

## Система команд і програмна модель AVR

Процес написання програм для МК AVR як і для будь-яких інших, складається з декількох етапів: підготовка вихідного тексту програми на якій-небудь мові програмування; компіляція програми; налагодження й тестування програми; остаточне програмування й підготовка до серійного виробництва.



Рисунок 1.11 – Етапи розробки програмного забезпечення мікроконтролерів AVR

Мікропрограма пристрою повинна бути написана на одній з мов програмування. На даний час для МК AVR існують декілька мов програмування, а також різних засобів підтримки розробки, що використовують одну мову, але різняться за функціональністю. На кожному з етапів необхідне застосування спеціальних програмних й апаратних засобів. Варто відзначити, що базовий набір програмного забезпечення (компілятор асемблера, ПЗ для програмування) поширюється фірмою Atmel безкоштовно. Однак за досить довгий період часу, що пройшов з моменту появи цих МК, з'явилася велика кількість програмного забезпечення сторонніх виробників [2].

Етапи розробки програмного забезпечення мікроконтролерів AVR наведені

на рисунку 1.11.

Інтегроване середовище розробки Arduino IDE – це кросплатформовий додаток на Java, що включає в себе редактор коду, компілятор та модуль передачі прошивки в плату.

Щоб почати використовувати Arduino IDE, треба зайти на сайт <https://www.arduino.cc>, перейти на вкладку SOFTWARE > DOWNLOADS та завантажити середовище розробки Arduino (рис.1.12).



Рисунок 1.12 — Завантаження середовища розробки

Далі необхідно інсталиувати драйвер для плати Arduino, з якою треба працювати (це робить тільки з підключеною платою через USB):

- 1) Підключіть плату до комп'ютера і дочекайтеся, поки Windows не почне процес установки драйверів. Незважаючи на всі зусилля системи, через кілька секунд процес завершиться невдачею.
- 2) Зайдіть в Пуск, відкрийте Панель керування.
- 3) У Панелі керування перейдіть на сторінку Система і безпека. Далі клацніть по пункту Система і відкрийте Диспетчер пристроїв.
- 4) Знайдіть розділ Порти (COM & LPT). У ньому ви побачите відкритий порт під ім'ям «Arduino UNO (COMxx)».
- 5) Клацніть правою кнопкою по пункту «Arduino UNO (COMxx)» і виберіть «Оновити драйвер».
- 6) Далі, у вікні, виберіть пункт «Виконати пошук драйверів на цьому комп'ютері»

- 7) На завершення, виберіть файл драйвера під ім'ям «*arduino.inf*», розташований в папці «Drivers» в директорії завантаженого програмного забезпечення Arduino. Windows завершить інсталяцію драйвера.
- 8) Відкрити середовище розробки Arduino та запустити тестову програму File > Examples > 1.Basics > Blink.
- 9) Тепер в меню Tools>Board необхідно вибрати пункт меню, що відповідає вашій моделі Arduino.
- 10) У меню Tools>Serial Port виберіть послідовний порт, до якого підключений лабораторний макет. Як правило, це COM-порт з номером 3 (COM3) або вище (COM1 і COM2 зазвичай асоційовані з апаратними портами). Тепер можна працювати та завантажувати скетчі в Arduino.

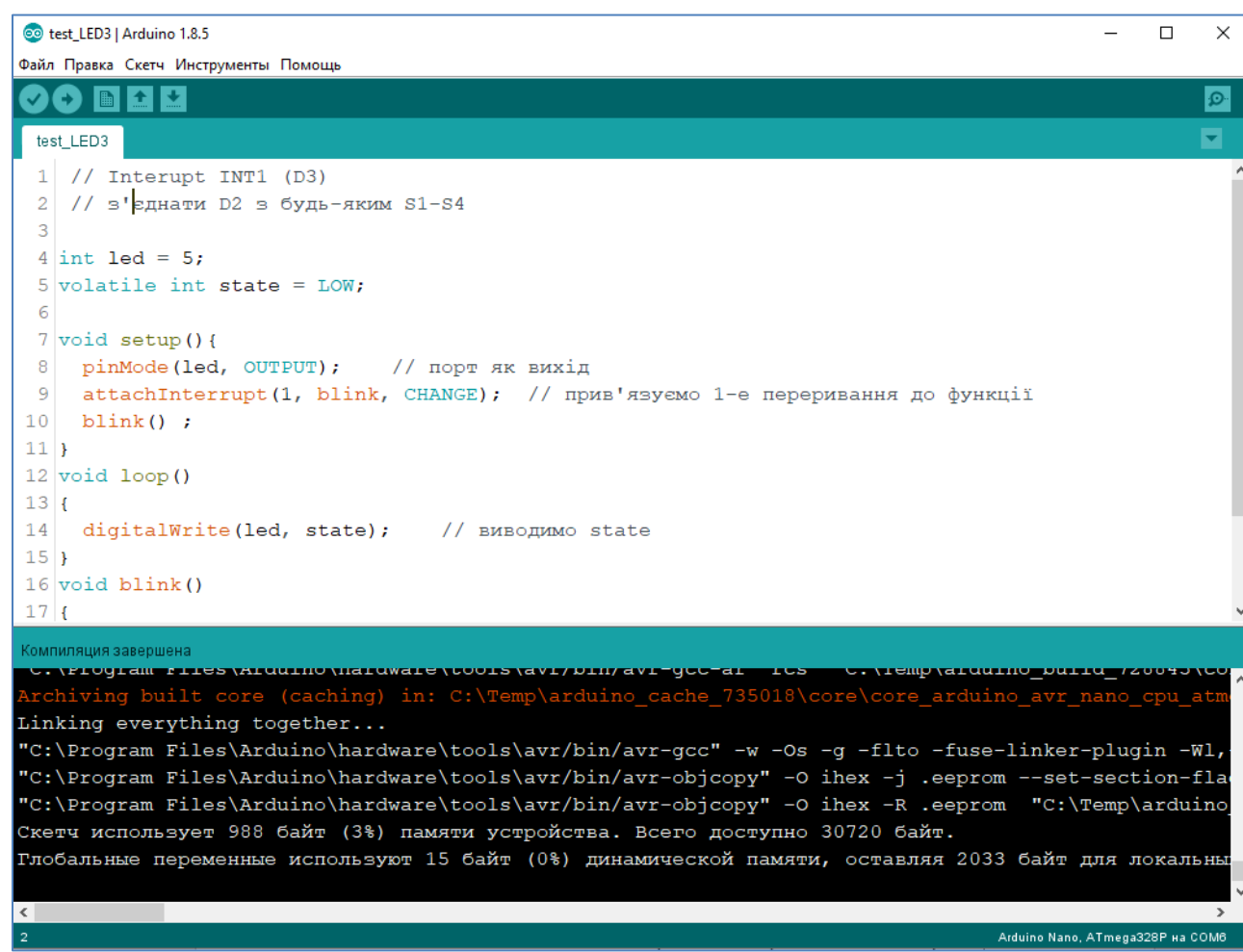


Рисунок 1.13 – Загальний вигляд програми Arduino IDE

Середовище розробки Arduino (рис.1.13) складається із вбудованого текстового редактора програмного коду, області повідомлень, вікна виведення тексту (консолі), панелі інструментів з кнопками часто

використовуваних команд і декількох меню. Для завантаження програм і зв'язку середовище розробки підключається до апаратної частини Arduino.

Програма, написана в середовищі Arduino, називається *скетч*. Скетч пишеться в текстовому редакторі, що має інструменти: вирізати / вставити, пошук / заміна тексту. Під час збереження та експорту проекту в області повідомлень з'являються пояснення, також можуть відображатися виникли помилки. Вікно виведення тексту (консоль) показує повідомлення Arduino, що включають повні звіти про помилки та іншу інформацію. Кнопки панелі інструментів дозволяють перевірити та записати програму, створити, відкрити та зберегти скетч, відкрити моніторинг послідовної шини.

### ***Блокнот (Sketchbook)***

Середовищем Arduino використовується принцип блокнота: стандартне місце для зберігання програм (скетчів). Скетчі з блокнота відкриваються через меню File → Sketchbook або кнопкою Open на панелі інструментів. При першому запуску програми Arduino автоматично створюється директорія для блокнота. Розташування блокнота змінюється через діалогове вікно Preferences.

### ***Закладки, Файли і Компіляція***

Дозволяють працювати з декількома файлами скетчів (кожен відкривається в окремій закладці). Файли коду можуть бути стандартними Arduino (без розширення), файлами C (розширення \* .c.), файлами C ++ (\*.cpp) або файлами заголовків (.h).

### ***Завантаження скетчу в Arduino***

Після вибору порту та платформи необхідно натиснути кнопку завантаження на панелі інструментів або вибрати пункт меню File → Upload to I/O Board. Сучасні платформи Arduino перезавантажуються автоматично перед завантаженням. На більшості плат під час процесу будуть мигати світлодіоди RX і TX. Середовище розробки Arduino виведе повідомлення про закінчення завантаження або про помилки.

При завантаженні скетчу використовується завантажувач (Bootloader) Arduino, невелика програма, що завантажується в мікроконтролер на платі.

Вона дозволяє завантажувати програмний код без використання додаткових апаратних засобів. Завантажувач (Bootloader) активний протягом декількох секунд при перезавантаженні платформи і при завантаженні будь-якого з скетчів в мікроконтролер. Робота завантажувача (Bootloader) розпізнається по миготінню світлодіода (13 вивід) (наприклад, при перезавантаженні плати).

### ***Експорт бінарного файлу***

Для роботи з віртуальним стендом «Arduino Learner Kit» у середовищі Proteus 8 необхідно підключити файл типу .HEX (рис. 1.14). Для його отримання потрібно вибрати пункт меню Sketch → Export Compiled Binary. HEX файл буде створений у директорії зі скетчим.

### ***Бібліотеки***

Бібліотеки додають додаткову функціональність скетчам, наприклад, при роботі з апаратною частиною або при обробці даних. Для використання бібліотеки необхідно вибрати меню Sketch → Include Library. Можна вибрати бібліотеки зі списку або завантажити через Library Manager (рис.1.14).

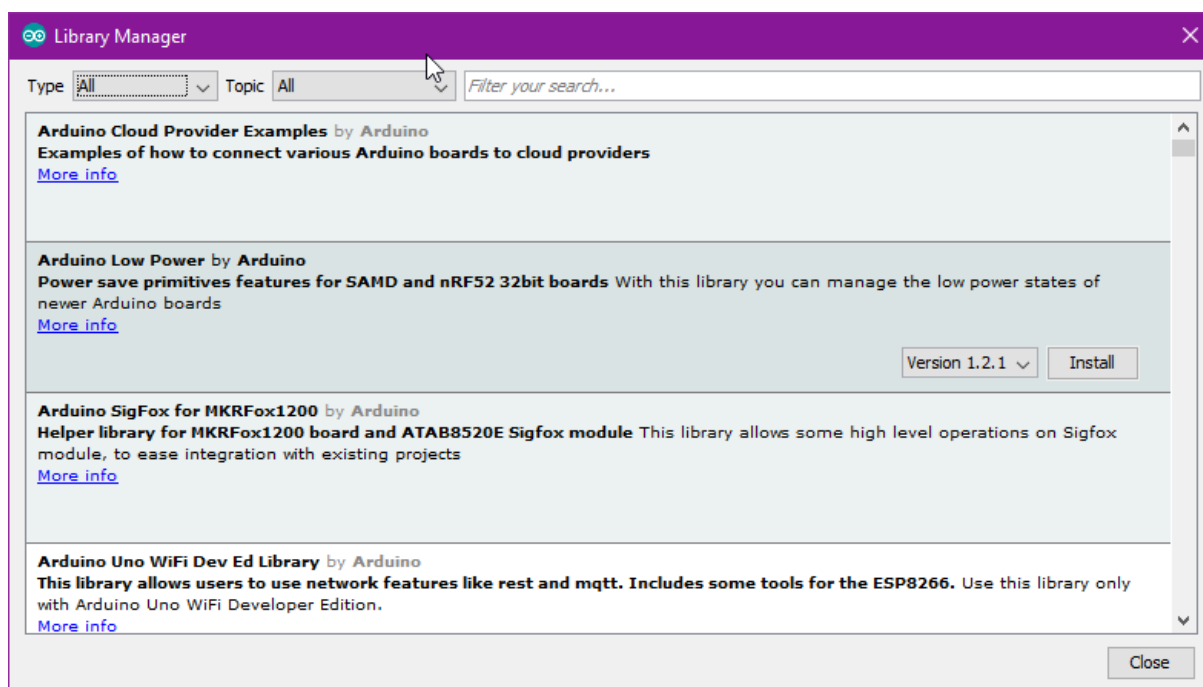


Рисунок 1.14 — Менеджер завантаження бібліотек

Одна або кілька директив *#include* будуть розміщені на початку коду скетчу з подальшою компіляцією бібліотек разом зі скетчем. Завантаження

бібліотек вимагає додаткового місця в пам'яті Arduino. Невикористані бібліотеки можна видалити з скетчу прибравши директиву *#include*.

Основні бібліотеки: EEPROM, SD, SPI, SoftwareSerial, Wire. Деякі бібліотеки включені в середовище розробки Arduino. Інші можуть бути завантажені з різних ресурсів. Для встановлення додаткових бібліотек необхідно створити директорію «libraries» у папці блокнота та потім розпакувати архів. Наприклад, для встановлення бібліотеки DateTime її файли повинні знаходитися в папці / libraries / DateTime папки блокнота.

## **Програмування в машинних кодах**

Середовище розробки засновано на мові програмування Processing і спроектовано для програмування новачками, які не знайомі близько з розробкою програмного забезпечення. Строго кажучи, це C/C++, доповнений деякими бібліотеками. Програми обробляються за допомогою препроцесора, а потім компілюються за допомогою AVR-GCC.

Мова Arduino має чотири складових: оператори, дані, функції, бібліотеки.

*Оператори:*

- setup (),
- loop (),
- оператори мови C.

*Дані:* типи даних з мови C.

*Функції:*

- цифрове введення / виведення,
- аналогове введення / виведення,
- час,
- математичні обчислення,
- тригонометрія,
- випадкові числа,
- біти і байти,
- зовнішні переривання,

- переривання.

*Бібліотеки:*

- EEPROM,
- SD,
- SPI,
- SoftwareSerial,
- Wire,
- допоміжні класи,
- клас Serial,
- клас Stream.

Оголошення змінної відбувається так: спочатку вказується тип даних для змінної, а потім назва змінної. Оператор присвоювання (=) — не є знаком рівності та не може використовуватися для порівняння значень. Оператор рівності записується так — ==. Присвоєння використовується для збереження певного значення в змінній. Наприклад, запис виду  $a = 10$  задає змінній  $a$  значення числа 10.

Якщо знаємо точну кількість дій (ітерацій) циклу, то можемо використовувати цикл *for*. Синтаксис його виглядає так:

*for* (дія до початку циклу; умова продовження циклу; дія в кінці кожної ітерації циклу)

```
{  
    інструкція    циклу    1;  
    інструкція    циклу    2;  
    інструкція циклу N;  
}
```

Ітерацією циклу називається один прохід цього циклу.

Коли не відомо скільки ітерацій повинен зробити цикл, тоді використовується цикл *while* або *do ... while*. Синтаксис циклу *while* виглядає так:

```
while (Умова) {  
    Тіло циклу;}
```

Даний цикл буде виконуватися, поки умова, вказана в круглих дужках є істиною.

*Оператор if* служить для того, щоб виконати будь-яку операцію в тому випадку, коли умова є вірною. Умовна конструкція завжди записується в круглих дужках після оператора *if*. У середині фігурних дужок вказується тіло умови. Якщо умова виконається, то почнеться виконання всіх команд, які знаходяться між фігурними дужками.

*Оператор else*. Кожному оператору *if* відповідає тільки один оператор *else*. Сукупність цих операторів — *else if* означає, що якщо не виконалася попередня умова, то перевірити дану. Якщо жодна з умов не є вірною, то виконується тіло оператора *else*.

#### Оператори порівняння

`==` (дорівнює);  
`!=` (не дорівнює);  
`<` (менше ніж);  
`>` (більше ніж);  
`<=` (менше або дорівнює);  
`>=` (більше або дорівнює).

#### Логічні оператори

`&&` (І);  
`||` (АБО);  
`!(НЕ);&` (побітове І);  
`|` (побітове АБО).

#### Бітові оператори

`^` (побітове XOR або виключне АБО);  
`~` (побітове НЕ);  
`<<` (побітовий зсув вліво);  
`>>` (побітовий зсув вправо).

#### Складні оператори++ (інкремент)

`--` (декремент);  
`+=` (складене додавання);  
`-=` (складене віднімання);  
`*=` (складене множення);



/ = (складене ділення);

& = (складене побітове І);

| = (складене побітове АБО).

Будь-яка функція має тип, як і будь-яка змінна. Функція може повертати значення, тип якого аналогічний типу самої функції. Якщо функція не повертає ніякого значення, то вона повинна мати тип *void* (такі функції іноді називають процедурами).

При оголошенні функції, після її типу має стояти ім'я функції і дві круглі дужки — відкриваюча і закриваюча, всередині яких можуть знаходитися один або кілька аргументів функції, яких також може не бути взагалі. Після списку аргументів функції ставиться відкриваюча фігурна дужка, після якої знаходиться саме тіло функції. В кінці тіла функції обов'язково ставиться фігурна дужка, що закриває його.

Під час виклику функції *setup()*, програма ініціалізується та встановлює початкові значення. Функція *setup()* викликається, коли стартує скетч. Використовується для ініціалізації змінних, визначення режимів роботи виводів, запуску використовуваних бібліотек. Функція *setup()* запускається тільки один раз, після кожної подачі живлення або скидання плати Arduino.

Функція *loop ()* забезпечує нескінченний робочий цикл програми. У циклі виконується опитування стану виводів, зміна їх стану, прийом-передача даних робота з АЦП та ін.

Приклад:

```
int buttonPin = 3;
void setup()
{
  // put your setup code here, to run once:
}
void loop()
{
  // ...
}
```

## Типи даних

*void* — ключове слово `void` використовується тільки при оголошенні функцій. Воно вказує на те, що оголошувана функція не повертає ніякого значення.

*boolean* — змінні типу `boolean` можуть приймати одне з двох значень: `true` або `false`. Кожна змінна типу `boolean` займає в пам'яті один байт.

*char* — тип даних, який займає в пам'яті 1 байт та зберігає символічне значення. Символи пишуться в одинарних лапках, наприклад: `'A'` (сукупність символів (рядки) пишуться у подвійних лапках: `"ABC"`).

*int* — цілочисельний тип даних. Це основний тип даних для зберігання чисел. В Arduino Uno змінні типу `int` зберігають 16-бітові (2-байтові) значення у діапазоні від `-32768` до `32767`.

*unsigned int* — беззнакові цілі містять двобайтові значення. Замість негативних чисел зберігаються лише позитивні значення в діапазоні від `0` до `65535`.

*long* — змінні типу `long` маю розширений розмір для зберігання чисел та мають розмірність 32 біта (4 байта), що дозволяє їм зберігати числа в діапазоні від `-2 147 483 648` до `2 147 483 647`.

*unsigned long* — мають розмірність 32 біта (4 байта). Змінні типу *unsigned long*, на відміну від звичайного `long`, зберігають тільки позитивні числа в діапазоні від `0` до `4 294 967 295`.

*short* — це 16-бітний тип даних.

*float* — тип даних для чисел з плаваючою точкою. Числа з плаваючою точкою часто використовуються для подання аналогових або безперервних величин, оскільки дають можливість окреслити їх більш точно, ніж цілі числа. Числа з плаваючою точкою мають 32 біта (4 байта) інформації та можуть досягати величезних значень від `-3.4028235E + 38` до `3.4028235E + 38`.

Точність дрібних чисел типу *float* становить 6-7 десяткових знаків. Мається на увазі загальна кількість цифр, а не кількість знаків після коми.

*double* — займають 4 байта. Аналогічні змінним *float*.

Створення (оголошення) *масиву*. Масив — це область пам'яті, де можуть послідовно зберігатися кілька значень.

```
int myInts[6];  
int myPins[] = {2, 4, 8, 3, 6};  
int mySensVals[6] = {2, 4, -8, 3, 2};  
char message[6] = "hello";
```

*string* — текстовий рядок. Може бути оголошений двома способами: можна використовувати тип даних *string* або оголосити рядок як *масив символів char* з нульовим символом в кінці.

```
char Str1 [15];  
char Str2 [8] = { 'a', 'r', 'd', 'u', 'i', 'n', 'o'};  
char Str3 [8] = { 'a', 'r', 'd', 'u', 'i', 'n', 'o', '\ 0'};  
char Str4 [] = "arduino";  
char Str5 [8] = "arduino";  
char Str6 [15] = "arduino".
```

Рядки завжди оголошуються в подвійних лапках ("Abc"), а символи завжди оголошуються в одинарних лапках ('A').