

ЗМІСТ

РОЗДІЛ 1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ARDUINO.....	5
1.1 Програмно-апаратний комплекс Arduino.....	5
1.2 Плати Arduino.....	7
1.3 Середовища розробки Arduino.....	20
РОЗДІЛ 2 ПРОГРАМУВАННЯ ARDUINO.....	25
2.1 Цифрове введення / виведення.....	31
2.2 Функції часу.....	32
2.3 Асинхронний послідовний обмін.....	35
2.4 Переривання.....	41
2.5 Клас String.....	44
РОЗДІЛ 3 ОСНОВНІ ІНТЕРФЕСИ ARDUINO.....	50
3.1 Двопроводний послідовний інтерфейс TWI (I ² C).....	50
3.2 Інтерфейс 1-Wire.....	57
РОЗДІЛ 4 ЛАБОРАТОРНИЙ ПРАКТИКУМ.....	67
4.1 Лабораторна робота №1. Вивчення роботи портів вводу-виводу плати Arduino.....	67
4.2 Лабораторна робота №2. Вивчення роботи послідовного порту мікроконтролера плати Arduino.....	70
4.3 Лабораторна робота №3. Робота з рядками в програмованих мікроконтролерах Arduino.....	74
4.4 Лабораторна робота №4. Вивчення роботи переривань, ШІМ та АЦП програмованого мікроконтролера Arduino.....	79
4.5 Лабораторна робота №5. Програмування в Arduino. дослідження функціоналу дальнометра HCSR04.....	85
4.6 Лабораторна робота №6. Програмування Arduino. Дослідження роботи датчика температури і вологості.....	90
4.7 Лабораторна робота №7. Програмування та налагодження	

інтерфейсу 1-Wire між платою Arduino та іншими пристроями.....	94
4.8 Лабораторна робота №8. Дослідження роботи сервоприводу та фотоеlementу.....	97
4.9 Лабораторна робота №9. Створення мелодії з використанням Arduino.....	102
4.10 Лабораторна робота №10. Робота з LCD – дисплеєм по протоколу I2C.....	108
4.11 Лабораторна робота №11. Програмування годинника реального часу DS3231.....	113
4.12 Лабораторна робота №12. Створення вимірювача швидкості реакції.....	119
4.13 Лабораторна робота №13. Віддалене керування приладами.....	122
4.14 Лабораторна робота №14. Програмування звукового підсилювача класу D.....	128
ПЕРЕЛІК ПОСИЛАНЬ	139

РОЗДІЛ 1 ЗАГАЛЬНІ ВІДОМОСТІ ARDUINO

1.1 Програмно-апаратний комплекс Arduino

Arduino – апаратно-програмна платформа, основними компонентами якої є мікроконтролерна плата вводу / виводу і середовище розробки на мові Processing / Wiring.

Мова програмування Arduino запозичена з середовища програмування мультимедіа «Processing».

Arduino, як і інші аналогічні засоби (Parallax Basic Stamp, Raspberry, Netmedia's BX-24, Phidgets, MIT's Handyboard і ін.), має на меті звільнити користувача від необхідності заглиблюватися в деталі внутрішнього устрою мікроконтролерів, надавши йому простий і зручний інтерфейс для їх програмування.

Плата Arduino Uno (Nano) містить мікроконтролер Atmel AVR (ATmega328 або ATmega168) та елементну обв'язку для програмування та інтеграції з іншими схемами. На кожній платі обов'язково присутні: лінійний стабілізатор напруги 5 В і 16 МГц кварцевий генератор (в деяких версіях керамічний резонатор). У мікроконтролер попередньо прошитий завантажувач, тому зовнішній програматор не потрібен.

У деяких варіантах, таких як Arduino Mini або неофіційної Boarduino, для програмування потрібне підключення окремої плати USB-to-serial з кабелем.

Плати Arduino дозволяють використовувати більшу частину I/O виводів мікроконтролера в зовнішніх схемах. Наприклад, в платі Diecimila доступно 14 цифрових вводів / виводів, 6 з яких можуть видавати ШІМ сигнал, і 6 аналогових входів. Ці виводи доступні у верхній частині плати через 0,1 дюймові роз'єми типу «мама». На ринку доступні зовнішні плати розширення, відомі як «shields».

Arduino на відміну від інших систем надає ряд **переваг**:

1. Просте і зручне середовище програмування. Середовище програмування Arduino зрозуміле і просте для початківців, але при цьому досить гнучке для просунутих користувачів. Воно засноване на середовищі програмування Processing, що може бути зручно для викладачів. Завдяки цьому, студенти, які вивчають програмування в середовищі Processing, зможуть легко освоїти Arduino.

2. Розширюване програмне забезпечення з відкритим вихідним кодом. Програмне забезпечення Arduino має відкритий вихідний код, завдяки цьому досвідчені програмісти можуть змінювати і доповнювати його. Можливості мови Arduino можна також розширювати за допомогою C++ бібліотек. Завдяки тому, що він заснований на мові AVR-C, просунуті користувачі, що бажають розібратися в технічних деталях, можуть легко перейти з мови Arduino на C або вставляти AVR-C код безпосередньо в програми Arduino.

3. Розширюване відкрите апаратне забезпечення. Пристрої Arduino побудовані на базі мікроконтролерів Atmel ATmega328 і ATmega168. Завдяки тому, що всі схеми модулів Arduino опубліковані під ліцензією Creative Commons, досвідчені інженери і розробники можуть створювати свої версії пристроїв на основі існуючих. І навіть звичайні користувачі можуть збирати дослідні зразки Arduino для кращого розуміння принципів їх роботи і економії коштів.

4. Низька вартість. У порівнянні зі схожими апаратними платформами, плати Arduino мають відносно низьку вартість: готові модулі Arduino коштують не дорожче 50 \$, а можливість зібрати плату вручну дозволяє максимально заощадити кошти і отримати Arduino за мінімальну ціну.

5. Кросплатформеність. Програмне забезпечення Arduino працює на операційних системах Windows, Macintosh OSX і Linux, в той час, як більшість подібних систем орієнтовані на роботу тільки в Windows.

1.2 Плати Arduino

Плати можна розділити на *контролери, шілди і аксесуари*. *Контролери* - це найважливіша частина - плата, яка містить мікроконтролер і в яку записується виконувана програма. *Шілди* - це плати розширення, які містять ту чи іншу периферію, керовану контролером. Шілд одягається зверху на контролер, утворюючи своєрідний «бутерброд».

Контролери Arduino Uno, Arduino Leonardo, Arduino Pro - пристрої на основі 8-розрядного мікроконтролера.

Arduino Due – це пристрій на основі мікропроцесора Atmel SAM3X8E ARM Cortex-M3. Це перша плата Arduino на базі 32-розрядного мікроконтролера ARM.

Завдяки використанню 32-розрядної ядра ARM, Arduino Due багато в чому перевершує типові плати на базі 8-розрядних мікроконтролерів.

Найбільш суттєві відмінності полягають в наступному:

- 32-бітове ядро дозволяє обробляти 4х-байтові дані всього за один такт. Тактова частота - 84 МГц.
- Обсяг оперативної пам'яті SRAM складає 96 КБ.
- Обсяг флеш-пам'яті програм - 512 КБ.

Наявність DMA-контролера, що дозволяє розвантажити центральний процесор від виконання ресурсномістких операцій з пам'яттю.

Arduino YUN – це контролер із вбудованим Wi-Fi модулем під управлінням ОС Linux і системою команд Arduino. Arduino YUN є комбінацією класичного Arduino Leonardo (на базі мікроконтролера ATmega32U4) і Wi-Fi системи на кристалі, що працює під управлінням Linino (дистрибутив ОС GNU / Linux на основі OpenWRT для мікропроцесорів MIPS).

Arduino Robot – перша офіційна версія Arduino, в конструкції якого передбачено колеса. Робот складається з двох плат, кожна з яких містить свій мікропроцесор. Плата приводів (Motor Board) контролює роботу двигунів, в

той час, як керуюча плата (Control Board) зчитує показання датчиків і приймає рішення про подальші операції. Кожна з двох плат є повноцінним пристроєм Arduino, програмованим за допомогою середовища розробки Arduino IDE.

Arduino Esplora – це мікропроцесорний пристрій, спроектований на основі Arduino Leonardo. Esplora відрізняється від усіх попередніх плат Arduino наявністю безлічі вбудованих, готових до використання датчиків для взаємодії. Esplora має вбудовані звукові і світлові індикатори (для виведення інформації), а також кілька датчиків (для введення інформації), таких, як джойстик, слайдер, датчик температури, акселерометр, мікрофон і світловий датчик.

Arduino ADK – це пристрій на основі мікроконтролера ATmega2560. У ньому реалізований USB-хост для підключення смартфонів на базі операційної системи Android.

Плати розширення: Arduino GSM, Arduino Ethernet, Arduino Wi-Fi, Arduino Motor, Arduino Proto і т.д.

Arduino Nano



Рис. 1.1 – Плата Arduino Nano

Arduino Nano - це повнофункціональний мініатюрний пристрій на базі мікроконтролера ATmega328 (Arduino Nano 3.0) або ATmega168 (Arduino Nano 2.x), адаптований для використання з макетної платі. Arduino Nano розроблено і випускається фірмою Gravitech.

Характеристики:

- Мікроконтролер: Atmel ATmega168 або ATmega328

- Робоча напруга (логічний рівень): 5В
- Напруга живлення (рекомендована): 7-12В
- Напруга живлення (гранична): 6-20В
- Цифрові входи / виходи: 14 (з яких 6 можуть використовуватися як ШІМ-виходи)
- Аналогові входи: 8
- Максимальний струм одного виведення: 40 мА
- Flash-пам'ять: 16 КБ (АТmega168) або 32 КБ (АТmega328) з яких 2 КБ використовуються завантажувачем
- SRAM: 1 КБ (АТmega168) або 2 КБ (АТmega328)
- EEPROM: 512 байт (АТmega168) або 1 КБ (АТmega328)
- Тактова частота: 16 МГц
- Розміри плати: 1.85 см х 4.3 см

Arduino Nano може живитися через кабель Mini-B USB, від зовнішнього джерела живлення з нестабілізованою напругою 6-20В або зі стабілізованою напругою 5В. Пристрій автоматично вибирає джерело живлення з найбільшим напругою.

Входи і виходи. З використанням функцій *pinMode ()*, *digitalWrite ()* і *digitalRead ()* кожен з 14 цифрових виводів Arduino Nano може працювати в якості входу або виходу. Робоча напруга виводів - 5В. Максимальний струм, який може віддавати або споживати один вивід, становить 40 мА. Всі виводи пов'язані з внутрішніми підтягуючими резисторами (за замовчуванням відключеними) номіналом 20-50 кОм. Крім основних, деякі виводи Arduino можуть виконувати додаткові функції:

Послідовний інтерфейс: виводи 0 (RX) і 1 (TX). Використовуються для отримання (RX) і передачі (TX) даних по послідовному інтерфейсу. Ці виводи з'єднані з відповідними виводами мікросхеми-перетворювача USB-UART від FTDI.

Зовнішні переривання: виводи 2 і 3. Дані виводи можуть бути налаштовані в якості джерел переривань, що виникають при різних умовах: при низькому рівні сигналу, по фронту, по спаду або при зміні сигналу.

ШИМ: виводи 3, 5, 6, 9, 10 і 11. За допомогою функції `analogWrite ()` можуть виводити 8-бітові аналогові значення в вигляді ШИМ-сигналу.

Інтерфейс SPI: виводи 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Дані виводи дозволяють здійснювати зв'язок по інтерфейсу SPI. У пристрої реалізована апаратна підтримка SPI, проте на даний момент мова Arduino поки її не підтримує.

Світлодіод: Вбудований світлодіод, приєднаний до цифрового виводу 13. При відправці значення HIGH світлодіод включається, при відправці LOW – вимикається.

I2C: виводи 4 (SDA) і 5 (SCL). З використанням бібліотеки `Wire` дані виводи можуть здійснювати зв'язок по інтерфейсу I2C (TWI).

Крім перерахованих, на платі існує ще кілька виводів:

- *AREF.* Опорна напруга для аналогових входів. Може бути задіяна функцією `analogReference ()`.
- *Reset.* Формування низького рівня (LOW) на цьому виводу призведе до перезавантаження мікроконтролера. Зазвичай цей вивід служить для функціонування кнопки скидання на платах розширення.

Arduino Uno



Рис. 1.2 – Плата Arduino Uno

Arduino Uno – це пристрій на основі мікроконтролера ATmega328. У його склад входить все необхідне для зручної роботи з мікроконтролером: 14 цифрових входів / виходів (з них 6 можуть використовуватися в якості ШІМ-виходів), 6 аналогових входів, кварцовий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм для внутрішньосхемного програмування (ICSP) і кнопка скидання. Для початку роботи з пристроєм досить просто подати живлення від AC / DC-адаптера або батарейки, або підключити його до комп'ютера за допомогою USB-кабелю.

На відміну від всіх попередніх плат Arduino, Uno в якості перетворювача інтерфейсів USB-UART використовує мікроконтролер ATmega16U2 (ATmega8U2 до версії R2) замість мікросхеми FTDI.

Характеристики

- Мікроконтролер: ATmega328
- Робоча напруга: 5 В
- Напруга живлення (рекомендована): 7-12 В
- Напруга живлення (гранична): 6-20 В
- Цифрові входи / виходи: 14 (з них 6 можуть використовуватися в якості ШІМ-виходів)
 - Аналогові входи: 6
 - Максимальний струм одного виводу: 40 мА
 - Максимальний вихідний струм виводу: 3.3V 50 мА
 - Flash-пам'ять: 32 КБ (ATmega328) з яких 0.5 КБ використовуються завантажувачем
- SRAM: 2 КБ (ATmega328)
- EEPROM: 1 КБ (ATmega328)
- Тактова частота: 16 МГц

Arduino Uno може живитися від USB або від зовнішнього джерела живлення - тип джерела вибирається автоматично.

У якості зовнішнього джерела живлення (не USB) може використовуватися мережевий AC / DC-адаптер або акумулятор / батарея. Штекер адаптера (діаметр - 2.1мм, центральний контакт - позитивний) необхідно вставити у відповідний роз'єм живлення на платі. У разі живлення від акумулятора / батареї, її провід необхідно під'єднати до виводів Gnd і Vin роз'єму POWER.

Напруга зовнішнього джерела живлення може бути в межах від 6 до 20 В. Однак, зменшення напруги живлення нижче 7В призводить до зменшення напруги на виводі 5V, що може стати причиною нестабільної роботи пристрою. Використання напруги більше 12В може призводити до перегріву стабілізатора напруги і виходу плати з ладу. З огляду на це, рекомендується використовувати джерело живлення з напругою в діапазоні від 7 до 12В.

Виводи живлення, що розташовані на платі:

VIN. Напруга, що надходить в Arduino безпосередньо від зовнішнього джерела живлення (не пов'язане з 5В від USB або іншою стабілізованою напругою). Через цей вивід можна як подавати зовнішнє живлення, так і споживати струм, коли пристрій живиться від зовнішнього адаптера.

5V. На вивід надходить напруга 5В від стабілізатора напруги на платі, незалежно від того, як живиться пристрій: від адаптера (7 - 12В), від USB (5В) або через вивід VIN (7 - 12В). Живити пристрій через виводи 5V або 3V3 не рекомендується, оскільки в цьому випадку не використовується стабілізатор напруги, що може привести до виходу плати з ладу.

3V3. 3.3В, що надходять від стабілізатора напруги на платі. Максимальний струм, споживаний від цього виводу, становить 50 мА.

GND. Загальний мінусовий вивід.

IOREF. Цей вивід надає платам розширення інформації про робочу напругу мікроконтролера Arduino. Залежно від напруги, зчитуваної з виводу IOREF, плата розширення може переключитися на відповідне джерело живлення або задіювати перетворювачі рівнів, що дозволить їй працювати як з 5В, так і з 3.3В-пристроями.

Пам'ять

Обсяг флеш-пам'яті ATmega328 становить 32 КБ (з яких 0.5 КБ використовуються завантажувачем). Мікроконтролер також має 2 КБ пам'яті SRAM і 1 КБ EEPROM (з якої можна зчитувати або записувати інформацію за допомогою бібліотеки EEPROM).

Входи і виходи

З використанням функцій *pinMode ()*, *digitalWrite ()* і *digitalRead ()* кожен з 14 цифрових виводів може працювати в якості входу або виходу. Рівень напруги на выводах обмежений 5В. Максимальний струм, який може віддавати або споживати один вивід, становить 40 мА. Усі виводи пов'язані з внутрішніми підтягуючими резисторами (за замовчуванням відключеними) номіналом 20-50 кОм. Крім цього, деякі виводи Arduino можуть виконувати додаткові функції:

Послідовний інтерфейс: виводи 0 (RX) і 1 (TX). Використовуються для отримання (RX) і передачі (TX) даних по послідовному інтерфейсу. Ці виводи з'єднані з відповідними выводами мікросхеми ATmega8U2, яка виконує роль перетворювача USB-UART.

Зовнішні переривання: виводи 2 і 3. Можуть служити джерелами переривань, що виникають при фронті, спаді або при низькому рівні сигналу на цих выводах. Для отримання додаткової інформації див. Функцію *attachInterrupt ()*.

ШИМ: виводи 3, 5, 6, 9, 10 і 11. За допомогою функції *analogWrite ()* можуть виводити 8-бітові аналогові значення в вигляді ШІМ-сигналу.

Інтерфейс SPI: виводи 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Із застосуванням бібліотеки SPI дані виводи можуть здійснювати зв'язок по інтерфейсу SPI.

Світлодіод: Вбудований світлодіод, приєднаний до висновку 13. При відправці значення HIGH світлодіод включається, при відправці LOW - вимикається.

В Arduino Uno є 6 аналогових входів (A0 - A5), кожен з яких може отримувати аналогову напругу у вигляді 10-бітного числа (1024 різних значень). За замовчуванням, вимір напруги здійснюється в діапазоні від 0 до 5 В. Проте, верхню межу цього діапазону можна змінити, використовуючи вивід AREF і функцію *analogReference()*.

TWI: вивід A4 або SDA і вивід A5 або SCL. З використанням бібліотеки *Wire* дані виводи можуть здійснювати зв'язок по інтерфейсу TWI.

AREF. Опорна напруга для аналогових входів. Може бути задіяна функцією *analogReference()*.

Reset. Формування низького рівня (LOW) на цьому виводу призведе до перезавантаження мікроконтролера. Зазвичай цей вивід служить для функціонування кнопки скидання на платах розширення

Зв'язок з комп'ютером

Arduino Uno надає ряд можливостей для здійснення зв'язку з комп'ютером, ще одним Arduino або іншими мікроконтролерами. У ATmega328 є універсальний асинхронний приймач-передавач (UART), що дозволяє здійснювати послідовну передачу даних за допомогою цифрових виводів 0 (RX) і 1 (TX) та перетворювача USB-UART. Мікроконтролер ATmega16U2 на платі виконує функції такого перетворювача, і при підключенні до ПК, дозволяє Arduino включитися, як віртуальний COM-порт. Прошивка мікросхеми 16U2 використовує стандартні драйвера USB-COM, тому установка зовнішніх драйверів не потрібна. На платформі Windows необхідний тільки відповідний .inf-файл. У пакет програмного забезпечення Arduino входить спеціальна програма, що дозволяє зчитувати і відправляти на Arduino прості текстові дані. При передачі даних через мікросхему-перетворювач USB-UART під час USB-з'єднання з комп'ютером, на платі будуть блимати світлодіоди RX і TX.

Бібліотека *SoftwareSerial* дозволяє реалізувати послідовний зв'язок на будь-яких цифрових виводах Arduino Uno.

У мікроконтролері ATmega328 також реалізована підтримка послідовних інтерфейсів I2C (TWI) і SPI. У програмне забезпечення Arduino входить бібліотека Wire, що дозволяє спростити роботу з шиною I2C. Для роботи з інтерфейсом SPI використовують бібліотеку SPI.

Автоматичне (програмне) скидання

Щоб кожен раз перед завантаженням програми не було потрібно натискати кнопку скидання, Arduino Uno спроектований таким чином, що дозволяє здійснювати його скидання програмно, з підключеного комп'ютера. Один з виводів ATmega8U2 / 16U2, який бере участь в управлінні потоком даних (DTR), з'єднаний з виводом RESET мікроконтролера ATmega328 через конденсатор номіналом 100 нФ. Коли на лінії DTR з'являється нуль, вивід RESET також переходить в низький рівень на час, достатній для перезавантаження мікроконтролера. Дана особливість використовується для того, щоб можна було прошивати мікроконтролер всього одним натисненням кнопки в середовищі програмування Arduino. Така архітектура дозволяє зменшити таймаут завантажувача, оскільки процес прошивки завжди синхронізований зі спадом сигналу на лінії DTR.

Однак ця система може призводити і до інших наслідків. При підключенні Uno до комп'ютерів, що працюють на Mac OS X або Linux, його мікроконтролер буде скидатися при кожному з'єднанні програмного забезпечення з платою. Після скидання на Arduino Uno активізується завантажувач на час близько пів секунди. Незважаючи на те, що завантажувач запрограмований ігнорувати сторонні дані (тобто всі дані, які не стосуються процесу прошивки нової програми), він може перехопити кілька перших байт даних з посилки, що відправляється платі відразу після встановлення з'єднання. Відповідно, якщо в програмі, що працює на Arduino, передбачено отримання від комп'ютера будь-яких налаштувань або інших даних при першому запуску, треба переконатися, що програмне забезпечення, з яким взаємодіє Arduino, здійснює відправку через секунду після установа з'єднання.

На платі Uno існує доріжка (зазначена як "RESET-EN"), розімкнувши яку, можна відключити автоматичне скидання мікроконтролера. Для повторного відновлення функції автоматичного скидання необхідно спаяти між собою виводи, розташовані по краях цієї доріжки. Автоматичне скидання також можна вимкнути, підключивши резистор номіналом 110 Ом між виводом RESET і 5В.

Захист USB від перевантажень

В Arduino Uno є відновлювані запобіжники, що захищають USB-порт комп'ютера від коротких замикань і перевантажень. Незважаючи на те, що більшість комп'ютерів мають власний захист, такі запобіжники забезпечують додатковий рівень захисту. Якщо від USB-портом споживається струм більше 500 мА, запобіжник автоматично розірве з'єднання до усунення причин короткого замикання або перевантаження.

Arduino Mega

Плата Arduino Mega 2560 призначена для створення проектів, в яких не вистачає можливостей звичайних Arduino Uno. У цьому пристрої максимальна з усіх плат сімейства Arduino кількість виводів і розширений набір інтерфейсів. Також у Arduino Mega більше вбудованої пам'яті.



Рис. 1.3 – Плата Arduino Mega

Плата має 54 цифрових входів / виходів (14 з яких можуть використовуватися як виходи ШІМ), 16 аналогових входів, 4 послідовних

портів UART, кварцовий генератор 16 МГц, USB коннектор, роз'єм живлення, роз'єм ICSP і кнопка перезавантаження.

Характеристики

- Мікроконтролер: ATmega2560
- Робоча напруга: 5 В
- Вхідна напруга (рекомендована): 7-12 В
- Вхідна напруга (гранична): 6-20 В
- Цифрові Входи / Виходи: 54 (14 з яких можуть бути сконфігуровані як виходи ШІМ)
- Аналогові входи: 16
- Постійний струм через вхід / вихід: 40 мА
- Постійний струм для виводу 3.3 В: 50 мА
- Флеш-пам'ять: 256 КБ (з яких 8 КБ використовуються для завантажувача)
- ОЗУ: 8 КБ
- Незалежна пам'ять: 4 КБ
- Тактова частота: 16 МГц

Arduino Mega може отримувати живлення як через підключення по USB, так і від зовнішнього джерела живлення. Джерело живлення вибирається автоматично.

Зовнішнє живлення (не USB) може подаватися через перетворювач напруги АС / DC (блок живлення) або акумуляторною батареєю. Перетворювач напруги підключається за допомогою роз'єму 2.1 мм з позитивним полюсом на центральному контакті. Провід від батареї підключаються до виводів Gnd і Vin роз'єму живлення (POWER).

Платформа може працювати при зовнішньому живленні від 6 В до 20 В. При напрузі живлення нижче 7 В, вивід 5 В може видавати менше 5 В, при цьому платформа може працювати нестабільно. При використанні напруги вище 12 В регулятор напруги може перегрітися і пошкодити плату. Рекомендований діапазон від 7 В до 12 В.

Для обміну даними по USB плата Mega2560 використовує мікроконтролер Atmega8U2, запрограмований як конвертер USB-UART.

Виводи живлення:

VIN. Вхід використовується для подачі живлення від зовнішнього джерела (за відсутності 5 В від роз'єму USB або іншого регульованого джерела живлення). Подача напруги живлення відбувається через даний вивід.

5V. Регульоване джерело напруги, що використовується для живлення мікроконтролера і компонентів на платі. Живлення може подаватися від виводу VIN через регулятор напруги, або від роз'єму USB, або іншого регульованого джерела напруги 5 В.

3V3. Напруга на виводі 3.3 В генерується мікросхемою FTDI на платформі. Максимальне споживання струму 50 мА.

GND. Виводи заземлення.

Пам'ять

Мікроконтролер ATmega2560 має: 256 КБ флеш-пам'яті для зберігання коду програми (4 КБ використовується для зберігання завантажувача), 8 КБ ОЗУ і 4 КБ EEPROM (яка читається і записується за допомогою бібліотеки EEPROM).

Входи і Виходи

Кожен з 54 цифрових висновків Mega, використовуючи функції *pinMode ()*, *digitalWrite ()*, і *digitalRead ()*, може налаштовуватися як вхід або вихід. Виводи працюють при напрузі 5 В. Кожен вивід має підтягуючий резистор (стандартно відключений) 20-50 кОм і може пропускати струм до 40 мА. Деякі виводи мають особливі функції: Atmega2560 підтримує 4 порти послідовної передачі даних UART, а також інтерфейси I2C (TWI) і SPI.

Послідовні порти: Послідовний порт 0 (RX) і 1 (TX); Послідовний порт 1: 19 (RX) і 18 (TX); Послідовний порт 2: 17 (RX) і 16 (TX); Послідовний порт 3: 15 (RX) і 14 (TX). Виводи використовуються для

отримання (RX) і передачі (TX) даних TTL. Виводи 0 і 1 підключені до відповідних виводів мікросхеми послідовного порту ATmega8U2.

Зовнішнє переривання: 2 (переривання 0), 3 (переривання 1), 18 (переривання 5), 19 (переривання 4), 20 (переривання 3), і 21 (переривання 2). Дані виводи можуть бути налаштовані на виклик переривання або на молодшому значенні, або на передньому чи задньому фронті, або при зміні значення.

PWM: 2 до 13 і 44-46. Будь-який з виводів забезпечує ШІМ з роздільною здатністю 8 біт за допомогою функції *analogWrite ()*.

SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). За допомогою даних виводів здійснюється зв'язок SPI, наприклад, використовуючи бібліотеку SPI.

LED: 13. Вбудований світлодіод, підключений до цифрового виводу 13. Якщо значення на виведення має високий потенціал, то світлодіод горить.

I2C: 20 (SDA) і 21 (SCL). За допомогою виводів здійснюється зв'язок по інтерфейсу I2C (TWI). Для створення використовується бібліотека Wire.

На платформі Mega2560 є 16 аналогових входів, кожен з роздільною здатністю 10 біт (тобто може приймати 1024 різних значення). Стандартно виводи мають діапазон вимірювання до 5 В відносно мінусу живлення, проте є можливість змінити верхню межу за допомогою використання AREF і функції *analogReference ()*.

Додаткові виводи:

AREF. Опорна напруга для аналогових входів. Використовується з функцією *analogReference ()*.

Reset. Низький рівень сигналу на виводі перезавантажує мікроконтролер. Зазвичай застосовується для підключення кнопки перезавантаження на платі розширення, що закриває доступ до кнопки на самій платі Arduino.

1.3 Середовище розробки Arduino

Інтегроване середовище розробки Arduino IDE – це кросплатформовий додаток на Java, що включає в себе редактор коду, компілятор і 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.1.4).

Download the Arduino IDE

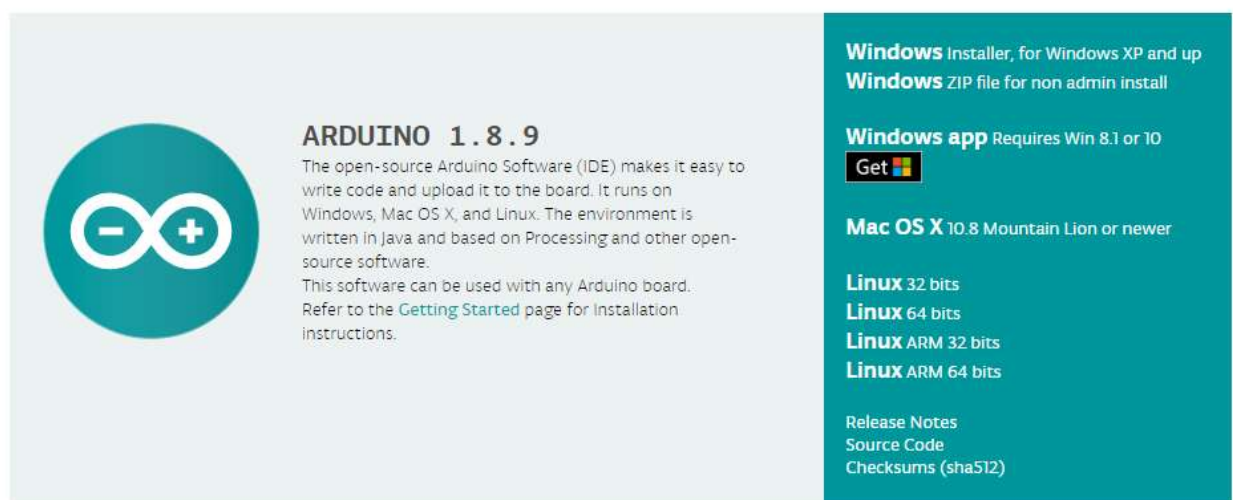


Рис.1.4 – Завантаження середовища розробки

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

- 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.
- Зайдіть в Пуск, відкрийте Панель керування.
- В Панелі керування перейдіть на сторінку Система і безпека. Далі клацніть по пункту Система і відкрийте Диспетчер пристроїв.
- Знайдіть розділ Порти (COM & LPT). У ньому ви побачите відкритий порт під ім'ям «Arduino UNO (COMxx)»

- Клацніть правою кнопкою по пункту «Arduino UNO (COMxx)» і виберіть «Оновити драйвер»
- Далі, у вікні, виберіть пункт «Виконати пошук драйверів на цьому комп'ютері»
- На завершення, виберіть файл драйвера під ім'ям «*arduino.inf*», розташований в папці «Drivers» в директорії завантаженого ПО Arduino. Якщо у вас стара версія IDE (1.0.3 або старіша) - вибирайте файл під ім'ям «*Arduino UNO.inf*»
- Windows завершить інсталяцію драйвера.
- Відкрити середовище розробки Arduino та запустити тестову програму File > Examples > 1.Basics > Blink.
- Тепер в меню Tools>Board необхідно вибрати пункт меню, що відповідає вашій моделі Arduino.
- У меню Tools>Serial Port виберіть послідовний порт, до якого підключена ваша плата. Як правило, це COM-порт з номером 3 (COM3) або вище (COM1 і COM2 зазвичай асоційовані з апаратними портами). Щоб дізнатися потрібний порт, можна тимчасово від'єднати Arduino і ще раз відкрити меню; зниклий порт і буде тим портом, з яким асоційований мікроконтролер. Назад підключіть пристрій до комп'ютера і виберіть з меню необхідний порт.
- Тепер можна працювати та завантажувати скетчі в Arduino (рис.1.4).

Опис Arduino IDE

Середовище розробки Arduino (рис.1.5) 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.

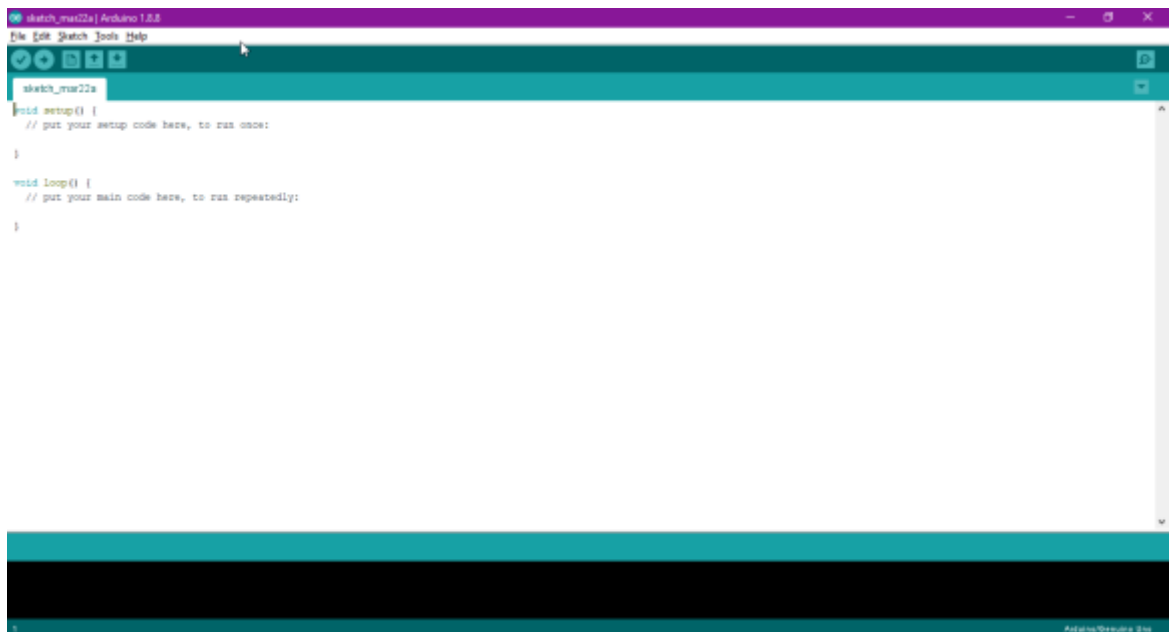


Рис.1.5 – Загальний вигляд програми

Програма, написана в середовищі Arduino, називається *скетч*. Скетч пишеться в текстовому редакторі, що має інструменти вирізки / вставки, пошуку / заміни тексту. Під час збереження і 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 і записати програму, створити, відкрити і зберегти скетч, відкрити моніторинг послідовної шини.

Блокном (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 і 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 старих платформах необхідно натиснути кнопку перезавантаження. 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).

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.1.6).

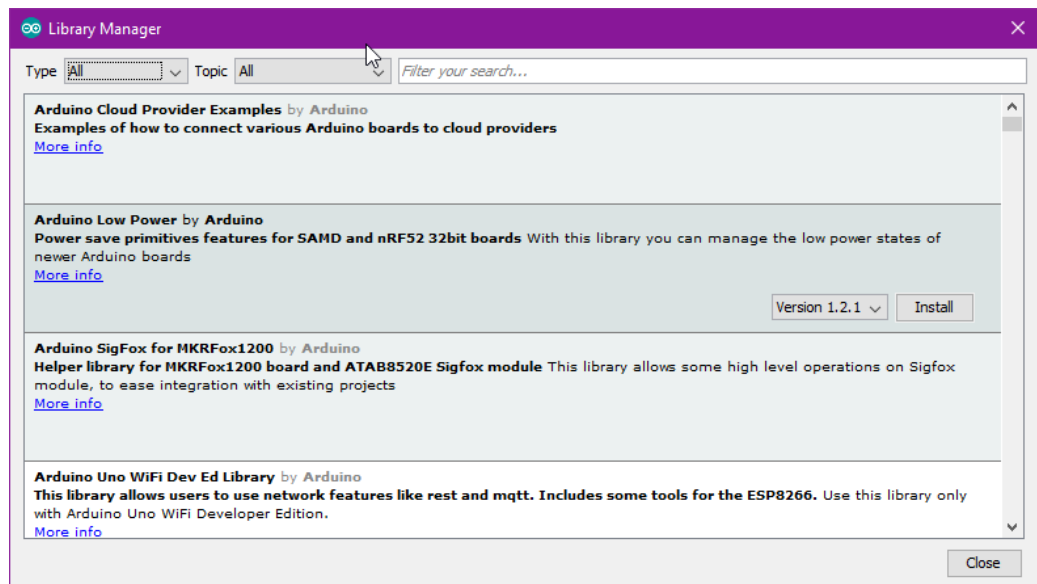


Рис. 1.6 – Менеджер завантаження бібліотек

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 викачаних бібліотек необхідно створити директорію «libraries» в папці блокнота і потім розпакувати архів. Наприклад, для встановлення бібліотеки DateTime її файли повинні знаходитися в папці / libraries / DateTime папки блокнота.

РОЗДІЛ 2 ПРОГРАМУВАННЯ ARDUINO

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

Разом з тим мова проста в освоєнні, і на даний момент Arduino – це, мабуть, найзручніший спосіб програмування пристроїв на мікроконтролерах.

Особливості мови Arduino

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

Оголошення змінної відбувається таким чином: спочатку вказується тип даних для цієї змінної а потім назва цієї змінної. Оператор присвоєння (=) – не є знаком рівності і не може використовуватися для порівняння значень. Оператор рівності записується як «подвійне одно» - ==. Присвоєння використовується для збереження певного значення в змінній. Наприклад, запис виду `a = 10` задає змінній `a` значення числа 10.

Цикли

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

```
for (дія до початку циклу; умова продовження циклу; дія в
кінці кожної ітерації циклу)
{
    інструкція циклу;
    інструкція циклу 2;
    інструкція циклу N;
}
```

Ітерацією циклу називається один прохід цього циклу

Коли ми не знаємо, скільки ітерацій повинен зробити цикл, нам знадобиться цикл *while* або *do ... while*. Синтаксис циклу *while* виглядає наступним чином:

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

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

Конструкція розгалуження

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)

Бітові оператори

& (Побітовое І)

| (Побітовое АБО)

^ (Побітовое XOR або виключаюче АБО)

~ (Побітовое НЕ)

<< (побітовий зсув вліво)

>> (побітовий зсув вправо)

Складні оператори

++ (інкремент)

-- (декремент)

+= (Складене додавання)

- = (складене віднімання)

* = (Складене множення)

/ = (Складене ділення)

& = (Складене побітовое І)

| = (Складене побітовое АБО)

Функції

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.

Функції *setup()*, *loop()*

Під час виклику функції *setup()*, програма ініціалізується і встановлює початкові значення. Функція *setup()* викликається, коли стартує скетч.

Використовується для ініціалізації змінних, визначення режимів роботи виводів, запуску використовуваних бібліотек і т.д. 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 Arduino

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 і зберігає символічне значення. Символи пишуться в одинарних лапках, наприклад: 'A' (сукупність символів - рядки - пишуться у подвійних лапках: "ABC").

unsigned char – Те ж саме, що і тип даних *byte*. Беззнаковий тип даних, що займає в пам'яті 1 байт.

int – цілочисельний тип даних. Це основний тип даних для зберігання чисел.

В Arduino Uno (і інших платах на базі 8-ти бітних мікроконтролерів) змінні типу `int` зберігають 16-бітові (2-байтові) значення. Така розмірність дає діапазон від -32768 до 32767.

unsigned int – в 8-ми бітних мікроконтролерів, змінні типу *unsigned int* (беззнакові цілі) містять двобайтові значення. Відмінність полягає в тому, що замість негативних чисел вони можуть зберігати лише позитивні значення в зручному діапазоні від 0 до 65535.

long – змінні типу *long* володіють розширеним розміром для зберігання чисел і 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 і 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 – для 8-ми бітних мікроконтролерів змінні типу *double* займають 4 байта. Аналогічні змінним *float*.

В Arduino Duo (32-х бітний) змінні *double* мають точність 8 байт (64 біта).

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`, який входить в ядро, починаючи з версії 0019; 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').

2.1 Цифрове введення / виведення

За замовчуванням всі порти Arduino визначаються як входи, і немає потреби описувати це в коді. Порти зазвичай прописуються в функції ініціалізації змінних.

Ініціалізація порту вводу-виводу *Arduino*:

1. *pinMode* (pin, mode)

- Параметри:

pin: номер виводу, режим роботи якого задається,

mode: приймає значення: INPUT – вхід, в цьому режимі відбувається зчитування даних з датчиків, стану кнопок, аналогового і 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 і видає максимальний струм (20 мА та 40 мА в піковому значенні) в навантаження, що до нього підключене. INPUT_PULLUP – порт

працює як вхід, але до нього підключається та званий підтягуючий резистор з номіналом 20 – 50 кОм.

- Значення, що повертаються – немає.

2. **digitalWrite (pin, value)**

- Параметри:

pin: номер виводу,

value: значення HIGH або LOW.

- Значення, що повертаються – немає.

3. **digitalRead (pin)**

- Параметри: pin: номер цифрового виводу, з якого необхідно вважати значення (int).

- Значення, що повертаються HIGH або LOW.

2.2 Функції часу

У ATmega328 передбачені три таймери/лічильники, на яких реалізовано функції часу, які використовують для формування і виміру часових інтервалів.

Основні функції часу:

1. **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). (В 1 секундi - 1000 мілісекунд.)

2. **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).

На даний момент найбільша кількість, що дозволяє сформувати точну затримку, - 16383. В майбутніх версіях Ардіуно цей показник може бути змінений. Для створення затримок тривалістю більше, ніж кілька тисяч мікросекунд, використовуйте функцію *delay ()*.

3. *millis ()*

- Параметри – Ні.
- 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, переповниться (скинеться в 0) через приблизно 50 днів.

4. *micros()*

- Параметри – Ні.
- 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, переповниться (скинеться в 0) через приблизно 70 хвилин. На платах Arduino з тактовою частотою 16 МГц (Duemilanove і Nano) роздільна здатність цієї функції становить чотири мікросекунди (тобто значення, що повертається, буде завжди кратне чотирьом). На платах Arduino з тактовою частотою 8 МГц (LilyPad), роздільна здатність функції становить вісім мікросекунд.

5. *pulseIn (pin, value)*

pulseIn (pin, value, timeout)

- Параметри: pin - номер виводу, на якому буде очікуватися сигнал; value: тип зчитує 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 таймаута.
- 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 і 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.

Емпіричним шляхом встановлено, що при використанні функції для вимірювання широких імпульсів можливе виникнення помилок. Функція працює з імпульсами тривалістю від 10 мікросекунд до 3 хвилин.

Приклад використання функції *millis ()*:

```
unsigned long time;
void setup () {
    Serial.begin (9600);
}
void loop () {
    Serial.print ( "Time:");
    time = millis ();
    // Виводимо час з моменту старту програми
    Serial.println (time);
    // Чекаємо 1 секунду, щоб не відправляти великий масив
даних
    delay (1000);
}
```


2.3 Асинхронний послідовний обмін

Найбільш розповсюджена форма в мікроконтроллерних системах — асинхронний обмін, при якому байт даних посиляється як пакет, що включає інформацію про початок і кінець передавання даних, а також інформацію для контролю помилок.

Першим передається не біт даних, а старт-біт, що вказує на початок передавання даних (початок пакета). Цей біт використовується приймачем для синхронізації процесу читання даних, що передаються за старт-бітом (молодший біт даних йде першим). Після бітів даних може впливати біт парності (контрольний біт), що використовується для перевірки правильності отриманих даних. Існує два типи перевірки на парність. Перевірка на непарність («odd») означає, що число одиниць у пакеті даних, включаючи біт парності, повинно бути непарним (наприклад, 0x55 буде мати біт парності рівним 1, щоб зробити число одиничних бітів рівним п'яти, тобто непарним). Перевірка на парність («even»), навпаки, означає, що число одиничних бітів повинно бути парним (наприклад, при передаванні числа 0x55 біт парності дорівнює 0).

За бітом парності передається стоп-біт, що використовується приймачем для обробки кінця передавання пакета.

Асинхронний пакет даних показаний на рис. 2.1. Існує набір параметрів, що повинний бути відомий при реалізації обміну. Одним з таких параметрів є число переданих бітів даних, що визначається типом приймального і передавального пристроїв. Пакет на рис. 2.1 містить тільки 5 біт даних (таке число бітів використовувалося в телетайпах), але можливі пакети довжиною до 8 біт.

Поряд з бітами парності («odd») чи непарності («even») можливі інші варіанти контрольних бітів: «no», «mark» і «space». «no» означає відсутність біта парності в пакеті. «mark» чи «space» означає, що замість біта парності завжди посиляється „1” («mark») чи „0” («space»), відповідно. Ці варіанти

контрольних бітів використовуються досить рідко - у тих випадках, коли необхідно дати приймачу додатковий час на оброблення пакета.

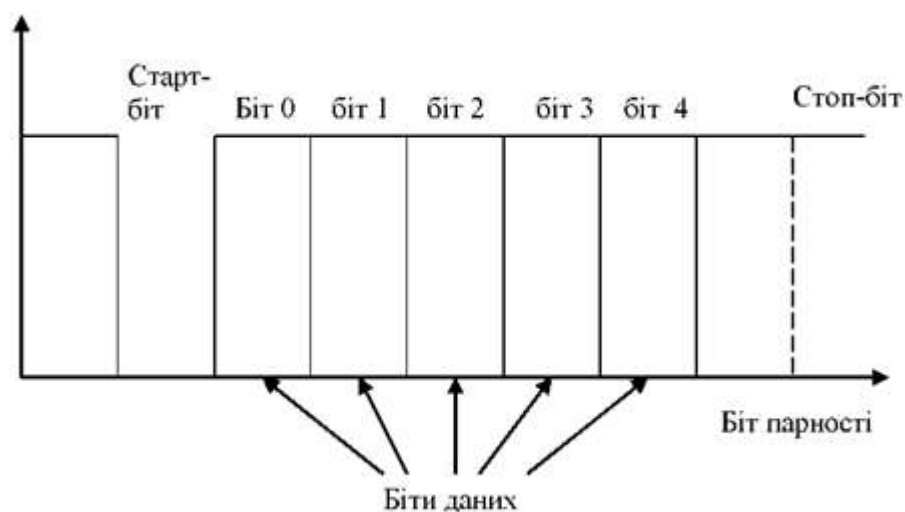


Рис. 2.1 – Асинхронне послідовне передавання даних

Кількість стоп-бітів також може бути різною. Другий стоп-біт може вводитися для тієї ж мети, що і контрольні біти «mark» і «space» - щоб дати приймачу більше часу для обробки прийнятого пакета.

Практично всі сучасні пристрої використовують для асинхронного обміну формат даних «8-N-1», що означає передачу 8 біт даних, відсутність біта парності й один стоп-біт. Біт парності і додатковий стоп-біт, звичайно, не потребуються для послідовного зв'язку.

Найбільш популярний протокол асинхронного послідовного зв'язку називається «RS-232», що у наш час є міжнародним стандартом. Це дуже старий стандарт, який використовували для зв'язку комп'ютерів через комунікаційні (COM) порти. Але і зараз стандарт «RS-232» широко використовується у вигляді з'єднання через віртуальні COM порти при використанні протоколу USB.

Організація обміну даними між платою Arduino і комп'ютером через USB

Для зв'язку з платою Arduino можна використовувати спеціальну програму моніторингу послідовного порту (Serial Monitor), вбудовану в програмне забезпечення Arduino. Монітор послідовної шини відображає дані, що посилаються в плату Arduino через віртуальний послідовний порт за допомогою USB. Для відправки даних вибирається швидкість передачі зі списку, відповідна значенню `Serial.begin` в скетчі. Потім необхідно ввести текст і натиснути кнопку Send або Enter.

Плата Arduino Nano має один послідовний порт (також відомий як UART або USART): Serial, що використовується для зв'язку з комп'ютером через USB. Він пов'язаний з цифровими виводами 0 (RX) і 1 (TX). Таким чином, під час роботи послідовного порту, виводи 0 і 1 не можуть використовуватися в якості цифрових входів або виходів.

Для забезпечення зв'язку плати Arduino з комп'ютером або іншими пристроями використовується клас Serial. Клас - це абстрактний тип даних. За допомогою класу описується деяка сутність (її характеристики і можливі дії). Наприклад, клас може описувати змінні, параметри і функції послідовного порту. Клас Serial містить близько 20 функцій. Розглянемо деякі.

Функції для роботи з послідовним портом плати Arduino:

1. *if (Serial)*

- Параметри – ні.
- Значення, що повертаються – boolean: повертає true, якщо вказаний послідовний порт готовий до роботи.
- Опис: дозволяє перевірити готовність певного послідовного порту.

2. *Serial.begin(speed)*

Serial.begin(speed, config)

- Параметри:

speed: швидкість в бітах на секунду (бодах) - long

config: задає кількість біт даних, перевірку парності и стопові біти:

- *SERIAL_5N1*
- *SERIAL_6N1*
- *SERIAL_7N1*
- *SERIAL_8N1* (за замовченням)
- *SERIAL_5N2*
- *SERIAL_8E2*
- *SERIAL_7O2*
- Опис: задає швидкість передачі даних по послідовному інтерфейсу в бітах в секунду (бодах). Для взаємодії з комп'ютером слід використовувати одну з попередньо встановлених швидкостей обміну: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 або 115200.

Другий аргумент цієї функції необов'язковий. Він дозволяє налаштувати кількість біт даних, перевірку парності і степових біти. За замовчуванням, посилка складається з 8 біт даних, без перевірки парності, з одним стоповим бітом.

3. *Serial.end ()*

- Параметри – ні.
- Значення, що повертаються – немає.
- Опис: функція розриває послідовний зв'язок, після чого виводи RX і TX знову можна використовувати як виводи загального призначення. Для відновлення послідовного з'єднання необхідно використовувати функцію *Serial.begin ()*.

4. *Serial.available()*

- Параметри – ні.
- Значення, що повертаються: кількість байт, доступних для зчитування.
- Опис: повертає кількість байт (символів) доступних для зчитування з буфера послідовного порту. Під символами розуміються дані, які вже

прийнятті і зберігаються в послідовному приймальному буфері (який може зберігати максимум 64 байта).

5. *Serial.read()*

- Параметри – ні.
- Значення, що повертаються: перший байт прийнятих даних (або -1, якщо таких нема) – int.
- Опис: зчитує дані, що надходять по послідовному інтерфейсу.

6. *Serial.print(val)*

Serial.print(val, format)

- Параметри: *val*: значення, яке необхідно вивести - будь-який тип даних; *format*: визначає систему числення (для цілочисельних типів), а також кількість десяткових знаків після коми (для чисел з плаваючою точкою).
- Значення, що повертаються: кількість виведених байт. Зчитування цього значення не є обов'язковим.
- Опис: функція виводить через послідовний порт заданий ASCII текст у вигляді, зрозумілому для людини. Ця команда може мати кілька різних форм. При виведенні числа кожній його цифрі відповідає один ASCII-символ. Дробові числа теж виводяться у вигляді ASCII-цифр, при цьому після коми за замовчуванням залишається два десяткових знака. Байти виводяться у вигляді окремих символів, а символи і рядки виводяться без змін – «як є».

Приклад:

Serial.print(78) - виведе "78"

Serial.print(1.23456) - виведе "1.23"

Serial.print('N') - виведе "N"

Serial.print("Hello world.") - виведе "Hello world."

Другий параметр необов'язковий. Він задає формат виведення; цей параметр може набувати таких значень: BIN (двійкова система з основою 2), OCT (восьмерична система з основою 8), DEC (десятькова система з основою 10), HEX (шістнадцятирічна система з основою 16). Для числа з плаваючою точкою цей параметр визначає кількість десяткових знаків після коми.

Приклад:

```
Serial.print (78, BIN) - виведе "1001110"  
Serial.print (78, OCT) - виведе "116"  
Serial.print (78, DEC) - виведе "78"  
Serial.print (78, HEX) - виведе "4E"  
Serial.println (1.23456, 0) - виведе "1"  
Serial.println (1.23456, 2) - виведе "1.23"  
Serial.println (1.23456, 4) - виведе "1.2346"
```

7. *Serial.println (val)*

Serial.println (val, format)

- Параметри: *val*: значення, яке необхідно вивести - будь-який тип даних; *format*: визначає систему числення (для цілочисельних типів), а також кількість десяткових знаків після коми (для чисел з плаваючою точкою).
- Значення, що повертаються: кількість виведених байт. Зчитування цього значення не обов'язково.
- Опис: виводить через послідовний порт ASCII-текст в зрозумілому для людини вигляді з символами повернення каретки (ASCII 13 або '\r') і нового рядка (ASCII 10 або '\n'). Ця команда має такі ж форми, як і *Serial.print ()*.

Приклад коду для роботи з послідовним портом

```
int incomingByte = 0;  
void setup() {  
    Serial.begin(9600);  
}  
void loop() {  
    if (Serial.available() > 0) {  
        incomingByte = Serial.read();  
        Serial.print("I received: ");  
        Serial.println(incomingByte, DEC);  
    }  
}
```

2.4 Переривання

Переривання – це сигнали, що переривають нормальний перебіг програми. Вони зазвичай 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`

і реалізовані в файлі:

`\ Hardware \ cores \ arduino \ WInterrupts.c`

Функції переривання Arduino:

1. *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.

Mode 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.

В Arduino Due є ще одне значення: HIGH – переривання буде спрацьовувати щоразу, коли на виводі присутній високий рівень сигналу (тільки для Arduino Due).

- Значення, що повертаються – немає.

2. *detachInterrupt (pin)*

- Опис: Забороняє задане переривання.
- Управління перериваннями *noInterrupts ()*.
- Параметри – ні.
- Значення, що повертаються – немає.

Забороняє переривання, для того, щоб відключити доступ до даних в процесі виконання переривання.

3. *interrupts ()*

- Параметри – ні.
- Значення, що повертаються – немає.
- Опис: повторно дозволяє переривання (після того, як вони були відключені функцією *noInterrupts ()*).

Аналогове введення / виведення

Функція ШІМ

analogWrite (pin, value)

- Параметри: *pin* – вивід, на якому буде формуватися напруга; *value* – коефіцієнт заповнення – лежить в межах від 0 (завжди вимкнений) до 255 (завжди включений).
- Значення, що повертаються: немає.
- Опис: Формує заданий аналогове напруга на виводі у вигляді ШІМ-сигналу. Може використовуватися для варіювання яскравості світіння світлодіода або управління швидкістю обертання двигуна. Після виклику

analogWrite (), на виводі буде безперервно генеруватися ШІМ-сигнал із заданим коефіцієнтом заповнення до наступного виклику функції *analogWrite()* (або до моменту виклику *digitalRead ()* або *digitalWrite ()*, взаємодіючих з цим же виводом). Частота ШІМ становить приблизно 490 Гц. На більшості плат Arduino (на базі мікроконтролерів ATmega168 або ATmega328) функція *analogWrite ()* працює з виводами 3, 5, 6, 9, 10 і 11. На Arduino Mega функція працює з виводами з 2 по 13. Функція *analogWrite ()* не має нічого спільного з аналоговими виводами і функцією *analogRead()*.

Програмування АЦП

Дуже корисний модуль в складі мікроконтролера - аналого-цифровий перетворювач. Він дозволяє мікроконтролеру вимірювати довільне напруження.

Аналого-цифровий перетворювач зчитує величину напруги на аналогових. Це дає можливість зчитувати дані з датчика освітленості, виміряти напругу живлення і т.д.

Основні команди для програмування АЦП в Arduino:

1. *analogReference (type)*

- Параметри: *type* - тип джерела опорної напруги (DEFAULT, INTERNAL, INTERNAL1V1, INTERNAL2V56 або EXTERNAL).
- Значення, що повертаються – ні.
- Опис: встановлює джерело опорної напруги, що використовується при зчитуванні аналогового сигналу (іншими словами, задає максимальне значення вхідного діапазону).

Для вибору джерела опорної напруги доступні наступні значення:

DEFAULT: опорна напруга за замовчуванням, рівна 5 В (на 5 В-платах Arduino) або 3.3 В (на 3.3 В-платах Arduino).

INTERNAL: внутрішня опорна напруга рівна 1.1 В в мікроконтролерах ATmega168 і ATmega328, або 2.56 В в мікроконтролері ATmega8 (не доступно в Arduino Mega).

INTERNAL1V1: внутрішнє опорна напруга 1.1 В (тільки для Arduino Mega).

INTERNAL2V56: внутрішнє опорна напруга 2.56 В (тільки для Arduino Mega).

EXTERNAL: як опорна напруга буде використовуватися напруга, прикладена до виводу AREF (від 0 до 5В).

2. *analogRead (pin)*

- Параметри: *pin*: номер виводу, з якого буде зчитуватися напруга (0 - 5 для більшості плат, 0 - 7 для Mini і Nano, 0 - 15 для Mega).
- Значення, що повертаються: ціле число *int* (від 0 до 1023).
- Опис: зчитує величину напруги з зазначеного аналогового виводу. АЦП перетворювач перетворює вхідну напругу з діапазону 0 - 5 В в цілочисельні значення в межах від 0 до 1023 відповідно. Роздільна здатність АЦП становить: 5 В / 1024 значення або 0.0049 В (4.9 мВ) на одне значення. Вхідний діапазон і роздільна здатність можуть змінюватися за допомогою функції *analogReference ()*.

Для зчитування значення з аналогового входу потрібно близько 100 мікросекунд (0.0001 с), тому максимальна частота опитування виводу приблизно дорівнює 10 000 разів в секунду.

Якщо аналоговий вхід ні до чого не підключений, значення, що повертається функцією *analogRead ()*, буде випадковим. Змінюватиметься під впливом декількох факторів: величина напруги на інших аналогових входах, наведення від руки поблизу плати і т.д.

2.5 Клас String

Клас String з'явився в ядрі, починаючи з версії 0019. Він дозволяє здійснювати більш складну обробку текстових рядків, ніж звичайні масиви символів. За допомогою класу String можливо об'єднувати і розширювати рядки, здійснювати пошук і заміну підрядків, і багато іншого. Він займає більше пам'яті, ніж простий масив символів, але надає більше можливостей.

Загально прийнято називати рядки з масивів символів з маленькою літери «s» (string), а екземпляри класу String - з великою «S». Слід мати на увазі, що рядкові константи, укладені в подвійні лапки, інтерпретуються як масиви символів, а не екземпляри класу String.

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

Опис *String()*

Створює екземпляр класу *String*. Існує кілька різних версій цієї функції, які створюють об'єкт String з різних типів даних (іншими словами, формують рядок з послідовності символів):

- рядкова константа в подвійних лапках (тобто масив символів);
- одиночний символ в одинарних лапках;
- другий примірник класу String;
- цілочисельна константа типу int або long;
- цілочисельна константа типу int або long із зазначенням підстави системи числення;;
- цілочисельна змінна типу int або long
- цілочисельна змінна типу int або long із зазначенням підстави системи числення.

При створенні об'єкта String з числа, результуючий рядок буде містити ASCII-представлення цього числа. За замовчуванням вважається, що число зазначено в десятковій системі, тому:

```
String string = String (13)
```

приведе до створення рядка "13". Можна вказувати підставу системи числення, наприклад:

```
String string = String (13, HEX)
```

приведе до створення рядка "D", що є шістнадцятковим поданням десяткового числа 13. Те ж саме в двійковій системі:

```
String string = String (13, BIN)
```

приведе до створення рядка "1101", що є двійковим поданням числа 13.

Текстові рядки

Опис: текстові рядки можуть бути оголошені двома способами: можна використовувати тип даних `String`, який входить в ядро, починаючи з версії 0019; або оголосити рядок як масив символів `char` з нульовим символом в кінці. Далі описано другий спосіб.

Приклади (нижче наведені приклади правильного оголошення рядків):

```
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".
```

Допустимі операції при оголошенні рядків

- Оголосити масив символів без його ініціалізації (`Str1`),
- оголосити масив символів з одним надлишковим елементом, компілятор сам додасть необхідний нульовий символ (`Str2`),
- додати нульовий символ явно (`Str3`),
- ініціалізувати масив за допомогою рядкової константи, укладеної в лапки; компілятор створить масив необхідного розміру з нульовим символом в кінці (`Str4`),
- ініціалізувати масив за допомогою рядкової константи, явно вказавши його розмір (`Str5`),
- ініціалізувати масив надлишкового розміру, залишивши місце для більш довгих рядків (`Str6`).

Нульовий завершальний символ

Як правило, всі рядки завершуються нульовим символом (ASCII код 0).

По суті, це означає, що довжина рядка повинна бути на 1 символ більше, ніж текст, який необхідно у ній зберігати. Саме тому `Str2` і `Str5` повинні бути довжиною 8 символів, не дивлячись на те, що слово «arduino» займає всього 7 – остання позиція автоматично заповнюється нульовим символом. Розмір `Str4` автоматично стане рівним 8 – один символ потрібно

для завершального нуля. У рядку Str3 було самостійно вказано нульовий символ (позначається '\0').

Слід мати на увазі, що в цілому можна оголосити рядок і без завершального нульового символу (наприклад, якщо задати довжину Str2 що дорівнює 7, а не 8). Однак, це призведе до непрацездатності більшості строкових функцій, тому не слід навмисно так робити.

Основні команди:

1. *String (val)*

String (eval, base)

- Параметри val: змінна, значення якої необхідно представити у вигляді об'єкта String - string, char, byte, int, long, unsigned int, unsigned long, base (необов'язково параметра) – основа системи числення, в якій необхідно представити ціле число.
- Значення, що повертається – немає.

2. *indexOf()*

- Опис: здійснює пошук символу або підрядка в рядку (String). За замовчуванням, пошук здійснюється з початку рядка, однак можна вказати і певну позицію, з якою необхідно почати пошук. Така можливість дозволяє знаходити всі входження підрядка в рядку.

Синтаксис

string.indexOf(val)

string.indexOf(val, from)

- Параметри: *string*: змінна типу String; *val*: шукане значення - char або String; *from*: початкова позиція для пошуку.
- Значення, що повертаються – індекс підрядка val в рядку String, або -1, якщо підрядок не знайдено.

3. *lastIndexOf()*

- Опис: здійснює пошук символу або підрядка в рядку (String). За замовчуванням, пошук здійснюється з кінця рядка, однак можна вказати і

певну позицію, з якої необхідно переглядати рядок у зворотному порядку. Така можливість дозволяє знаходити всі входження підрядка в рядку.

Синтаксис

string.lastIndexOf (val)

string.lastIndexOf (val, from)

- Параметри: *string*: змінна типу String; *val*: шукане значення - char або String; *from*: початкова позиція для пошуку в зворотному порядку.
- Значення, що повертаються: індекс підрядка *val* в рядку String, або -1, якщо підрядок не знайдено.

4. *length ()*

- Опис: повертає довжину рядка String в символах. (Зверніть увагу, що при підрахунку кількості символів функція не враховує завершальний нульовий символ).

Синтаксис

string.length ()

- Параметри: *string*: змінна типу String.
- Значення, що повертаються – довжина рядка String в символах.

5. *substring ()*

- Опис: повертає підрядок в рядку (String). Функція приймає два параметри: початковий індекс і кінцевий індекс в рядку (необов'язковий параметр). При цьому початковий індекс є включеним (відповідний символ буде включений в підрядок), а кінцевий індекс - НЕ включеним (відповідний символ не включається в результуючий підрядок). Якщо кінцевий індекс відсутній, то значення, що повертається функцією, буде містити всі символи від початкового індексу і до кінця рядка.

Синтаксис

string.substring (from)

string.substring (from, to)

- Параметри: *string*: змінна типу String; *from*: початковий індекс в рядку; *to* (необов'язковий параметр): кінцевий індекс в рядку.

- Значення, що повертаються – підрядок.

6. *charAt()*

- Опис: повертає вказаний символ з рядка String.

Синтаксис

string.charAt(n)

- Параметри: *string*: змінна типу String; *n*: номера символу.
- Значення, що повертаються – n-ний символ рядка String.

7. *compareTo()*

- Опис: перевіряє два рядки типу String на рівність, а також дозволяє визначити, який з порівнюваних рядків йде раніше іншого в алфавітному порядку. Рядки порівнюються посимвольно по ASCII-коду кожного символу. Наприклад, символ 'a' йде до символу 'b', але після символу 'A'. Всі цифри йдуть до букв.

Синтаксис

string.compareTo(string2)

- Параметри: *string*: змінна типу String; *string2*: друга змінна типу String.
- Значення, що повертаються: -1: якщо рядок string йде до рядка string2 в алфавітному порядку; 0: якщо рядок string еквівалентний рядку string2; 1: якщо string йде після string2.

8. *toInt()*

- Опис: перетворює рядок (String) в ціле число. Рядок повинен починатися з символьного запису цілого числа.

Синтаксис

string.toInt()

- Параметри: *string*: змінна типу String.
- Значення, що повертаються: long.

РОЗДІЛ 3 ОСНОВНІ ІНТЕРФЕЙСИ ARDUINO

3.1 Двопровідний послідовний інтерфейс TWI (I²C)

Двопроводовий послідовний інтерфейс TWI ідеально підходить для типових додатків на мікроконтролерах і вимагає тільки дві лінії зв'язку. Протокол TWI дозволяє проектувальнику системи зовні зв'язати до 128 різних пристроїв через одну двухпроводну двосторонню шину, де одна лінія – лінія синхронізації SCL і одна – лінія даних SDA. В якості зовнішніх апаратних компонентів, які потрібні для реалізації шини, потрібен лише підтягуючий до плюса живлення резистор на кожній лінії шини. Всі пристрої, які підключені до шини, мають свої індивідуальні адреси, а механізм визначення вмісту шини підтримується протоколом.

Відмінні особливості:

- Гнучкий, простий, ефективний послідовний комунікаційний інтерфейс, що вимагає тільки дві лінії зв'язку;
- підтримка як Master (ведучого або майстра), так і Slave (веденого) режимів роботи;
- можливість роботи, як приймача, так і як передавача;
- 7-розрядний адресний простір дозволяє підключити до шини до 128 підлеглих пристроїв;
- швидкість передачі даних до 400 кГц;
- схема шумозниження, яка підвищує стійкість до завад на лініях шини;
- програмована адреса для Slave режиму з підтримкою загального виклику;
- пробудження мікроконтролера з режиму сну при визначенні заданої адреси на шині.

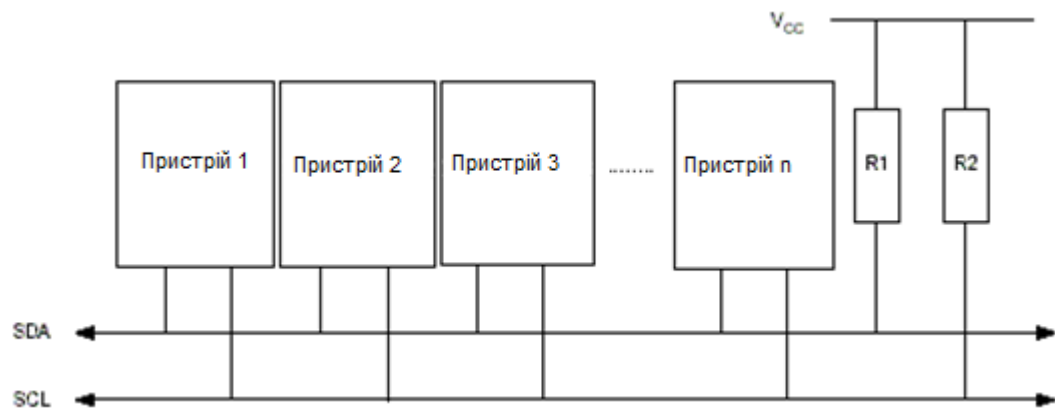


Рис. 3.1 – Система протоколу TWI

Як показано на рис. 3.1, обидві лінії шини підключені до шини живлення через підтягуючі резистори. У всіх сумісних з TWI пристроях, як драйвер шини, використовуються транзистор або з відкритим стоком, або з відкритим колектором. Так реалізована функція, яка дуже важлива для двобічної роботи інтерфейсу. Низький логічний рівень на лінії шини TWI генерується, якщо один або більше з TWI-пристроїв виводить логічний 0. Високий рівень на лінії присутній, якщо всі TWI-пристрої перейшли у третій високоімпедансний стан, дозволяючи підтягуючим резисторам задати рівень логічної 1. Зверніть увагу, що при підключенні до шини TWI декількох AVR-мікроконтролерів, для роботи шини, усі ці мікроконтролери повинні бути підключені до живлення.

Кількість пристроїв, яка може бути підключено до однієї шини обмежується гранично допустимою ємністю шини (400 пФ) і 7-розрядним простором адрес. Підтримуються два різних набори технічних вимог, де один набір - для шин зі швидкістю передачі даних нижче 100 кГц, а інший дійсний для швидкостей понад 400 кГц.

Уся передача даних складається із стартової послілки, бітів і стопової послілки. Початок передачі визначається Start послідовністю - провал SDA при високому рівні SCL.

При передачі інформації від Master до Slave, Master генерує такти на SCL і видає біти на SDA. Які Slave зчитує коли SCL стає 1.

При передачі інформації від Slave до Master, Master генерує такти на SCL і дивиться що там Slave робить з лінією SDA – зчитує дані. А Slave, коли SCL йде в 0, виставляє на SDA біт, який Master зчитує коли підніме SCL назад.

Закінчується все STOP послідовністю. Коли при високому рівні на SCL лінія SDA переходить з низького на високий рівень.

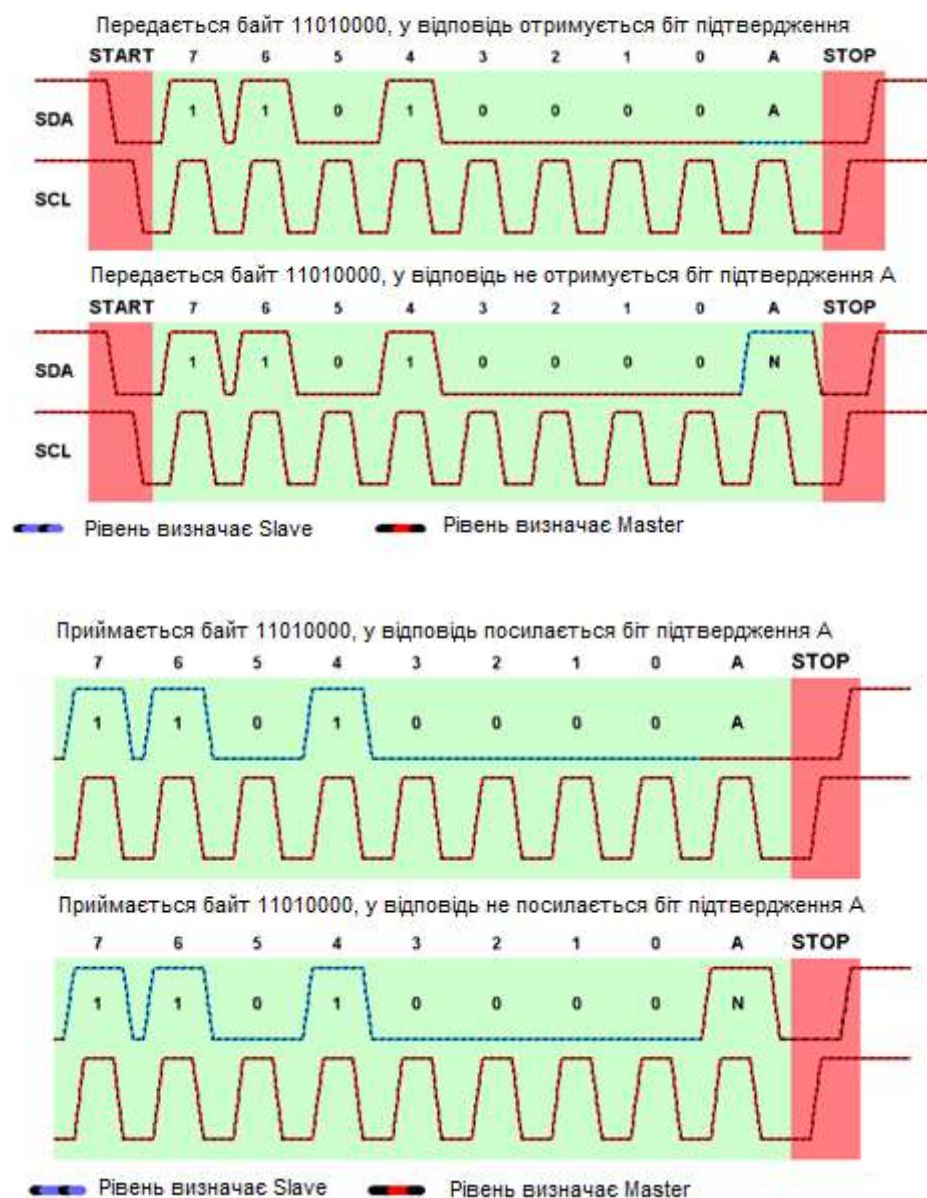


Рис. 3.2 – Приймання/передавання байтів інформації в I2C

Перший пакет від Master до Slave – це фізична адреса пристрою і біт напрямку.

Сама адреса складається з семи біт (ось чому до 127 пристроїв на шині), а восьмий біт означає що буде робити Slave на наступному байті - приймати або передавати дані. Дев'ятим бітом йде біт підтвердження ACK. Якщо Slave розпізнав свою адресу повністю, то на дев'ятому такті він переведе лінію SDA в 0, згенерувавши ACK - тобто зрозумів. Тоді Master продовжить передачу даних. Якщо Slave не відповів, тобто SDA на дев'ятому такті не буде переведено в 0 (не буде ACK), то майстер припинить свої спроби під'єднатися.

Після адресного пакета йдуть пакети з даними в ту або іншу сторону, в залежності від біта RW в заголовному пакеті.

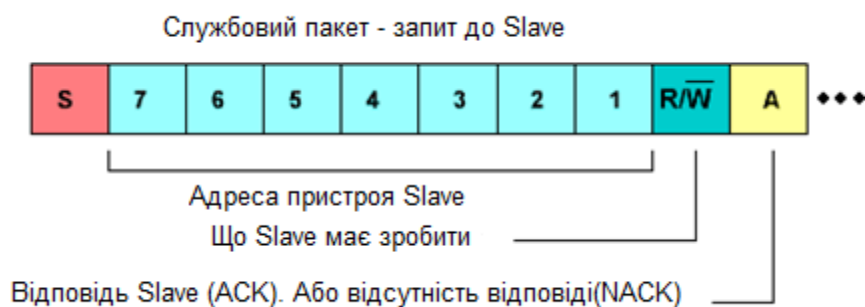


Рис. 3.3 – Схема передачі пакетів

Робота з I2C / TWI на платах Arduino

Табл. 3.1 – Виводи I2C на платах Arduino

	Контакт SDA	Контакт SCL
На Arduino UNO, Nano, Pro Mini.	A4	A5
На Arduino Mega, Due.	20	21
На Arduino Leonardo, Pro Micro.	2	3

Бібліотека Wire.h дозволяє Arduino взаємодіяти з різними пристроями по інтерфейсу I2C / TWI. Згідно з протоколом I2C, адреса пристрою може складатися як з 7, так і з 8 біт. Як правило, 7 біт ідентифікують пристрій, в той час, як восьмий біт задає напрям передачі даних: від пристрою (читання) або до нього (запис). Всі функції бібліотеки Wire.h використовують 7-бітну адресацію, тим самим обмежуючи діапазон можливих адрес в межах 0 – 127.

Функції I2C / TWI на платах Arduino:

1. *Wire.begin()*

Wire.begin(address)

- Параметри: *address* – 7-бітова адреса Slave-пристрою (необов'язковий параметр); якщо адреса не вказана, то Arduino виступає в ролі Master-пристрою.
- Значення, що повертаються – немає.
- Опис: ініціалізує бібліотеку Wire і підключає Arduino до шини I2C в ролі ведучого (master) або веденого (slave) пристрою. Як правило, ця функція викликається тільки один раз.

2. *Wire.requestFrom (address, quantity)*

Wire.requestFrom (address, quantity, stop)

- Параметри: *address* – 7-бітова адреса відомого пристрою, у якого запитуються дані; *quantity*: кількість запитуваних байт; *stop*: boolean. При значенні true буде відправлений запит з стоповим бітом, що дозволить звільнити шину. При значенні false – з'єднання буде підтримуватися в активному стані.
- Значення, що повертаються – *byte*: кількість байт, повернутих веденим пристроєм.
- Опис: функція запрошує дані у веденого пристрою (slave); як правило, використовується тільки ведучим пристроєм (Master). Після виклику *requestFrom ()* запитувані дані повинні бути зчитані за допомогою функцій *available ()* і *read ()*.

Починаючи з версії Arduino 1.0.1, функція *requestFrom* () може приймати третій параметр – логічне значення, що забезпечує кращу сумісність з деякими I2C-пристроями. Якщо цей параметр дорівнює true, то функція *requestFrom* () відправить запит із стоповим бітом, що дозволить звільнити шину I2C. Якщо цей параметр дорівнює false, то після відправлення запиту шина як і раніше буде зайнята, що запобігає надсиланню сторонніх повідомлень іншими провідними пристроями. Цей режим дозволяє Master`у відправляти по декілька запитів за один сеанс.

- Значення за замовчуванням – true.

3. *Wire.beginTransmission(address)*

- Параметри: *address*: 7-бітова адреса пристрою
- Значення, що повертаються немає.
- Опис: починає процедуру передачі даних по інтерфейсу I2C веденому пристрою з вказаною адресою. Для подальшої відправки даних, необхідно спершу поставити їх в чергу за допомогою функції *write* (), після чого здійснити, безпосередньо, передачу функцією *endTransmission* ().

4. *Wire.endTransmission()*

Wire.endTransmission(stop)

- Параметри *stop*: boolean. При значенні true буде відправлений запит зі стоповим бітом, що дозволить звільнити шину. При значенні false – з'єднання буде підтримуватися в активному стані, що запобігає надсиланню сторонніх повідомлень іншими ведучими пристроями. Цей режим дозволяє ведучому відправляти по декілька запитів за один сеанс.

- Значення, що повертаються: byte, байт даних, що характеризує статус передачі:

0: передача успішна

1: обсяг даних занадто великий для буфера передачі

2: отриманий NACK при передачі адреси

3: отриманий NACK при передачі даних

4: інша помилка

- Опис: завершує процедуру передачі даних веденому пристрою, ініційовану функцією *beginTransaction* (). При цьому функція відправляє байти, поставлені в чергу функцією *write* ().

- Значення за замовчуванням – true.

5. **Wire.write(value)**

Wire.write(string)

Wire.write(data, length)

- Параметри: *value*: значення, яке необхідно відправити у вигляді одиночного байта; *string*: рядок, який необхідно відправити у вигляді послідовності байт; *data*: масив даних, який необхідно відправити у вигляді декількох байт; *length*: кількість переданих байт.

- Значення, що повертаються: byte.

- Опис: повертає кількість записаних байт. Зчитування цього значення не обов'язково.

6. **Wire.available ()**

- Параметри: немає.

- Значення, що повертаються: кількість байт, доступних для зчитування.

- Опис: повертає кількість байт, доступних для зчитування функцією *read* (). На ведучому пристрої (Master), ця функція має викликатися після функції *requestFrom* (), а на веденому (Slave) - всередині обробника *onReceive* ().

- Функція *available* () є спадкоємцем допоміжного класу *Stream*.

7. **Wire.read ()**

- Параметри: немає.

- Значення, що повертаються: черговий отриманий байт.

- Опис: функція зчитує байт даних, отриманий ведучим пристроєм від веденого (або навпаки) в результаті виконання функції *requestFrom* (). Функція *read* () є спадкоємцем допоміжного класу *Stream*.

8. *Wire.onReceive (handler)*

- Параметри: *handler*: функція, яку необхідно викликати при надходженні даних від ведучого пристрою; ця функція не повинна повертати ніяких значень і може приймати тільки один параметр (int), що описує кількість надійшли байт. Наприклад: *void myHandler (int numBytes)*.
- Значення, що повертаються: немає.
- Опис: на веденому пристрої дозволяє призначити функцію, яка буде автоматично викликатися при надходженні даних від Master`а.

9. *Wire.on Request (handler)*

- Параметри: *handler*: викликається, без параметрів, не повинна повертати ніяких значень. Наприклад: *void myHandler ()*.
- Значення, що повертаються: немає.
- Опис: на веденому пристрої дозволяє призначити функцію, яка буде автоматично викликатися при отримання запиту від Master`а.

3.2 Інтерфейс 1-Wire

Інтерфейс 1-Wire був запропонований фірмою Dallas Semiconductor в кінці 90-х років минулого століття. Системи 1-Wire привабливі завдяки легкості монтажу, низької вартості пристроїв, можливості вибирати користувача при підключенні до функціонуючої мережі, великому числу пристроїв в мережі і т.д.

Типова система 1-Wire складається з керуючого контролера (майстра або ведучого) і одного або декількох пристроїв (ведених), приєднаних до загальної шини (рис.3.4).

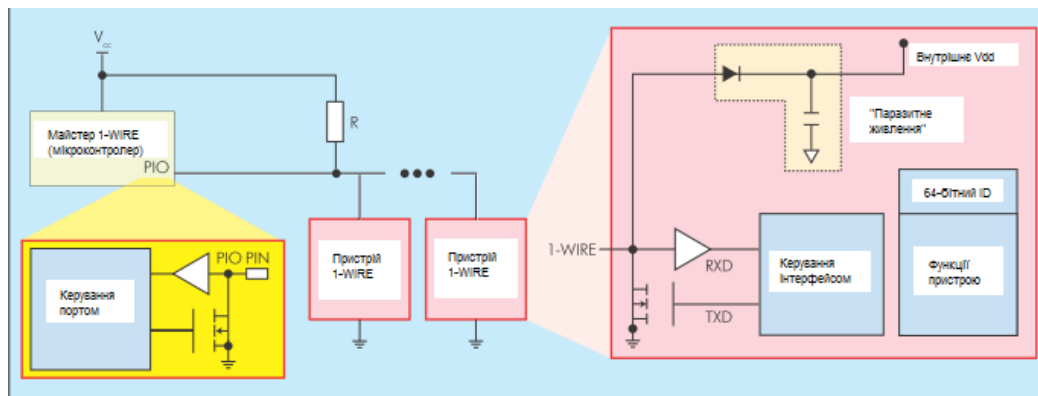


Рис.3.4 – Конфігурація системи 1-Wire

Пристрої підключаються до шини за схемою з відкритим колектором і підтягуючим резистором (див. рис.3.4). Рівень сигналів в шині - від 3 до 5 В. У пасивному стані в лінії підтримується високий рівень напруги. Всі сигнали формуються за допомогою замикання сигнальної шини на землю (низький рівень напруги).

Головна особливість шини 1-Wire в тому, що вона використовує лише два дроти, один – сигнальний, інший – для заземлення пристроїв. По сигнальному проводу можливо і електроживлення пристроїв 1-Wire – так зване паразитне живлення. Джерелом живлення служить конденсатор, який заряджається від сигнальної лінії, що входить до складу відомих пристроїв кола (див. рис.3.4).

Більшість пристроїв 1-Wire підтримують дві швидкості передачі даних: стандартну – близько 15 кбіт / с і підвищену (overdrive) – близько 111 кбіт / с.

Зрозуміло, що чим вище швидкість, тим більше обмежень на довжину шини і число підключаються до неї пристроїв.

Режим передачі даних по шині 1-Wire – напівдуплексний: майстер і ведені пристрої передають дані по черзі. Кожна транзакція через інтерфейс 1-Wire починається з того, що майстер передає імпульс Reset. Для цього він переводить напругу в шині на низький рівень і утримує його в цьому стані протягом 480 мкс (рис.3.5). Потім майстер відпускає шину, і підтягуючий резистор повертає напругу до високого логічного рівня. Всі ведені пристрої,

виявивши сигнал Reset і дочекавшись його закінчення, передають свій сигнал - Presence. Він являє собою сигнал низького рівня тривалістю 100-200 мс. Пристрій може генерувати сигнал Presence і без імпульсу Reset – наприклад, у такий спосіб він повідомляє про себе при підключенні до шини.

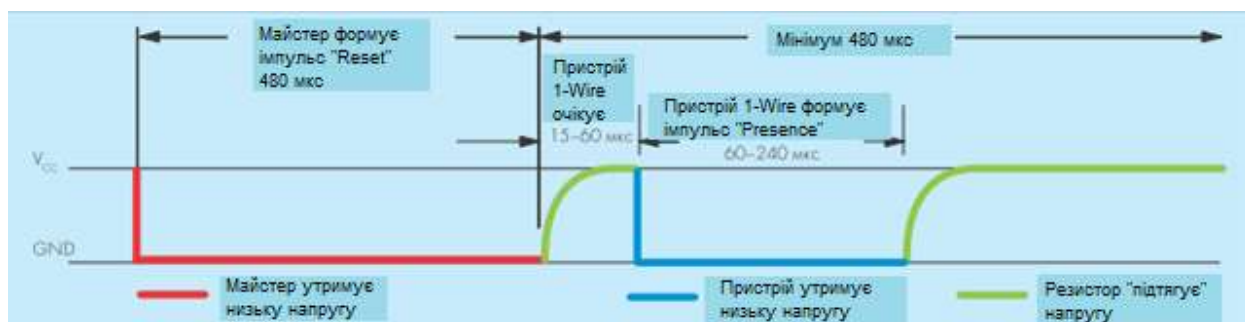


Рис.3.5 – Послідовність ініціалізації шини 1-Wire

Після передачі імпульсу Presence пристрій 1-Wire готовий до прийому команд. Весь інформаційний обмін в шині відбувається під керуванням майстра. Для передачі кожного біта виділяється спеціальний часовий проміжок (таймслот) тривалістю близько 80 мкс. На початку кожного таймслота майстер переводить лінію на нульовий рівень. Якщо далі майстер хоче передати 0, він утримує напругу на низькому рівні як мінімум 60 мкс (рис.3.6). При передачі одиниці майстер утримує нульову напругу 5-6 мкс, а потім відпускає лінію і вичікує приблизно 60 мкс до початку формування наступного таймслота.

Якщо майстер очікує дані від ведених пристроїв, він також позначає початок таймслота, обнулити лінію на 5-6 мкс, після чого перестає утримувати низьку напругу і протягом короткого часу слухає лінію (рис.3.6 б). Якщо пристрій хоче передати нуль, воно саме обнуляє лінію відразу після реєстрації імпульсу початку тайм-слота. Якщо пристрою потрібно передати одиницю, воно ніяких дій не робить. Відзначимо, що наведені значення тимчасових інтервалів відповідають стандартній швидкості передачі даних через інтерфейс 1-Wire. У режимі overdrive ці інтервали відповідно зменшуються.

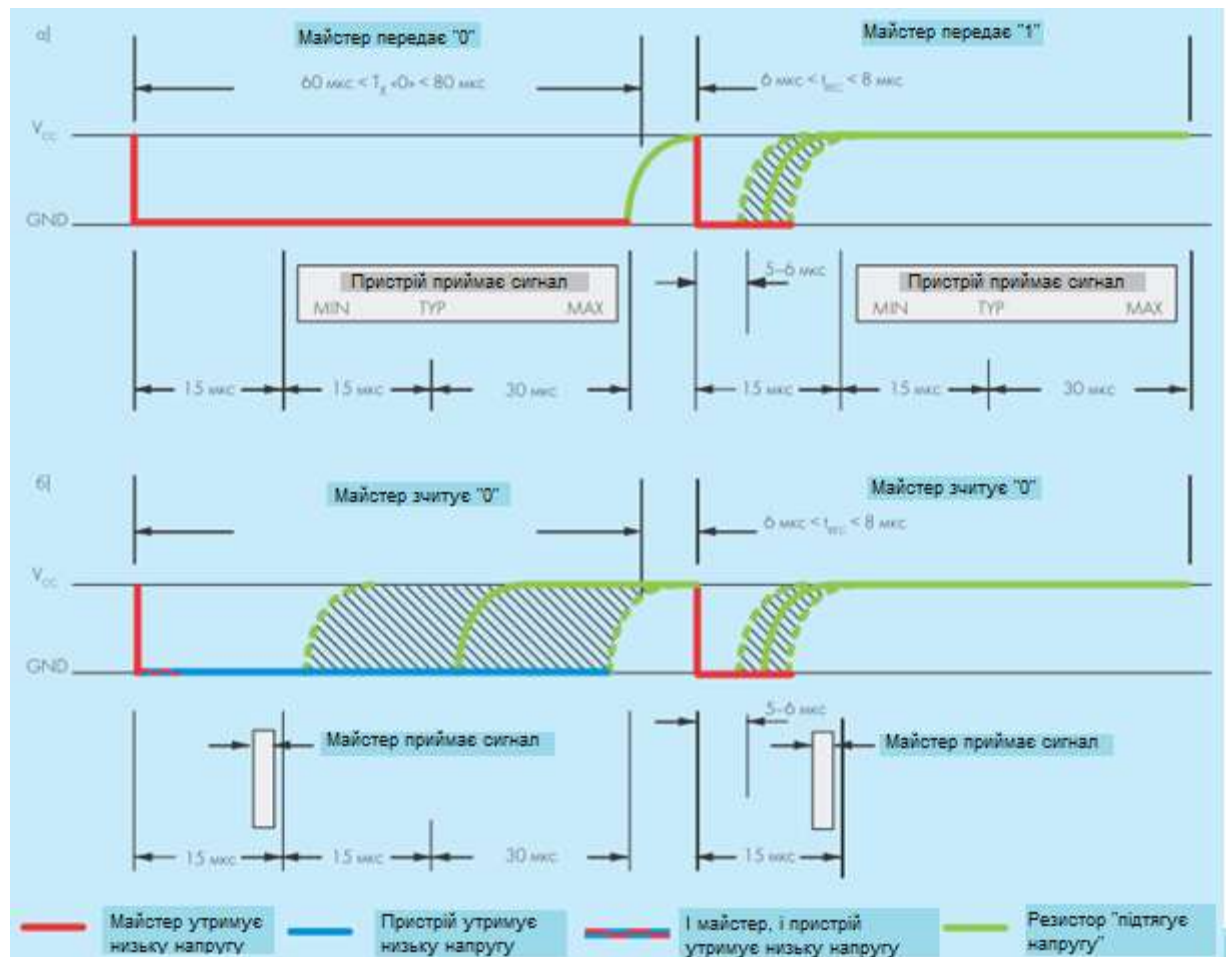


Рис.3.6 – Передача інформаційних бітів по шині 1-Wire

Весь обмін на шині 1-Wire відбувається за допомогою спеціальних команд. Їх число для кожного типу пристроїв по-різному. Але є і мінімальний набір стандартних команд, які підтримують всі 1-Wire-пристрої – так звані ROM-команди.

Формат команд простий – ідентифікатор команди (1 байт), за яким можуть слідувати дані (ідентифікатор пристрою, корисні дані і т.п.). Всі пристрої в мережі знають довжину кожної команди.

У кожного пристрою 1-Wire є 64-розрядний ідентифікатор (ID). Він складається з 8-розрядного коду сімейства, який ідентифікує тип пристрою і та функції, які він підтримує, 48-розрядної серійного номера і 8-бітного поля коду циклічного надлишкового контролю (CRC-8). ID вводиться при виготовленні пристрою і зберігається в ПЗУ.

Увесь обмін командами ініціює майстер. Початок нового циклу транзакцій він зазначає командою Reset, і, отримавши підтвердження, вибирає пристрій спеціальною командою MATCH ROM, передаючи її ідентифікатор (55_{16}) і 64 біта ID адресованому пристрою. Отримавши таку команду, ведений пристрій з даним ID очікує нових команд від майстра, а всі інші залишаються в пасивному стані до наступної команди Reset. В системі з одним пристроєм можна не передавати ID, використовуючи команду SKIP ROM. В результаті ведений пристрій вважає себе обраним без отримання адреси.

Після того, як майстер вибрав пристрій для взаємодії, можна починати процес управління цим пристроєм і обміну даними з ним. Для цього використовуються команди, які специфічні для кожного типу пристроїв.

Але щоб почати роботу з певним пристроєм, майстер повинен знати його ID. Якщо в системі тільки один ведений пристрій, його адресу можна визначити за допомогою команди READ ROM. У відповідь на команду READ ROM пристрій передає свій 64-бітову адресу (рис.3.7) .

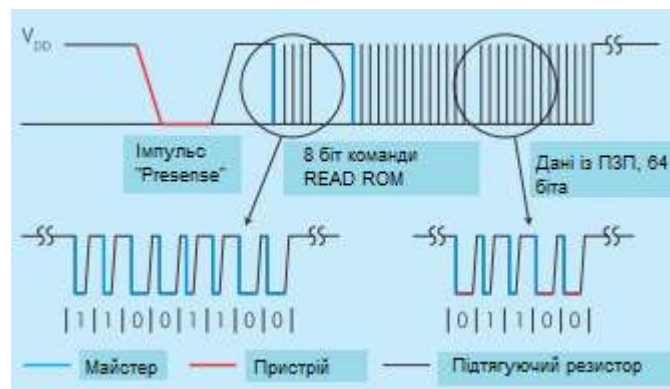


Рис.3.7 – Читання адреси пристрою

Якщо ж в системі декілька пристроїв з невідомими ID, спроба використати команду READ ROM призводить до колізії. У цьому випадку для визначення адреси використовується спеціальний алгоритм пошуку, в основі якого лежить команда SEARCH ROM. Майстер передає команду

SEARCH ROM. У відповідь всі пристрої, підключені до шини, висилають молодший біт своєї адреси. Властивості шини 1-Wire такі, що при одночасній передачі сигналів усіма пристроями результат буде дорівнювати логічній 1 значень всіх надісланих бітів. Отже, сумарний відгук дорівнює 1, тільки коли сигнали від всіх пристроїв рівні 1. Після прийому першого біта адреси, майстер ініціює наступний таймслот, в якому пристрій передає інвертований перший біт. Зіставляючи значення результатів запиту істинного і інверсного бітів, можна отримати певну інформацію про значеннях перших бітів адрес пристроїв (див. табл. 3.2).

Табл. 3.2 – Значення бітів адрес пристроїв

Істинний біт	Інверсний біт	Інформація
0	0	У поточному біті адреси є як 0, так і 1. Це так звана «розбіжність» (discrepancy)
0	1	У біті адреси присутні тільки нулі.
1	0	У біті адреси присутні тільки одиниці
1	1	У пошуку не бере участі жоден пристрій

Таким чином, при комбінаціях 0 1 і 1 0 майстер знає значення першого біта адреси, фіксує його і по тій же схемі може переходити до визначення наступного. Після отримання інверсного біта майстер передає певний біт веденим пристроям. Якщо його значення збігається із значенням поточного біта з адреси пристрою, то пристрій продовжує брати участь в пошуку і видає у відповідь наступний біт своєї адреси. Якщо не було «розбіжності», то значення, що виставляється майстром біта визначено. У разі розбіжності майстер посилає нульовий біт. Така послідовність – читання біта адреси та інверсного біта, передача біта майстром – повторюється для наступних 63 бітів адреси. Таким чином, алгоритм пошуку послідовно виключає всі пристрої, поки не залишається один останній – його адреса і визначається в першому циклі пошуку.

Після того, як адресу першого пристрою визначено, пошук триває для наступного пристрою. Алгоритм запам'ятовує місце останньої розбіжності і вибирає іншу гілку дерева пошуку (майстер посилає в цьому місці біт з іншим значенням). Процес триває до тих пір, поки не буде пройдена гілка, відповідна останньому пристрою. В результаті пошуку стають відомі адреси всіх пристроїв, під'єднаних до шини, і їх число.

Можливість ідентифікації і швидкого включення в мережу тільки-но підключеного пристрою, робить 1-Wire ефективним рішенням для багатьох додатків. На практиці це означає, що прилад досить просто підключити до мережі, і всі подальші транзакції відбудуться автоматично. Наприклад, так можна вважати дані з пам'яті датчика, прочитати код електронної мітки або електронного ключа, прийняти масив значень від приладової мережі і т.п.

Не менш важливо, що мережа 1-Wire не вимагає окремої лінії для передачі тактових сигналів. І, звичайно, величезне число ID пристроїв, що підключаються вигідно виділяє її на тлі інших послідовних мереж.

Застосування 1-Wire

Наявність унікальних 64-бітних адрес дозволяє широко використовувати пристрої 1-Wire в системах аутентифікації. Тут вони часто застосовуються в пристроях iButton. Це мікросхема з введенням на етапі виробництва 64-бітною адресою, укладена в круглий корпус з нержавіючої сталі діаметром 16 мм (MicroCAN). Такі пристрої функціонують, наприклад, в домофонних ключах.

Мікросхеми з підтримкою 1-Wire (наприклад, DS2401, DS2431, DS28E01-100) використовуються також для ідентифікації картриджів принтерів, медичних сенсорів, ємностей з реагентами та ін. Перевага мікросхем 1-Wire в тому, що для контролю ідентифікованого пристрою потрібен всього один контакт. Ще одне поширене застосування 1-Wire - системи автоматизації. В першу чергу це системи багатоточкового вимірювання температури різних середовищ і моніторингу теплового режиму приміщень.

Програмування 1-Wire

Для початку роботи, необхідно імпортувати бібліотеку OneWire.h.

Основні функції:

- ***OneWire myWire (pin)***

За допомогою зазначеного виводу створює об'єкт OneWire. Хоча до одного виводу можна підключити відразу декілька пристроїв, найчастіше це важко реалізувати. Тому, якщо використовується кілька пристроїв, треба підключати їх групами не до одного, а до кількох виводів. Ви можете створити кілька об'єктів OneWire - по одному на кожен вивід.

- ***myWire.search (addrArray)***

Шукає наступний пристрій. Масив *addrArray* - це масив з 8 байтів. Якщо пристрій виявлено, *addrArray* заповнюється адресою цього пристрою, після чого функція повертає true. Якщо більше пристроїв не знайдено, функція повертає false.

- ***myWire.reset_search ()***

Починає новий пошук. Наступне використання пошуку почнеться на першому пристрої.

- ***myWire.reset ()***

Робить скидання шини 1-Wire. Як правило, це потрібно перед комунікацією з будь-яким пристроєм.

- ***myWire.select (addrArray)***

Вибирає пристрій, чия адреса вказана в дужках. Потрібна після скидання - щоб вибрати пристрій, який потрібно використовувати. Подальша комунікація буде саме з цим пристроєм, але до нового скидання.

- ***myWire.skip ()***

Пропускає вибір пристрою. Це працює лише тоді, коли підключено лише один пристрій, тобто ви можете не використовувати пошук, а відразу отримати доступ до вашого пристрою.

- ***myWire.write (num)***

Записує байт.

- ***myWire.write (num, 1)***

Записує байт і залишає включеним паразитне живлення, що підключене до шини 1-Wire.

- ***myWire.read ()***

Зчитує байт.

- ***myWire.crc8 (dataArray, length)***

Розраховує перевірочний CRC для масиву даних.

Приклад програми для зчитування даних з датчика:

```
#include <OneWire.h>
OneWire ds (10); // чіп DS18S20 на 10-му виводі
void setup (void) {
    // запускаємо послідовний порт:
    Serial.begin (9600);
}
void loop (void) {
    byte i;
    byte present = 0;
    byte data [12];
    byte addr [8];
    ds.reset_search ();
    if (! ds.search (addr)) {
        Serial.print ( "No more addresses. \ N"); // "Адрес більше
немає. \ N")
        ds.reset_search ();
        return;
    }
    Serial.print ( "R =");
    for (i = 0; i <8; i++) {
        Serial.print (addr [i], HEX);
        Serial.print ( "");
    }
    if (OneWire :: crc8 (addr, 7) != addr [7]) {
        Serial.print ( "CRC is not valid! \ N"); // "CRC не
коректне! \ N")
        return;
    }
    if (addr [0] == 0x10) {
        Serial.print ( "Device is a DS18S20 family device. \ N");
        // "Пристрій належить сімейству DS18S20. \ N")
    }
    else if (addr [0] == 0x28) {
        Serial.print ( "Device is a DS18B20 family device. \ N");
        // "Пристрій належить сімейству DS18B20. \ N")
    }
}
```

```

    }
    else {
        Serial.print ( "Device family is not recognized: 0x"); //
        "Сімейство пристрою не розпізнано. \ N")
        Serial.println (addr [0], HEX);
        return;
    }
    ds.reset ();
    ds.select (addr);
    ds.write (0x44,1); // запускаємо конверсію і включаємо
    паразитное живлення
    delay (1000); // 750 мілісекунд може вистачити, а може і ні
    // тут можна використовувати ds.depower (), але про це подбає
    скидання
    present = ds.reset ();
    ds.select (addr);
    ds.write (0xBE); // зчитуємо scratchpad-пам'ять
    Serial.print ( "P =");
    Serial.print (present, HEX);
    Serial.print ( "");
    for (i = 0; i <9; i ++) { // нам потрібно 9 байтів
        data [i] = ds.read ();
        Serial.print (data [i], HEX);
        Serial.print ( "");
    }
    Serial.print ( "CRC =");
    Serial.print (OneWire :: crc8 (data, 8), HEX);
    Serial.println ();
}

```


РОЗДІЛ 4 ЛАБОРАТОРНИЙ ПРАКТИКУМ

4.1 Лабораторна робота №1. Вивчення роботи портів вводу-виводу плати Arduino

Мета: навчитися програмувати Arduino і дослідити роботу портів вводу-виводу мікроконтролера.

Завдання: створити програму керування світлодіодами, використовуючи порт вводу-виводу Arduino.

Обладнання: мікроконтролер Arduino; проводи; датчик температури і вологості; макетна плата; USB – кабель.

Загальні відомості

Функції цифрового введення / виведення

1. *pinMode (pin, mode):*

- Параметри: *pin*: номер виводу, режим роботи якого буде конфігуруватися; *mode*: приймає значення INPUT, OUTPUT або INPUT_PULLUP.

- Значення, що повертаються – немає.

2. *digitalWrite (pin, value):*

- Параметри: *pin*: номер виводу; *value*: значення HIGH або LOW.
- Значення, що повертаються – немає.

3. *digitalRead (pin):*

- Параметри: *pin*: номер цифрового виводу, з якого необхідно вважати значення.
- Значення, що повертаються HIGH або LOW.

Приклад коду:

Конфігурація виводів плати:

```
int switchPin = 5;  
int ledPin = 9;  
  
void setup()
```

```

{
pinMode(switchPin, INPUT);
pinMode(ledPin, OUTPUT);
}
    // Основний цикл програми:
void loop()
{
if(digitalRead(switchPin) == HIGH)
    digitalWrite(ledPin, LOW);
else
    digitalWrite(ledPin, HIGH);
}

```

Хід виконання роботи

1. Зібрати макет відповідно схеми на рис. 4.1.
2. Підключити схему до комп'ютера через USB порт плати Arduino.
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи програми.

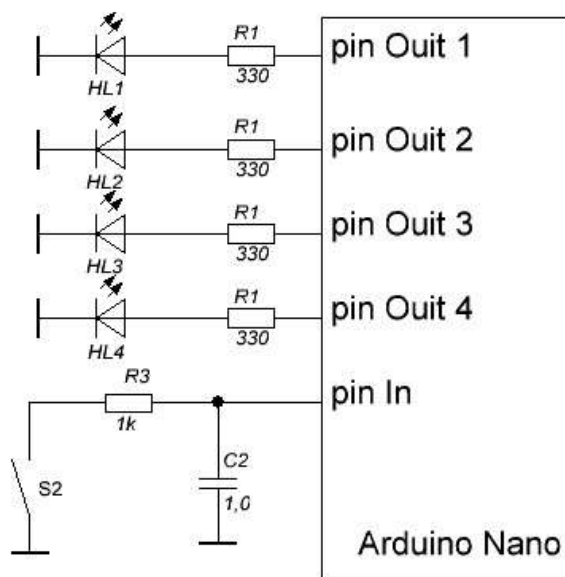


Рис. 4.1 – Схема макету

Завдання

Зібрати схему (рис. 3). Варіанти підключення компонентів до входів-виходів взяти з таблиці 1. Написати програму включення-виключення світлодіодів на виходах мікроконтролера в залежності від кількості натискань кнопки на вході. Всього натискань має бути три. При включенні

світлодіоди починають миготіти із заданою частотою й у порядку згідно варіанту в табл. 4.1. Після першого натискання кнопки частота збільшується вдвічі, після другого ще вдвічі. Після третього повертається до початкового значення.

Налагодити програму в середовищі Arduino і перевірити на макеті.

Табл. 4.1 – Завдання відповідно до варіанту

№ варіанта	Завдання						
	Номера виводів плати Arduino					Початкова частота миготіння, Гц	Порядок включення світлодіодів
	Pin In	Pin Out 1	Pin Out 2	Pin Out 3	Pin Out 4		
1	7	11	6	10	9	0,5	одночасно
2	4	10	5	3	6	1	збільшення
3	12	9	3	6	11	2	зменшення
4	6	11	3	9	10	0,5	через один
5	5	10	6	8	11	1	одночасно
6	3	9	5	7	10	2	збільшення
7	8	9	10	6	11	0,5	зменшення

Підготувати звіт згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

Контрольні питання

1. Зазначте основні сфери застосування Arduino?
2. Які є основні функції цифрового введення та виведення?
3. Які значення має параметр value функції *digitalWrite ()*?
4. Чи повертає значення функція *pinMode (pin, mode)*?
5. Описати алгоритм ініціалізації порту вводу-виводу Arduino?

4.2 Лабораторна робота №2. Вивчення роботи послідовного порту плати Arduino

Мета: дослідити роботу послідовного порту плати Arduino з використанням класу Serial.

Завдання: навчитися приймати и передавати дані з комп'ютера в плату Arduino через послідовний порт.

Обладнання: мікроконтролер Arduino; проводи; макетна плата; USB – кабель.

Загальні відомості

Клас Serial використовується для зв'язку плати Arduino з комп'ютером або іншими пристроями. Плата Arduino Nano має один послідовний порт, також відомий як UART або USART. Він пов'язаний з цифровими виводами 0 (RX) і 1 (TX), і використовується для зв'язку з комп'ютером через USB. Таким чином, під час використання послідовного порту, виводи 0 і 1 не можуть використовуватися в якості цифрових входів або виходів.

Для зв'язку з Arduino можна використовувати спеціальну програму моніторингу послідовного порту, вбудовану в програмне забезпечення Arduino. Для виклику програми натисніть відповідну кнопку на панелі інструментів і встановіть ту ж швидкість передачі, що вказується в програмі при виклику методу *begin* ().

Приклади функцій для роботи з послідовним портом плати Arduino:

1. *Serial.begin(speed)*

Serial.begin(speed, config)

- Параметри:

speed: швидкість в бітах на секунду (бодах) – long;

config: задає кількість біт даних, перевірку чотності и стопові біти.

- Опис: задає швидкість передачі даних по послідовному інтерфейсу в бітах в секунду (бодах). Для взаємодії з комп'ютером слід використовувати одну з попередньо встановлених швидкостей обміну: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 або 115200.

Другий аргумент цієї функції необов'язковий. Він дозволяє налаштувати кількість біт даних, перевірку парності і степових біти. За замовчуванням, посилка складається з 8 біт даних, без перевірки парності, з одним стоповим бітом.

2. *Serial.available()*

- Параметри – нема.
- Значення, що повертаються: кількість байт, доступних для зчитування.
- Опис: повертає кількість байт (символів) доступних для зчитування з буфера послідовного порту. Під символами розуміються дані, які вже прийняті і зберігаються в послідовному приймальному буфері (який може зберігати максимум 64 байта).

3. *Serial.read()*

- Параметри – нема.
- Значення, що повертаються: перший байт прийнятих даних (або -1, если таких нема) – int.
- Опис: зчитує дані, що надходять по послідовному інтерфейсу.

4. *Serial.print(val)*

Serial.print(val, format)

- Параметри:
val: значення, яке необхідно вивести – будь-який тип даних;
format: визначає систему числення (для цілочисельних типів), а також кількість десяткових знаків після коми (для чисел з плаваючою точкою).
- Значення, що повертаються: кількість виведених байт. Зчитування цього значення не обов'язково.
- Опис: функція виводить через послідовний порт заданий ASCII текст у вигляді, зрозумілому для людини. Ця команда може мати кілька різних форм. При виведенні числа кожній його цифрі відповідає один ASCII-символ. Дробові числа теж виводяться у вигляді ASCII-цифр, при цьому після коми

за замовчуванням залишається два десяткових знака. Байти виводяться у вигляді окремих символів, а символи і рядки виводяться без змін – «як є».

Приклад коду для роботи з послідовним портом:

```
int incomingByte = 0;
void setup() {
    Serial.begin(9600);
}
void loop() {
    if (Serial.available() > 0) {
        incomingByte = Serial.read();
        Serial.print("I received: ");
        Serial.println(incomingByte, DEC);
    }
}
```

Хід виконання роботи

1. Зібрати макет відповідно завдання.
2. Підключити схему до живлення(5В).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи програми.

Завдання

Зібрати схему, зазначену на рис. 4.2.

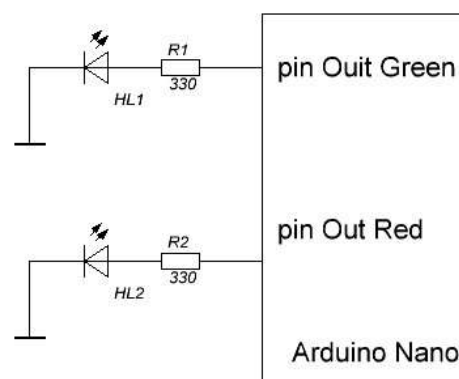


Рис. 4.2 – Схема макету

Написати програму включення світлодіодів в різних режимах по команді комп'ютера. Початковий стан роботи світлодіодів контролера наведено в таблиці 1.

Контролер Arduino повинен виконувати три команди:

1. Команду перевірки з'єднання між комп'ютером і контролером. На певний символ контролер повинен відповісти підтвердженням.

2. Команду управління світлодіодами. При отриманні якої встановлюється режим, зазначений в таблиці 4.2. Отримання команди підтвердити

3. Команду виключення світлодіодів, при отриманні якої світлодіоди вимикаються.

Команди складаються мінімум з одного символу. Індивідуальні завдання для вибору режимів роботи світлодіодів наведені в табл. 4.2.

Табл. 4.2 – Індивідуальне завдання

№ варіанта	Режим до отримання команди		Режим після отримання команди	
	Червоний світлодіод	Зелений світлодіод	Червоний світлодіод	Зелений світлодіод
1	ввімкнений постійно	мигтить	мигтить	ввімкнений постійно
2	вмикаються-вимикаються протифазно		ввімкнений постійно	мигтить
3	ввімкнений постійно	ввімкнений постійно	вмикаються-вимикаються протифазно	
4	мигтить	ввімкнений постійно	вмикаються-вимикаються одночасно	
5	вмикаються-вимикаються одночасно		ввімкнений постійно	ввімкнений постійно
6	мигтить	ввімкнений постійно	ввімкнений постійно	мигтить

Алгоритм роботи програми:

1. Мікроконтролер отримує команду від комп'ютера.
2. Виконує аналіз команди
3. В залежності від команди відсилає символ підтвердження, змінює алгоритм роботи світлодіодів або вимикає їх.

Підготувати звіт згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

Контрольні питання

1. Який клас та функції застосовують для роботи з послідовним портом плати Arduino?
2. Опишіть основні особливості функції *Serial.available()*.
3. Зазначте параметри функції *Serial.print()*.
4. Які команди виконує контролер Arduino згідно з завданням?
5. Які дві характеристики має функція *Serial.begin()*?

4.3 Лабораторна робота №3. Робота з рядками в програмованих мікроконтролерах Arduino

Мета: організувати обмін інформацією з платою Arduino із застосуванням класу String.

Завдання: завантажити байт даних в EEPROM а потім зчитати його, використовуючи окремі паролі на запис та зчитування.

Обладнання: мікроконтролер Arduino; проводи; макетна плата; USB – кабель.

Загальні відомості

Клас String дозволяє здійснювати більш складну обробку текстових рядків, ніж звичайні масиви символів. За допомогою класу String можна об'єднувати і розширювати рядки, здійснювати пошук і заміну підрядків, і багато іншого.

На платі Arduino є 6-канальний (8-канальний – в Mini і Nano, 16 – в Mega) 10-бітний аналогово-цифровий перетворювач, який перетворює вхідну напругу з діапазону 0 – 5 В у цілочисельні значення в межах від 0 до 1023 відповідно. Роздільна здатність АЦП становить: 5 В / 1024 значення або 0.0049 В (4.9 мВ) на одне значення. Вхідний діапазон і роздільна здатність можуть змінюватися за допомогою функції *analogReference()*.

Для зчитування значення з аналогового входу потрібно близько 100 мікросекунд (0.0001 с), тому максимальна частота опитування виведення приблизно дорівнює 10 000 разів на секунду. Якщо аналоговий вхід ні до чого не підключений, значення, що повертається функцією *analogRead()*, буде випадковим. Змінюватиметься під впливом декількох факторів: величина напруги на інших аналогових входах, наведення від руки поблизу плати і т.д.

Приклади функцій Arduino для роботи з рядками:

1. *String(val)*

String(val, base)

- Параметри:

val: змінна, значення якої необхідно представити у вигляді об'єкта String - string, char, byte, int, long, unsigned int, unsigned long;

base (не обов'язково параметра) - основа системи числення, в якому необхідно представити ціле число.

- Значення, що повертаються – немає.
- Опис: створює екземпляр класу String. Існує кілька різних версій цієї функції, які створюють об'єкт String з різних типів даних (іншими словами, формують рядок з послідовності символів). При створенні об'єкта String з числа, результуючий рядок буде містити ASCII-представлення цього числа. За замовчуванням вважається, що число зазначено в десятковій системі, тому:

2. *String.indexOf (val)*

String.indexOf (val, from)

- Параметри:

string: змінна типу String;

val: шукане значення - char або String;

from: початкова позиція для пошуку.

- Значення, що повертаються:
індекс підрядка *val* в рядку String, або -1, якщо підрядок не знайдено.
- Опис: здійснює пошук символу або підрядка в рядку (String). За замовчуванням, пошук здійснюється з початку рядка, однак можна вказати і певну позицію, з якою необхідно почати пошук. Така можливість дозволяє знаходити всі входження підрядка в рядку.

Функції Arduino для запису/зчитування інформації з EEPROM:

1. *read ()*

Опис: зчитує байт з EEPROM. Ті байти, які ніколи не піддавалися записи, мають значення 255.

Синтаксис:

EEPROM.read (address)

- Параметри: *address*: адреса байта, значення якого необхідно зчитати (int).
- Значення, що повертаються: значення зазначеного байта (byte).

2. *write ()*

- Опис: записує байт в EEPROM.

Синтаксис:

EEPROM.write (address, value)

- Параметри:

address: адреса байта (нумерація з 0), значення якого необхідно записати (int);

value: записується значення, від 0 до 255 (byte).

- Значення, що повертаються немає.

Хід виконання роботи

1. Скласти макет згідно завданню.
2. Підключити схему до живлення (5 В).
3. Завантажити програму в програмований мікроконтролер Arduino.
4. Перевірити правильність роботи макету.

Завдання

Скласти схему, вказану на рис. 4.3.

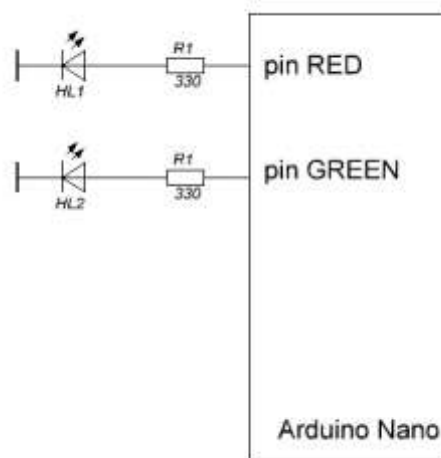


Рис. 4.3 – Схема макету

Написати програму завантаження байта даних в EEPROM а потім зчитування його, використовуючи окремі паролі на запис та зчитування.

Алгоритм роботи програми:

1. Після перезавантаження плати світиться червоний світлодіод. Мікроконтролер отримує від комп'ютера пароль і команду на завантаження байта даних в EEPROM одним рядком. Для поділу пароля, команди і значення байта використовувати символ '/' або '\'. Для позначення закінчення повідомлення використовувати символ '?'.

2. Виконується верифікацію пароля.

3. Якщо пароль не підходить, то відсилається інформація про неправильний пароль і червоний світлодіод мигає декілька разів.

4. Якщо пароль збігається, то червоний світлодіод гасне, а зелений включається на 0,5 – 1,5 с, значення байта зчитується та записується в EEPROM.

5. Аналогічно для зчитування виконуються пункти 2-4 та використовується інший пароль. У результаті байт зчитується з EEPROM та передається в комп'ютер з супровідним повідомленням.

Запит (пароль) – ім'я студента або інше слово латиницею. Варіанти підключення світлодіодів згідно табл. 4.3.

Табл. 4.3 – Варіанти завдань

№ варіанта	Завдання	
	Pin RED	Pin GREEN
1	2	6
2	3	7
3	4	8
4	5	9
5	6	10
6	7	11

Підготувати звіт згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань)

Контрольні питання

1. Описати основні характеристики класу String.
2. Які існують варіанти створення рядка в мові Arduino?
3. Які особливості має функція *String.indexOf()*?
4. Які є основні функції Arduino для запису/зчитування інформації з EEPROM та описати їх характеристики.
5. Коротко зазначити основні етапи роботи програми даної лабораторної.

4.4 Лабораторна робота №4. Вивчення роботи переривань, ШІМ та АЦП програмованого мікроконтролера Arduino

Мета: дослідити роботу переривань, АЦП та ШІМ в контролерах Arduino.

Завдання: зібрати схему і написати програму зміни коефіцієнту заповнення імпульсів ШІМ на виході плати Arduino за допомогою переривань і здійснити вимірювання напруги на виході RC-фільтра.

Обладнання: мікроконтролер Arduino; проводи; макетна плата; USB – кабель.

Загальні відомості

Переривання – це сигнали, що переривають нормальний перебіг програми. Вони зазвичай використовуються для апаратних пристроїв, що вимагають негайної реакції на появу подій. Наприклад, послідовний порт або UART (універсальний асинхронний приймач) мікроконтролера повинен бути обслугований при отриманні нового символу. Якщо цього не зробити до приходу наступного символу, то даний символ може бути втрачений.

При надходженні нового символу UART генерує переривання. Мікроконтролер зупиняє виконання основної програми і переходить на програму обробки переривань (ISR), призначену для даного переривання. В даному випадку це переривання за отриманим символом. Програма обробки ISR зчитує символ з UART, поміщає в буфер, потім очищає прапор переривання і виконує повернення в основну програму, продовжуючи її з точки виклику. Все це відбувається у фоновому режимі і не впливає безпосередньо на основний код роботи програми.

Якщо в процесі роботи запускається багато переривань або переривання генерує швидкодіючий таймер, основна програма буде виконуватися повільніше, так як мікроконтролер розподіляє свій машинний час між основною програмою і всіма функціями обробки переривань.

Arduino так само надає свої функції для роботи з перериваннями.

Ці функції оголошені у файлі:

\ *Hardware* \ *cores* \ *arduino* \ *wiring.h*

і реалізовані в файлі:

\ *Hardware* \ *cores* \ *arduino* \ *WInterrupts.c*

Функції для роботи з перериванням:

1. *attachInterrupt (interrupt, function, mode)*

- Параметри:

interrupt: номер переривання (int). У більшості плат Arduino існує два зовнішніх переривання: номер 0 (цифровий вивід 2) і 1 (цифровий вивід 3).

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 спрацювати переривання.

- Може приймати одне з чотирьох визначених значень:

LOW - переривання буде спрацювати щоразу, коли на виводі присутній низький рівень сигналу

CHANGE - переривання буде спрацювати 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.

- Опис: *attachInterrupt* задає функцію обробки зовнішнього переривання, тобто функцію, яка буде викликана за зовнішнім перериванням.

Якщо до цього була задана інша функція, то призначається нова.

2. *detachInterrupt (interrupt)*

- Параметри: *interrupt*: номер переривання (0 або 1).

Опис: вимикає обробку зовнішнього переривання.

Команди для роботи з АЦП та ШІМ

1. *analogReference (type)*

- Параметри: *type* – тип джерела опорної напруги (DEFAULT, INTERNAL, INTERNAL1V1, INTERNAL2V56 або EXTERNAL).

- Значення, що повертаються: нема.

Опис: встановлює джерело опорної напруги, що використовується при зчитуванні аналогового сигналу (іншими словами, задає максимальне значення вхідного діапазону). Для вибору джерела опорної напруги доступні наступні значення:

DEFAULT: опорна напруга за замовчуванням, рівна 5 В (на 5 В-платах Arduino) або 3.3 В (на 3.3 В-платах Arduino).

INTERNAL: внутрішньо опорна напруга, рівна 1.1 В в мікроконтролерах ATmega168 і ATmega328, або 2.56 В в мікроконтролері ATmega8 (недоступно в Arduino Mega).

INTERNAL1V1: внутрішня опорна напруга 1.1 В (тільки для Arduino Mega).

INTERNAL2V56: внутрішня опорна напруга 2.56 В (тільки для Arduino Mega).

EXTERNAL: як опорна напруги буде використовуватися напруга, прикладена до виводу AREF (від 0 до 5В).

2. *analogRead (pin)*

- Параметри:

pin: номер виводу, з якого буде зчитуватися напруга (0 - 5 для більшості плат, 0 - 7 для Mini і Nano, 0 - 15 для Mega).

- Значення, що повертаються: ціле число int (від 0 до 1023).

- Опис: зчитує величину напруги з зазначеного аналогового виводу.

3. *analogWrite (pin, value)*

- Параметри:

pin – вивід, на якому буде формуватися напругу (ШИМ виводи 3, 5, 6, 9, 10 і 11);

value – коефіцієнт заповнення – лежить в межах від 0 (завжди вимкнений) до 255 (завжди включений).

- Значення, що повертаються: немає
- Опис: формує задану аналогову напругу на виводі у вигляді ШІМ-сигналу. Може використовуватися для варіювання яскравості світіння світлодіода або управління швидкістю обертання двигуна. Після виклику *analogWrite ()*, на виводі буде безперервно генеруватися ШІМ-сигнал з заданим коефіцієнтом заповнення до наступного виклику функції *analogWrite()* (або до моменту виклику *digitalRead ()* або *digitalWrite ()*, взаємодіючих з цим же виводом).

Приклад коду з обробкою переривань:

```
// світлодіод, підключений до digital pin 13 буде змінювати свій
// стан при зміні напруги на digital pin 2
int pin = 13;
volatile int state = LOW;

void setup() {
    pinMode(pin, OUTPUT);           // порт як вихід
    attachInterrupt(0, blink, CHANGE); // прив'язуємо 0-е
    переривання до                // функції
    blink().
}

void loop() {
    digitalWrite(pin, state);       // виводимо state
}

void blink() {
    state = !state;                 // міняємо значення на протилежне
}
```

Хід виконання роботи

1. Зібрати макет відповідно завдання.
2. Підключити схему до живлення(5V).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи програми.

Завдання

Зібрати схему зображену на рис. 4.4.

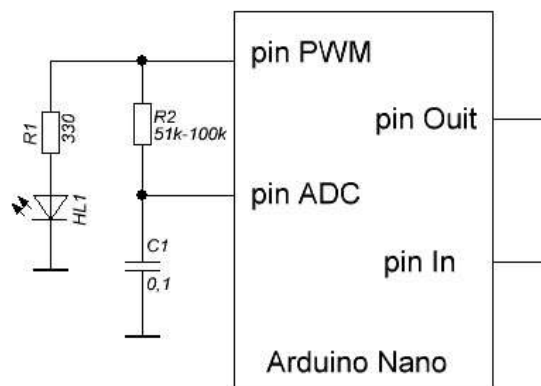


Рис. 4.4 – Схема макету

Написати програму зміни довжини імпульсів на виході ШІМ за допомогою переривань і вимірювання середнього значення цієї напруги.

Варіанти підключення входів-виходів взяти з табл. 4.4.

Табл. 4.4 – Варіанти підключень входів-виходів

№ варіанта	Виводи				
	Pin PWM	Pin ADC	Pin In	Pin Out	Тип переривання
1	3	A1	int 0 (вивід 2)	13	LOW
2	5	A2	int 0 (вивід 2)	12	CHANGE
3	6	A3	int 0 (вивід 2)	11	RISING
4	9	A4	int 1 (вивід 3)	5	FALLING
5	10	A5	int 1 (вивід 3)	6	LOW
6	11	A6	int 1 (вивід 3)	7	FALLING

Алгоритм роботи програми:

1. При ініціалізації виставити на виводі *Pin Out* рівень сигналу, необхідний для роботи по перериванню. Наприклад, виставити рівень HIGH, якщо використовується переривання LOW.
2. З комп'ютера передається команда та значення ШІМ (0-255) у вигляді рядка.
3. Після прийому команди з комп'ютера на виводі *Pin Out* встановлюється рівень сигналу, який запускає переривання.
4. У перериванні необхідно встановити ШІМ сигнал на виході *Pin PWM* з прийнятим коефіцієнтом та виставити початковий рівень сигналу на виводі *Pin Out*.
5. Через 200 мс після закінчення обробки переривання АЦП зчитує напругу з конденсатора C1 (діапазон значень 0-1023).
6. Код з АЦП перераховується у вольти та у вигляді рядка: "Voltage: X.XX v" передається в комп'ютер.

Підготувати звіт згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань)

Контрольні питання

1. Що таке переривання?
2. Які значення може видавати функція *attachInterrupt ()*?
3. Чи повертаються будь-які значення при роботі з функцією *analogReference ()*?
4. Зазначте основні команди для роботи з АЦП та ШІМ?
5. Опишіть, коротко, алгоритм роботи програми.

4.5 Лабораторна робота №5. Вимірювання відстані за допомогою ультразвукового дальноміра HCSR04

Мета роботи: дослідження функціоналу дальнометра HCSR04 за допомогою мікроконтролерного модуля Arduino.

Завдання: зібрати схему і написати програму для вимірювання відстані за допомогою ультразвукового дальноміра.

Обладнання: мікроконтролер Arduino; проводи; макетна плата; USB – кабель; ультразвуковий дальномір HC-SR04.

Загальні відомості

Дія ультразвукового дальноміра HC-SR04 заснована на принципі ехолокації. Він випромінює звукові імпульси в простір і приймає відбитий від перешкоди сигнал. По часу відбивання звукової хвилі до перепони і назад визначається відстань до об'єкта. Відбиття звукової хвилі починається з подачі імпульсу довжиною не менше 10 мікросекунд на вивід **TRIG** дальномера. Як тільки імпульс закінчується, дальномір випромінює в простір перед собою пачку звукових імпульсів частотою 40 кГц. В цей же час на виводі **ECHO** дальнометра з'являється логічна одиниця. Як тільки датчик приймає відбитий сигнал, на виводі ECHO з'являється логічний нуль. По довжині логічної одиниці на виводі ECHO («Затримка ехо» на рис. 4.5) визначається відстань до перешкоди.

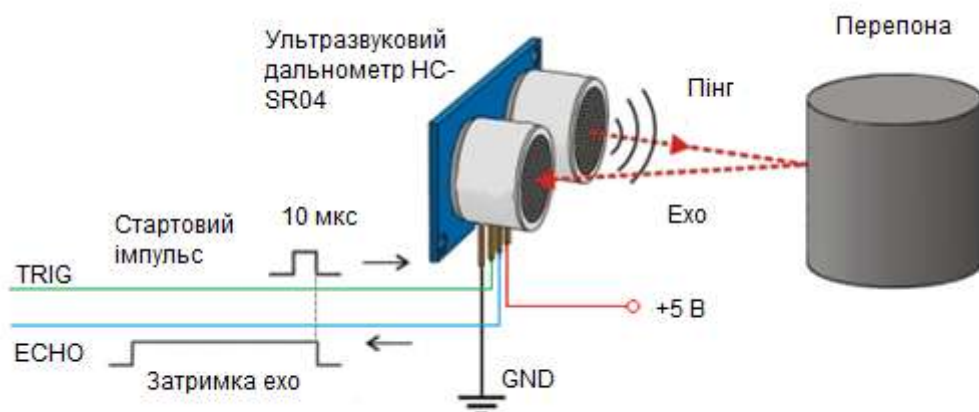


Рис. 4.5 – Принцип роботи ультразвукового дальнометра

Діапазон вимірів відстані дальнометра HC-SR04 – до 4 метрів з мінімальною градацією шкали в 0,3 см. Кут спостереження – 30°, ефективний кут – 15°. Струм живлення в режимі очікування 2 мА, при роботі – 15 мА.

Опис виводів

- 1 — **VCC** – живлення,
- 2 — **TRIG** – вхід для подачі строба запуску,
- 3 — **ECHO** – вивід для зняття імпульса у відповідь,
- 4 — **GND** – земля.

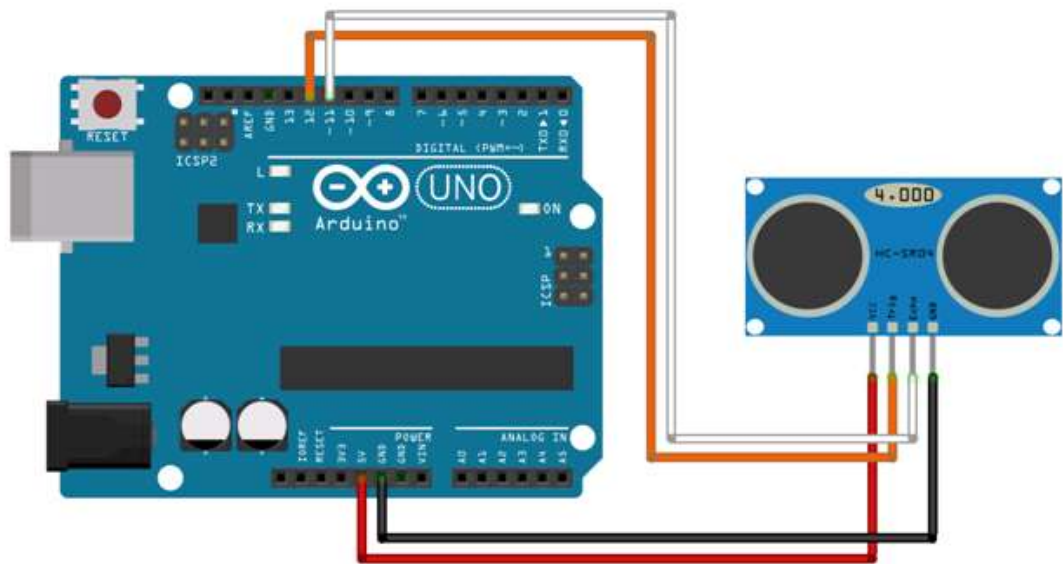


Рис. 4.6 – Схема підключення дальнометра до плати Arduino

Підключення дальнометра без використання бібліотеки NewPing

Скетч для визначення відстані до перешкоди і виводу визначеного значення в послідовний порт без використання бібліотек наведено нижче. Виводи дальнометра TRIG и ECHO – підключені до виводів 12 и 11 плати Arduino. В кожному повторенні циклу *loop()* обраховується дистанція і її значення виводиться в порт.

Функція *getEchoTiming()* генерує імпульс запуску. Вона створює 10-мікросекундний імпульс, який є тригером для початку випромінювання дальнометром звукового пакета в простір. Далі вона запам'ятовує час від початку передачі звукової хвилі до прихода еха.

Функція *getDistance()* розраховує дистанцію до об'єкта. Відстань дорівнює швидкість помножити на час: $S = V \times t$.

Швидкість звуку в повітрі 340 м/сек, час в мікросекундах ми знаємо (змінна *duration*). Щоб отримати час *duration* в секундах, необхідно розділити його на 1 000 000. Так як звук проходить подвійну відстань – до об'єкта і назад – треба ще розділити результат на два. Ось і виходить, що відстань до об'єкта складає:

$$S = 34000 \text{ см/сек} \times \text{duration} / 1\,000\,000 \text{ сек} / 2 = 1,7 \text{ см/сек} / 100.$$

Приклад програми:

```
const int trigPin = 12;
const int echoPin = 11;
void setup() {
    pinMode(trigPin, OUTPUT); // тригер - вихідний пін
    pinMode(echoPin, INPUT);  // ехо - вхідний
    Serial.begin(9600);       // ініціалізація послід. порта
}
void loop() {
    long distance = getDistance(); // отримуємо дистанцію з
    датчика
    Serial.println(distance); // виводимо в послідовний порт
    delay(100);
}
// Визначення дистанції до об'єкта в см
long getDistance() {
    long distacne_cm = getEchoTiming() * 1.7 * 0.01;
    return distacne_cm;
}
// Визначення часу затримки
long getEchoTiming() {
```

```

digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH); // генеруємо імпульс запуску
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// визначення на пині echoPin довжині рівня HIGH, мксек:
long duration = pulseIn(echoPin, HIGH);
return duration;
}

```

Підключення дальнометра з використанням бібліотеки NewPing

Конструктор класу NewPing:

NewPing sonar (trigger_pin, echo_pin, max_distance_cm).

Ця функція визначає 12 пін як вихід тригера, 11 пін як ехо вхід, з максимальною відстанню вимірювання 200 см.

Основні функції бібліотеки *NewPing*:

- *sonar.ping()* – посилає пінг, повертає час ехо в мікросекундах,
- *sonar.ping_in()* – посилає пінг, повертає відстань в дюймах,
- *sonar.ping_cm()* – посилає пінг, повертає дистанцію в сантиметрах,
- *sonar.ping_median(iterations)* – посилає серію пінгів (по замовчуванню = 5), повертає середнє значення,
- *sonar.convert_in(echoTime)* – конвертує мікросекунди в дистанцію в дюймах,
- *sonar.convert_cm(echoTime)* – конвертує мікросекунди в дистанцію в дюймах,
- *sonar.ping_timer(function)* – посилає пінг та викликає функцію, що тестує чи пінг працює,
- *sonar.check_timer()* – перевіряє чи пінг повернувся у межах вказаної максимальної відстані,
- *timer_us(frequency, function)* – викликає функцію з заданою частотою (в мікросекундах),

- *timer_ms(frequency, function)* – викликає функцію з заданою частотою (в мілісекундах),
- *timer_stop()* – зупиняє таймер.

Приклад програми ініціалізації:

```
#include <NewPing.h>
#define TRIGGER_PIN 12 // Arduino pin прив'язаний до тригера
на ультразвуковому датчику..
#define ECHO_PIN 11 // Вивід Arduino прив'язаний до echo-
виводу на ультразвуковому датчику.
#define MAX_DISTANCE 200 // Максимальна відстань, з якою ми
хочемо працювати, тобто пінгувати (у см).
//Максимальна відстань датчика оцінюється в 400-500 см.
// Нлаштування контактів і максимальної відстані.
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
void setup() {
    Serial.begin(115200); // Відкриття послідовного порту 115200.
}
void loop() {
    delay(50); // Чекаємо 50 мс між пінгами (близько 20
пінг/с).
    //29мс має бути найкоротшою затримкою між пінгами.
    .....
}
```

Хід виконання роботи

1. Зібрати макет відповідно завдання.
2. Підключити схему до живлення(5В).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи програми.

Завдання

Створити скетч програми, що буде використовувати функції бібліотеки *NewPing* задані згідно з номером варіанту у табл. 4.5.

Табл. 4.5 – Завдання на лабораторну роботу згідно з номером варіанту

Варіант 1,3,6	Варіант 2,4,7	Варіант 3,5,8
<i>sonar.ping()</i>	<i>sonar.ping_in()</i>	<i>sonar.ping_cm()</i>
<i>sonar.convert_cm(echoTime)</i>	<i>sonar.check_timer()</i>	<i>sonar.ping_timer(function)</i>

Підготувати звіт згідно з ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

Контрольні питання

1. Які основні характеристики та опишіть призначення дальномера HC-SR04?
2. Опишіть виводи, що використовують для підключення дальнометра.
3. Які функції використовують при підключенні дальнометра без використання бібліотеки NewPing та сформулюйте алгоритм роботи програми?
4. Зазначте основні особливості бібліотеки NewPing?
5. Які функції бібліотеки NewPing використовуються для підключення та якісної роботи дальноміра?

4.6 Лабораторна робота №6. Вимірювання температури і вологості повітря за допомогою датчика DHT

Мета: ознайомитись з принципом роботи та передачі даних датчика температури і вологості.

Завдання: написати програму зчитування та передачі показників температури і вологості.

Обладнання: мікроконтролер Arduino; проводи; датчик температури і вологості DHT11; макетна плата; USB – кабель.

Загальні відомості

Датчик DHT11 – це цифровий датчик температури і вологості, що дозволяє калібрувати цифровий сигнал на виході. Складається з ємнісного датчика вологості і термістора. Також, датчик містить в собі АЦП для перетворення аналогових значень вологості і температури.

Характеристики:

- Визначення вологості: 20-90% RH \pm 5% (макс.)
- Визначення температури: 0-50 °C \pm 2% (макс.)
- Частота опитування: не більше 1 Гц
- Розміри 15.5 x 12 x 5.5 мм
- 4 виводи з відстанню між контактами 0.1"
- Живлення 3.5-5.5 В

Виводи:

1. VCC (3 – 5 В живлення)
2. Data Out - Вивід даних
3. NC - не використовується
4. Загальний

При підключенні до мікроконтролеру, можна між виводами Vcc і Data розмістити підтягаючий pull-up резистор номіналом 10 кОм. Плата Arduino має вбудовані pull-up, однак вони дуже слабкі – близько 100 кОм.

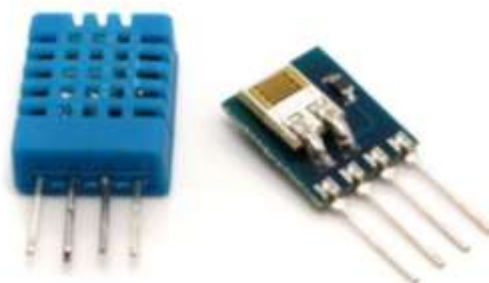


Рис. 4.7 – Зовнішній вигляд DHT11

Програмування датчика DHT

Для роботи з датчиком використовують клас DHT містить декілька функцій:

1. *DHT.begin()*

- Параметри: немає.
- Опис: ініціалізує роботу датчика.

2. *DHT.readTemperature(bool Scale)*

- Параметри: *Scale: false* – температура по Цельсію, *true* – температура по Фаренгейту.
- Значення, що повертаються: *temp (float)*: температура.
- Опис: вимірює температуру повітря.

3. *DHT.convertFtoC(float temp)*

- Параметри: *temp* – температура по Фаренгейту.
- Значення, що повертаються: *temp (float)*: температура по Цельсію.
- Опис: перетворює значення температури по Фаренгейту в температуру по Цельсію.

4. *DHT.convertCtoF(float temp)*

- Параметри: *temp* – температура по Цельсію.
- Значення, що повертаються: *temp (float)*: температура по Фаренгейту.
- Опис: перетворює значення температури по Цельсію в температуру по Фаренгейту.

5. *DHT.readHumidity()*

- Параметри: немає.
- Значення, що повертаються: *hum (float)*: вологість.
- Опис: вимірює вологість повітря.

Приклад коду ініціалізації:

```
// Оголошення змінних:
#include "DHT.h"
#define DHTPIN 2          // вказуємо вивід, до якого підключений
датчик
#define DHTTYPE DHT22    // DHT 22  (AM2302)
#define fan 4
  int maxHum = 60;
int maxTemp = 40;
  DHT dht(DHTPIN, DHTTYPE);

// Ініціалізація інтерфейсів:

void setup() {
  pinMode(fan, OUTPUT);
  Serial.begin(9600);
  dht.begin();
}
```

Хід виконання роботи

1. Зібрати макет відповідно завдання.
2. Підключити схему до живлення(5В).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи програми.

Завдання

Зібрати схему (рис. 4.8).

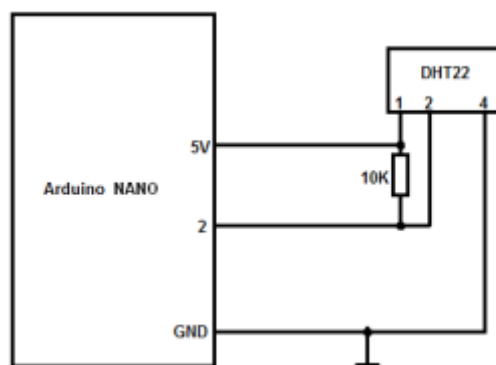


Рис. 4.8 – Схема макету

В середовищі Arduino написати і налагодити програму зчитування показників температури і вологості. Варіанти формату виводу значень температури (за Цельсієм або за Фаренгейтом) взяти з табл. 4.5.

Табл. 4.5 – Значення для різних варіантів

Номера варіантів	Формат
парні	за Цельсієм
непарні	за Фаренгейтом

Підготувати звіт згідно з ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

Контрольні питання

1. Призначення та підключення датчика DHT
2. Опишіть формат передачі даних з датчика DHT
3. Який інтерфейс використовується для передачі даних?
4. Опишіть основні функції класу DHT.
5. Який алгоритм роботи має програма для зчитування показників температури і вологості?

4.7 Лабораторна робота №7. Програмування та налагодження інтерфейсу 1-Wire між платою Arduino та датчиком DS18S20

Мета: організувати обмін інформацією з платою Arduino із застосуванням інтерфейсу 1-Wire.

Завдання: прочитати унікальний ID код та температуру датчика DS18S20 та передати їх в комп'ютер через інтерфейс USB.

Обладнання: мікроконтролер Arduino; проводи; датчик температури DS18S20; макетна плата; USB – кабель.

Загальні відомості

Системи 1-Wire привабливі завдяки легкості монтажу, низької вартості пристроїв, можливості вибирати користувача при підключенні до функціонуючої мережі, великому числу пристроїв в мережі і т.д.

Типова система 1-Wire складається з керуючого контролера (майстра або ведучого) і одного або декількох пристроїв (ведених), приєднаних до загальної шини.

Головна особливість шини 1-Wire в тому, що вона використовує лише два проводи, один – сигнальний, інший – для заземлення пристроїв. По сигнальному проводу можливе і електроживлення пристроїв 1-Wire – так зване паразитное живлення.

Опис основних функцій бібліотеки OneWire.h:

- ***OneWire myWire (pin)*** – за допомогою зазначеного виводу створює об'єкт OneWire.
- ***myWire.search (addrArray)*** – шукає наступний пристрій.
- ***myWire.reset_search ()*** – починає новий пошук.
- ***myWire.reset ()*** – робить скидання шини 1-Wire.
- ***myWire.select (addrArray)*** – вибирає пристрій, чия адреса вказана в дужках.
- ***myWire.skip ()*** – пропускає вибір пристрою.

- ***myWire.write (num)*** – записує байт.
- ***myWire.write (num, 1)*** – записує байт і залишає включеним паразитне живлення, що підключене до шини 1-Wire.
- ***myWire.read ()*** – зчитує байт.
- ***myWire.crc8 (dataArray, length)*** – розраховує перевірочний CRC для масиву даних.

Хід виконання роботи

1. Скласти макет згідно завданню (рис. 4.9).
2. Підключити схему до живлення (5 В).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи макету.

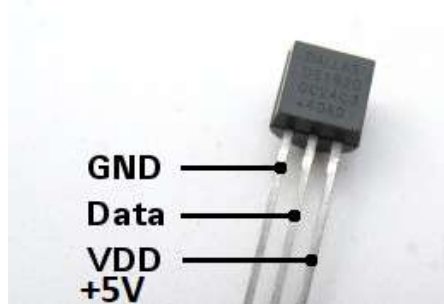


Рис. 4.9 – Призначення виводів датчика температура DS18S20

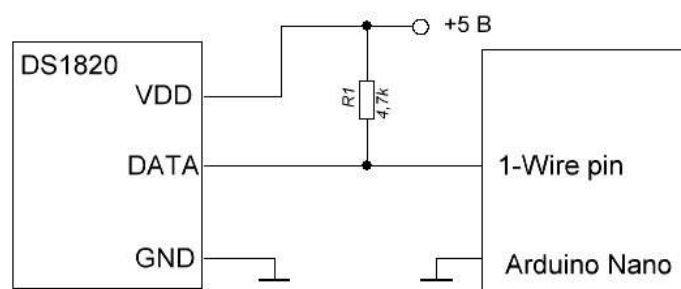


Рис. 4.10 – Схема макету

У режимі паразитного живлення контакт Vdd з датчика підключається до GND на Arduino - в цьому випадку використовуються лише два провoda. Роботу в паразитному режимі краще не використовувати без необхідності, так як можуть погіршитися швидкодія і стабільність.

Завдання

Скласти схему, вказану на рис. 4.10. Написати програму зчитування унікального ID коду та температури датчика DS18S20 та передати його в комп'ютер через інтерфейс USB.

Можна написати свої функції, або використовувати готові бібліотеки з інтернету.

Основні функції, які мають бути реалізовані в програмі:

1. Можливість прочитати унікальний ID код мікросхеми DS1820.
2. Можливість прочитати виміряну температуру.
3. Можливість прочитати дані з пам'яті.
4. Можливість записати дані в пам'ять.

Варіанти виконання роботи:

Для парних підключити всі три виводи мікросхеми DS1820. Для непарних підключити мікросхему DS1820 у паразитному режимі: виводи «GND» та «VDD» з'єднуються між собою та підключаються до «мінусу» живлення. Вивід «DATA» підключається як на Рис. 4.10.

Підготувати звіт згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань)

Контрольні питання

1. Які особливості має система 1-Wire?
2. Яку швидкість передачі даних мають пристрої 1-Wire?
3. Опишіть основні функції бібліотеки OneWire.h.
4. Який режим передачі даних використовується в шині 1-Wire?
5. Опишіть алгоритм програми виконання завдання.

4.8 Лабораторна робота №8. Дослідження роботи сервоприводу та фотоелементу

Мета: дослідити роботу сервоприводу, з використанням фотоелементу та мікроконтролера Arduino.

Завдання: зібрати схему і написати програму зміни положення кута сервоприводу відповідно до роботи фотоелементу.

Обладнання: мікроконтролер Arduino, сервопривід, 2 фотоелемента, проводи-конектори.

Загальні відомості

Сервопривід (серводвигун, сервомашина) є пристроєм, котрий містить електричний двигун. Він дозволяє точно керувати рухом (наприклад, положенням лопасті, як на рис. 4.11) за допомогою негативного зворотного зв'язку. Серводвигун також складається з датчика та блока керування. Блок керування підтримує певні значення на датчику за зовнішнім параметром.



Рис. 4.11 – Зовнішній вигляд сервомашини

Принцип роботи

Спочатку сервопривід отримує імпульсний сигнал — управляючий, котрий і визначає кут повороту валу серводвигуна. Далі блок керування починає порівнювати отриманий параметр із значенням на його датчику. Залежно від результату порівняння, система управління продукує сигнал,

який визначає, яку дію треба виконати: прискорення чи уповільнення, поворот, щоб порівняні метрики стали однаковими.

Конструкція сервомашини

Більшість сучасних рульових машинок побудовані за тим самим принципом. Вони складаються з таких компонентів: вихідний вал, редуктор, двигун постійного струму, потенціометр, друкована плата і керуюча електроніка.



Рис. 4.12 – Будова серводвигуна

Редуктор разом з мотором формують привід. Для перетворення вхідної напруги в механічне обертання, необхідний електродвигун. Редуктор складається із шестерень, він перетворює крутний момент і служить для зниження швидкості обертання двигуна. Це потрібно через те, що часто швидкість буває зовелика та не підходить для практичного використання.

Одночасно із вмиканням та вимиканням електродвигуна, обертається і вихідний вал. До нього кріпиться качалка. Вона з'єднується з кермом моделі тягою. Саме ця задає рух моделі, для цього у конструкції сервомашини передбачений потенціометр. Цей датчик перетворити кут повороту назад в електричний сигнал.

Однак, одним з основних та найважливіших елементів є плата керування. Вона отримує електричний імпульс, аналізує його з даними потенціометра та вмикає/вимикає електродвигун.

У якості двигуна в механізмі сервоприводу застосовуються колекторні, колекторні Coreless (без осердя) та безколекторні двигуни.

Фотоелемент – електричний пристрій, що виконує роль перетворювача частини світлової енергії в електричну за допомогою фотоефекту.

Принцип роботи сучасних фотоелементів заснований на напівпровідниковому р-п переході. Коли фотон поглинається в області, яка прилягає до р-п переходу, утворюється пара носіїв заряду: електрон та дірка. Одна із цих часток являє собою неосновний зарядод і з великою ймовірністю проникає крізь перехід. У результаті, за рахунок поглинання енергії фотонів, заряди розділені в просторі й не можуть бути рекомбінованими. Унаслідок цього порушується рівновага густини зарядів. При підключення елемента до зовнішнього навантаження, по колу протікає струм.

Бібліотека Servo.h

Дана бібліотека дозволяє Arduino управляти роботою серводвигунів. Стандартні сервомашини дозволяють задавати кут повороту валу в діапазоні від 0 до 180 градусів. У двигунах з безперервним обертанням валу можна задавати швидкість його обертання.

У бібліотеці Servo реалізована можливість одночасного управління декількома двигунами: на більшості плат Arduino – до 12, на Arduino Mega – до 48. При цьому, на всіх платах Arduino (за винятком Arduino Mega) використання цієї бібліотеки призводить до непрацездатності ШІМ-функції `analogWrite()` на висновках 9 і 10, незалежно від того, підключений серводвигун до цих висновків, чи ні. На платі Arduino Mega діють інші правила: якщо до Arduino підключено менше 12 двигунів, то бібліотека Servo не впливає на ШІМ-функції; в разі підключення від 12 до 23 двигунів - функція ШІМ стане недоступна на виводах 11 і 12.

Деякі команди:

- ***attach()*** – асоціює вихід Arduino зі змінною типу Servo. Примітка: у версіях Arduino до 0016 бібліотека Servo може працювати тільки з серводвигунами, підключеними до виходів 9 і 10.

- ***read()*** – повертає поточний кут повороту валу серводвигуна (значення, передане функції *write()* при останньому виклику).
- ***write()*** – функція відправляє сервоприводу значення, величина якого впливає на стан вала двигуна. У звичайних сервоприводах це значення задає кут повороту вала в градусах. У сервоприводах безперервного обертання таке значення задає швидкість обертання валу (0 – максимальна швидкість в одному напрямку, 180 – максимальна швидкість в іншому напрямку; відправка значення, приблизно рівного 90, призводить до зупинки двигуна).
- ***writeMicroseconds()*** – відправляє серводвигунів сигнал завдання, виражений в мікросекундах, який характеризує стан вала. У звичайних сервоприводах відправляється значення визначає кут повороту вала. При цьому числа 1000 означає крайнє ліве положення вала (проти годинникової стрілки), 2000 – крайнє праве (за годинниковою стрілкою), а 1500 – положення по середині.

Хід виконання роботи

1. Зібрати макет відповідно завдання.
2. Підключити схему до живлення(5В).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи програми.

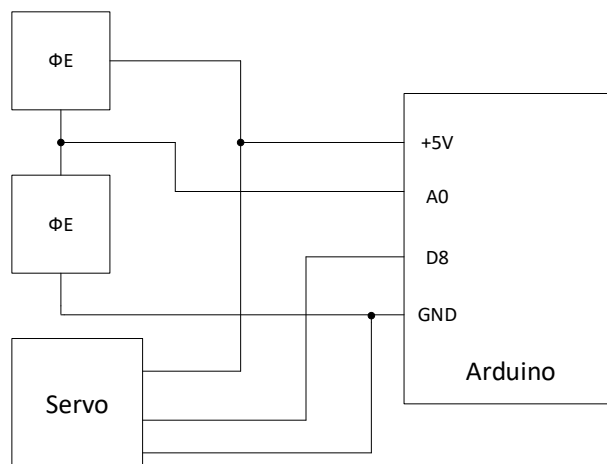


Рис. 4.13 – Схема макету

Завдання

Зібрати схему зображену на рис. 4.13.

Написати програму зміни положення валу сервоприводу відповідно до сигналів, поданих із фотоелементів. Має вийти такий собі «сонячник», котрий тягнеться до сонця (світла). При цьому додатково використати команди, що наведені в табл. 4.6.

Табл. 4.6 – Додаткове завдання

№ варіанта	
Парні	<i>writeMicroseconds()</i>
Непарні	<i>read()</i> з виведенням значень на екран

Підготувати звіт згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

Контрольні питання

1. Що таке сирвопривід, які складові даної системи? Опишіть принцип роботи.
2. Що таке фотоелемент та на чому побудований його принцип роботи?
3. Яка бібліотека використовується для керування сервопиводом?
4. Який кут повороту валу задають сучасні сервомашини?
5. Які функції виконує команда *writeMicroseconds()*?

4.9 Лабораторна робота №9. Створення мелодії з використанням Arduino

Мета: дослідити можливості створення мелодії, за допомогою спеціальних бібліотек, в Arduino.

Завдання: зібрати схему і написати програму створення мелодії, а також відтворення нот.

Обладнання: плата Arduino; маленький звуковий динамік із опором 8 Ом; резистор 100 Ом; з'єднувальні проводи; ультразвуковий дальнометр; кнопки.

Загальні відомості

Для відтворення звукового сигналу, використовуючи мікроконтролер Arduino, використовують команду *tone()*.

tone() генерує на виводі мікроконтролера прямокутний сигнал заданої частоти (з коефіцієнтом заповнення 50%). Функція також дозволяє задавати тривалість сигналу. Однак, якщо тривалість сигналу не вказана, він буде генеруватися доти, поки не буде викликана функція *noTone()*. Для відтворення звуку вивід можна підключити до зумеру або динаміку.

У кожен момент часу може генеруватися тільки один сигнал заданої частоти. Якщо сигнал вже генерується на будь-якому виводі, то використання функції *tone()* для цього виводу просто призведе до зміни частоти цього сигналу. У той же час виклик функції *tone()* для іншого висновку не матиме ніякого ефекту.

Використання функції *tone()* може впливати на ШІМ-сигнал на виводах 3 і 11 (на всіх платах, крім Mega).

Для відтворення різних звуків на кількох виводах, необхідно спершу викликати *noTone()* на одному і тільки після цього використовувати функцію *tone()* на наступному.

Синтаксис:

tone (pin, frequency),

tone (pin, frequency, duration).

Параметри:

pin: вивід, на якому буде генеруватися сигнал;

frequency: частота сигналу в Герцах – unsigned int;

duration: тривалість сигналу в мілісекундах (опціонально) – unsigned long.

Значення, що повертає – немає.

Приклад коду програми відтворення звуку

Наведений нижче код використовує додатковий файл, `pitches.h`. Цей файл містить всі значення кроку для простих звуків. Наприклад, `NOTE_C4` є нотою Сі. `NOTE_FS4` – фа-дієз, і так далі.

Отож, спочатку треба створити файл `pitches.h`, у котрому зазначена нота та відповідна їй частота, й зберегти його у папці з проектом. Файл має в собі містити наступний код:

```
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
```

```
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
```



```

#define NOTE_G7 3136
#define NOTE_G5 3322
#define NOTE_A7 3520
#define NOTE_A5 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978

```

Основний код скетчу має наступний вигляд:

```

#include «pitches.h»

// ноти в мелодії:
int melody[] = {
NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4};
// тривалість ноти: 4 = четверть ноти, 8 = восьма частина,
и т.д.:
int noteDurations[] = {
4, 8, 8, 4, 4, 4, 4, 4 };
void setup () {
//перебір нот мелодії:
for (int thisNote = 0; thisNote < 8; thisNote++) {
// для розрахунку тривалості мелодії була взята одна секунда
//розділена на тип ноти.
// четверть ноти = 1000 / 4, восьма частина = 1000/8, і т.д..
int noteDuration = 1000/noteDurations[thisNote];
tone (8, melody[thisNote],noteDuration);
//для того, щоб розділити ноти, встановимо мінімальну паузу між
ними.
// наприклад, тривалість ноти + 30% :
int pauseBetweenNotes = noteDuration * 1.30;
delay (pauseBetweenNotes);
// stop the tone playing:
noTone (8);
}
}
void loop () {
// нема потреби повторювати мелодію.
}

```

Хід виконання роботи

1. Зібрати макет відповідно завдання.
2. Підключити схему до живлення(5V).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи програми.

Завдання

Зібрати схему зображену на рис.1. Кожна бригада має створити свою унікальну мелодію, використавши приклад, що наведений вище. Та виконати додаткові завдання, відповідно до номеру бригади.

Парні номери:

Створити додаткову функцію, котра дозволить відтворювати певну ноту, відповідно до відстані, котру зчитує дальнометр (схема підключення на рис.2). Наприклад, 5 см – нота До, 15 см – Ре і т.д. Основи роботи з дальнометром описані в лабораторній роботі №5.

Непарні номери:

Під'єднати 8 кнопок та запрограмувати їх, при натисканні, видавати певну ноту(одна кнопка – одна нота).

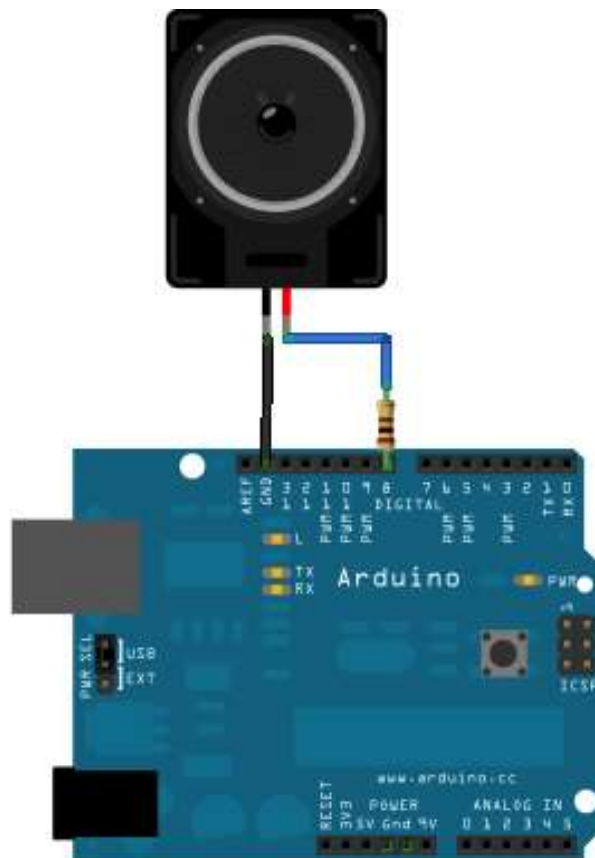


Рис. 4.14 – Схема підключення звукового динаміку

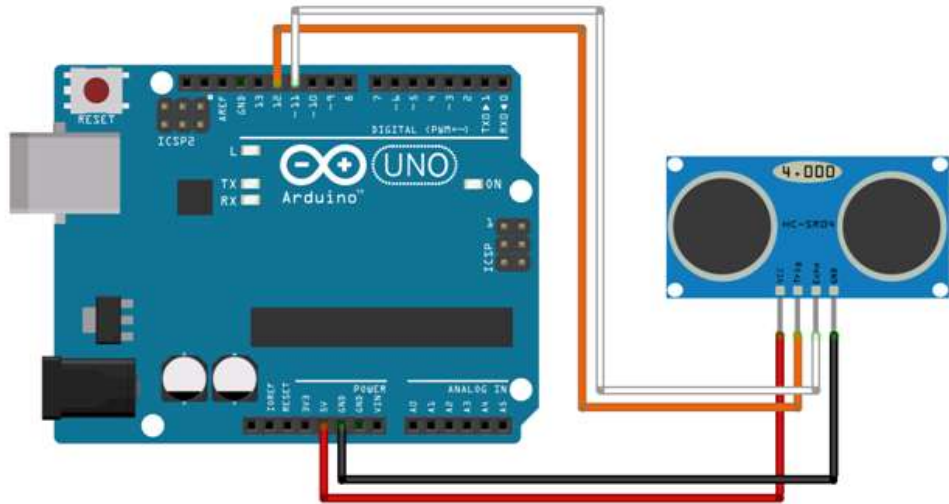


Рис. 4.15 – Схема підключення ультразвукового дальнометра

Підготувати звіт згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

Контрольні питання

1. Яку команду використовує Arduino для відтворення звуку? Опишіть основні її особливості.
2. Які параметри має функція *tone()*?
3. Які дані має містити файл *pitches.h* та опишіть його призначення?
4. Опишіть алгоритм програми створення мелодії, а також відтворення нот.
5. З чого складається макет роботи?

4.10 Лабораторна робота №10. Робота з LCD – дисплеєм по протоколу I2C

Мета: написати програму для виводу даних і керування LCD – дисплеєм через інтерфейс I2C.

Завдання: зібрати схему з використанням перехідника I2C і написати програму виведення на дисплей символів.

Обладнання: плата Arduino; дисплей LCD 1602 (або 2004) з підключеним перехідником I2C; проводи; макетна плата; кабель USB.

Загальні відомості

LCD – дисплей

LCD екран на базі 1602 (або 2004) з керуванням по шині I2C (TWI, PC) дуже зручний при нестачі вільних виводів на Arduino. Досить підключити всього два провідники (само-собой не рахуючи живлення) і можна повністю управляти дисплеєм. Працювати з подібними дисплеями можна через бібліотеку LiquidCrystal_I2C.

Підключення стандартне, як і для всієї лінійки подібних дисплеїв на цьому контролері (рис. 4.16). Контраст екрану налаштовується змінним резистором зі зворотного боку плати. Живлення - 5В.

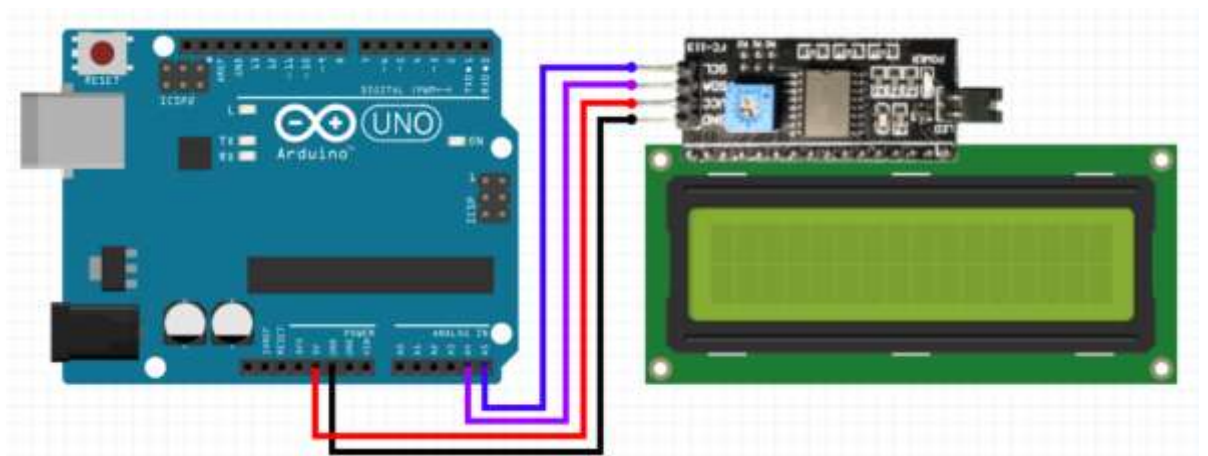


Рис. 4.16 – Схема підключення LCD – дисплею

I2C – адреса може бути: *0x27 0x20* или на *0x3F*.

Табл. 4.7 – співвідношення виводів, що відповідають за інтерфейс I2C на платах Arduino на базі різних контролерів

LCD I2C модуль	Arduino Nano	Leonardo	MEGA, ADK, DUE
SCL	A5	D3	D21
SDA	A4	D2	D20
VCC	+5V	+5V	+5V
GND	GND	GND	GND

Бібліотеки для роботи з LCD – дисплеєм

Для взаємодії Arduino з LCD 1602 (2004) по шині I2C необхідно як мінімум дві бібліотеки:

- Бібліотека Wire.h для роботи з I2C вже є в стандартній програмі Arduino IDE.
- Бібліотека LiquidCrystal_I2C.h, яка включає в себе велику різноманітність команд для управління монітором по шині I2C і дозволяє зробити скетч простішим і коротшим.

Після підключення до скетчу всіх необхідних бібліотек треба створити програмний об'єкт LCD-дисплей та використовувати всі його функції.

Опис функцій і методів бібліотеки LiquidCrystal_I2C:

- ***init ()*** – ініціалізація дисплею.
- ***backlight ()*** – підключення підсвічування.
- ***setCursor ()*** – установка курсора.
- ***print ()*** – набір тексту.
- ***home ()*** і ***clear ()*** – перша функція дозволяє повернути курсор в початок екрану, друга теж, але при цьому видаляє все, що було на моніторі до цього.
- ***write (ch)*** – дозволяє вивести одиночний символ *ch* на екран.

- ***cursor ()*** і ***noCursor ()*** – показує / приховує курсор на екрані.
- ***blink ()*** і ***noBlink ()*** – курсор блимає / не блимає (якщо до цього було включено його відображення).
- ***display ()*** і ***noDisplay ()*** – дозволяє підключити / відключити дисплей.
- ***scrollDisplayLeft ()*** і ***scrollDisplayRight ()*** – прокручує екран на один знак вліво / вправо.
- ***autoscroll ()*** і ***noAutoscroll ()*** – дозволяє включити / вимкнути режим автопрокручування. У цьому режимі кожен новий символ записується в одному і тому ж місці, витісняючи раніше написане на екрані.
- ***leftToRight ()*** і ***rightToLeft ()*** – установка напрямку виведеного тексту - зліва направо або справа наліво.
- ***createChar (ch, bitmap)*** – створює символ з кодом *ch* (0 - 7), використовуючи масив бітових масок *bitmap* для створення чорних і білих точок.

Приклад програми:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h> // Підключення бібліотеки

LiquidCrystal_I2C lcd (0x27,16,2); // Вказуємо I2C адресу
(найбільш поширене значення), а також параметри екрану (в цьому
випадку LCD 1602 - 2 рядки по 16 символів в кожному)

void setup ()
{
    lcd.init (); // Ініціалізація дисплею
    lcd.backlight (); // Підключення підсвічування
    lcd.setCursor (0,0); // Установка курсора в початок першого
рядка
    lcd.print ( "Hello"); // Набір тексту на першому рядку
    lcd.setCursor (0,1); // Установка курсора в початок другого
рядка
```

```

    lcd.print ( "ArduinoMaster"); // Набір тексту на другому рядку
}
void loop ()
{
}

```

Якщо після завантаження скетчу не з'явилося жодної написи на дисплеї, треба виконати наступні дії:

- збільшити або зменшити контрастність монітора,
- перевірити чи правильно підключено контакти, чи підключено живлення підсвічування,
- може бути вказано неправильна адреса. Треба змінити в скетчі адресу пристрою з 0x27 0x20 або на 0x3F. У різних виробників можуть бути зашиті різні адреси за замовчуванням.

Створення власних символів

Варто зазначити, що літери англійського алфавіту зашиті в пам'ять контролера всередині дисплея і з ними проблем немає. Проте літери кирилиці модуль екрана не підтримує, тому необхідно їх створити самому.

Дисплей підтримує до 8 користувацьких символів розміром 5x8 пікселів. Зовнішній вигляд кожного користувацького символу задається масивом з восьми байт, кожен з яких характеризує відповідний рядок.

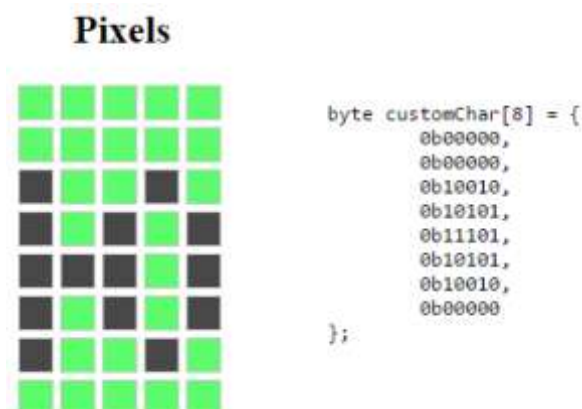


Рис. 4.17 – Приклад створення масиву станів пікселів

П'ять молодших біт кожного байта визначають стан пікселів у відповідному рядку. На рис. 4.17 приведений приклад створення масиву. На рис. добре видно, що дивлячись на масив, можна легко зрозуміти, який символ буде створено.

Після запису станів пікселів до масиву, за допомогою цього ж масиву створюється відповідний символ.

Хід виконання роботи

1. Зібрати макет відповідно завдання.
2. Підключити схему до живлення(5В).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи програми.

Завдання

Зібрати схему зображену на рис. 4.16.

Кожна бригада має створити програму, котра виводить на дисплей номер бригади та прізвища її членів українською мовою. Невистачаючи символи кирилиці створити власноруч.

Підготувати звіт згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

Контрольні питання

1. Опишіть особливості протоколу I2C / ІІС (Inter-Integrated Circuit) та зазначте основні його переваги та недоліки.
2. Що таке LCD – дисплей та як з ним працювати?
3. Які існують бібліотеки для роботи з LCD – дисплеєм?
4. Які особливості функцій *scrollDisplayLeft ()* і *scrollDisplayRight ()*, *autoscroll ()* і *noAutoscroll ()*?
5. Опишіть принцип створення власних символів для виводу їх на LCD – дисплей.

4.11 Лабораторна робота №11. Програмування годинника реального часу DS3231

Мета: дослідити можливості програмування годинника реального часу з використанням інтерфейсу I2C.

Завдання: зібрати схему і написати програму виведення на дисплей параметрів годинника, а саме: час, дата та ін.

Обладнання: плата Arduino; дисплей LCD 1602 (або 2004) з підключеним перехідником I2C; проводи; макетна плата; кабель USB; годинник реального часу DS3231.

Загальні відомості

DS3231 – це годинник реального часу з екстремально точним ходом, завдяки вбудованому кварцовому резонатору з температурною компенсацією. Інтерфейс передачі даних – I2C. У мікросхемі є також вхід для підключення резервної батареї. При відключенні основного живлення мікросхема автоматично перемикається на роботу від резервної батареї, точність ходу від резервної батареї не порушується.

У DS3231 підтримується підрахунок секунд, хвилин, годин, днів місяця (дати), днів тижня, місяців і років (з урахуванням високосного року для місяців). Підтримується робота в 12 і 24 годинному форматі. Є 2 будильника з можливістю їх налаштування і відстеження стану. Підстроювання точності температурної компенсації. А також два виходи – на 32 кГц (вихід становить 32.768 кГц) і програмований вихід від 1 Гц до 8.192 кГц. Є також вивід скидання – RST. Мікросхема випускається в корпусі SO-16.

Що важливо знати при роботі з мікросхемою годинника реального часу, так це ж як витягти з неї дані або записати їх туди. Годинник має інтерфейс I2C. Для того щоб здійснити запис даних (а це потрібно і для того, щоб прочитати дані), потрібно передати умову старту (ці команди здійснюються за рахунок апаратного або програмного I2C для мікроконтролера), далі передати адресу мікросхеми з бітом запису, далі

передати адресу регістру, до якого будемо звертатися і потім передати в цей регістр байт даних. Якщо слідом передати ще байт даних, він запишеться в наступний регістр і т.д. По закінченню потрібно передати умову зупинки.

Графічне зображення вищесказаного представлено на рис. 4.18:

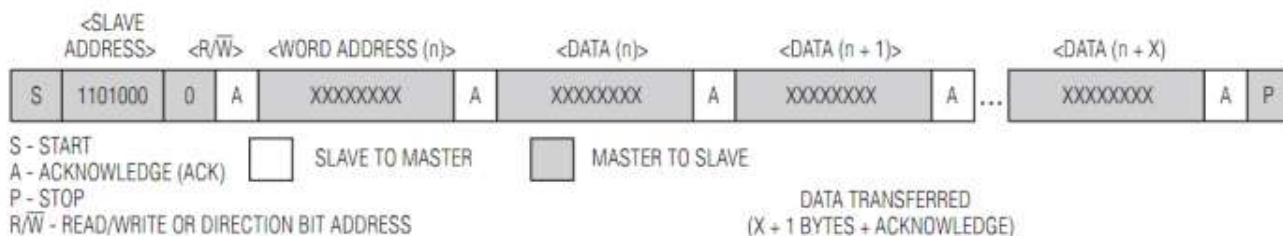


Рис. 4.18 – Процес зчитування/запису для DS3231

Запис даних необхідний для початкового налаштування, а також для налаштування поточного часу. Далі нам потрібно постійно отримувати дані про поточний час і дату. Для цього необхідно здійснювати читання з регістрів зберігання цієї інформації. Читання складається з двох процедур – встановити покажчик на потрібний регістр і прочитати його. Щоб встановити покажчик на потрібний регістр, потрібно передати умову старту, потім передати адресу мікросхеми з бітом запису і байт з адресою регістру. Далі або умова зупинки і слідом умова старту, або просто рестарт. Тепер друга процедура – безпосередньо читання з регістрів. Старт переданий, далі потрібно відправити адресу мікросхеми з бітом читання і зчитувати регістри в необхідній кількості, по закінченню передати умову зупинки. Якщо інформація з регістра була прочитана, то покажчик автоматично переходить на наступний за ним регістр без зайвих дій з боку мікроконтролера (майстер пристрою). На рис. 4.19 проілюстровано все вище сказане щодо читання регістрів за рахунок I2C інтерфейсу.

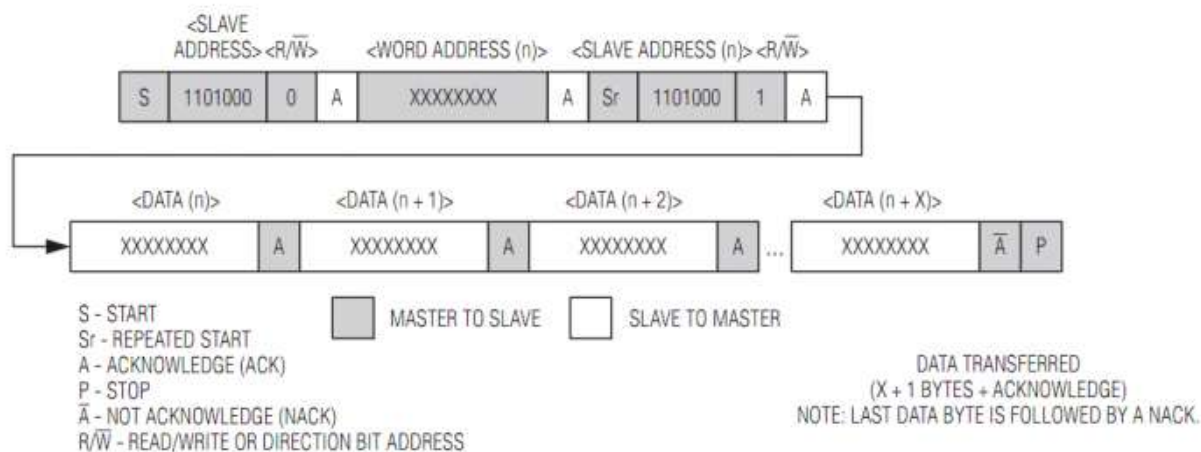


Рис. 4.19 – Читання регістрів за рахунок I2C-інтерфейсу

Підключення годинника до Arduino

Годинник підключається по шині ІІС(I2C) (Inter-IntegratedCircuit) через виводи SDA і SCL до SDA і SCL Arduino відповідно. Також необхідно підключити живлення + 5В і GND.

Табл. 4.8 – Співвідношення виводів шини I2C в залежності від типу контролера, що розташовані на різних виводах

	Контакт SDA	Контакт SCL
На Arduino UNO, Nano, Pro Mini.	A4	A5
На Arduino Mega, Due.	20	21
На Arduino Leonardo, Pro Micro.	2	3

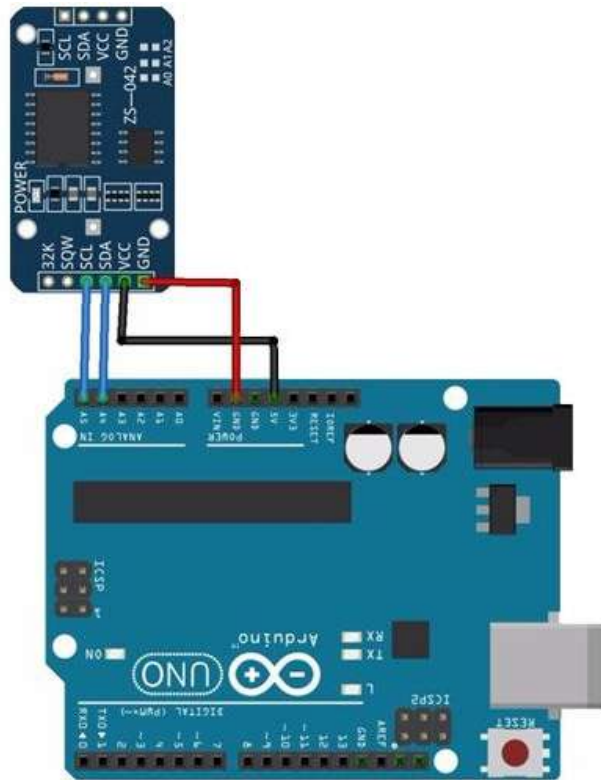


Рис. 4.20 – Схема підключення годинника реального часу DS3231

Бібліотеки для роботи DS3231

Для підключення RTC годинника реального часу DS3231 було розроблено декілька бібліотек. Варто звернути увагу на `iarduino_RTC.h`. Вона універсальна (підходить для DS3231, DS1302, DS1307), а також використовую менше ОЗП та флеш-пам'яті.

Опис функцій і методів бібліотеки `iarduino_RTC.h`:

- `#include <iarduino_RTC.h>` – підключення бібліотеки.
- `iarduino_RTC Об'єкт (НАЗВА [, ВИВІД_RST [, ВИВІД_CLK [, ВИВІД_DAT]]])` – створення об'єкта.
- Функція `begin ()` – ініціалізація роботи RTC модуля.
- Функція `settime (СЕК [, ХВ [, ГОД [, ДЕНЬ [, МІС [, РІК [, ДТ]]]]])`; - встановлення часу.
- Функція `gettime ([РЯДОК])` – отримання часу.
- Функція `blinktime (ПАРАМЕТР [ЧАСТОТА])` – змушує функцію `gettime` «блимати» зазначеним параметром часу.

- Функція ***period (ХВИЛИНИ)*** – вказує мінімальний період звернення до модуля в хвилинах.
- ***seconds*** – повертає секунди від 0 до 59.
- ***minutes*** – повертає хвилини від 0 до 59.
- ***hours*** – повертає годинник від 1 до 12.
- ***hours*** – повертає годинник від 0 до 23.
- ***midday*** – повертає опівдні 0 або 1 (0-am, 1-pm).
- ***day*** – повертає день місяця від 1 до 31.
- ***weekday*** – повертає день тижня від 0 до 6 (0-неділю, 6-субота).
- ***month*** – повертає місяць від 1 до 12.
- ***year*** – повертає рік від 0 до 99.

Приклад програми:

У даному прикладі наведена програма виводу часу годинника DS1307 в Монітор порта.

```
#include <iarduino_RTC.h> // Підключаємо бібліотеку
iarduino_RTC time (RTC_DS1307); // Оголошуємо об'єкт time для
модуля на базі чіпа DS1307
void setup () {
    Serial.begin (9600); // Почнемо передачу даних в монітор
    послідовного порту
    time.begin (); // Ініціалізуємо RTC модуль
}
void loop () {
    if (millis () % 1000 == 0) { // якщо пройшла 1 секунда
        Serial.println (time.gettime ("d-m-Y, H: i: s, D")); //
        ВИВОДИМО час в монітор, одним рядком
        delay (2); // призупиняємо роботу скетчу на 2 мс.
    }
}
```

Хід виконання роботи

1. Зібрати макет відповідно завдання.
2. Підключити схему до живлення(5V).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи програми.

Завдання

Зібрати схему зображену на рис.3. Виконати наступні завдання по бригадах:

1, 3 варіанти:

Установити поточне значення часу на годинник та виводити його значення в Монітор послідовного порту кожні 5 сек.

2, 5 варіанти:

Установити поточне значення дати на годинник та виводити його в монітор порту за командою.

4, 6 варіанти:

Виставити значення часу та будильник. При спрацьовуванні останнього, виводити текстове повідомлення в Монітор послідовного порту.

Підготувати звіт згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

Контрольні питання

1. Що таке DS3231, які має характеристики та який інтерфейс використовує?
2. Коротко опишіть принцип підключення годинника до Arduino.
3. Які бібліотеки існують для роботи DS3231?
4. Які основні функції має, описана в роботі, бібліотека DS3231?
5. Які значення та в яких діапазонах повертаються годинником?

4.12 Лабораторна робота №12. Створення вимірювача швидкості реакції

Мета: закріпити знання роботи з LCD – дисплеєм, світлодіодами та кнопками.

Завдання: зібрати схему і написати програму виведення на дисплей швидкості натискання кнопки, як реакцію на мигтіння світлодіода чи звукового динаміка.

Обладнання: плата Arduino; дисплей LCD 1602 (або 2004) з підключеним перехідником I2C; проводи; макетна плата; кабель USB; 6 світлодіодів; звуковий динамік; кнопка.

Загальні відомості

Рефлексометри були дуже популярні у минулому. Стверджували, що подібні, тільки більш складні прилади застосовують при тренуванні космонавтів. І це насправді так. Рефлексометрія вимірює швидкість реакції людини на звуковий або світловий сигнал. Якщо в ньому передбачено декілька видів сигналів, помилятися не можна – потрібно натиснути кнопку, що відповідає тому сигналу, який прозвучав або засвітився.

Принцип роботи полягає в наступному: натискають кнопку, після чого пристрій починає подавати різні сигнали. RGB-світлодіод може світитися різними кольорами, звуковий динамік – видавати звуки. Зреагувати, натиснувши кнопку, потрібно, наприклад, на червоний сигнал, не звертаючи уваги на інші. Вимірюється швидкість реакції на сигнал.

Хід виконання роботи

1. Зібрати макет відповідно завдання.
2. Підключити схему до живлення(5В).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи програми.

Завдання

Грунтуючись на знаннях, отриманих з попередніх лабораторних робіт, самостійно придумати схему підключення та скласти програму, а саме:

Парні варіанти:

До мікроконтролера Arduino підключити кнопку, LCD – екран та звуковий динамік.

Необхідно, щоб динамік відтворював 3 різні частоти. Відтворювання має проходити швидко, але щоб можна було розрізнити сигнали. При програванні певної (однієї з трьох) частоти, необхідно натискати кнопку. На LCD – дисплей виводити час швидкості реакції на цей сигнал. Після натискання робити затримку в 5 с. Після закінчення цієї перерви, має заграти шуканий сигнал, щоб нагадати своє звучання. Але на цей випадок натискати кнопку не треба. Це сигналізує про продовження роботи.

До початку, на дисплеї має зображуватися текст: «Почнемо?». Звук вимкнений.

Щоб почати програвання, потрібно один раз натиснути кнопку. У цей час має заграти сигнал, який й буде визначатися. Це сигналізує про початок роботи.

Щоб вимкнути гру, треба натиснути кнопку 2 рази підряд. Після цього все має стати, як до початку гри.

Непарні варіанти:

До мікроконтролера Arduino підключити кнопку, LCD – екран та 6 світлодіодів (1 червоний та 5 зелених).

Необхідно, щоб світлодіоди миготіли дуже швидко. Червоний світлодіод має вмикатися набагато рідше, ніж зелені. Коли він засвітиться, треба натиснути кнопку. На LCD – дисплей виводити час швидкості реакції на цей сигнал. Після натискання робити затримку в 5 с. Після закінчення цієї

перерви, має увімкнутися шуканий світлодіод. Але на цей випадок натискати кнопку не треба. Це сигналізує про продовження роботи.

До початку, на дисплеї має зображуватися текст: «Поїхали?». Світлодіоди вимкнені.

Щоб почати програвання, потрібно два рази натиснути кнопку. У цей час має загорітися червоний світлодіод, який й буде визначатися. Це сигналізує про початок роботи.

Щоб вимкнути гру, треба натиснути кнопку та тримати протягом 2 с. Після цього все має стати, як до початку гри.

Підготувати звіт згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

Контрольні питання

1. Опишіть що таке рефлексометр та зазначте його принцип роботи.
2. Які функції було використано для реалізації завдання?
3. Опишіть алгоритм роботи програми.
4. Які бібліотеки використовують для роботи з LCD – дисплеями, звуковими динаміками та світлодіодами?
5. Який інтерфейс використовується для підключення дисплею? Опишіть його особливості.

4.13 Лабораторна робота №13. Віддалене керування приладами

Мета: дослідити роботу таких пристроїв: датчик освітленості, датчик звуку, приймач та передавач та написати програму роботи з ними плати Arduino.

Завдання: зібрати схему і написати програму передавання даних по радіочастотах з датчика освітленості та мікрофона, на базі двох мікроконтролерів Arduino.

Обладнання: дві плати Arduino; датчик освітленості Flying Fish – MH Sensor Series; радіомодулі 433 МГц FS1000a та XY-MK-5V(приймач та передавач); датчик звуку LM393 з мікрофоном CZN-15E; проводи; гучномовець; світлодіод.

Загальні відомості

Опис радіомодуля

Організувати безпроводовий зв'язок між платформами Arduino можна за кількома технологіями. Для реалізації простого і дешевого бездротового підключення, можна застосувати технологію в діапазоні LPD433, що працює на частоті 433,920 МГц. З огляду на те що частоту даного діапазону не потрібно ліцензувати, вона знайшла широке застосування в світі, серед малопотужних пристроїв. В даному діапазоні працюють авто сигналізації, пульти відкривання гаражних дверей і включення люстр, а так само малопотужні рації. У Україні дозволено вільне використання подібних пристроїв, якщо потужність передавача не перевищує 10 мВт. Для Arduino випускається комплект з передавального (FS1000A) і приймального (MX-RM-5V) радіомодулів, які в залежності від напруги живлення і умов зв'язку, можуть забезпечити зв'язок на відстані до 100м в умовах прямої видимості. Подібні модулі випускаються на 315, 330 і 433 МГц.



Рис. 4.21 – Зовнішній вигляд радіомодулів

Для роботи з радіомодулями скористаємося бібліотекою `RCSwitch.h`. Для прикладу, напишемо програму, яка буде кожну секунду по черзі відправляти два різних повідомлення.

Програмний код для передавача:

```
#include <RCSwitch.h>

RCSwitch mySwitch = RCSwitch(); //оголошуємо об'єкту для роботи
з передавачем
void setup() {
    mySwitch.enableTransmit(2); // вмикаємо передавач і вказуємо
до якого виводу він підключений
}
void loop() {
    mySwitch.send(B0100, 4); // відправляємо повідомлення;
перший аргумент – текст повідомлення (у даному випадку у
двійковій системі, на це вказує символ «B»; може бути й в
десятковому вигляді та у вигляді двійкового рядка), а другий –
розмір пакету даних, що відправляється.
    delay(1000);
    mySwitch.send(B1000, 4);
    delay(1000);
}
```

Програмний код для приймача:

Для демонстрації факту передачі ми будемо запалювати світлодіод, підключений до виводу №3 на платі Arduino. Якщо приймач отримав код B1000 - увімкнемо світлодіод, а якщо B0100 - вимкнемо.

```
#include <RCSwitch.h>
RCSwitch mySwitch = RCSwitch();// оголошуємо об'єкту для роботи
з передавачем
void setup() {
    pinMode( 3, OUTPUT );
    mySwitch.enableReceive(0); // вмикаємо приймач і вказуємо
номер переривання, у даному випадку переривання 0 знаходиться на
виводі 2
}
void loop() {
    if( mySwitch.available() ){ // mySwitch.available() -
повертає істину, якщо приймач прийняв які-небудь дані
        int value = mySwitch.getReceivedValue(); //
getReceivedValue() - витягує із потоку даних один пакет даних і
декодує його в число.
        if( value == B1000 )
            digitalWrite( 3, HIGH );
        else if( value == B0100 )
            digitalWrite( 3, LOW );
        mySwitch.resetAvailable();
    }
}
```

Датчик освітленості

Даний пристрій видає напругу +5 В на виході, якщо на фоточутливий елемент не потрапляє світло, і навпаки: видає 0 В, якщо світло потрапляє на датчик. Можна використовувати для автоматичного регулювання рівня освітлення електричних пристроїв. Значення логічного 0 або 1 можна

побачити через Монітор послідовного порту, попередньо підключивши прилад до Arduino.

Табл. 4.9 – Параметри підключення Flying Fish – MH Sensor Series

AO	Аналогове значення датчика
DO	1 (+5V) – якщо темно, 0 – якщо світло – цифрове значення датчика
GND	Земля
VCC+	Живлення +5V

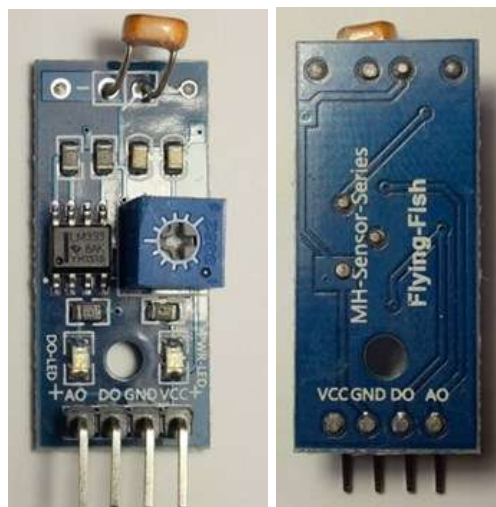


Рис. 4.22 – Зовнішній вигляд сенсора освітленості Flying Fish – MH Sensor Series

Датчик звуку LM393

Акустичний датчик модуль використовується в проектах, де потрібно вловлювати і вимірювати звукові хвилі.

Для використання акустичного датчика його потрібно підключити до Arduino. Потім створити програму для роботи з датчиком і відкалібрувати його чутливість.

В основу роботи датчика покладений принцип дії звукових коливань на тонку мембрану мікрофона. Коливання внутрішньої мембрани мікрофону породжують електричні коливання. Напруга, що виникає в процесі роботи датчика, подається на інтерфейс для збору даних.

На корпусі модуля розташований регулятор чутливості (змінний резистор). На платі датчика є два світлодіоди позначені L1 і L2. Світлодіод L1 горить, коли на датчик подається напруга живлення. Світлодіод L2 горить, коли датчик уловлює звукові коливання.

Акустичний сенсор має 4-піновий роз'єм для підключення контролера і живлення:

- контакт «+» для підключення напруги живлення 5В;
- контакт АО аналоговий вихідний сигнал;
- контакт DO цифровий вихідний сигнал;
- контакт «-»(GND) загальний.

Живлення може здійснюватися або від Arduino контролера, або від іншого керуючого мікропроцесорного пристрою, або від зовнішнього джерела живлення (блоку живлення батареї).



Рис. 4.23 – Зовнішній вигляд датчика звуку

Хід виконання роботи

1. Зібрати макет відповідно завдання.
2. Підключити схему до живлення(5В).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи програми.

Завдання

Зібрати схему відповідно до рис. 4.24.

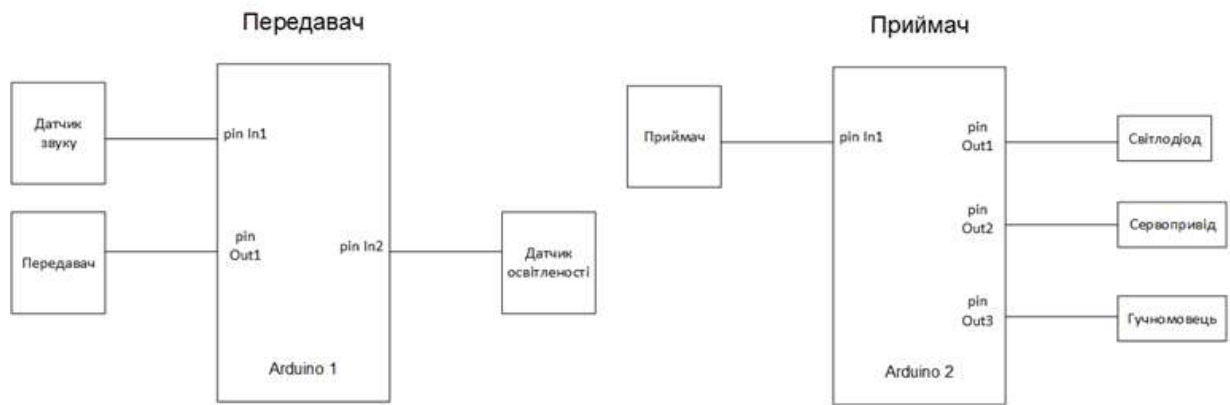


Рис. 4.24 – Структурна схема підключення макету

Скориставшись прикладом налаштування передавача та приймача, наведеним вище, створити програму, котра дозволить передавати та приймати дані з сенсорів, що вказані за варіантом бригади.

1,3 бригади:

Датчик звуку (мікрофон). Мікрофон має зчитувати хлопок. Отриманий сигнал має передаватися через радіомодуль. На боці приймача має загорітися світлодіод. Ще один такий прийнятий сигнал його вимикає.

2,5 бригади:

Датчик освітленості. При накриванні фотоелемента датчика, має створюватися сигнал, котрий відсилається через передавач на приймач. Прийнятий сигнал вмикає обертання сервоприводу. Ще один така прийнята команда сервомотор вимикає. Принцип роботи з сервомашиною наведений у попередніх лабораторних роботах.

3,6 бригади:

Введення команди «ON» через Монітор послідовного порту має вмикати на стороні приймача звуковий динамік. Команда «OFF» вимикає звук.

Підготувати звіт згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

Контрольні питання

1. Які основні особливості радіомодулів в Arduino?
2. Опишіть алгоритм роботи радіомодулів в Arduino та поясніть як працює перепивання в даному алгоритмі.
3. Поясніть основні функції бібліотеки *RCSwitch.h*, що використовуються в роботі.
4. Опишіть особливості датчику освітленості та зазначте параметри підключення Flying Fish – MH Sensor Series.
5. Які особливості має датчик звуку LM393 та які функції і бібліотеки треба використовувати при роботі з ним в Arduino?

4.14 Лабораторна робота №14. Програмування звукового підсилювача класу D

Мета: Отримати навички роботи зі звуковими підсилювачами, їх програмуванням, використовуючи плату Arduino.

Завдання: Реалізувати підсилення та керування гучномовцем на основі підсилювача класу D.

Обладнання: плата Arduino; проводи; гучномовець; adafruit 20W stereo amplifier MAX9744; потенціометр; 2 кнопки.

Загальні відомості

Функція звукового підсилювача полягає у відтворенні вхідного сигналу з необхідною гучністю і потужністю, з мінімальним розсіюванням енергії та малими спотвореннями. Підсилювач повинен мати хороші характеристики в діапазонах звукових частот, які знаходяться в області 20–20000 Гц (для вузькосмугових динаміків, наприклад, сабвуфера або високочастотної голівки, діапазон менше). Вихідна потужність варіюється в широких межах в залежності від призначення підсилювача – від міліват у головних телефонів, до декількох ват в телевізорі і персональному комп'ютері (ПК); сотні ват в найбільш потужних домашніх або комерційних аудіосистем для театрів і концертних залів.

Найпростішим варіантом реалізації підсилювача звуку – використання транзисторів в лінійному режимі, що дозволяє отримати на виході збільшену вхідну напругу.

У звичайному підсилювачу вихідний каскад містить транзистори, що забезпечують необхідне миттєве значення вихідного току. У багатьох аудіосистемах вихідні каскади працюють в класах А, В і АВ. У порівнянні з вихідним каскадом, що працює в класі D, потужність розсіювання в лінійних каскадах велика навіть у випадку їх ідеальної реалізації. Це забезпечує класу D значну перевагу у багатьох випадках, внаслідок меншого тепловиділення,

зменшення розмірів і відповідно вартості приладу, збільшення роботи автономних пристроїв.

Підсилювачі класу D не є, в загальному випадку, цифровими пристроями. Найпростіша і найбільш розповсюджена схема упідсилювача класу D з синхронною широтно-імпульсною модуляцією (ШІМ) – це повністю аналогова схема. В її основі – задаючий генератор сигналів трикутної форми, частота яких достатньо висока, наприклад, 500 кГц, швидкодіючий компаратор, і формувач імпульсів, що відкриває вихідні транзистори. Якщо миттєве значення вхідної напруги перевищує напругу на виході генератора, компаратор подає сигнал на відкриття транзисторів верхнього плеча, якщо ж ні – то на відкриття транзисторів нижнього плеча. Формувач імпульсів підсилює ці сигнали, почергово відкриваючи транзистори верхнього і нижнього плечей, а включений між ними і навантаженням, LC-фільтр зглажує струм, що віддає в навантаження. На виході підсилювача – посилена і демодульована, очищена від високочастотних завад копія вхідної напруги.

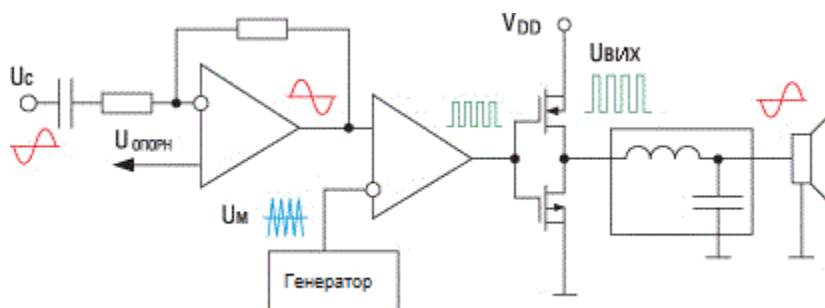


Рис. 4.25 – Типова структурна схема підсилювача класу D

Опис підсилювача MAX9744

Двоканальний підсилювач класу D MAX9744 зібраний на невеликій платі, забезпечує вихідну потужність до 20 Вт на канал на навантаження 4-8 Ом. Має роз'єм 3,5 мм для навушників. Також має як аналогові, так і цифрові регулятори гучності. До мікроконтролера підключається по інтерфейсу I2C.

Технічні характеристики:

- Живлення від напруги 4,5В-14В постійного струму;
- Енергоефективність до 93% (88-93% у типових підсилювачах);
- Струм спокою 20 мА;
- Максимальне підсилення до 29,5 дБ;
- Цифровий та аналоговий вхід;
- Забезпечує зниження електромагнітних без фільтра;
- Вихід стерео, потужністю до 20 Вт на канал;
- Захист від КЗ та теплового перевантаження.

Розташування виводів на платі наведено на рис. 4.26, без конденсатора живлення та роз'ємів для підключення звукових колонок.

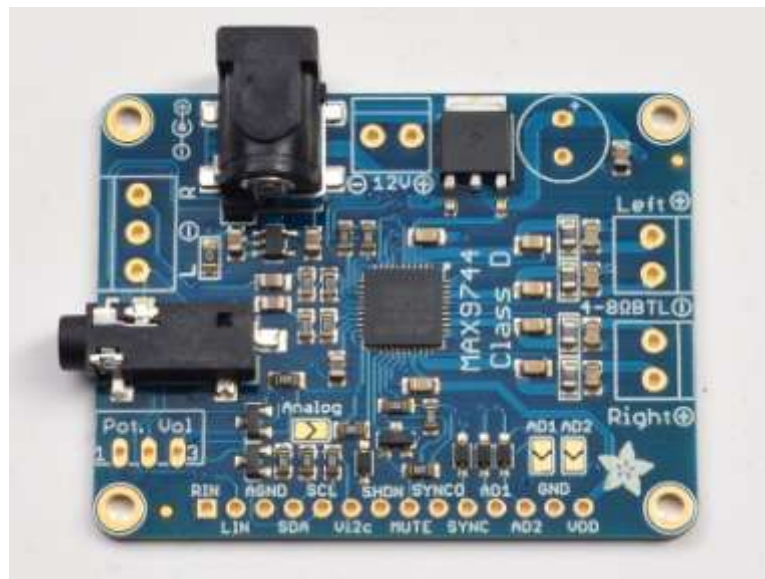


Рис. 4.26 – Розташування портів на платі

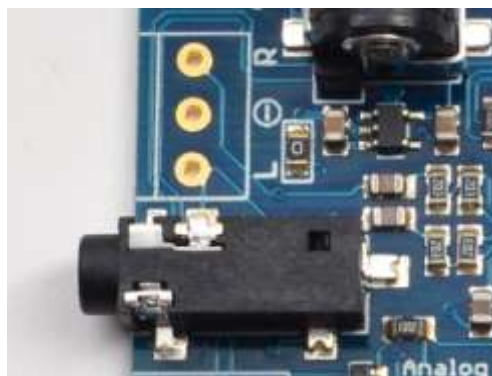


Рис. 4.27 – Розташування аудіо входів на платі

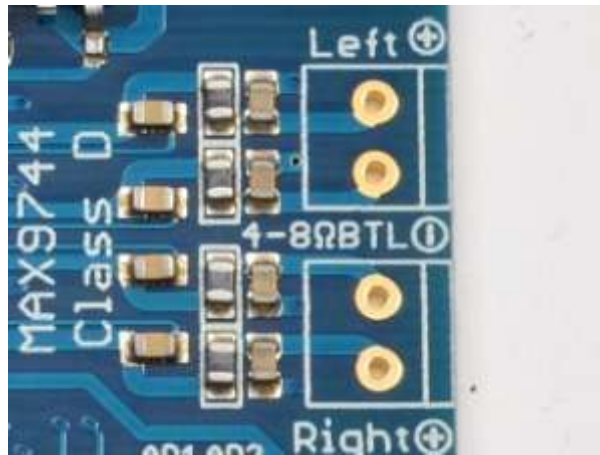


Рис. 4.28 – Розташування виходів гучномовців

Динаміки можуть бути імпедансом від 4 до 8 Ом, максимальна потужність 20 Вт. Оскільки це підсилювач класу D, сигнал на його виходах є високочастотним (~ 300 кГц) імпульсами прямокутної форми. Індуктивність динаміка згладжує цей сигнал як фільтр на звукових частотах 20-20 кГц.



Рис. 4.29 – Додаткові контакти

Якщо підсилювач використовується в аналоговому режимі, то до Pot. Vol контактів підключається потенціометр 1 кОм, а до додаткових контактів (рис. 4.29) немає потреби підключатися. Перемички AD1, AD2, Analog замкнено, для того, щоб повідомити мікроконтролеру, що буде використано потенціометр для регулювання гучності.

У підсилювачі використовують наступні контакти:

- RIN - дублікат правого вхідного аудіоканалу.
- LIN - дублікат лівого вхідного аудіоканалу.

- AGND – аналогова земля.
- SDA - цифровий режим для керування гучністю через i2c.
- SCL - цифровий режим для керування гучністю через i2c.
- Vi2c - посилення напруги I2C для логіки. Використовується тільки в цифровому режимі - підключають до 3В або 5В, незалежно від того, який мікроконтролер використовують.
- SHDN - цифровий вимикач. Підключають до землі, щоб вимкнути всю мікросхему і перевести її в режим низького енергоспоживання.
- MUTE - цифровий вимикач. Підключають до землі, щоб вимкнути лише аудіовиходи, це швидше, ніж вимикання та зберігає цифровий рівень звуку I2C.
- SYNCO - Синхронізований вихід - високочастотний сигнал від генератора ШІМ, близько 1,4 МГц.
- SYNC - Синхронізований вхід, для досвідчених користувачів, які хочуть використовувати свою частоту ШІМ, тримати її вище 800 кГц.
- AD1 - I2C адреса контакту №1.
- AD2 - I2C адреса контакту №2.
- GND – земля.
- VDD – живлення 5-12 В постійного струму. Можна використовувати для живлення інших пристроїв, що підтримують напругу 5-12 В постійного струму.

Перемикач ззаду дозволяє перемикати аналогову землю (AGND) на цифрову (DGND).

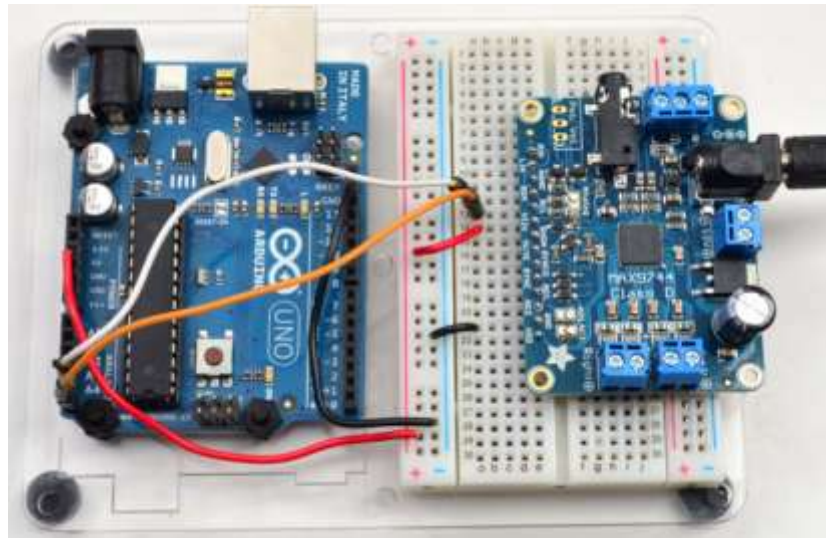


Рис. 4.30 – Вигляд зібраного макету

Хід виконання роботи

1. Скласти макет згідно завданню та схеми на рис.6.
2. Підключити схему до живлення (5 В).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи макету.

Тестовий код регулювання гучності

```
#include <Wire.h>
// 0x4B адреса i2c за замовчуванням
#define MAX9744_I2CADDR 0x4B
// Будемо відстежувати рівень гучності у цій змінній.
int8_t thevol = 31;
void setup() {
    Serial.begin(9600);
    Serial.println("MAX9744 demo");
    Wire.begin();
    if (! setvolume(thevol)) {
        Serial.println("Failed to set volume, MAX9744 not found!");
        while (1);
    }
}
// Щоб налаштувати гучність, треба передати 6-бітний код
гучності в шину i2c.
```

```

boolean setvolume(int8_t v) {
    // cant be higher than 63 or lower than 0
    if (v > 63) v = 63;
    if (v < 0) v = 0;
    Serial.print("Setting volume to ");
    Serial.println(v);
    Wire.beginTransmission(MAX9744_I2CADDR);
    Wire.write(v);
    if (Wire.endTransmission() == 0)
        return true;
    else
        return false;
}
// Зчитування «+» та «-» для задавання гучності.
void loop() {
    if (! Serial.available()) return;
    char c = Serial.read();
    if (c == '+') {
        thevol++;
    }
    else if (c == '-') {
        thevol--;
    }
    else
        return;
    if (thevol > 63) thevol = 63;
    if (thevol < 0) thevol = 0;
    setvolume(thevol);
}

```

Завдання

1. Провести тестування підсилювача: підключити звукові колонки, підключити плату до живлення, під'єднати звуковий плеєр та перевірити звучання.

2. Аналоговий режим: перевести схему у аналоговий режим; змінювати гучність за допомогою потенціометра. Оцінити програвання аудіофрагмента.
3. Цифровий режим: підключити I2C виходи плати до Arduino, за прикладом минулих лабораторних робіт.
4. Завантажити в Arduino тестовий код зміни гучності через команди послідовного порта за допомогою протоколу I2C (використовуючи бібліотеку Wire.h), та перевірити роботу підсилювача.
5. Написати програму для регулювання гучності динаміків *згідно варіантів завдання*. Адреса пристрою за замовчуванням: 0x4B .

Варіанти виконання роботи:

Для парних: через окремий потенціометр, який підключити до Arduino.

Для непарних: через натискання кнопок, які будуть відповідати «+» та «-».

Оформлення звіту згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

Контрольні питання

1. Назвіть основні переваги використаного в роботі підсилювача.
2. Які звукові пристрої можуть підключатися до плати підсилювача та які вони повинні мати характеристики для цього?
3. У яких режимах працює схема та що для цього потрібно?
4. Куди підключається потенціометр на платі підсилювача та які характеристики він повинен мати?
5. Опишіть принцип роботи протоколу I2C.

Список посилань

1. Офіційний сайт Arduino [Електронний ресурс]. - Режим доступу: <https://www.arduino.cc>.
2. Сайт Arduino.ua Плати Arduino Nano [Електронний ресурс]. - Режим доступу: <https://doc.arduino.ua/ru/hardware/Nano>
3. Сайт Arduino.ua Плати Arduino Uno [Електронний ресурс]. - Режим доступу: <https://doc.arduino.ua/ru/hardware/Uno>
4. Сайт Arduino.ua Плати Arduino Mega2560 [Електронний ресурс]. - Режим доступу: <https://doc.arduino.ua/ru/hardware/Mega2560>
5. Специфікації Philips I²C [Електронний ресурс]. - Режим доступу: http://www.nxp.com/acrobat_download/literature/9398/39340011.pdf
6. Reading and Writing 1-Wire® Devices Through Serial Interfaces [Електронний ресурс]. - Режим доступу: <https://www.maximintegrated.com/en/app-notes/index.mvp/id/74>
7. Будова та принцип роботи сервоприводу. URL: <https://radiomodel.in.ua/budova-ta-printsip-roboti-servoprivodu/> (дата звернення: 16.03.2019).
8. ServoAttach Программирование в Украине. URL: <https://doc.arduino.ua/ru/prog/Servo> (дата звернення: 16.03.2019).
9. Arduino - ToneMelody. URL: <https://www.arduino.cc/en/Tutorial/toneMelody> (дата звернення: 16.03.2019).
10. Подключение дисплея LCD 1602 к Arduino по I2C/IIC. URL: <https://arduinomaster.ru/datchiki-arduino/lcd-i2c-arduino-display-ekran/> (дата звернення: 17.03.2019).
11. LCD I2C модуль подключение к Arduino. URL: <http://zelectro.cc/LcdI2Cmodule> (дата звернення: 17.03.2019).
12. Универсальная библиотека iarduino_RTC. URL: <http://iarduino.ru/file/235.html> (дата звернення: 17.03.2019).
13. Подключение часов DS3231 к Arduino. URL: <https://voltiq.ru/connecting-ds3231-to-arduino/> (дата звернення: 17.03.2019).
14. 10. Рефлексометр. URL: <https://usamodelkina.ru/13191-refleksometr.html> (дата звернення: 17.03.2019).
15. Беспроводная связь для Arduino на 433МГц (FS1000A и MX-RM-5V). URL: <http://radiolis.pp.ua/arduino/49-besprovodnaya-svyaz-arduino-fs1000a-mx-rm-5v> (дата звернення: 17.03.2019).

16. Ардуино: радиомодуль на 433 МГц. URL:
<http://robotclass.ru/tutorials/arduino-radio-433mhz/> (дата звернення:
17.03.2019).
17. Flying Fish – MH Sensor Series – двоичный датчик освещенности. URL:
<http://musuk.guru/blog/flying-fish-mh-sensor-series> (дата звернення:
17.03.2019).
18. Универсальный звуковой датчик (аналог и цифра). URL:
<https://arduino.ua/prod1211-universalnii-zvykovoi-datchik-analog-i-cifra> (дата
звернення: 17.03.2019).