НАВЧАЛЬНО-МЕТОДИЧНА ЛІТЕРАТУРА

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет

імені Івана Пулюя

Кафедра автоматизації технологічних

процесів та виробництв

Методичні вказівки

до лабораторної роботи №10 «Використання програмного середовища Arduino IDE для програмування мікроконтролерів AVR»

з дисципліни

«Проектування мікропроцесорних систем керування технологічними процесами»

Тернопіль 2022

Методичні вказівки до лабораторної роботи №10 "Використання програмного середовища Arduino IDE для програмування мікроконтролерів AVR" з дисципліни "Проектування мікропроцесорних систем керування технологічними процесами"/Медвідь В.Р., Пісьціо В.П. – Тернопіль: ТНТУ, 2022. – 22 с.

Відповідальні за випуск доцент, к.т.н. Медвідь В.Р.,
асистент Пісьціо В.П.

Методичні вказівки призначені для студентів спеціальності 151 "Автоматизація та комп'ютерно-інтегровані технології", котрі вивчають курс "Проектування мікропроцесорних систем керування технологічними процесами". Методичні вказівки можуть бути корисні для інших приладобудівних спеціальностей, а також для інженерів, що займаються розробкою мікропроцесорних систем на основі Arduino.

Лабораторна робота №10

Використання програмного середовища Arduino IDE для програмування мікроконтролерів AVR

Середовище Arduino IDE

Процес написання програм для МК AVR як і для будь-яких інших, складається з декількох етапів:

- ◊ підготовка вихідного тексту програми на вибраній мові програмування;
- ◊ компіляція програми;
- ◊ налагодження й тестування програми;
- ◊ остаточне програмування мікроконтролера.

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.

Базовий набір програмного забезпечення (компілятор асемблера, ПЗ для програмування) поширюється фірмою Atmel безкоштовно.

Етапи розробки програмного забезпечення мікроконтролерів AVR наступні:

- ◊ написання програми на вибраній мові програмування;
- ◊ програмування віртуального мікроконтролера AVR з використанням програмних симуляторів, відлагодження програми;
- ◊ перевірка роботи програми на реальному пристрої;
- ◊ програмування реального мікроконтролера.

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

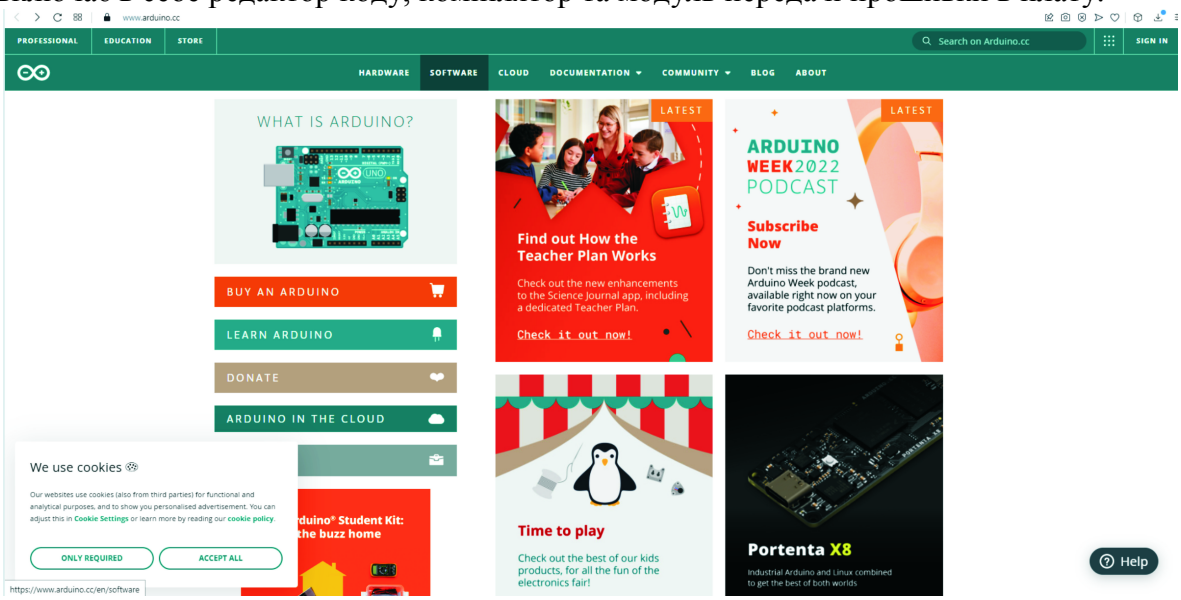

Рис. 1

Щоб почати використовувати Arduino IDE, потрібно зайти на сайт https://www.arduino.cc (рис. 1), перейти на вкладку SOFTWARE, на якій вибрати на полі «DOWVNLOAD OPTIONS» потрібну операційну систему (рис. 2), та завантажити середовище розробки Arduino, скориставшись вкладкою «JUST DOWNLOAD» (рис.3).

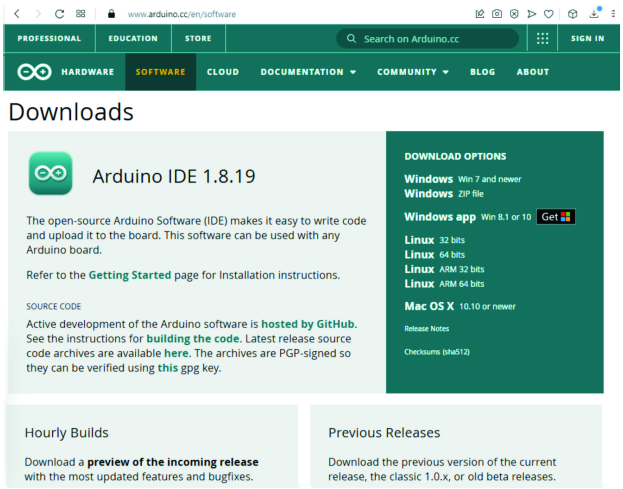

Рис. 2

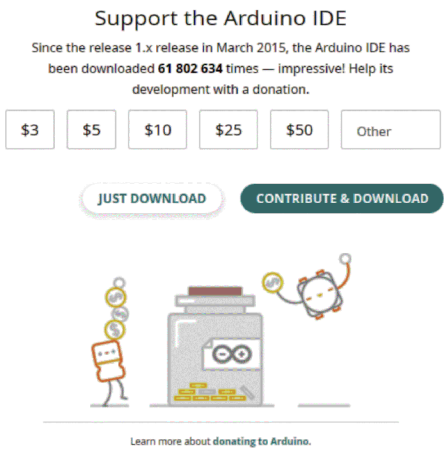

Рис. 3

Після завантаження програми її необхідно інсталювати у запропоновану на початку інсталяції директорію (рис.4)

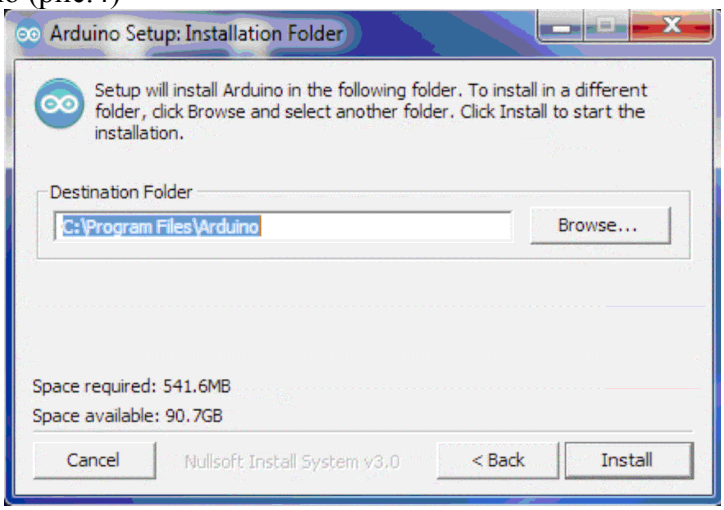

Рис. 4

Після закінчення інсталяції закрити вікно «Arduino Setap» (рис.5).

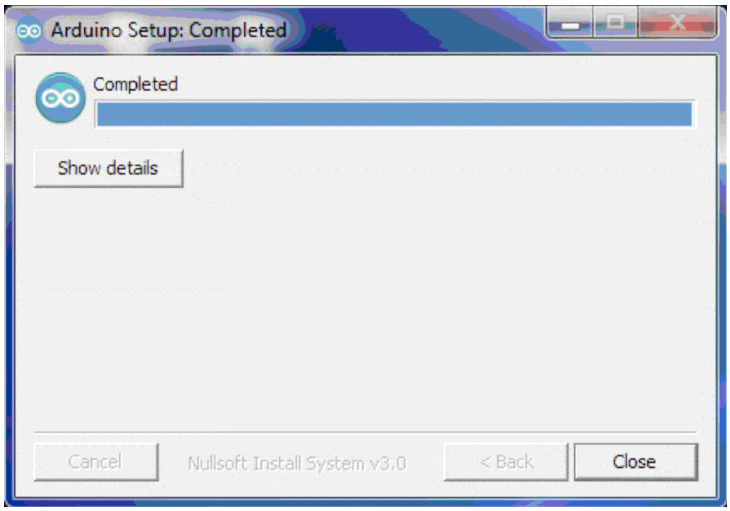

Рис. 5

На робочому столі комп’ютера з'явиться іконка Arduino Ide. 

Далі необхідно інсталювати драйвер для плати Arduino, яка має бути підключена до комп’ютера через порт USB.

Для цього потрібно:

◊ відкрити програму Arduino і натиснути на вкладку «Tools» («Інструменти»). У вікні, що розкриється, вибрати потрібну версію плати (рис.6).

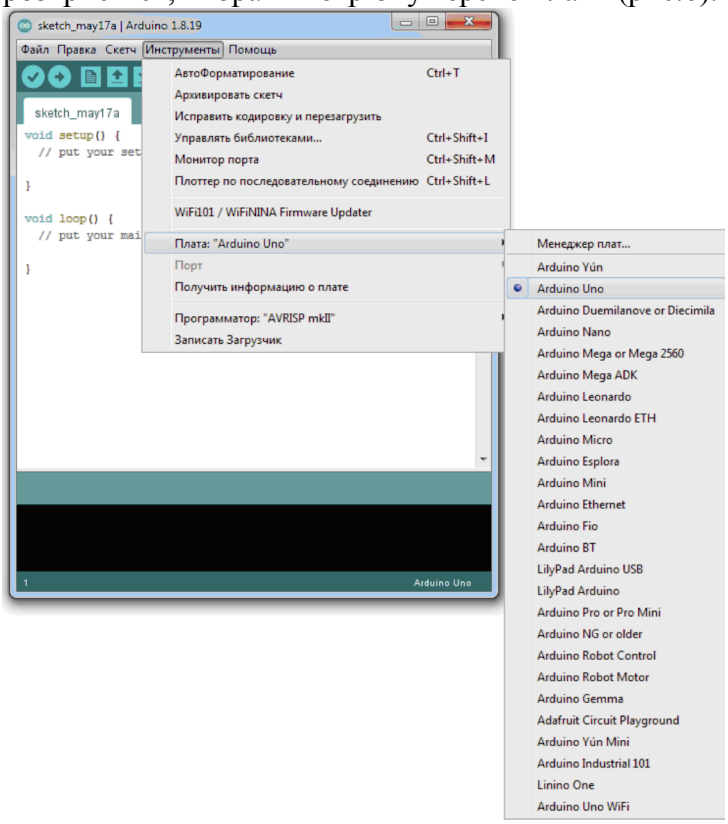

Рис. 6

У прикладі це плата «Arduino Uno», зображення якої подано на рис.7;

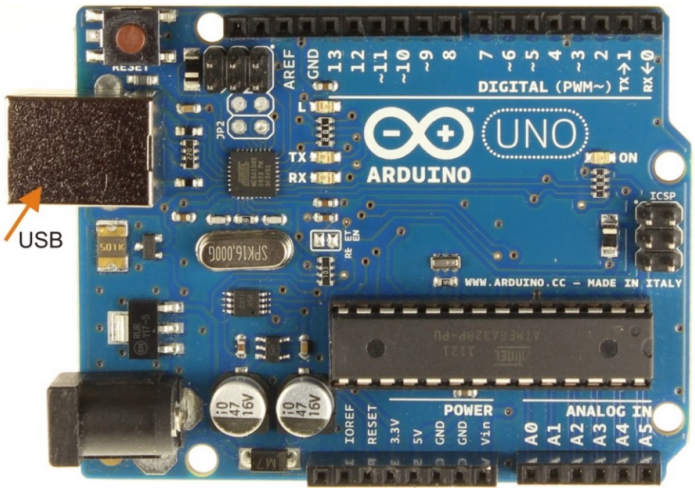

Рис. 7

◊ підключити плату до комп'ютера через порт USB і дочекатися, поки Windows не почне процес встановлення драйверів. Після встановлення драйверів плата буде готовою до роботи (рис. 8).

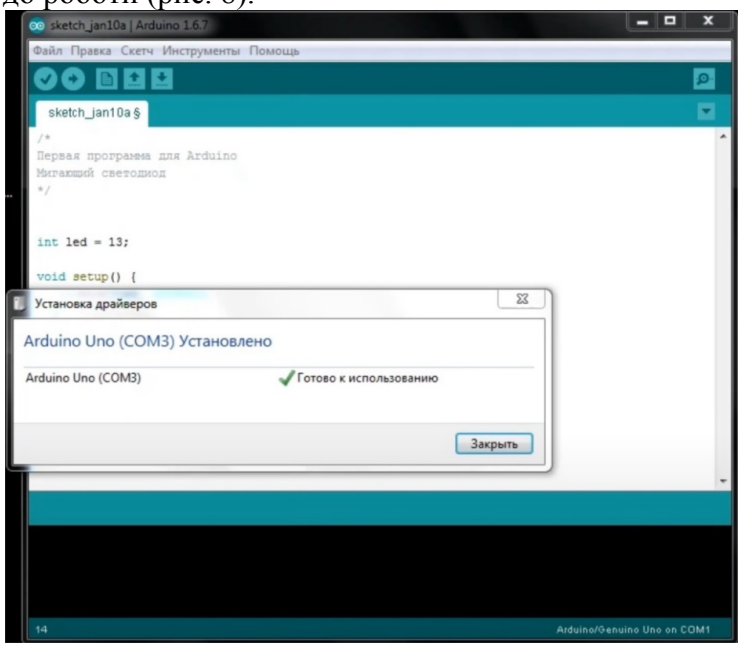

Рис. 8

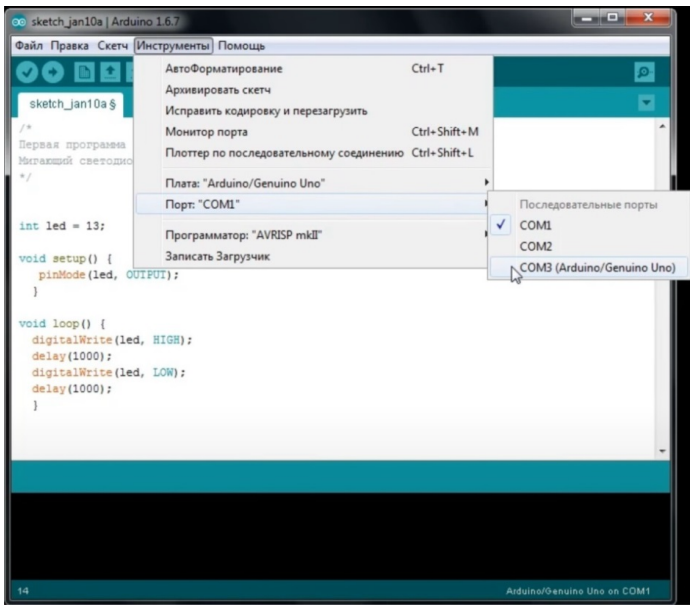

Рис. 9

Порт для під’єднання плати вибирається через вкладку «Tools», як показано на рис. 9.

Як правило, вибирають порт COM3.

Якщо ж системі не вдається під’єднати плату, тоді необхідно:

1) Зайти на вкладку «Пуск» комп’ютера і відкрити «Панель керування»;

2) У «Панелі керування» перейти на сторінку «Система і безпека» та клацнути по пункту «Система» і відкрити «Диспетчер пристроїв»;

3) Знайти розділ «Порти (COM & LPT)». У ньому буде видно відкритий порт під ім'ям «Arduino UNO (COMxx)»;

4) Клацнути правою кнопкою по пункту «Arduino UNO (COMxx)» і вибрати «Оновити драйвер»;

5) У вікні, що відкриється, вибрати пункт «Виконати пошук драйверів на цьому комп'ютері»;

6) Вибрати файл драйвера під ім'ям «*arduino.inf*», що розташований в папці «Drivers» в директорії завантаженого програмного забезпечення Arduino. Windows завершить інсталяцію драйвера;

7) У меню Tools>Board необхідно вибрати пункт меню, що відповідає потрібній моделі Arduino;

8) У меню Tools>Serial Port вибрати послідовний порт, до якого буде підключена плата Arduino (COM-порт з номером 3 (COM3) або вище (COM1 і COM2 асоційовані з апаратними портами);

9) Відкрити середовище розробки Arduino та вибрати тестову програму File > Examples > 1.Basics > Blink (рис. 10). Ця програма змушує мигати світлодіод, розташований на платі Arduino Uno, який апаратно під’єднаний до виводу 13 мікроконтролера.

10) Запустити програму на її компілювання та виконання, натиснувши на вкладку із стрілкою вправо;

11) Програма буде завантажена в мікроконтролер плати і виконана. Діод на платі почне мигати.

Тепер можна завантажувати скетчі в Arduino.

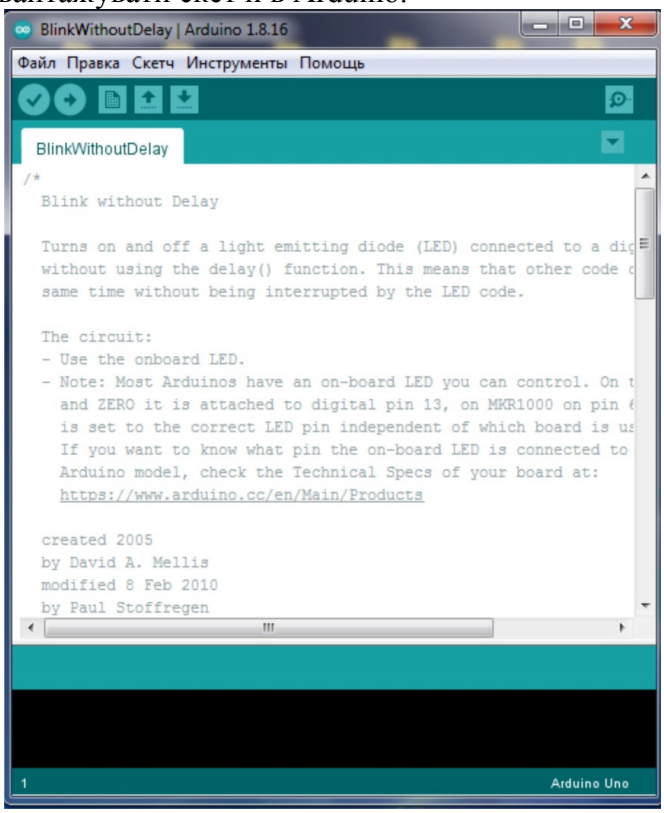

Рис. 10

Робота у середовищі Arduino Ide

Середовище розробки Arduino Ide (рис. 11) 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 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 виниклі 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.

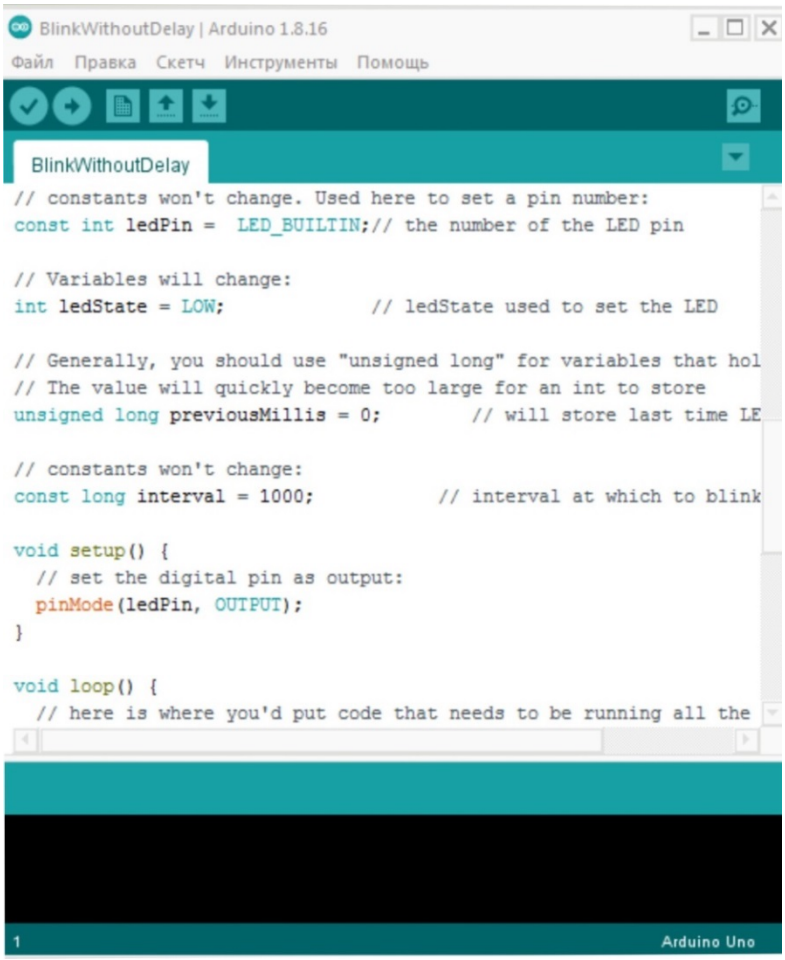

Рис. 11. Загальний вигляд програми Arduino IDE

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 необхідно підключити файл типу .HEX. Для його отримання потрібно вибрати пункт меню Sketch →Export Compiled Binary. HEX файл буде створений у директорії зі скетчем.

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. 12).

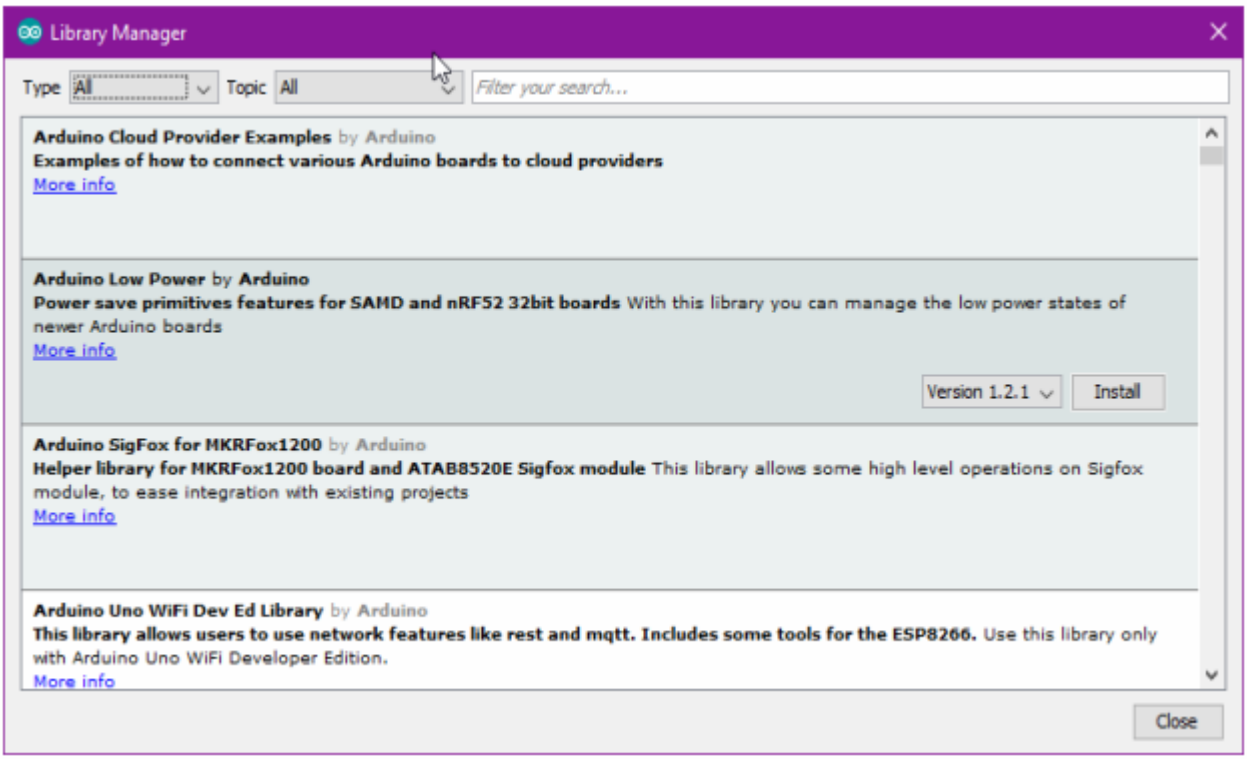

Рис. 12. Менеджер завантаження бібліотек

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 знаходиться в папці / libraries / DateTime папки блокнота.

Програмування

Середовище розробки засновано на мові програмування Processing, котра, власне, є мовою C/C++, доповнена 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 С.
Дані:
типи даних з мови С.
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 як ==. 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*. Усередині 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 1:

```
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 символьне 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 зберігаются 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 байти), 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 додатні 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 байти) 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 байти. 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 - в одинарних лапках ('A').

Апаратне забезпечення

Порти вводу/виводу AVR. Програмний ввід/вивід інформації

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 наступні 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 одиниці або нуля. Вихід стає свого роду керованим джерелом живлення та видає максимальний струм (20 мА та 40 мА в піковому значенні) 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 прочитати значення (int);

– значення, що повертаються HIGH або LOW.

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) (рис.13).

За замовчуванням (після ініціалізації мікроконтролера) всі порти налаштовані як входи (INPUT).

Кожен порт контролюється трьома регістрами, кожен з яких відповідає за певний стан.

Регістр **DDR** визначає, яка лінія порту буде працювати на ввід, а яка - на вивід.

Регістр **PORT** встановлює біт порту у відповідний стан HIGH або LOW.

Регістр **PIN** містить поточний стан вхідного порту.

Регістри DDR та PORT можуть бути як прочитані, так і записані.

Регістр PIN містить стан вхідних портів, тому може бути лише прочитаний.

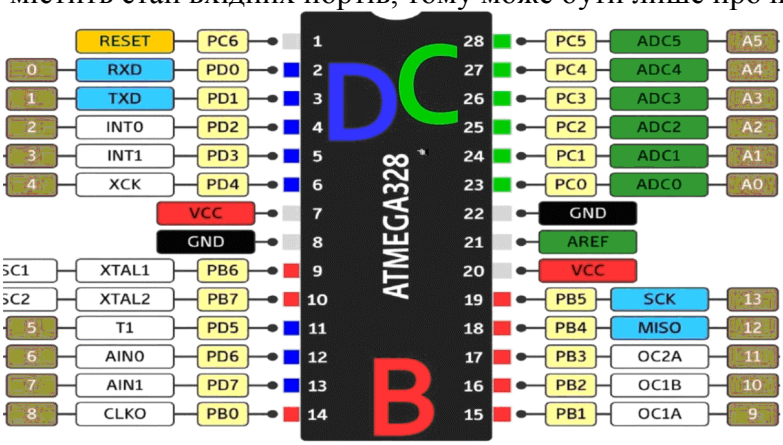

Рис. 13. Виводи мікроконтролера ATMEGA 328

Наприклад:

DDRD - регістр напряму пересилання даних через порт D.

PORTD – регістр, в який записуються дані для пересилання через лінії 0…7 порту D.

PIND – регістр, що містить вхідні дані порту D.

PORTB містить шість ліній вводу/виводу. Два старших біти (6 та 7), до яких під’єднуються виводи кварцу, для обміну даними не використовуються.

DDRB - регістр напряму пересилання даних через порт B.

PORTB - регістр, в який записуються дані для пересилання через лінії 0…5 порту B.

PINB - регістр, що містить вхідні дані порту B.

PORTC є лініями аналогових виводів 0…5.

DDRC - регістр напряму обміну даними через порт C.

PORTC - регістр даних порту C.

PINC - регістр вхідних даних порту С.

Слід пам'ятати, що молодші виводи порту D (D0 та D1) задіяні послідовним портом, тому робота з ними можлива тільки в тому випадку, якщо налагодження та послідовний порт не потрібні.

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;
```

Таймери/лічильники. Модуль переривань

У ATMega 328 передбачені три таймери/лічильники, на яких реалізовані функції часу, які використовують для формування та вимірювання часових інтервалів.

Основні функції часу:

delay (ms).

Параметри:

– ms - кількість мілісекунд, на які необхідно призупинити програму;

– значення, що повертаються - немає;

– опис: припиняє виконання програми на вказаний проміжок часу (в мілісекундах).

delayMicroseconds (us).

Параметри:

– us - кількість мікросекунд, на які необхідно призупинити програму;

– значення, що повертаються – немає;

опис: припиняє виконання програми на вказаний проміжок часу (в мікросекундах). Найбільше число для формування затримки - 16383.

millis ():

Параметри - немає;

– значення, що повертаються - кількість мілісекунд, що пройшли з моменту старту програми;

– опис: повертає кількість мілісекунд, що пройшли з моменту старту програми Arduino. Число, що повертається, скинеться в 0 через приблизно 50 днів.

micros():

Параметри - немає;

– значення, що повертаються. кількість мікросекунд, що минули з моменту старту програми;

– опис: повертає кількість мікросекунд, що минули з моменту початку виконання програми Arduino. Число, що повертається, скинеться в 0 через приблизно 70 хвилин. Роздільна здатність цієї функції становить чотири мікросекунди.

pulseIn (pin, value) / pulseIn (pin, value, timeout):

Параметри:

– pin : номер виводу, на якому буде очікуватися сигнал;

– value: тип імпульсу (HIGH або LOW);

– timeout (опціонально): час очікування імпульсу в мікросекундах (значення за замовчуванням - одна секунда);

– значення, що повертаються - тривалість імпульсу (в мікросекундах) або 0 в разі відсутності імпульсу протягом тайм-аута;

– опис: зчитує тривалість імпульсу (будь-якого. HIGH або LOW) на виведення.

Наприклад, якщо задане значення value = HIGH, то функція *pulseIn ()* очікує появи на виведення сигналу HIGH, потім вимірює час та чекає перемикання в стан LOW, після чого зупиняє відлік часу.

Функція повертає тривалість імпульсу в мікросекундах, або 0 в разі відсутності

імпульсу протягом певного часу очікування.

Функція працює з імпульсами тривалістю від 10 мікросекунд до 3 хвилин.

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);
  }
}
```

Переривання

Це процедури, що переривають нормальний перебіг програми. Вони використовуються для апаратних пристроїв, що вимагають негайної реакції на появу подій. Обробка переривань у мікроконтролері відбувається за допомогою модуля переривань, який приймає запити переривання й організовує перехід до виконання визначеної програми.

Запити переривання можуть надходити як від зовнішніх джерел, так і від джерел, розташованих у різних внутрішніх модулях мікроконтролера.

Як входи для прийому запитів від зовнішніх джерел, найчастіше використовуються виводи паралельних портів вводу/виводу, для яких ця функція є альтернативною.

Джерелами запитів зовнішніх переривань також можуть бути будь-які зміни зовнішніх сигналів на деяких спеціально виділених лініях портів вводу/виводу.

Arduino надає свої функції для роботи з зовнішніми перериваннями. Ці функції оголошені у файлі:\ *Hardware* \ *cores* \ *arduino* \ *wiring.h* та реалізовані в файлі: \ *Hardware* \ *cores* \ *arduino* \ *WInterrupts.c.*

Також дозволяється використовувати всі переривання конкретного мікроконтролера за допомогою макроса IRS. Однак останній метод потребує від користувача знання про наявні переривання та їх особливості у конкретній системі.

Функції переривання Arduino

attachInterrupt (interrupt, function, mode):

Параметри:

– interrupt: номер переривання (0 - pin D2, 1- pin D3);

– function: функція, яку необхідно викликати при виникненні переривання; ця функція повинна бути без параметрів і не повертати ніяких значень (таку функцію іноді називають оброблювачем переривання);

– mode: визначає умову, за якої має спрацьовувати переривання. Може приймати одне з чотирьох визначених значень: LOW. переривання буде спрацьовувати щоразу, коли на виводі присутній низький рівень сигналу, CHANGE. Переривання буде спрацьовувати щоразу, коли змінюється стан виводу, RISING. переривання спрацює, коли стан виводу зміниться з низького рівня на високий, FALLING. Переривання спрацює, коли стан виводу зміниться з високого рівня на низький;

– значення, що повертаються - немає.

detachInterrupt (pin):

Параметри - немає;

– значення, що повертаються - немає;

– опис: забороняє задане переривання. Забороняє переривання для того, щоб відключити доступ до даних в процесі виконання переривання.

interrupts ():

Параметри - немає;

– значення, що повертаються - немає;

– опис: повторно дозволяє переривання.

Таймери, як і зовнішні переривання, працюють незалежно від основної програми. У стандартних платах Arduino є три таймера Timer0, Timer1 і Timer2.

Timer0 є 8 бітним таймером, це означає, що його рахунковий регістр може зберігати числа до 255. Timer0 використовується стандартними часовими функціями Arduino такими як delay() і millis(), так що краще його не використовувати у своїх проектах.

Timer1 - це 16- бітний таймер з максимальним значенням 65535. Timer2 є 8-бітний і дуже схожий на Timer0. Він використовується в функції tone () Arduino.

Для обробки переривань у мові програмування Arduino використовується функція ISR ().

Програмування Arduino Ide на асемблері

Зазвичай, реальні випадки використання asm-вставок в Arduino не є частими. Але, якщо скетч написано у вигляді досить великої програми і при її компіляції з'являється ряд помилок, чи потрібно збільшити швидкість виконання програми, тоді це можна реалізувати за допомогою асемблера.

В принципі, Arduino Ide використовує в якості компілятора avr-gcc, тому всі команди асемблера для AVR ATMega працюють і тут.

Для вставки команди асемблера (asm-коду) необхідно використовувати конструкцію:

```
asm volatile
( "asm-code " "\n" );
```

Приведемо приклад написання скетчу з asm-вставкою (рис. 14).

Слід пам'ятати, що командою LDI можна завантажувати константи в регістри мікроконтролера від r16 до r31, які займають фізичні адреси в його пам'яті від 0x10 до 0x1F.

Для запису операнда в регістр, адреса якого є меншою за 0x10, необхідно спочатку записати його в один з регістрів r16…r31, а далі командою OUT вивести в потрібний регістр.

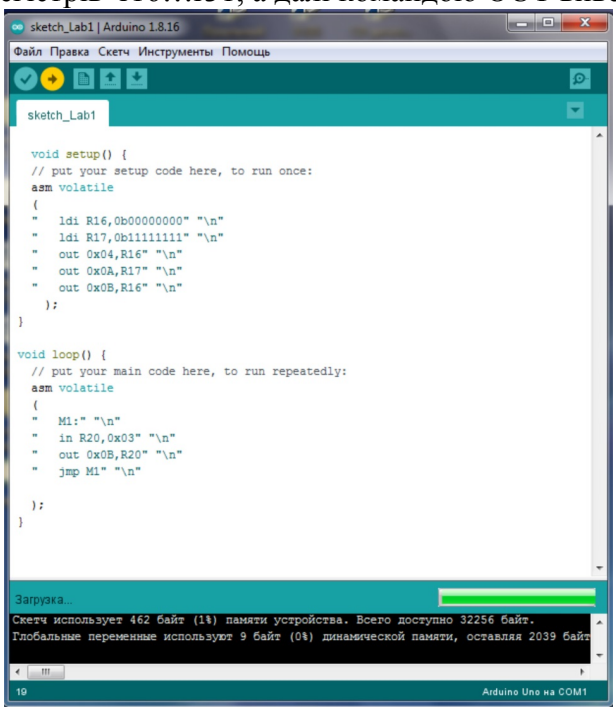

Рис. 14. Скетч для Arduino з asm-вставкою

Наприклад:

```
ldi r17,0b11111111      ; записати константу FF в регістр r17
out 0x0A,r17            ; вивести вміст r17 в регістр DDRD.
```

Для адресації регістрів в командах, які використовуються в тому числі для програмування портів чи обміну даними через них, необхідно використовувати їх фізичні адреси (наприклад, 0x0A для регістра DDRD), а не позначення (DDRD).

Тобто, якщо другу команду попереднього прикладу записати як

```
out DDRD,r17
```

компілятор Arduino Ide сприйме це як помилку.

Фізичні адреси всіх регістрів мікроконтролерів можна знайти в документації (DATASHEET). Для мікроконтролера ATMega 328 адреси портів приведені на рис. 15.

PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PINB – The Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

PORTC – The Port C Data Register

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

DDRC – The Port C Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PINC – The Port C Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

DDRD – The Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PIND – The Port D Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Рис. 15. Адреси регістрів портів мікроконтролера ATMega 328

Для відлагодження скетчу для схеми, побудованої на основі Arduino Uno за допомогою, наприклад, програмного середовища Proteus VSM чи іншого програмного симулятора, необхідно використовувати створений в результаті компілювання скетчу файл з розширенням .hex.

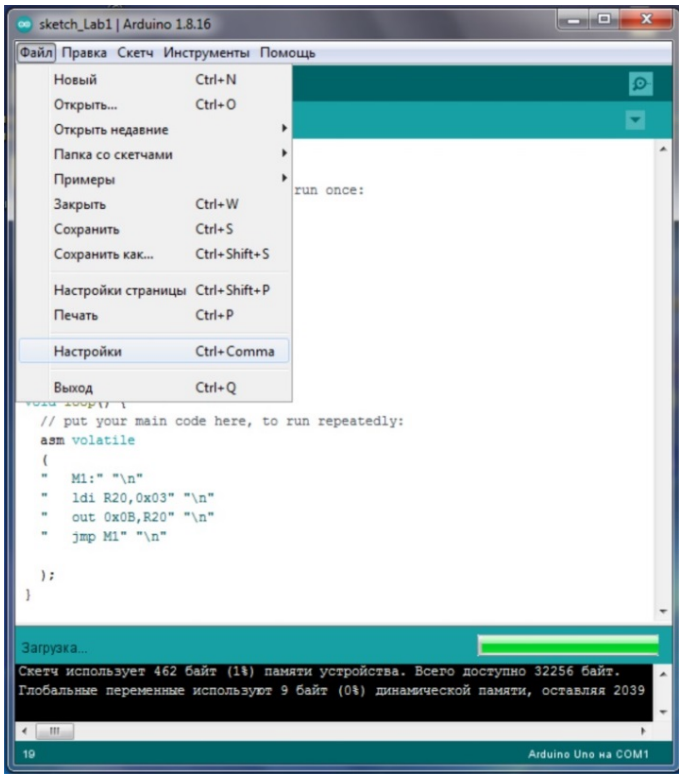

Рис. 16

Для цього необхідно перед компілюванням скетчу на Arduino Ide зайти на вкладку «Файл» і далі «Налаштування» (рис. 16).

У вікні, що відкриється (рис. 17), поставити галочку навпроти опції «Показати детальний вивід: Компіляція».

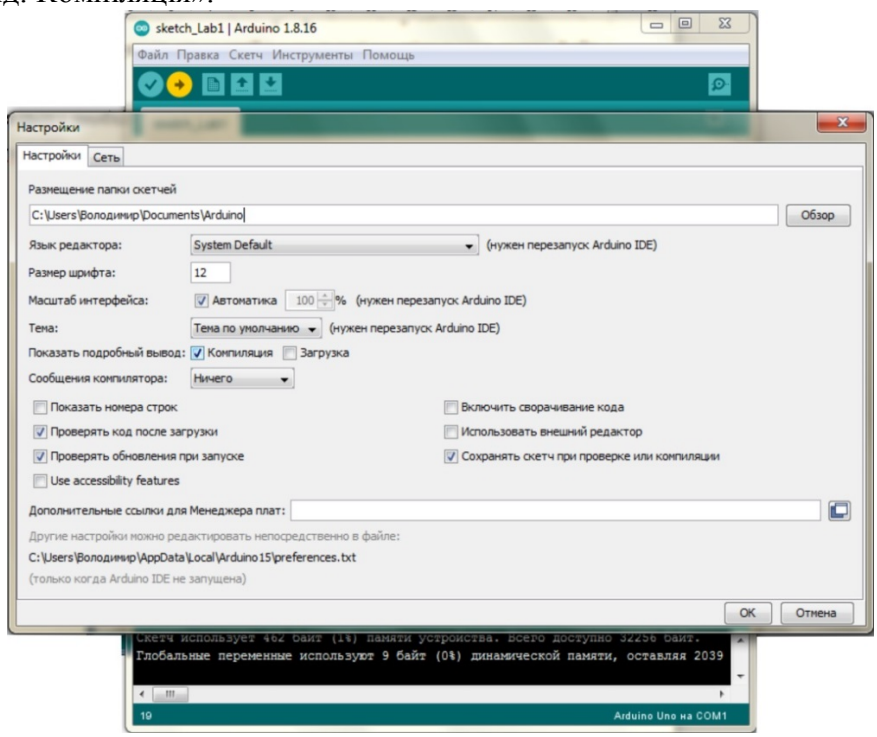

Рис. 17

Далі компілюємо скетч, натиснувши курсором на вкладці «Перевірити» (рис. 18), і серед ряду файлів, створених в процесі компілювання (вікно повідомлень Arduino Ide), знаходимо потрібний з розширенням .hex.

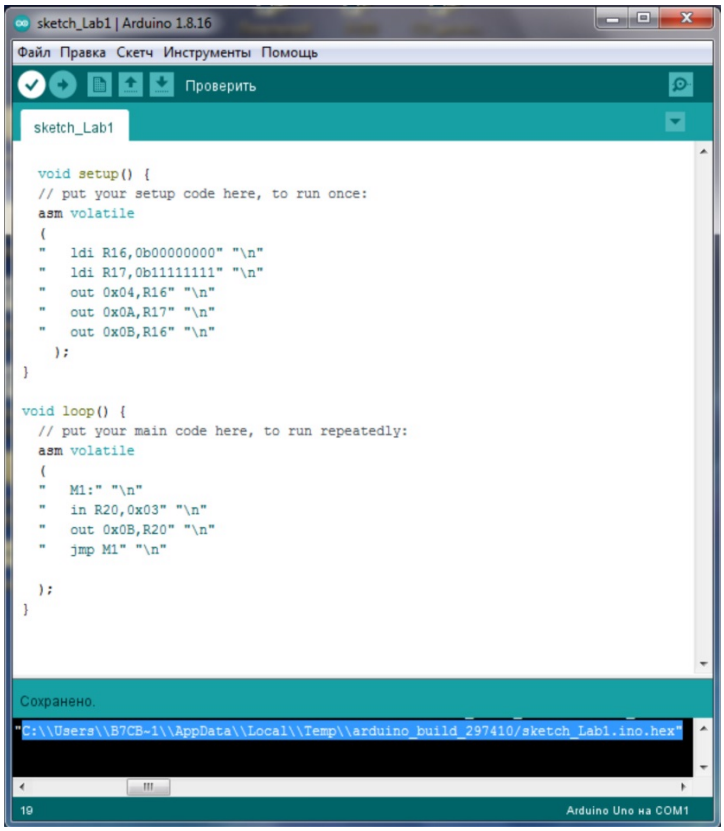

Рис. 18.

Копіюємо його адресу:

C:\\Users\\B7CB~1\\AppData\\Local\\Temp\\arduino_build_250134/sketch_Lab1.ino.hex"

В подальшому даний файл використовується для симуляції роботи схеми на основі Arduino, побудованої в середовищі програмного симулятора.

Виконання лабораторної роботи

Завдання №1

1. Запустити Arduino Ide і створити скетч за прикладом програми (рис. 14). Скомпілювати його, знайти серед файлів з результатами компілювання потрібний файл з розширенням .hex. Скопіювати посилання на нього (клавішами Ctrl-C; Ctrl-V).
2. Зробити «Print Screen» екрану монітора з результатом виконання лабораторної роботи і додайте його у звіт.
3. Скопіювати в текстовий файл скетч з робочою програмою і додати до кожної команди коментар про її призначення. Файл з програмою також додайте у звіт.

Завдання №2

1. Запустити Arduino Ide і створити скетч за прикладом програми (рис. 14), в якому запрограмувати лінії порту D на ввід, а лінії порту B - на вивід. Скомпілювати його, знайти серед файлів з результатами компілювання потрібний файл з розширенням .hex. Скопіювати посилання на нього (клавішами Ctrl-C; Ctrl-V).
2. Зробити «Print Screen» екрану монітора з результатом виконання лабораторної роботи і додайте його у звіт.
3. Скопіювати в текстовий файл скетч з робочою програмою і додати до кожної команди коментар про її призначення. Файл з програмою також додайте у звіт.

Завдання №3

1. Створити скетч для запалювання світлодіоду, під’єднаного до виводу 13 мікроконтролера за Прикладом 1.
2. Зробити «Print Screen» екрану монітора з результатом виконання лабораторної роботи і додайте його у звіт.
3. Скопіювати в текстовий файл скетч з робочою програмою і додати до кожної команди коментар про її призначення. Файл з програмою також додайте у звіт.

Завдання №4

1. Запустити Arduino Ide і створити скетч за прикладом програми з прикладу 2.

Програма демонструє роботу з 4-позиційним семисегментним індикатором з загальним анодом та регістрами портів Arduino: до порту D під'єднуються сегменти чотирьох індикаторів, які увімкнені паралельно, сегмент запалюється логічною «1», аноди індикаторів під'єднуються до порту B і вибираються почергово з часовою затримкою, причому, активним рівнем на виході порту B є логічний «0». Для семисегментних індикаторів із спільним катодом активним рівнем для вибору однієї з чотирьох його позицій буде логічна «1».

Алгоритм програми виводить на індикатор число 0123 в режимі динамічної індикації з частотою 200 Гц (50 Гц на кожну позицію).

Скомпілювати його, знайти серед файлів з результатами компілювання потрібний файл з розширенням .hex. Скопіювати посилання на нього (клавішами Ctrl-C; Ctrl-V).

Приклад 2 Лістинг програми

```
/* *--A--*
   F     B
   *--G--*
   E     C
   *--D--*  /G
A-PD0, B-PD1, C-PD2, D- PD3, E-PD4, F-PD5
G-PD6, D1-PB0, D2-PB1, D3-PB2, D4-PB3 */
#define P0 0B11111110;
#define P1 0B11111101;
#define P2 0B11111011;
#define P3 0B11110111;
void setup() {
  DDRD  = 0xFF;
  DDRB  = 0xFF;
  PORTD = 0x00;
  PORTB = 0<<0;
}
void loop() {
  PORTD=0B00111111; //0
  PORTB=P0;
  delay (500);
  PORTD=0B00000110; //1
  PORTB=P1;
  delay (500);
  PORTD=0B01011011; //2
  PORTB=P2;
  delay (500);
  PORTD=0B01001111; //3
  PORTB=P3;
  delay (500);
}
```

2. Зробити «Print Screen» екрану монітора з результатом виконання лабораторної роботи і додайте його у звіт.

3. Скопіювати в текстовий файл скетч з робочою програмою і додати до кожної команди коментар про її призначення. Файл з програмою також додати у звіт.

Завдання №5

1. Запустити Arduino Ide і створити скетч за прикладом програми (Приклад 3). Програма демонструє роботу з АЦП Arduino.

Алгоритм програми: значення напруги з потенціометра RV1 зчитується внутрішнім АЦП і приймає значення в діапазоні від 0 до 1023. Це значення використовується для формування частоти мигання світлодіоду HL1 та регулювання яскравості світіння світлодіоду HL2 (для цього отримане значення ділимо на 4 і отримаємо діапазон

регулювання яскравості від 0 до 255).

Приклад 3 Лістинг програми

```
void setup() {
 pinMode(12, OUTPUT); //підключення світлодіоду HL2
 pinMode(13, OUTPUT); // підключення світлодіоду HL1
 pinMode(A1, INPUT) ; // до входу A1 підключаємо потенціометр
}
void loop() {
 int val1 = analogRead(A1);
 int val2 = val1 / 4;
 analogWrite(12, val2);
 digitalWrite (13, HIGH);
 delay(val1) ;
 digitalWrite (13, LOW);
 delay(val1);
}
```

2. Зробити «Print Screen» екрану монітора з результатом виконання лабораторної роботи і додайте його у звіт.

3. Записати скетч Приклад 3 для випадку, коли світлодіоди під'єднуються до молодштх розрядів порту B, а напруга з потенціометра подається на вхід A0 мікроконтролера. Виконати програму.

4. Скопіювати в текстові файли скетчі з робочими програмами і додати до кожної команди коментар про її призначення. Файли з програмами також додати у звіт.

Література

1. Мікропроцесорні системи управління: навч. посіб./ В.О.Денисюк, С.М.Цирульник; Вінн. нац. аграр. ун-т. Вінниця: ТВОРИ, 2021. 204.

2. Методичні вказівки до лабораторних робіт з дисципліни «Електротехніка та електроніка» для студентів спеціальності 122 Комп'ютерні науки і інформаційні технології» денної форм навчання / Укл.: А.В. Пархоменко, О.М. Гладкова. – Запоріжжя: ЗНТУ, 2016. – 41 с.

3. Розробка програмних модулів для обміну даними у промислових мережах: [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» / Укладачі: А. В. Сагун, В. В. Хайдуров, І. А. Поліщук; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 16,2 МБайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 103 с.

Зміст