Міністерство освіти і науки України

Національний університет водного господарства та природокористування

Кафедра автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій

04-03-251

МЕТОДИЧНІ ВКАЗІВКИ

до практичних занять
з навчальної дисципліни
«Розробка пристроїв на базі Arduino»
для здобувачів вищої освіти першого (бакалаврського) рівня
усіх освітньо-професійних програм спеціальностей НУВГП
денної форми навчання

Схвалено науково-методичною
радою НУВГП
Протокол № 5 від 25.09.2019 р.

Рівне – 2019

Методичні вказівки до практичних занять з навчальної дисципліни «Розробка пристроїв на базі Arduino» для здобувачів вищої освіти першого (бакалаврського) рівня усіх освітньо-професійних програм спеціальностей НУВГП денної форми навчання [Електронне видання] / Реут Д. Т. – Рівне : НУВГП, 2019. – 24 с.

Укладач: Реут Д. Т. – старший викладач кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій;

Відповідальний за випуск – Древецький В. В., д.т.н., професор, завідувач кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

Вчений секретар
науково-методичної ради Костюкова Т. А.

Зміст

Вступ

Практичні роботи з навчальної дисципліни «Розробка пристроїв на базі Arduino» дають змогу засвоїти принцип дії пристроїв на базі мікроконтролерів, можливості мікроконтролерних плат Arduino й навчитись їх використовувати при розробці прототипів нових пристроїв, а також отримати навички вибору компонентів і програмування пристроїв на базі цих плат.

Практична робота 1. Опитування кнопок пристрою та дискретне керування зовнішнім навантаженням

Мета роботи: ознайомитися з апаратною та програмною частинами середовища Arduino. Навчитись опитувати зовнішні дискретні вхідні сигнали та формувати дискретні вихідні сигнали.

Теоретичні відомості

Arduino – це відкрита електронна платформа, що базується на простому у використанні апаратному та програмному забезпеченні.

Апаратна частина включає мікроконтролерну плату, а програмна – середовище Arduino IDE. Платформа Arduino є відкритою, тому існує ряд аналогів Arduino-сумісних мікроконтролерних плат різного виконання.

Arduino Uno

Arduino Uno – це пристрій на основі мікроконтролера ATmega328P. До його складу 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 достатньо просто подати живлення від AC/DC-адаптера або батарейки, або підключити його до комп'ютера за допомогою USB-кабелю.

Входи і виходи

Кожен з 14 цифрових виводів може працювати в якості входу або виходу. Рівень напруги на виводах обмежений 5В.

Максимальний струм, який може віддавати або споживати один вивід, становить 40 мА. Усі виводи з'єднані з внутрішніми підтягуючими резисторами (за замовчуванням відключеними) номіналом 20-50 кОм. Крім цього, деякі виводи Arduino Uno 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 ATmega8U2, що 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 виводах.

ШІМ: виводи 3, 5, 6, 9, 10 і 11. За допомогою функції analogWrite() можуть виводити 8-бітові аналогові значення у 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: 13. Вбудований світлодіод, приєднаний до виводу 13.

В Arduino Uno є 6 аналогових входів (A0 - A5), на кожен з яких можна подати напругу в межах 0 – 5В та отримати аналогово-цифрове перетворення вхідного сигналу у вигляді 10-бітного числа (1024 різних значення). Верхню межу вхідної напруги можна змінити, використовуючи вивід AREF і функцію analogReference().

Таблиця 1.1

Основні характеристики Arduino Uno

Мікроконтролер	ATmega328
Робоча напруга	5V
Вхідна напруга (рекомендована)	7-12V
Дискретні (цифрові) входи/виходи	14 (6 із них ШІМ-виходи)
Аналогові входи	6
Максимальний струм одного вивода	40 mA
Максимальний струм вивода 3,3В	50 mA
Флеш-пам'ять	32 KB
SRAM	2 KB

EEPROM	1 KB
Тактова частота	16 MHz

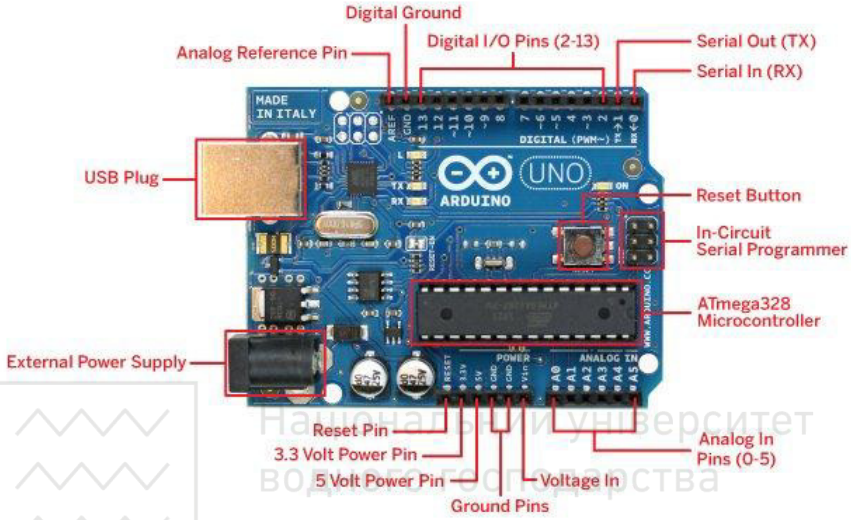

Рис. 1. Зовнішній вигляд та призначення виводів плати Arduino Uno

Програмування Arduino

Програмування мікроконтролерних плат Arduino здійснюється у вільно розповсюджуваному середовищі Arduino IDE.

Мікроконтролери в платах Arduino випускаються з прошитим завантажувачем (bootloader), що дозволяє завантажувати в мікроконтролер нові програми без необхідності використання зовнішнього програматора. Взаємодія з ним здійснюється за оригінальним протоколом STK500.

Також мікроконтролер можна прошити і через роз'єм для внутрішньосхемного програмування ICSP (In-Circuit Serial Programming), не звертаючи уваги на завантажувач.

Мова програмування Arduino – це видозмінена мова C/C++ із розширеним набором функцій.

Кожна програма для пристроїв Arduino містить дві основні функції:

1) void setup() – функція, що виконується о дин раз, після кожної подачі живлення або скидання плати Arduino;

2) void loop() – виконується постійно в циклі після виконання функції void setup().

Перед завантаженням програми в мікроконтролер слід спочатку вибрати тип плати Arduino та порт, до якого вона підключена.

Деякі функції для роботи з входам/виходами Arduino

pinMode(pin, value) – налаштування виводу з номером pin як дискретного входу (value=INPUT) або дискретного виходу (value=OUTPUT). Наприклад: pinMode(13, OUTPUT).

digitalRead(pin) – функція зчитує із заданого входу значення HIGH або LOW.

digitalWrite(pin, value) – подає на цифровий вхід/вихід значення HIGH або LOW;

analogRead(pin)– зчитує величину напруги з аналогового входу і видає результат у вигляді цілого числа від 0 до 1023.

Завдання до виконання практичної роботи

1. Підключити за допомогою макетної плати та провідників механічні кнопки до контактів 10 та 11. Аналогічно до прикладу “Button” в середовищі Arduino IDE написати програму, що запалюватиме світлодіод на платі при натисканні однієї кнопки й гаситиме при натисканні іншої кнопки. Вивантажити програму в мікроконтролер на платі та перевірити її роботу. Використати вбудовані підтягуючі резистори (INPUT_PULLUP).

2. Підключити реле до 13-го контакту плати. Перевірити замикання його контактів при після натискання однієї кнопки та розмикання при натисканні іншої.

3. Скласти схему та запрограмувати плату так, щоб при натисканні однієї кнопки вмикалось реле та утримувалось у ввімкнутому стані 25 с, після чого вимикалось.

Практична робота 2. Зчитування сигналів з аналогових датчиків фізичних величин та виведення даних на світлодіодні індикатори

Мета роботи: навчитися програмувати плату Arduino Uno для зчитування сигналу з давачів і виведення індикатори.

Теоретичні відомості

Для вимірювання фізичних величин використовують датчики — пристрої, які перетворюють вимірювану величину в вихідний сигнал, параметри якого залежать від виміряної величини. Наприклад, напруга на виході інтегрального датчика температури залежить від температури, тому можна дізнатись температуру, вимірюючи напругу з такого типу датчика. Для вимірювання напруги використовується аналогово-цифровий перетворювач.

Аналогово-цифровий перетворювач (АЦП, англ. Analog-todigital conventer, ADC) - пристрій, що перетворює вхідний аналоговий сигнал в цифровий код (цифровий сигнал).

У платі Arduino Uno в мікроконтролері ATmega328P є 10-розрядний АЦП. Вхідна напруга конвертується в 10-розрядне двійкове значення. Мінімальне значення відповідає 0, а максимальне - опорній напрузі. Отриманий результат перетворення записується в 2 регістра: ADCH і ADCL та повертається функцією analogRead(pin) як число в діапазоні 0…1023. Результат перетворення можна розрахувати за формулою

$$ADC = \frac{V_{in} \cdot 1024}{V_{ref}}$$

Опорна напруга V_{ref} в Arduino за замовчуванням рівна напрузі живлення, проте її джерело можна змінити функцією analogReference().

Внаслідок перехідних процесів швидкість АЦП є обмеженою.

Завдання до виконання практичної роботи

1. Підключити підстроювальний або змінний резистор за схемою подільника напруги (потенціометра). Струм від позитивного полюса живлення протікатиме послідовно через 2 опори до контакту GND, тому зміна одного з опорів призведе до зміни спаду напруги на ньому, що можна виміряти АЦП.

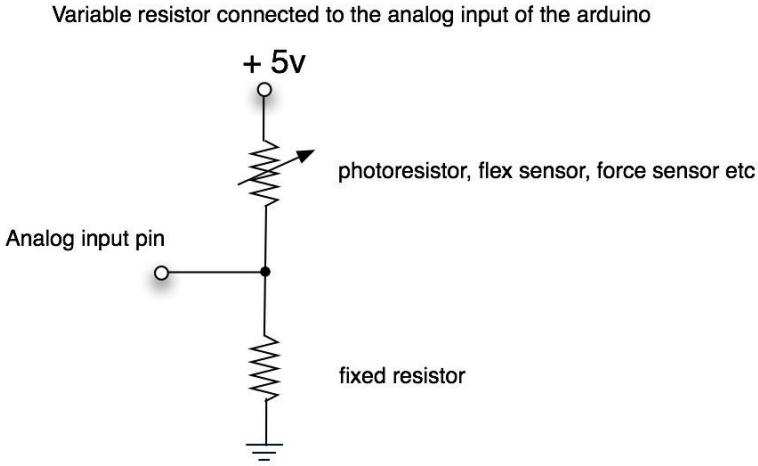

Рис. 2. Включення фоторезистора за схемою подільника напруги

2. Відповідно до напруги на ковзному контакті (0...5 В) передавати у послідовний інтерфейс значення змінної *norm*, яка зберігає число від 0 до 100, яке прямопропорційно залежить від напруги.

```
int sensorPin = A0;    // аналоговий вхід
int sensorValue = 0;  // значення сигналу від давача
    int norm=0; // нормоване значення сигналу з давача
void setup() {
Serial.begin(9600);    // налаштовуємо послідовний порт для зв'язку з ПК
}
void loop() {
  sensorValue = analogRead(sensorPin);        // зчитуємо значення з давача
   norm=map(sensorValue, 0, 1023, 0, 100); // нормуємо сигнал в межах від 0 до 100
   Serial.println(norm);    // виводимо нормоване значення сигналу в порт
    delay(500);  // затримка 0,5 с
  }
```

3. Замість змінного резистора підключити до аналогового входу подільник, утворений постійним резистором й давачем освітленості (фоторезистором). Змінюючи освітленість фоторезистора, спостерігати за значенням змінної norm.

4. Вивести на світлодіодний дисплей значення змінної norm за допомогою бібліотеки SevenSeg. При використанні плати розширення кафедри АЕКІТ потрібно вказати контакти 5, 7, A2, A4, A5, 6, 4 для керування сегментами та 9, 2, 8 для керування розрядами.

Практична робота 3. Формування сигналів на сервоприводи. Керування роботизованим маніпулятором

Мета роботи: навчитися використовувати сервоприводи окремо та в складі роботизованого маніпулятора.

Теоретичні відомості

Під сервоприводом найчастіше розуміють механізм з електродвигуном, який можна попросити повернутися в заданий кут і утримувати це положення. Однак, це не зовсім повне визначення. Якщо сказати повніше, сервопривід - це привід з управлінням через негативний зворотний зв'язок, що дозволяє точно керувати параметрами руху.

Сервоприводом є будь-який тип механічного приводу, що має в складі давач (положення, швидкості, зусилля тощо) і блок керування приводом, який автоматично підтримує необхідні параметри згідно заданому завданню. Тобто сервопривод отримує на вхід значення керуючого параметра, наприклад, кут повороту. Блок управління порівнює це значення зі значенням на своєму датчику. На основі результату порівняння привод виконує певнку дію, наприклад: поворот, прискорення або сповільнення так, щоб значення з внутрішнього давача стало якомога ближче до значення зовнішнього керуючого параметра. Найбільш поширені сервоприводи, які утримують заданий кут і сервоприводи, що підтримують задану швидкість обертання. Отже, сервопривод – це регульований редукторний електродвигун. Він зазвичай складається (рис. 3) з приводного механізму з двигуном постійного струму, плати управління і потенціометра, котрий забезпечує зворотний зв'язок.

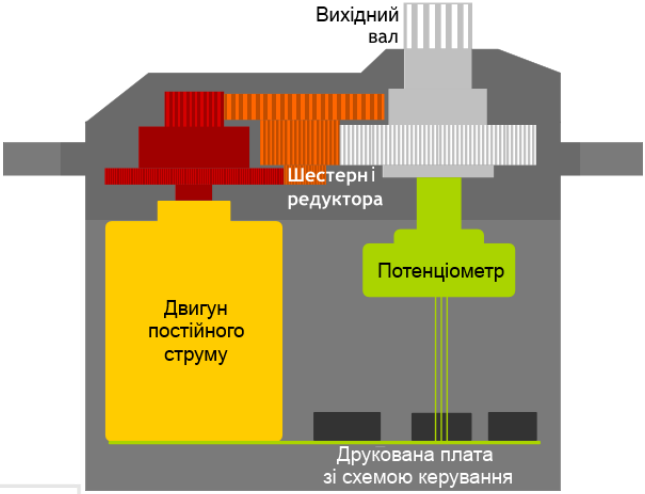

Рис. 3. Будова сервопривода

Управління сервоприводом. Інтерфейс керуючих сигналів

Керуючий сигнал для сервопривода – імпульси постійної частоти і змінної ширини. Те, яке положення повинен зайняти сервопривод, залежить від довжини імпульсів. Коли сигнал надходить в керуючу схему, наявний у ній генератор імпульсів виробляє свій імпульс, тривалість якого визначається через потенціометр. Інша частина схеми порівнює тривалість двох імпульсів. Якщо тривалість різна, включається електродвигун. Напрямок обертання визначається тим, який з імпульсів коротший. Якщо довжини імпульсів рівні, електродвигун зупиняється. Найчастіше в аматорських сервоприводів імпульси виробляються з частотою 50 Гц (рис. 4). Це означає, що імпульс випускається приймається раз в 20 мс. Зазвичай при цьому тривалість імпульсу 1520 мкс означає, що сервопривод повинен зайняти середнє положення. Збільшення або зменшення довжини імпульсу змусить сервопривод повернутися за годинниковою або проти годинникової стрілки відповідно. При цьому існують верхня і нижня межі тривалості імпульсу.

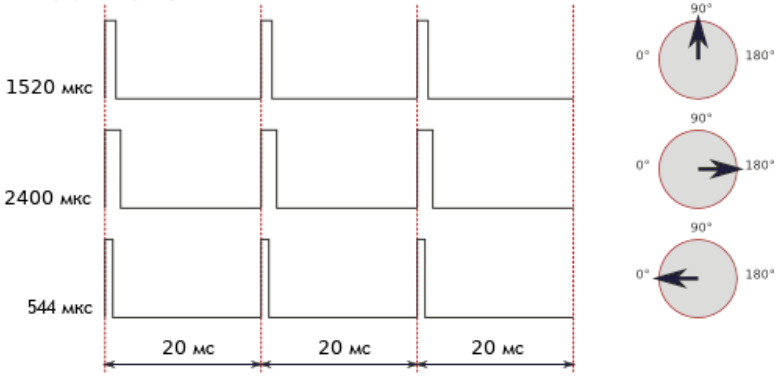

Рис. 4. Залежність кута повороту сервопривода від тривалості керуючих імпульсів

У бібліотеці Servo для Arduino за замовчуванням виставлені наступні значення довжин імпульсу: 544 мкс – для 0° і 2400 мкс – для 180°. На конкретному пристрої заводські налаштування можуть відрізнятися від стандартних. Також варто відзначити, що це всього лише загальноприйняті довжини. Навіть у рамках однієї і тієї ж моделі сервоприводу може існувати похибка, що допускається при виробництві, яка призводить до того, що робочий діапазон довжин імпульсів трохи відрізняється. Для точної роботи кожен конкретний сервопривод повинен бути відкалібрований: шляхом експериментів необхідно підібрати коректний діапазон, характерний саме для нього. При формуванні сигналу керування для сервопривода важливою є довжина імпульсів, а не частота їх появи. 50 Гц – це норма, але сервопривод буде працювати коректно і при 40, і при 60 Гц.

Сервоприводи малої потужності (наприклад, SG90) можна безпосередньо підключати до плати Arduino (рис. 5), для підключення більш потужних – слід використовувати зовнішнє джерело живлення. Генерувати ШІМ-сигнал на платі Arduino Uno можуть виводи, позначені ‘~’ або ‘PWM’.

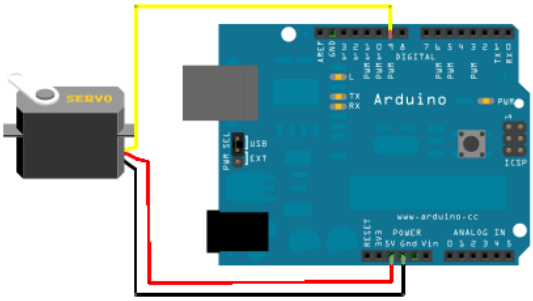

Рис. 5. Підключення сервопривода малої потужності до мікроконтролерної плати Arduino Uno

Бібліотека Servo дозволяє здійснювати програмне керування сервоприводами. Управління здійснюється наступними функціями:

attach() – приєднує об‘єкт до конкретного виводу плати. Можливі два варіанти синтаксису для цієї функції: servo.attach(pin) і servo.attach (pin, min, max). При цьому pin - номер виводу, до якого приєднують сервопривод, min і max - довжини імпульсів в мікросекундах, що відповідають за кути повороту 0° і 180°. За замовчуванням виставляються рівними 544 мкс і 2400 мкс відповідно.

write() - віддає команду сервоприводу прийняти деяке значення параметра. Синтаксис наступний:

servo.write (angle), де angle - кут, на який повинен повернутися сервопривод.

writeMicroseconds() - віддає команду надіслати на сервопрівод імульс певної довжини, є низькорівневим аналогом попередньої команди. Синтаксис наступний:

servo.writeMicroseconds(uS), де uS – довжина імпульсу в мікросекундах.

read() - читає поточне значення кута, в якому знаходиться сервопривод. Синтаксис наступний: servo.read(), повертається ціле значення від 0 до 180.

attached() - перевірка, чи був приєднаний об‘єкт до конкретного виводу. Синтаксис наступний: servo.attached(),

повертається true, якщо об‘єкт був приєднаний до якого-небудь виводу, або false в зворотному випадку.

detach() - виконує дію, зворотну дії attach(), тобто від'єднує об‘єкт від виводу, до якого він був приписаний. Синтаксис наступний: servo.detach().

Приклад програми з використанням бібліотеки Servo

```
#include <Servo.h>
Servo myservo;
void setup()
{
        myservo.attach(9); // сигнал керування на сервопривод генеруємо з контакту 9
}
void loop()
{
        myservo.write(90);// встановлюємо сервопривод в середнє положення
        delay(500);
        myservo.write(0); // встановлюємо сервопривод в крайнє ліве положення
        delay(500);
        myservo.write(180);// встановлюємо сервопривод в крайнє праве положення
        delay(500);
}
```

Опис лабораторної установки

Використовуваний маніпулятор, зображений на рис. 6, має 6 ступенів свободи. Для приведення в рух ланок маніпулятора в кожному із суглобів встановлено сервопривод MG996R. Перший ступінь рухомості забезпечує основа маніпулятора, другий – плече, третій – лікоть, четвертий – обертання кисті, п’ятий – поворот кисті, шостий – захват.

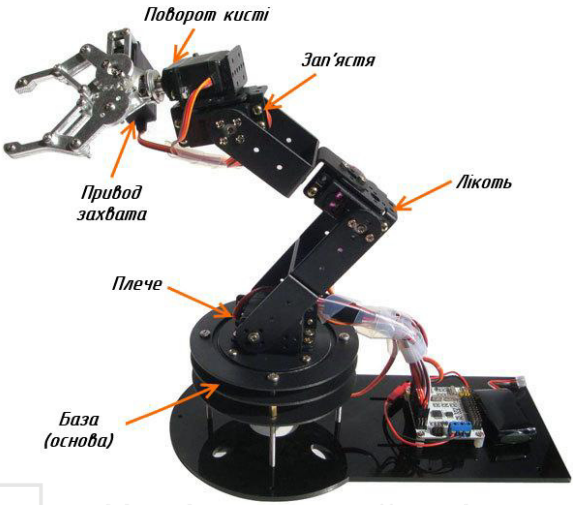

Рис. 6. Зовнішній вигляд досліджуваного маніпулятора

Параметри сервоприводів MG996R подані нижче (рис. 7).

Рис. 7. Зовнішній вигляд та розміри сервопривода MG996R

Основні характеристики:

З'єднувальний провід: довжина 300мм.

Розміри: 40.7 мм x 19,7 мм x 42.9 мм.

Маса: 55 г

Робоча швидкість: 0.17 с / 60 градусів (4.8В без навант.)

Робоча швидкість: 0.14 с / 60 градусів (6.0В без навант.)

Обертальний момент: 9.4 кгс · см (4.8В), 11 кгс · см (6 В)

Джерело живлення: через зовнішній адаптер.

Робоча напруга: 4,8 – 7,2В

Редуктор : металеві шестерні.

Робочий струм при русі: 500 – 900 мА (6 В).

Струм утримання: 2.5 А (6 В).

Підключення: оранжевий — сигнал керування, червоний — +5В, чорний — GND.

Завдання до виконання практичної роботи

1. Підключити сервопривод SG-90 до контакту 9 плати Arduino та вивантажити в плату приклад програми з методичних вказівок. Перевірити роботу програми.

2. Використовучи бібліотеку VarSpeedServo, запрограмувати плату Arduino для керування двома сервоприводами роботизованого маніпулятора — сервоприводом основи та повороту кисті. В програмі використовувати наступні тривалості імпульсів керування сервоприводами:

myservo1.attach(2,900,2365);
myservo2.attach(3,720,1000);
myservo3.attach(4,720,2365);
myservo4.attach(5,720,2300);
myservo5.attach(10,670,2300);
myservo6.attach(11,720,2300);

Практична робота 4. Розробка системи контролю доступу з використанням ключів iButton та RFID Mifare Classic

Мета роботи: ознайомитися з принципом роботи систем контролю доступу та навчитись використовувати ключі iButton або Mifare Classic з платою Arduino.

Теоретичні відомості

Зчитування контактних ключів сімейства DS1990a

платою Arduino дозволяє реалізовувати системи контролю доступу на її основі. Кожен електронний ключ (контактний або безконтактний) містить ідентифікатор, який може бути в переліку дозволених для доступу ідентифікаторів. В такому випадку особа, що володіє даним ключем, матиме доступ до приміщення, яке захищається системою контролю доступу. В протилежному випадку особа не зможе потрапити до нього.

Як правило, в таких системах використовують соленоїдні засувки, що при подачі на обмотку напруги живлення надають або закривають можливість відкрити двері. Для зв'язку Arduino з соленоїдним замком використовують реле або транзистори.

Завдання до виконання практичної роботи

1. Завантажити бібліотеку OneWire та встановити її за допомогою менеджера бібліотек. Вивантажити код нижче в плату Arduino Uno.

```
OneWire  ds(10);  //ключ підключати через резистор до 10-го цифрового контакту

void setup(void) {
  Serial.begin(9600);
}

void loop(void) {
  byte i;
  byte present = 0;
  byte data[12];
  byte addr[8];
  if ( !ds.search(addr)) {
     Serial.print("No more addresses.\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");
        return;
    }
    if ( addr[0] != 0x01) {
        Serial.print("Device is not a DS1990A family device.\n");
        return;
    }
    Serial.println();
    ds.reset();
    delay(1000);
}
```

2. Підключити до плати Arduino Uno ключ наступним чином: боковий циліндричний контакт до GND, а центральний плоский контакт — до 10-го цифрового контакту плати, який потрібно підтягнути до напруги живлення резистором 2,2 кОм. Відкрити монітор порту й перевірити співпадіння зчитуваних даних з нанесеним на ключ ідентифікатором.

3. Завантажити бібліотеку для роботи з RFID-модулем за посиланням https://github.com/miguelbalboa/rfid. Підключити завантажений ZIP-архів у середовище Arduino IDE в меню Скетч-Додати бібліотеку-Додати ZIP-бібліотеку. Після підключення бібліотеки в прикладах знайти DumpInfo та вивантажити в плату Arduino Uno.

4. Підключити до Arduino RFID-модуль на базі MFRC522 так, як показано на рис. 8. Для підключення RFID-модуля MFRC522 може використовуватись інтерфейс I^2C або SPI. В цій схемі використовується 4-провідний інтерфейс SPI та окремий провідник для сигналу скидання Reset (RST).

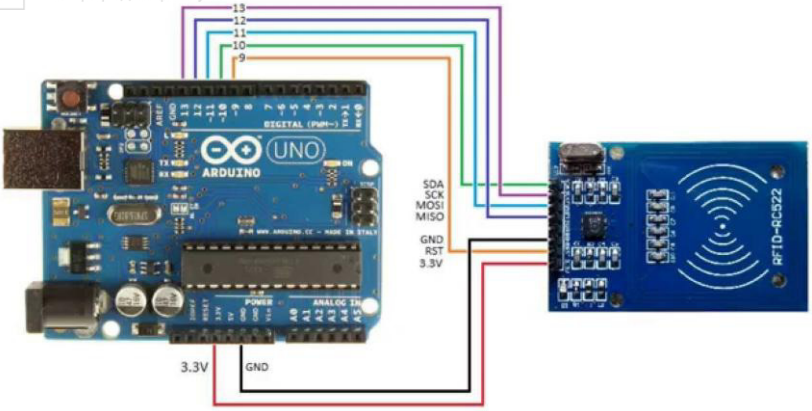

Рис. 8. Підключення модуля MFRC522 до Arduino Uno

5. Зчитати RFID-мітку (картку або брелок), спостерігаючи виведений вміст картки у моніторі порту. Знайти UID мітки.

6. Розробити програму, що зчитуватиме UID та порівнюватиме його із заданим. Якщо ідентифікатори не співпадають — запалювати червоний світлодіод (“доступ заборонено”), а якщо співпадають — зелений (“доступ дозволено”).

Практична робота 5. Обмін даними за допомогою інтерфейсу I2C. Визначення положення пристрою в просторі за допомогою акселерометра й гіроскопа.

Мета роботи: навчитись працювати з модулями та мікросхемами, що обмінюються даними через інтерфейс I2C. Навчитись використовувати мікросхему акселерометра-гіроскопа MPU-6050 для визначення положення та прискорення об’єктів.

Теоретичні відомості

Набільшого поширення сьогодні набули акселерометри та гіроскопи, виконані за MEMS технологією.

MEMS-датчики широко застосовуються в смартфонах, автомобільній промисловості для управління подушками безпеки, і в охоронної сигналізації, в навігаційних системах для

обчислення пройденого шляху або визначення маршруту проходження.

Найчастіше для побудови таких сенсорів використовують вимірювання зміщення інерційної маси всередині під дією зовнішніх прискорень. Вимірюваною величиною є напруга внаслідок п'єзоелектричного ефекту або зміна ємності конденсаторів.

Модуль акселерометра-гіроскопа GY-521

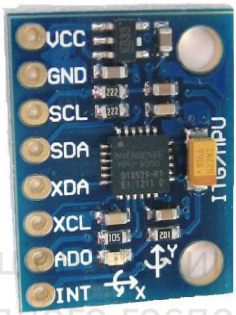

Рис. 9. Зовнішній вигляд модуля акселерометра-гіроскопа GY-521

Модуль GY-521 (рис. 9) на мікросхемі MPU6050 - це 3-осьовий гіроскоп і акселерометр на три координати. На платі модуля GY-521 розміщені всі необхідні для її надійного функціонування елементи, в тому числі і підтягуючі резистори. Обмін даними з контролером здійснюється по шині I^2C.

Необхідну напругу живлення для модуля MPU6050 регулює вбудований стабіллізатор напруги 3,3V.

Серцем модуля GY-521 є мікросхема MPU-6050

Характеристики його наступні:

- Напруга живлення: 3-5 В
- Зв'язок: I^2C протокол
- Діапазон вимірювань гіроскопа: ± 250, 500, 1000, 2000 °/с
- Діапазон вимірювань акселерометра: ± 2, 4, 8, 16 g
- Крок між контактами: 2.54 мм
- Вбудований 16-бітний АЦП

- Розмір плати: 20 мм х 16 мм

Приклад програми, що виводить через послідовний інтерфейс значення прискорення

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

void setup()
{
  Serial.begin(115200);
  Serial.println("Initialize MPU6050");
  while(!mpu.begin(MPU6050_SCALE_2000DPS,
MPU6050_RANGE_2G))
  {
    Serial.println("Could not find a valid MPU6050 sensor, check
wiring!");
    delay(500);
  }

  // If you want, you can set accelerometer offsets
  // mpu.setAccelOffsetX();
  // mpu.setAccelOffsetY();
  // mpu.setAccelOffsetZ();

  checkSettings();
}

void checkSettings()
{
  Serial.println();
    Serial.print(" * Sleep Mode:            ");
  Serial.println(mpu.getSleepEnabled() ? "Enabled" : "Disabled");
   Serial.print(" * Clock Source:          ");
  switch(mpu.getClockSource())
  {
    case MPU6050_CLOCK_KEEP_RESET:
Serial.println("Stops the clock and keeps the timing generator in reset");
break;
    case MPU6050_CLOCK_EXTERNAL_19MHZ:
Serial.println("PLL with external 19.2MHz reference"); break;
```

```
            case MPU6050_CLOCK_EXTERNAL_32KHZ:
Serial.println("PLL with external 32.768kHz reference"); break;
            case MPU6050_CLOCK_PLL_ZGYRO:      Serial.println("PLL
with Z axis gyroscope reference"); break;
            case MPU6050_CLOCK_PLL_YGYRO:      Serial.println("PLL
with Y axis gyroscope reference"); break;
            case MPU6050_CLOCK_PLL_XGYRO:      Serial.println("PLL
with X axis gyroscope reference"); break;
            case MPU6050_CLOCK_INTERNAL_8MHZ:
Serial.println("Internal 8MHz oscillator"); break;
          }

          Serial.print(" * Accelerometer:         ");
          switch(mpu.getRange())
          {
            case MPU6050_RANGE_16G:          Serial.println("+/- 16 g");
break;
            case MPU6050_RANGE_8G:           Serial.println("+/- 8 g");
break;
            case MPU6050_RANGE_4G:           Serial.println("+/- 4 g");
break;
            case MPU6050_RANGE_2G:           Serial.println("+/- 2 g");
break;
          }

          Serial.print(" * Accelerometer offsets: ");
          Serial.print(mpu.getAccelOffsetX());
          Serial.print(" / ");
          Serial.print(mpu.getAccelOffsetY());
          Serial.print(" / ");
          Serial.println(mpu.getAccelOffsetZ());

          Serial.println();
        }

        void loop()
        {
          Vector rawAccel = mpu.readRawAccel();
          Vector normAccel = mpu.readNormalizeAccel();

          Serial.print(" Xraw = ");
```

```
Serial.print(rawAccel.XAxis);
Serial.print(" Yraw = ");
Serial.print(rawAccel.YAxis);
Serial.print(" Zraw = ");
Serial.println(rawAccel.ZAxis);
Serial.print(" Xnorm = ");
Serial.print(normAccel.XAxis);
Serial.print(" Ynorm = ");
Serial.print(normAccel.YAxis);
Serial.print(" Znorm = ");
Serial.println(normAccel.ZAxis);
delay(200);}
```

Завдання до виконання практичної роботи

1. Підключити в середовищі Arduino IDE бібліотеку MPU6050 (*https://github.com/jarzebski/Arduino-MPU6050*). Скласти програму для визначення прискорення об'єкта, на якому закріплений модуль з мікросхемою MPU-6050.

2. Скласти програму для визначення положення об'єкта, на якому закріплений модуль з мікросхемою MPU-6050. Перевірити дрейф нуля під час роботи.

Практична робота 6. Використання Wi-Fi модуля для обміну даними через мережу Інтернет

Мета роботи: Ознайомитися з АТ-командами, що використовуються для керування Wi-Fi модулем ESP8266. Навчитися створювати програми для обміну даними через інтернет та дистанційного керування.

Завдання до виконання практичної роботи

1. Ознайомитись з набором АТ-команд для модуля ESP8266, розташованих на файловому сервері кафедри AKIT/Stud_doc/2-k/Розробка пристроїв на базі Arduino/esp8266_at_instruction_set_en_v1.5.4_0.pdf або на сайті виробника https://espressif.com.

2. Підключити модуль за допомогою USB-UART перетворювача до комп'ютера та під'єднатись до якоїсь з мереж Wi-Fi, використовуючи АТ-команди.

3. Додати в налаштуваннях Arduino IDE новий URL менеджера додаткових плат https://arduino.esp8266.com/stable/package_esp8266com_index.json . Відкрити менеджер плат та встановити нову платформу — ESP8266. Встановити бібліотеку PubSubClient за допомогою менеджера бібліотек.

4. Використовуючи приклади, розробити програму, що видаватиме на цифровий вихід логічну 1 при отриманні в топіку *relay* повідомлення “on” по протоколу MQTT, та логічний 0 при отриманні повідомлення “off”. В якості сервера використати Cute Cat від https://www.cloudmqtt.com, а клієнта - IoT MQTT Dashboard з Google Play.

Список рекомендованої літератури

1. Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino. Санкт-Петербург : БХВ-Петербург, 2012. 256 с.

2. Margolis Michael. Arduino Cookbook. O'Reilly Media, 2011. 662 p.

3. Evans B. Arduino programming notebook. First edition. 2007. 38 p. URL: https://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf.