

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до виконання самостійної роботи
з дисципліни
“Операційні системи”
для студентів
спеціальності 6.05010301
“Програмне забезпечення систем” та
спеціальності 6.05010102
“Інженерія програмного забезпечення”
усіх форм навчання

2017

Методичні вказівки до виконання самостійної роботи з дисципліни “Операційні системи” для студентів спеціальності 6.05010301 “Програмне забезпечення систем” та спеціальності 6.05010102 “Інженерія програмного забезпечення”, усіх форм навчання / Укл.: С.М. Сердюк – Запоріжжя: ЗНТУ, 2017. – 26с.

Укладачі: С. М. Сердюк, доцент, к.т.н.,

Рецензент: С.К. Корнієнко, доцент, к.т.н.

Відповідальний
за випуск: С.М. Сердюк, доцент, к.т.н.,

Затверджено
на засіданні кафедри
“Програмних засобів”

Протокол № _____
від _____ 2017

ЗМІСТ

ВСТУП.....	4
1 Загальні відомості про дисципліну.....	8
2 Теоретичні питання з дисципліни.....	9
3 Перелік лабораторних робіт.....	11
4 Прийом і передача даних з використанням інтерфейсів I2C та SPI	13
ЛІТЕРАТУРА.....	25

ВСТУП

Самостійна робота студентів – це спланована пізнавальна, організаційно і методично направлена на досягнення результату діяльність студента, яка здійснюється без прямої допомоги викладача. Вона є основним способом оволодіння студентами навчальним матеріалом у час, вільний від обов'язкових аудиторних занять [1].

Мета виконання самостійної роботи – поглиблення, узагальнення і закріплення теоретичних знань і практичних умінь студентів з дисципліни, що вивчається, шляхом вироблення вміння самостійної роботи з навчальною і фаховою науково-технічною літературою.

Завдання самостійної роботи студентів:

- розвиток творчих здібностей та активізація розумової діяльності студентів;
- формування у студентів потреби безперервного самостійного поповнення знань;
- розвиток морально-вольових зусиль студентів;
- самостійна робота студентів як результат їх морально-вольових зусиль;
- навчити студентів самостійно працювати з літературою;
- навчити студентів творчо сприймати навчальний матеріал і осмислювати його;
- •сформувати навички щоденної самостійної роботи з метою одержання та узагальнення знань, умінь і навичок.

Форми самостійної роботи студентів:

- підготовка до лекцій і лабораторних занять;
- опрацювання теоретичних основ прослуханого лекційного матеріалу;
- вивчення окремих тем або питань, що передбачені для самостійного опрацювання;
- опрацювання навчальної та методичної літератури;
- опрацювання наукової літератури та періодичних видань;
- виконання розрахунково-графічного завдання,
- написання рефератів,
- контрольної роботи – для студентів заочної форми навчання.

Зміст самостійної роботи студентів з дисципліни визначається навчальною програмою дисципліни та робочою навчальною програмою вивчення дисципліни. На самостійну роботу виноситься: частина теоретичного матеріалу, менш складного за змістом, окремі практичні завдання та роботи, що не потребують безпосереднього керівництва викладача.

Складовими самостійної роботи студента є обов'язкова та вибіркова частини. Обов'язкова складова передбачає опанування програмного матеріалу дисципліни. Вибіркова складова передбачає виконання завдань, які студент вибирає з метою підвищення свого професійного рівня, особистого рейтингу. Як правило, до таких завдань відноситься робота науково-дослідницького і творчого характеру.

Навчально-методичні засоби самостійної роботи студентів:

- основна література (підручник, конспект лекцій, навчальні та методичні посібники);
- додаткова література (наукова, фахова, періодична);
- методичні матеріали.

Місцями виконання самостійної роботи з дисципліни є бібліотека, навчальні кабінети, комп'ютерні класи та лабораторії університету, а також дім студента.

При використанні студентами складного обладнання чи устаткування, складних систем доступу до інформації передбачаються можливості отримання необхідної консультації або допомоги з боку викладача.

Вимоги до самостійної роботи студента:

– робота має бути виконана особисто студентом або групою студентів, де кожен її член самостійно виконує свою частку колективної роботи;

– робота повинна являти собою закінчену розробку (чи її етап), де розкриваються й аналізуються актуальні проблеми з певної теми або її окремих аспектів;

– робота має демонструвати достатню компетентність автора (авторів) у розкритті питань, що досліджуються;

– робота повинна мати навчальну, наукову й (або) практичну спрямованість і значимість, містити певні елементи новизни (при виконанні науководослідної роботи).

Оформлення звітів зі самостійної роботи студентів здійснюється відповідно до вимог, розроблених кафедрою, та інших нормативних документів, що стосуються виконання та оформлення наукових, навчально-методичних та інших робіт.

Керівництво самостійною роботою студентів – це індивідуально-консультативна робота – це форма організації навчальної роботи викладача зі студентами, яка здійснюється шляхом створення необхідних умов для виявлення і розвитку індивідуальних здібностей студента на основі особистіснодіяльнісного підходу. Вона проводиться з метою посилення мотивації студентів до пізнавальної діяльності і спрямування її в необхідному напрямку.

Індивідуально-консультативна робота, як правило, проводиться у вигляді консультацій, інколи – індивідуальних занять у формі: діалогу з різних навчальних проблем; перевірки виконання завдань; виконання індивідуальних завдань (курскових та дипломних проєктів (робіт), розрахункових, творчих робіт тощо); евристичної бесіди; наукової роботи та ін.

Консультація – одна з форм організації навчального процесу, що проводиться з метою отримання студентом відповіді на окремі теоретичні чи практичні питання, пояснення певних теоретичних положень та їх практичного застосування. При цьому виділяють такі види консультацій: тематичні – проводяться за певними темами дисципліни або найбільш складними питаннями програмного матеріалу; цільові – використовуються перед проведенням модульної контрольної роботи або іншого виду поточного чи підсумкового контролю; активні – консультації з використанням активних методів навчання (наприклад, у формі прес-конференції); з самостійної роботи – проводяться при підведенні підсумків самостійної роботи.

Формами контролю студентів за якістю оволодіння навчальним матеріалом є: самоконтроль за допомогою контрольно-тестових завдань та контроль з боку викладача, який здійснюється за допомогою методів поточного і підсумкового контролю.

Методами поточного контролю є:

- усне опитування студентів на лабораторних заняттях;
- перевірка практичних завдань, виконаних студентами індивідуально;
- перевірка рефератів та організація їх презентацій;
- співбесіда.

Формами підсумкового контролю є:

- проведення контрольних робіт в аудиторії (за тематичними модулями);
- екзамен (залік) за передбаченими програмою курсу питаннями та практичними завданнями.

Облік успішності студентів з виконання самостійної роботи здійснюють викладачі у журналах обліку успішності.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ДИСЦИПЛІНУ

Метою дисципліни є вивчення: ролі, задач, структури сучасних операційних систем (ОС) та принципів їх проектування; засобів використання програмних інтерфейсів додатків (API); засобів керування загальними зовнішніми пристроями комп'ютерних систем на рівні апаратних засобів; базової системи вводу-виводу.

Внаслідок вивчення дисципліни студенти повинні:

- вивчити структури, набір команд та можливості сучасних ОС та мікропроцесорів;
- вивчити принципи організації паралелізму (багатозадачності);
- вивчити принципи планування та диспетчеризації процесів;
- вивчити принципи керування оперативною та віртуальною видами пам'яті;
- вивчити структури та засоби підключення до ОС додаткових драйверів зовнішніх пристроїв та резидентних програм;
- одержати навички роботи із жорстким диском та досвід по використанню сервісних (обслуговуючих) програм.

На основі вивчення дисципліни "Операційні системи" студенти повинні знати:

- методи структуризації та абстракції, ресурси систем, функції програмних інтерфейсів додатків, механізм переривань, організацію режимів роботи користувача/супервізору та захисту, принципи синхронізації, стани та діаграми станів, паралельне виконання та блокування, розподіл фізичної пам'яті, оверлеї, сторінкову організацію пам'яті та сегментації, стратегії свопінгу та вивантаження сторінок, кешування, файлові системи.

вміти:

- здійснювати трансформацію інформації, отриманої від користувачів в чітко та однозначно визначені програмні вимоги, визначати високорівневі вимоги (стратегічні цілі), для досягнення яких створюється програмна система, використовувати можливості апаратного забезпечення, використовувати можливості операційних систем, проектувати системне програмне забезпечення.

2 ТЕОРЕТИЧНІ ПИТАННЯ З ДИСЦИПЛІНИ

2.1 Операційна система MS-DOS

Структура та завантаження ОС MS-DOS. Прошарки програм в оперативній пам'яті персонального комп'ютера (ПК). Карта пам'яті ПК в реальному режимі. Види пам'яті ПК. Стандарти керування пам'яттю – EMS, XMS, VCPI, DPMS, RSIS.

Література [1, с.117-135; 2, с.177-194].

2.2 Основні відомості про жорсткі диски

Стандарти інтерфейсів жорстких дисків. Обслуговування жорсткого диску. Логічна організація жорсткого диска. Таблиця параметрів жорсткого диска.

Література [1, с. 15-55, с. 111-145; 3, с. 94-118; 4, с. 315-321, с. 322-391; 6, с. 82-124].

2.3 Захищений режим

Архітектура процесора Intel 386 та його основні технічні показники. Робота процесора Intel 386 в дійсному режимі та режимі віртуального процесора 8086. Команди процесора Intel 386. Особливості захищеного режиму процесора Intel 386. Селектори та дескриптори. Способи адресації. Сторінкова організація пам'яті та сегментація.

Література [3, с. 18-22, с. 27-30; 4, с. 386-394, с. 411-414].

2.4 Характеристика ОС WINDOWS

Відміни ОС WINDOWS від ОС MS-DOS. Файлова система WINDOWS 9x: особливості роботи з довгими іменами файлів; монопольне блокування тому; кеширування в WINDOWS 9x; файлове відображення (file mapping); види дискових/файлових драйверів WINDOWS. Файлова система NTFS.

Поняття процесів і потоків. Багатозадачність. Пріоритети та планування. Керування процесами: створення процесів; одержання інформації щодо процесу; контроль над процесом; знищення процесу. Керування потоками: створення потоків; одержання інформації щодо потоку; контроль над потоком; знищення потоку; синхронізація виконання багато поточкових прикладних програм.

Глобальна та локальна віртуальна пам'ять. Стандартні функції керування пам'яттю мови С. Функції WINAPI для роботи з віртуальною пам'яттю. Захист пам'яті.

Література [5, с. 525-640; 6, с. 419-493].

2.5 Основи ОС UNIX

Початкове завантаження ОС. Особливості запуску в різноманітних ОС Solaris, HP-UX, IRIX, Sun OS, OSF/ 1 2.0 , BSD /OS 1.1. Вхід користувача, ввід простих команд. Поняття процесу початок роботи з процесами. Фонові процеси. Процеси-демони. Основні демони: init; cron; inetd. Конфігурування inetd. Файл services. Перезапуск демону inetd. Знищення процесів. Вихід із системи.

Література [7 , с.13-22; 8 ,с.27-57 , с.758-763; 9 , с.42-64].

2.6 Файлова система ОС UNIX

Структура файлової системи, іменування файлів. Створення зв'язку з файлом. Механізм захисту файлів і змін прав доступу. Означення прав доступу. Блоки файлової системи, структура каталогу.

Література [7 , с.28-57; 8 , с.73-86; 9 , с.114-139]

2.7 Командний інтерпретатор bash

Взаємодія з інтерпретатором bash, змінні оболонки. Арифметичні дії, аналіз ланцюга дій. Умонтовані команди, керування програмами. Основи написання сценаріїв (скриптів).

Література [9 , с.437-501].

2.8 Комунікації, які підтримуються системою UNIX

Системи електронної пошти. Призначення , особливості, аналіз версій і режими роботи програми транспортування електронної пошти sendmail. Конфігурування програми sendmail. Забезпечення захисту і секретності інформації. Мережева модель TCP/IP. INTERNET: адресація, служби, довідкова інформація.

Література [8 , с.271-317, с.345-398, с. 457-500; 9 , с. 183-382].

3 ПЕРЕЛІК ЛАБОРАТОРНИХ РОБІТ

3.1 Лабораторна робота №1. Функції операційної системи MS-DOS: визначення версії ОС.

Мета роботи: Практичне вивчення роботи основних функцій операційної системи MS-DOS.

3.2 Лабораторна робота № 2. Використання звукових можливостей IBM PC

Мета - вивчити способи генерації звука, навчитися використовувати звукові можливості IBM-сумісних ПЕОМ та практично реалізувати запропоновані способи генерації звука.

3.3 Лабораторна робота № 3. Функції припинення процесу.

Мета роботи: вивчити функції припинення обчислювального процесу, навчитися їх практичному використанню.

3.4 Лабораторна робота № 4. Основи Windows. Графічний інтерфейс користувача. Визначення параметрів жорсткого диску.

Мета роботи: вивчити принципи створення графічного інтерфейсу в ОС Windows, навчитися використовувати функції WinAPI для визначення параметрів логічних дисків системи.

3.5 Лабораторна робота № 5. Базові операції Linux-системи. Основні команди ОС.

Мета роботи: Придбати основні навички роботи в командному рядку ОС Linux. Освоїти основні команди керування файлами, каталогами та потоками даних в ОС Linux.

3.6 Лабораторна робота № 6. Командний інтерпретатор bash. Основи написання сценаріїв (скриптів).

Мета роботи: Освоїти командний інтерпретатор bash. Набути початкових навичок написання командних файлів (скриптів).

3.7 Лабораторна робота № 7. Системне адміністрування Linux.

Мета роботи: освоєння програмного забезпечення, призначеного для заведення і видалення користувача і групи користувачів, зміни пароля користувача, зміни облікових записів про користувача і групу. Пакет sudo. Монтування файлових систем.

4 ПРИЙОМ І ПЕРЕДАЧА ДАНИХ З ВИКОРИСТАННЯМ ІНТЕРФЕЙСІВ I2C ТА SPI

4.1 Стислі теоретичні відомості

4.1.1 Апаратна обчислювальна платформа Arduino

Основними компонентами Arduino [10] є плата вводу/виводу та середовище розробки на мові Processing/Wiring. Arduino може використовуватися як для створення автономних інтерактивних об'єктів, так і підключатися до програмного забезпечення, яке виконується на комп'ютері (наприклад: Adobe Flash, Processing, Max/MSP, Pure Data, SuperCollider).

Апаратна частина. Плата Arduino складається з мікроконтролера Atmel AVR, а також елементів обв'язки для програмування та інтеграції з іншими пристроями. На багатьох платах наявний лінійний стабілізатор напруги +5В або +3,3В. Тактування здійснюється на частоті 16 або 8 МГц кварцовим резонатором. У мікроконтролер записаний завантажувач (bootloader), тому нема потреби у зовнішньому програматорі.

На концептуальному рівні усі плати програмуються через RS-232 (послідовне з'єднання), але реалізація даного способу різниться від версії до версії. Новіші плати програмуються через USB, що можливо завдяки мікросхемі конвертера USB-to-Serial FTDI FT232R. У версії платформи Arduino Uno в якості конвертера використовується контролер Atmega8 у SMD-корпусі. Дане рішення дозволяє програмувати конвертер таким чином, щоб платформа відразу розпізнавалася як миша, джойстик чи інший пристрій за вибором розробника зі всіма необхідними додатковими сигналами керування. У деяких варіантах, таких як Arduino Mini або неофіційній Boarduino, для програмування потрібно підключити до контролера окрему плату USB-to-Serial або кабель.

Плати Arduino дозволяють використовувати значну кількість I/O виводів мікроконтролера у зовнішніх схемах. Наприклад, у платі Decimila доступно 14 цифрових входів/виходів, 6 із яких можуть видавати ШІМ сигнал, і 6 аналогових входів. Ці сигнали доступні на платі через контактні площадки або штирьові розніми. Також існує декілька видів зовнішніх плат розширення, які називаються "shields" ("щити"), які приєднуються до плати Arduino через штирьові розніми.

Програмне забезпечення. Інтегроване середовище розробки Arduino це багатоплатформовий додаток на Java, що включає в себе редактор коду, компілятор і модуль передачі прошивки в плату. Середовище розробки базується на мові програмування Processing та спроектоване для програмування новачками. Мова програмування аналогічна мові Wiring. Строго кажучи, це C++, доповнений деякими бібліотеками. Програми обробляються за допомогою препроцесора, а потім компілюється за допомогою AVR-GCC.

Програми Arduino пишуться на мові програмування C або C++. Середовище розробки Arduino поставляється разом із бібліотекою програм, яка називається "*Wiring*", яка бере початок від проекту Wiring, який дозволяє робити багато стандартних операцій вводу/виводу набагато простіше. Користувачам необхідно визначити лише дві функції, для того щоб створити програму, яка буде працювати за принципом циклічного виконання:

- `setup()`: функція виконується лише раз при старті програми і дозволяє задати початкові параметри;
- `loop()`: функція виконується періодично доки плата не буде вимкнена.

Приклад 1. Розглянемо типову програму для мікроконтролеру, яка посилає команду блимати світловому діоду в середовищі Arduino [11]. Для цього потрібні наступні апаратні засоби:

- плата Arduino або Genuino
- світловий діод
- резистор 220 Ом.

Для побудови схеми, один кінець резистора підключають до 13-го контакту Arduino, а довгу ніжку світлодіода (анод) на інший кінець резистора. Коротку ніжку світлодіода (катод) підключають до Arduino GND (земля), як показано на на рис. 4.1.

Програма.

```
#define LED_PIN 13
void setup () // обов'язкова процедура, що запускається на
початку програми
{
    pinMode (LED_PIN, OUTPUT); // оголошення
використовуваного порту, led - номер порту, другий аргумент - тип
використання порту - на вхід (INPUT) або на вихід (OUTPUT)
}
```

```

void loop () //обов'язкова процедура loop, що запускається
циклічно після процедури setup
{
    digitalWrite (LED_PIN, HIGH); // Включити світлодіод - ця
команда використовується для включення або виключення напруги на
цифровому порту; led - номер порту, другий аргумент - включення
(HIGH) або вимикання (LOW)
    delay (1000); // Зачекати одну секунду (1000 мілісекунд)
    digitalWrite (LED_PIN, LOW); // Вимкнути світлодіод
    delay (1000); // Зачекати одну секунду
}

```

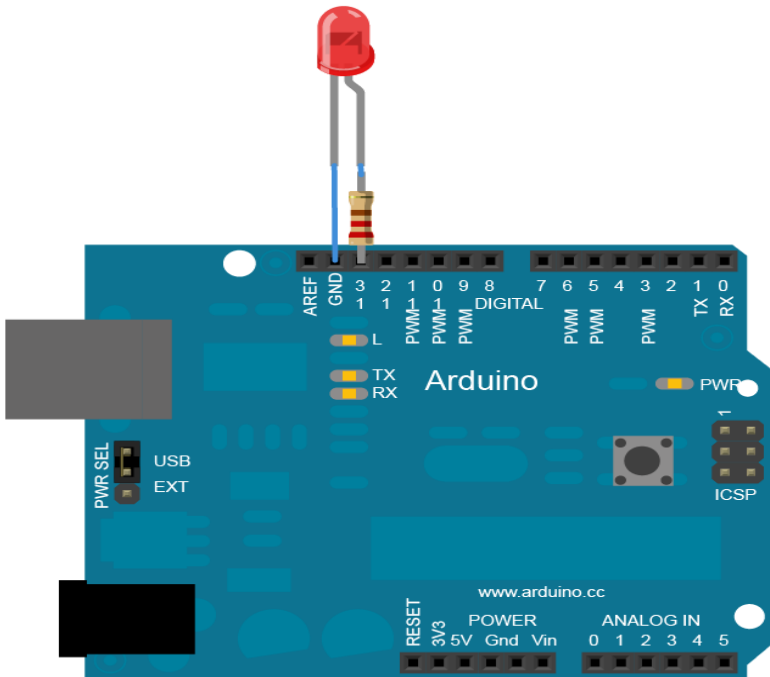


Рисунок 4.1 – Схема підключення світлодіоду на Arduino

4.1.2 Ультразвуковий далекомір PING)))

PING))) - це ультразвуковий далекомір від компанії Parallax. Він визначає дистанцію до найближчого об'єкта, який знаходиться перед ним (пошуковий діапазон від 2 см до 3 м). Принцип його роботи - він

відсилає пучок ультразвукових хвиль, а потім чекає відлуння, що створюваного «відпружинюванням» цього сигналу від об'єкту, що знаходиться попереду. Arduino відсилає віддалеміру короткий імпульс, щоб запустити процес виявлення, а потім через той же контакт чекає імпульсу за допомогою функції *pulseIn()*. Тривалість відповідного імпульсу еквівалентна часу, який йде на те, щоб ультразвук дійшов до об'єкта і повернувся до датчика. За допомогою швидкості звуку цей час можна конвертувати в дистанцію.

Приклад 2. Розглянемо програму для мікроконтролеру, яка зчитує дані далекоміра (PING))), а потім повертає дані про дистанцію до найближчого об'єкта [12]. Апаратні засоби:

- плата Arduino;
- ультразвуковий далекомір (PING)));
- провода-перемички.

Електрична схема підключення (PING))) до Arduino наведена на рис. 4.2. 5-вольтний контакт (PING))) підключений до 5-вольтового контакту на Arduino, а контакт з «землею» на (PING))) з'єднаний з «землею» на Arduino. Контакт SIG (тобто сигнал) на (PING))) приєднаний до сьомого цифровому контакту на Arduino. Результатом є ланцюг наведений на рис.4.3.

Програма.

```
const int pingPin = 7; // Задали контакт для датчика
void setup() {
  // Ініціалізуємо послідовну передачу даних:
  Serial.begin(9600);
}
void loop() {
  // Задаємо змінні для тривалості імпульсу, а також підсумкову
  дистанцію - в дюймах і сантиметрах:
  long duration, inches, cm;
  // (PING))) активується імпульсом HIGH тривалістю 2 або більше
  μs

  // Перед цим дамо короткий імпульс LOW, щоб «змити» імпульс
  HIGH (якщо той залишився з минулого разу):
```

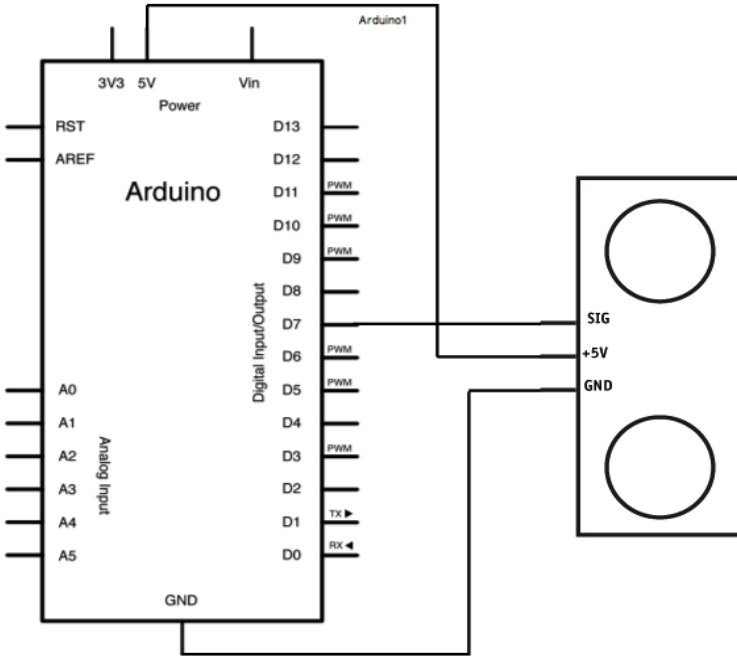


Рисунок 4.2 – Електрична схема підключення PING))) до Arduino

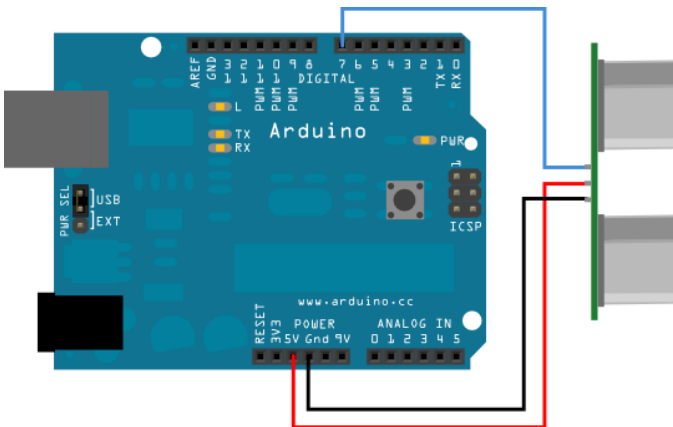


Рисунок 4.3 – Ланцюг підключення PING))) до Arduino

```

pinMode(pingPin, OUTPUT);
digitalWrite(pingPin, LOW);
delayMicroseconds(2);
digitalWrite(pingPin, HIGH);
delayMicroseconds(5);
digitalWrite(pingPin, LOW);
// Скористаємося тим же контактом для зчитування
сигналу від PING))
// Тривалість сигналу HIGH - це і є час (в  $\mu$ s) від
відправки ультразвукових хвиль до отримання відлуння
//від об'єкту
pinMode(pingPin, INPUT);
duration = pulseIn(pingPin, HIGH);
// Перетворимо час в дистанцію:
inches = microsecondsToInches(duration);
cm = microsecondsToCentimeters(duration);
Serial.print(inches);
Serial.print("in, ");
Serial.print(cm);
Serial.print("cm");
Serial.println();
delay(100);
}
long microsecondsToInches(long microseconds)
{
// Згідно з паспортними даними до PING)),
швидкість ультразвуку 73,746  $\mu$ s/дюйм це дозволяє
// нам визначити дистанцію, пройдену сигналом (туди і
назад)
// Тому ми ділимо її на 2, щоб отримати реальну
дистанцію до об'єкта
return microseconds / 74 / 2;
}
long microsecondsToCentimeters(long microseconds)
{
// Швидкість звуку - 331 м/с або 30  $\mu$ s/см
// Імпульс рухається туди і назад, тому для
визначення реальної дистанції потрібно поділити
// отримане значення на 2
return microseconds / 30 / 2;
}

```

4.1.3 Використання інтерфейсу SPI. Датчик температури та атмосферного тиску SCP1000

Датчик SCP1000 може зчитувати як атмосферний тиск, так і температуру, а потім передавати ці дані через SPI. Більш докладно про регістри управління дивіться в datasheet SCP1000 [13].

Для підключення SCP1000 потрібні наступні апаратні засоби:

- плата Arduino;
- плата з датчиком SCP1000;
- макетна плата Breadboard або плата для прототипування.

Електрична схема підключення SCP1000 до Arduino наведена на рис. 4.4.

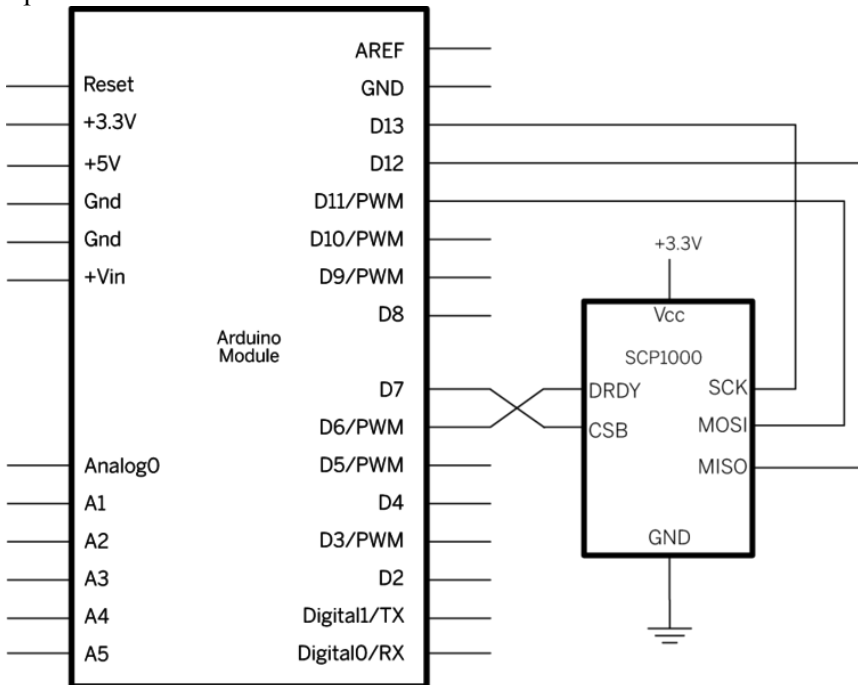


Рисунок 4.4 – Електрична схема підключення SCP1000 до Arduino

Датчик потрібно під'єднати до другого, сьомого, одинадцятого, дванадцятого і тринадцятого контактів тандему «Arduino і Shield», а також підключити живлення через вихідний контакт 3,3-вольта. Контакт DRDY (Data Ready - дані готові для зчитування) підключіть до цифровому контакту 6, а контакт CSB (Chip Select - вибір веденого

пристрою) - до 7-го. Контакт MOSI (Master Out, Slave In; для передачі даних від ведучого пристрою до веденого) повинен бути підключений до цифровому контакту 11, а MISO (Master In, Slave Out - для передачі даних від веденого пристрою до ведучого) - до 12. Контакт SCK (Serial Clock - для передачі послідовного тактового сигналу) підключіть до цифрового контакту 13. Переконайтеся, що обидва пристрої підключені до однієї і тієї ж «землі». Результатом є ланцюг наведений на рис.4.5.

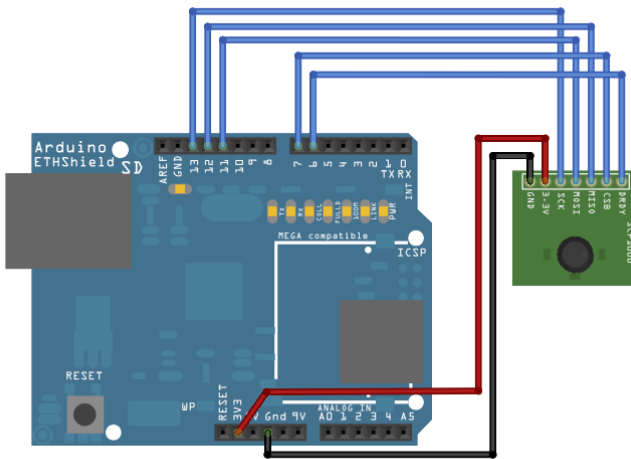


Рисунок 4.5 – Ланцюг підключення SCP1000 до Arduino

4.1.4 Використання бібліотеки Wire для роботи з I2C

Мікроконтролери (МК) Atmega (Ардуино) мають апаратну підтримку інтерфейсу I2C (TWI). Лінії інтерфейсу SDA і SCL у МК Atmega8/168/328, розташовані на пінах с номерами 27 (PC4) і 28 (PC5), відповідно.

На платах Arduino, лінія даних - SDA (data line) виведена на аналоговий пін 4, а лінія тактірованія - SCL (clock line) виведена на аналоговий пін 5. На Arduino Mega, SDA - це цифровий пін 20, а SCL - цифровий пін 21.

Для роботи з протоколом I2C, у Arduino є штатна бібліотека Wire [14], яка дозволяє взаємодіяти з I2C / TWI-пристроями, як в режимі Master, так і в режимі Slave.

У Arduino 1.0, бібліотека успадковується від Stream,
 class TwoWire : public Stream

{

що робить її використання схожим з іншими бібліотеками читання/запису (read/ write libraries). У зв'язку з цим, методи *send()* і *receive()* були замінені на *read()* і *write()*.

Розглянемо методи бібліотеки.

Методи:

```
void begin ();
void begin (uint8_t address);
void begin (int address)
```

здійснюють ініціалізацію бібліотеки Wire і підключення до шини I2C в якості Master або Slave. Як правило, викликаються тільки один раз.

Параметри:

Address - 7-бітна адреса пристрою (в режимі Slave). Якщо не вказано, то контролер підключається до шини в ролі Master.

Значень, що повертаються немає.

Метод

```
uint8_t requestFrom (uint8_t address, uint8_t quantity;
```

Використовується майстром для запиту байта від веденого пристрою. Байти можуть бути отримані за допомогою методів *available()* і *read()*.

Параметри:

Address - 7-бітний адресу пристрою для запиту байтів даних

Quantity - кількість запитаних байт

Повертає число лічених байт.

Метод

```
void beginTransmission (uint8_t address);
```

Починає передачу I2C для веденого пристрою (Slave) із заданою адресою. Потім, потрібно викликати метод *write()* для додавання послідовності байт в чергу призначених для передачі, і виконати саму передачу даних методом *endTransmission()*.

Параметри:

Address - 7-бітна адреса пристрою для передачі

Значень, що повертаються немає.

Метод

```
uint8_t endTransmission (void);
```

завершує передачу даних для веденого пристрою, яке було розпочато

beginTransmission() і, фактично, здійснює перечу байт, які були поставлені в чергу методом *write()*.

Повертається байт, який вказує статус передачі:

0: успіх

1: даних занадто багато і вони не поміщаються в буфер передачі.

Розмір буфера задається визначенням

```
#define BUFFER_LENGTH 32
```

2 отримали NACK на передачі адреси

3 отримали NACK на передачі даних

4: інша помилка

Методи

```
size_t write(uint8_t data);
```

```
size_t write(const uint8_t *data, size_t quantity);
```

Виклик:

```
Wire.write(value);
```

```
Wire.write(string);
```

```
Wire.write(data, length);
```

Ці методи записують дані від веденого пристрою у відповідь на запит майстра, або записують чергу байт для передачі від майстра до веденого пристрою (в проміжках між викликами *beginTransmission()* і *endTransmission()*).

Параметри:

value - значення для відправлення як одиничний байт;

string - рядок для відправлення як послідовність байт;

data -масив байт для відправлення;

length -число байт для передачі.

Повертається число записаних байт.

Приклад.

```
#include <Wire.h>
```

```
byte val = 0;
```

```
void setup()
```

```
{
```

```
    Wire.begin(); // підключення до шини I2C
```

```
}
```

```
void loop()
```

```
{
```

```
    Wire.beginTransmission(44); // передача для пристрою #44
```

(0x2c)

```
    // адреса пристрою вказується в документації (datasheet)
```

```
    Wire.write(val); // відправка байта val
```

```

Wire.endTransmission(); // передача даних
val++; // інкремент значення
if(val == 64) // якщо досягли 64 (max)
{
    val = 0; // починаємо з мінімального значення
}
delay(500);
}

```

Метод

```
void onReceive (void (* function) (int));
```

Реєструє функцію, яка викликається, коли ведений пристрій отримує дані від майстра.

Параметри:

`function` - функція, яка викликається, коли ведений отримує дані; обробник повинен приймати один параметр - `int` (число байт, зчитаних від майстра) і нічого не повертати.

Наприклад:

```
void MyHandler (int numBytes);
```

Значень, що повертаються немає.

Метод

```
void onRequest (void (* function) (void));
```

Реєструє функцію, яка викликається, коли майстер запрошує дані з цього веденого пристрою.

Параметри:

`function` - функція, яка буде викликатися; не має параметрів і нічого не повертає.

Наприклад:

```
void MyHandler();
```

4.2 Завдання до самостійної роботи

4.2.1 Вивчити бібліотеку SPI [15], необхідну для обміну даних між SCP1000 та Arduino за допомогою шини SPI.

4.2.2 Підключити датчик SCP1000 згідно рис. 4.5.

4.2.3 Написати на мові C++ програму для відображення даних атмосферного тиску та температури від датчика на Serial Monitor з використанням бібліотеки SPI. Одиниці вимірювання атмосферного тиску – Паскалі, а температури – градуси Цельсія.

4.2.4 Вивчити бібліотеку Wire (п.4.1.4), необхідну для роботи з шиною I2C.

4.2.5 Написати програму роботи з годинником DS3231 по шині I2C з використанням бібліотеки Wire. Програма повинна забезпечувати установку часу (секунди, хвилини, години) і дати (день тижня, місяць, рік), а також зчитування даних про поточний час і дату.

Для визначення адреси мікросхеми DS3231SN на шині I2C необхідно запустити сканер пристроїв [16]. Більш докладну інформацію щодо годинника дивіться в datasheet DS3231 [17].

4.3 Контрольні питання

4.3.1. Які основні компоненти платформи Arduino?

4.3.2. На чому базується інтегроване середовище розробки Arduino?

4.3.3. Принцип роботи ультразвукового далекоміру (PING)).

4.3.4. Що таке інтерфейс SPI? Особливості його використання.

4.3.5. Що таке інтерфейс I2C? Особливості його використання.

4.3.6 Яке призначення бібліотеки Wire?

4.3.7. Назвіть основні методи бібліотеки Wire.

ЛІТЕРАТУРА

Основна література

1. Гудмэн Дж. Управление памятью для всех. – К.: Диалектика, 1995. – 528 с.
2. Гудмэн Дж. Секреты жесткого диска. – К.: Диалектика, 1994. – 247с.
3. Брамм П., Брамм Д. Микропроцессор 80386 и его программирование. –М.: Мир, 1990. –346 с.
4. Юров В. Assembler.– СПб.: Питер, 2000. – 624 с.
5. Велнум К. Секреты программирования в Windows 98.– М.: Диалектика, 1999. –855с.
6. Руководство программиста по Microsoft Windows 98.– М.: Русская редакция, 1997.– 600с.
7. Столлингс В. Операционные системы. Внутреннее устройство и принципы проектирования / В.Столлингс. М.: Вільямс, 2002. –346 с.
8. Э.Немет, Г. Снайдер. UNIX: руководство системного администратора.– К.: BHV,1996.–832с.
9. Р. Петерсон. LINUX: руководство по операционной системе. – К.: BHV, 1997.–618с.

Додаткова література

10. Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino / У. Соммер. – СПб: БХВ, 2012. – 238 с.
11. Blink. [Електрон. ресурс]. – Режим доступу: <https://www.arduino.cc/en/Tutorial/Blink>.
12. Ping. [Електрон. ресурс]. – Режим доступу: <http://www.arduino.cc/en/Tutorial/Ping>
13. SCP1000 Datasheet. [Електрон. ресурс]. – Режим доступу: <http://www.alldatasheet.com/view.jsp?Searchword=Scp1000>
14. Wire Library. [Електрон. ресурс]. – Режим доступу: http://arduino.net.ua/file_archive/Arduino%20Library/Arduino%20Wire%20Library/
15. SPI library. [Електрон. ресурс]. – Режим доступу: <https://www.arduino.cc/en/Reference/SPI>
16. i2c_scanner. [Електрон. ресурс]. – Режим доступу: <http://playground.arduino.cc/Main/I2cScanner>

17. DS3231SN (Dallas). [Электрон. ресурс]. – Режим доступа: <http://www.allcomponents.ru/dallas/ds3231sn.htm>.