D7).

Кожен порт контролюється трьома регістрами, кожен з яких відповідає за певний стан. Регістр DDR визначає, який біт порта вхідний, а який вихідний. Регістр PORT встановлює біт порта у відповідний стан HIGH або LOW, регістр PIN читає стан вхідного порта.

Регістри DDR та PORT можуть бути як прочитані, так і записані. Регістр PIN відповідає за стан вхідних портів, тому може бути лише прочитаний.

PORTD відповідає за виводи 0 - 7.
DDRD — регістр напряму порту D;
PORTD — регістр даних порту D;
PIND — регістр вхідних даних порту D

PORTC відповідає за аналогові виводи 0 - 5.
DDRC — регістр напряму порту C;
PORTC — регістр даних порту C;
PINC — регістр вхідних даних порту C

PORTB відповідає за виводи 8 - 13. Два старших біта (6 та 7), що відповідають за виводи кварцу, не використовуються.
DDRB — регістр напряму порту B;
PORTB — регістр даних порту B;
PINB — регістр вхідних даних порту B

# Порти введення/виведення AVR

## Приклад роботи з портом D

```
// призначаємо виводи Arduino 1-7 вихідними, вивід 0-вхідним
DDRD = B11111110;
// виводи з 2 по 7 вихідні, стан виводів 0 та 1 не змінюється
DDRD = DDRD | B11111100;
// встановлюємо рівень HIGH на цифрових виводах 7,5,3
PORTD = B10101000;
```

## Приклад роботи з портом B

```
void setup() {
//виставляємо всі біти порту B як вихід, PB4 як вхід
DDRB = B11101111;
PORTB = B00000000; //скидаємо всі біти порту B
}
void loop() {
if (PINB==B00010000) {
PORTB |= 1 << 5 // PB5=1
delay (1000) // очікуємо секунду
PORTB &= ~(1 << 5)// PB5=0
delay (1000) }
}
```

# Порти введення/виведення AVR

При роботі з портами можна використовувати вбудовані в Arduino функції для роботи з бітами порта bitRead(), bitWrite(), bitSet(), bitClear().

bitRead(x, n) зчитує стан зазначеного біта числа, де x: регістр вхідних даних порту (PINB, PIND, PINC) біт якого буде зчитуватись; n: номер біта, стан якого необхідно зчитати (стан біту (0 або 1)).

bitWrite(x, n, b) змінює стан зазначеного біта змінної, де x: числова змінна, у якій необхідно змінити біт (DDRx або PORTx); n: номер біта, стан якого необхідно змінити; b: нове значення біта (0 або 1).

bitSet(x, n) встановлює зазначений біт (записує 1) числової змінної, де x: числова змінна, у якій необхідно змінити біт (DDRx або PORTx); n: номер біта, стан якого необхідно змінити.

bitClear(x, n) скидає вказаний біт (записує 0) числової змінної, де x: числова змінна, у якій необхідно змінити біт (DDRx або PORTx); n: номер біта, стан якого необхідно змінити.

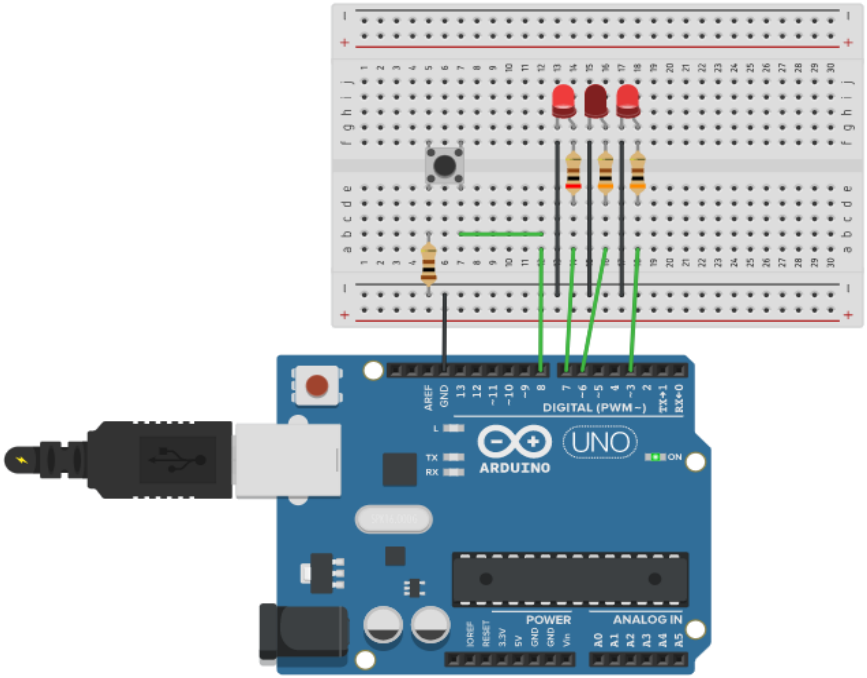
Використання регістрів портів суттєво зменшує розмір коду та збільшує швидкодію на декілька порядків

# Порти введення/виведення AVR

Приклад читання стану цифрового порта (перевірка натиснення кнопки):

```
void setup(){
pinMode (8, INPUT_PULLUP);
DDRD=0xFF;
  PORTD = 0b010101010;}

void loop(){
  if (digitalRead(8)==0){
     delay (200);
     if (digitalRead(8)==1)
        PORTD = ~PORTD; }
  }
```

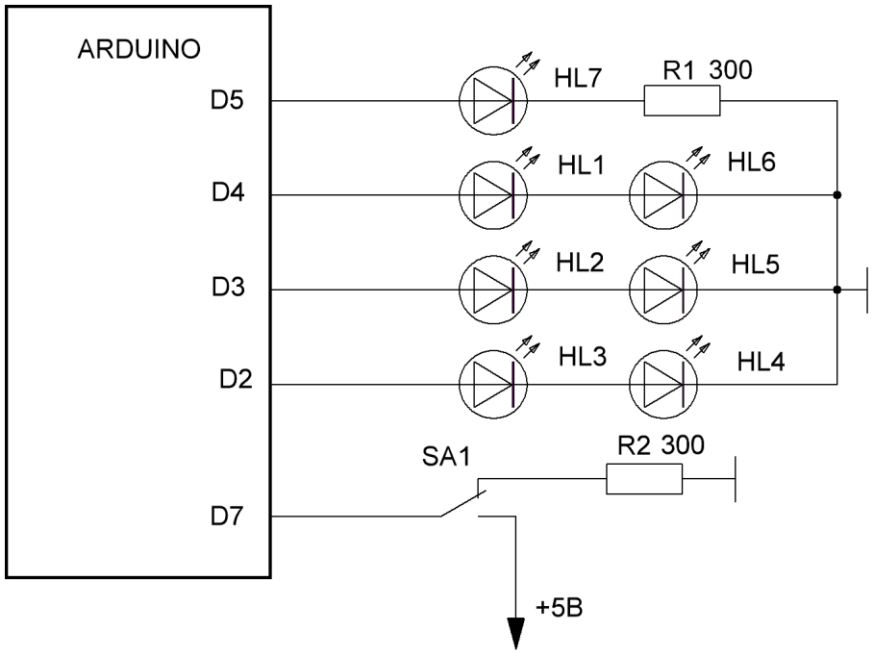


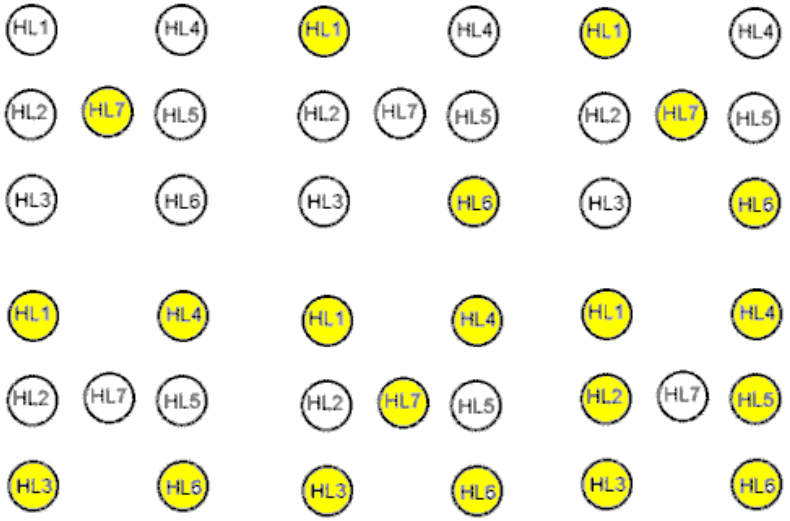

```
void setup(){
pinMode (8, INPUT_PULLUP);
DDRD=0xFF;
  PORTD = 0b010101010;}

void loop(){
    while(digitalRead(8)==1){}
    while(digitalRead(8)==0){}
    PORTD = ~PORTD;
  }
```

# Індивідуальне завдання

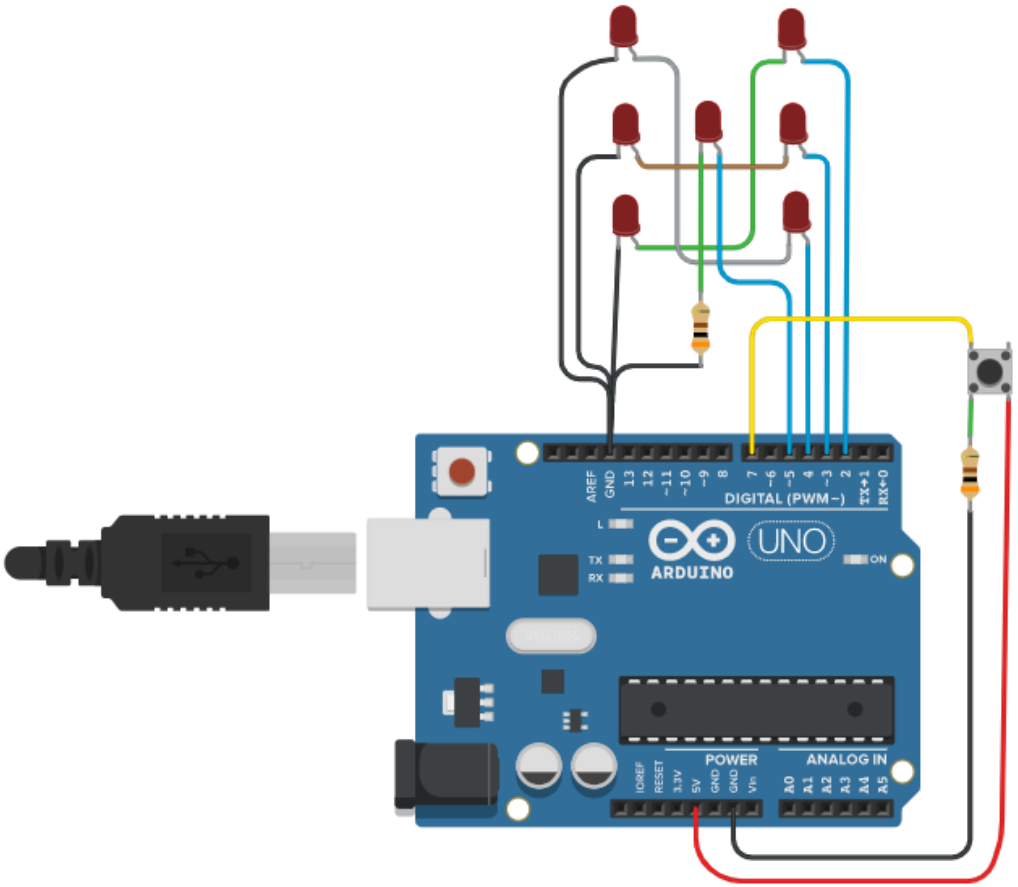
Реалізувати електронний кубик на платформі Arduino



# Індивідуальне завдання

Реалізувати електронний кубик на платформі Arduino



# Етапи реалізації

1. Скласти схему у TinkerCad
2. Написати програму (code 1), щоб по черзі відображались цифри 1, 2, 3, 4, 5, 6 (використати bitRead, digitalRead).
3. Ввести зміни у програму (code 2), щоб з кожним натисненням кнопки по черзі відображались цифри 0 (усе погашено), 1, 2, 3, 4, 5, 6, 0....
4. Ввести зміни у програму (code 3), щоб генерувалось випадкова комбінація count = random(1,7).
5. Внести зміни у схему та програму, щоб використати Interrupt (code4)
5. Оформити звіт. Він містить code 1, code 2, code 3, code 4 та скріни роботи електронного кубика. За можливістю надати посилання на проект у TinkerCad