

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ТЕХНОЛОГІЇ ІНТЕРНЕТУ РЕЧЕЙ В ЕЛЕКТРОНІЦІ КОМП'ЮТЕРНИЙ ПРАКТИКУМ

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для студентів,
які навчаються за спеціальністю 171 «Електроніка»,
спеціалізацією «Електронні компоненти і системи»*

Київ
КПІ ім. Ігоря Сікорського
2018

Технології інтернету речей в електроніці: Комп'ютерний практикум [Електронний ресурс]: навч. посіб. для студ. спеціальності 171 «Електроніка», спеціалізації «Електронні компоненти і системи» / КПІ ім. Ігоря Сікорського; уклад.: Ю. С. Ямненко, Ю. В. Хохлов. – Електронні текстові данні (1 файл: X,XX Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 76 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 9 від 24.05.2018 р.) за поданням Вченої ради факультету електроніки (протокол № 05/2018 від 21.05.2018 р.)

Електронне мережне навчальне видання

ТЕХНОЛОГІЇ ІНТЕРНЕТУ РЕЧЕЙ В ЕЛЕКТРОНІЦІ КОМП'ЮТЕРНИЙ ПРАКТИКУМ

Укладачі: *Ямненко Юлія Сергіївна, д-р техн. наук, проф.*
Хохлов Юрій Віталійович, канд. техн. наук, доц.

Відповідальний редактор *Терещенко Т.О., професор кафедри промислової електроніки, д.т.н., проф.*

Рецензенти: *Найда С.А., заст. декана факультету електроніки, Професор кафедри акустики та акустoeлектроніки, д.т.н., проф.*

Методичне видання містить роз'яснення щодо виконання 6 лабораторних робіт, передбачених робочою навчальною програмою кредитного модуля «Технології інтернету речей в електроніці».

Кожна робота містить теоретичні відомості щодо архітектури, особливостей налаштування та програмування, завдання до роботи, зміст звіту по роботі, що вимагається від студента для успішного зарахування роботи. Для кожної роботи наводяться приклади програмної реалізації. В кінці кожної роботи пропонуються контрольні питання для самоперевірки. Для самостійної роботи студентів наводиться список рекомендованої літератури.

Для студентів, що навчаються за освітньо-науковою магістерською програмою «Електронні пристрої та системи» спеціальності 171 Електроніка денної форми навчання.

© КПІ ім. Ігоря Сікорського, 2018

Зміст

Лабораторна робота №1. Знайомство із Arduino. Принципи роботи зі світлодіодами	4
Лабораторна робота №2. Простий пристрій моніторингу температури.....	13
Лабораторна робота №3. Послідовний інтерфейс, РК-дисплей, інтерфейс 1-wire	28
Лабораторна робота №4. Простий радіо-зв'язок точка-точка	35
Лабораторна робота №5. Однокристальний мікроконтролер ESP8266 з WiFi та протокол MQTT	44
Лабораторна робота №6. Робота з OpenHab та підключення MQTT broker	57

Лабораторна робота №1.

Знайомство із Arduino. Принципи роботи зі світлодіодами

Мета роботи: ознайомитись з платою Arduino. Оволодіти основними принципами роботи та підключення світлодіодів до плати, ознайомитись з різновидами функцій та команд для створення прототипу світлофора.

1. Теоретичні відомості

1.1 Arduino Arduino – це електронний конструктор і зручна платформа швидкої розробки електронних пристроїв для новачків і професіоналів. Переваги платформи: зручна у використанні, простота мови програмування, а також відкрита архітектура і програмний код. Пристрій програмується через USB без використання програматорів.

Пристрої на базі Arduino можуть отримувати інформацію про навколишнє середовище за допомогою різних датчиків, а також можуть управляти різними виконавчими пристроями.

Існує декілька версій платформ Arduino. В цьому курсі буде розглядатися платформа Arduino Uno (рис. 3.1)

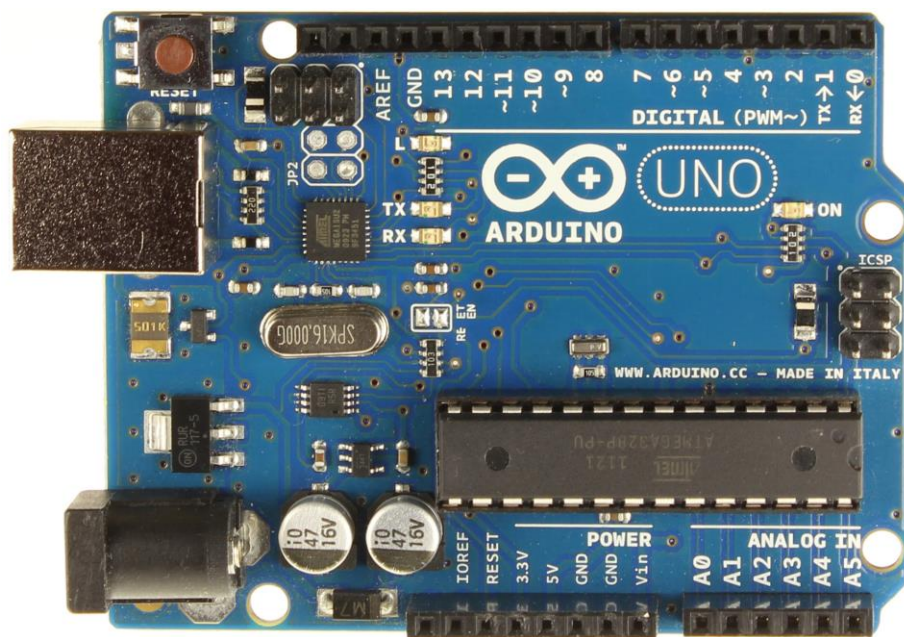


Рисунок 3.1 – Зовнішній вигляд Arduino Uno

Arduino Uno базується на мікроконтролері Atmel ATmega328. Мікроконтролер на платі програмується за допомогою мови Arduino (заснований на мові Wiring) і середовища розробки Arduino (заснована на середовищі Processing). Проекти пристроїв, засновані на Arduino, можуть працювати самостійно, або ж взаємодіяти з програмним забезпеченням на комп'ютері (напр.: Flash, Processing, MaxMSP). Плати можуть бути зібрані користувачем самостійно або куплені в зборі. Програмне забезпечення доступне для безкоштовного скачування. Вихідні креслення схем (файли САД) є загальнодоступними, користувачі можуть застосовувати їх на свій розсуд.

Платформа має 14 цифрових вхід/виходів (6 з яких можуть використовуватися як виходи ШІМ), 6 аналогових входів, кварцовий генератор 16 МГц, роз'єм USB, силовий роз'єм, роз'єм ICSP і кнопку перезавантаження. Кожен з 14 цифрових виводів Uno може налаштований як вхід або вихід, використовуючи функції `pinMode ()`, `digitalWrite ()`, і `digitalRead ()`. Кожен вивід має навантажувальний резистор (за замовчуванням відключений) 20-50 кОм і може пропускати до 40 мА. Для роботи необхідно підключити платформу до комп'ютера за допомогою кабелю USB, або подати живлення за допомогою адаптера АС/DC або батареї.

Платформа може працювати при зовнішньому живленні від 6 В до 20 В. При напрузі живлення нижче 7 В, вивід 5V 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.

Деякі виводи мають особливі функції:

- VIN. Вхід використовується для подачі живлення від зовнішнього джерела (в відсутність 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 вбудованим регулятором на платі. Максимальне споживання струму 50 мА.

- GND. Виводи заземлення.

- Послідовна шина: 0 (RX) і 1 (TX). Виводи використовуються для отримання (RX) і передачі (TX) даних TTL. Дані виводи підключені до відповідних роз'ємів мікросхеми послідовної шини ATmega8U2 USB-to-TTL.

- Зовнішнє переривання: 2 і 3 Дані висновки можуть бути налаштовані на виклик переривання або на молодшому значенні, або на передньому чи задньому фронті, або при зміні значення. Детальна інформація знаходиться в описі функції `attachInterrupt ()`.

- ШІМ: 3, 5, 6, 9, 10, і 11. Будь-який з виводів забезпечує ШІМ з роздільною здатністю 8 біт за допомогою функції `analogWrite ()`.

- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). За допомогою даних виводів здійснюється зв'язок SPI, для чого використовується бібліотека SPI.

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

На платформі Uno встановлені 6 аналогових входів (позначених як A0 .. A5), кожен роздільною здатністю 10 біт (тобто може приймати 1024 різних значення). Стандартно виводи мають діапазон вимірювання до 5 В відносно землі, проте є можливість змінити верхню межу за допомогою виводу AREF і функції `analogReference ()`.

Деякі виводи мають додаткові функції:

- I2C: 4 (SDA) і 5 (SCL). За допомогою виводів здійснюється зв'язок I2C (TWI), для створення якої використовується бібліотека `Wire`.

Додаткова пара виводів платформи:

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

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

1.2 Основні функції для роботи зі світлодіодами

Для роботи зі світлодіодом необхідно знати і вміти володіти такими функціями і константами:

- оператор `setup()`;
- оператор `loop()`;
- функція `pinMode()`
- функція `digitalWrite()`;
- функція `delay()`;
- константи `OUTPUT`, `HIGH`, `LOW`.

Далі наведений код програми найпростішого прикладу мерехтіння вбудованим у плату Arduino світлодіодом, який підключено до 13 виводу:

```
void setup()  
{ pinMode(13, OUTPUT);  
}
```

Ця функція виконується на початку роботи програми (після запуску мікроконтролеру). Тобто послідовно виконується кожна команда, яка знаходиться між фігурними скобками цієї функції. Наприкінці кожної строки необхідно поставити символ закінчення команди “;”. Тут функція `setup` містить одну єдину команду – `pinMode(13, OUTPUT)` . Ця команда налаштовує 13 порт Arduino, як вивід. Порт № 13 знаходиться на верхній колодці портів Arduino.

Після функції `setup` виконується функція `loop`.

```
void loop()  
{
```

```
digitalWrite(13, HIGH); // вмикаємо світлодіод  
delay(1000); // чекаємо секунду  
digitalWrite(13, LOW); // вимикаємо світлодіод  
delay(1000); // чекаємо секунду  
}
```

На відміну від `setup`, функція `loop` постійно повторюється – як тільки послідовно виконані всі команди в скобках, функція запускається знову. Функція `loop` для цього прикладу складається з чотирьох команд:

На порт 13 подається напруга (5 В) – світлодіод вмикається.

Затримка до виконання наступної команди 1000 мілісекунд (одну секунду)

Порт 13 з'єднується з землею – світлодіод вимикається.

Ще одна затримка на 1 секунду.

Після виконання усіх чотирьох команд, знову виконується перша команда (включення світлодіоду) и так продовжується до тих пір, поки Arduino включена або поки не буде натиснута кнопка RESET.

2. Завдання до лабораторної роботи

2.2 Підключіть світлодіоди до плати Arduino згідно зі схемою, яка зображена на рисунку 3.2.

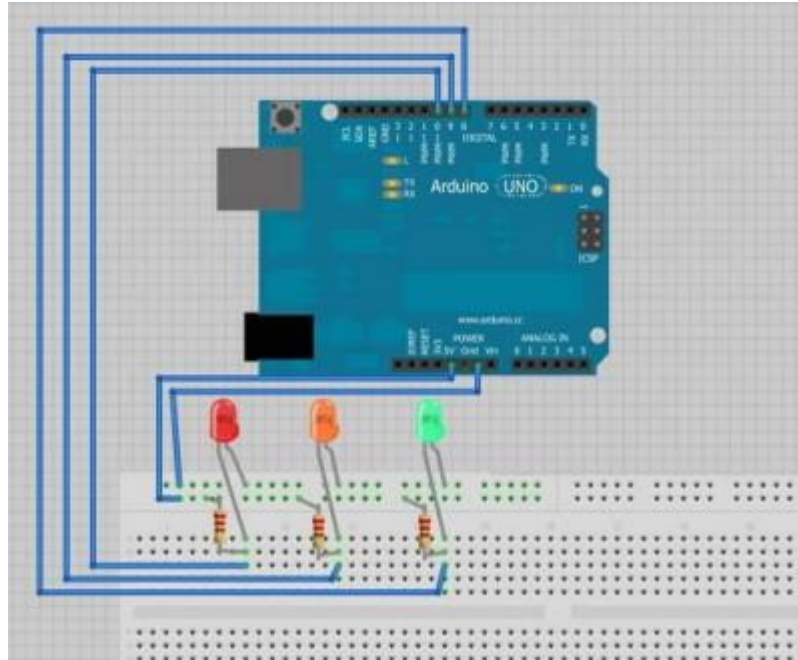


Рисунок 3.2 – Схема підключення світлодіодів

2.3 Запустіть на комп'ютері середовища розробки Arduino.

2.4 Напишіть код згідно з алгоритмом представленим на рисунку 3.3.

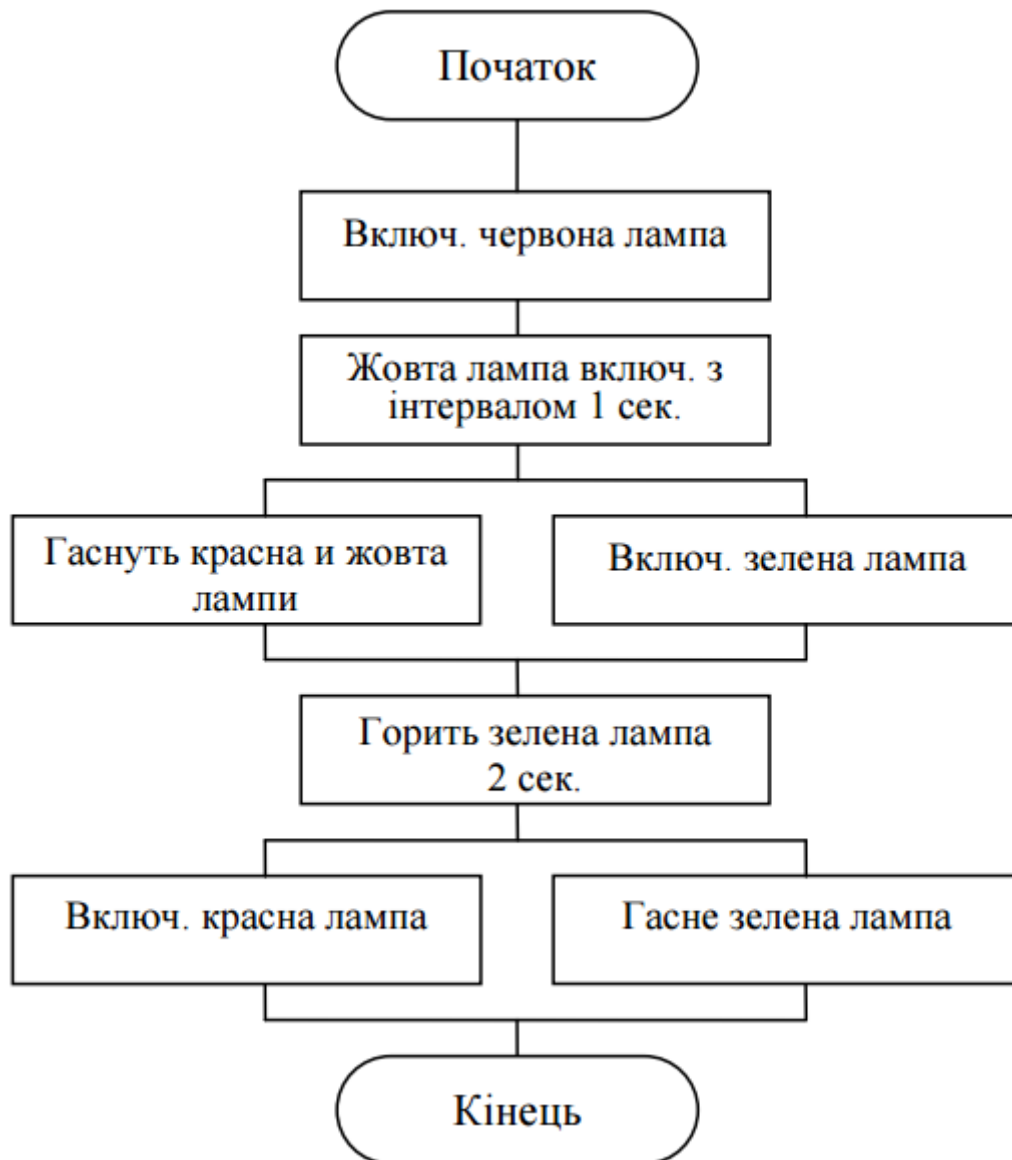


Рисунок 3.3 – Схема алгоритму роботи світлофора

2.5 Підключіть плати Arduino через USB до ПК.

2.6 Натисніть кнопку Verify и переконайтесь, що у нижній частині вікна з'явився надпис Done Compiling. Це значить, що у написаній програмі не знайдено помилок.

2.7 Виберіть у Tools->Board ваш тип плати. Перевірте, чи правильно обрано USB-порт в Tools->Serial port. Після натисніть на кнопку Upload.

2.8 Якщо внизу з'явився надпис “Done uploading” – процес запису прошов успішно.

3. Зміст звіту

- 3.1. Тема та мета роботи.
- 3.2. Схема електрична-принципова світлофору.
- 3.3. Код програми.
- 3.4. Висновки з роботи.

5. Контрольні запитання

1. Що таке Arduino?
2. Які версії платформи Arduino ви знаєте?
3. На якому мікроконтролері базується Arduino Uno?
4. Скільки цифрових вход/виходів має платформа?
5. Які функції мають виводи?
6. Назвіть основні функції та константи для роботи зі світлодіодом.
7. Чим відрізняється функція “setup” від “loop”?
8. Назвіть переваги платформи Arduino

6. Список рекомендованої літератури

1. Шегедин, О.І. Теоретичні основи електротехніки : навч.посіб. / О.І. Шегедин, В.С. Маляр.; Львів : Магнолія 2006, 2012. – 167с.
2. Коруд, В.І. Електротехніка : підручник для ВНЗ / В.І.Коруд, О.Є. Гамола.; за заг. ред. В.І. Коруда.- 3-тє вид., перероб. і доп.- Львів:Магнолія, 2007.- 447с.

3. Мілих, В.І. Електротехніка, електроніка та мікропроцесорна техніка: підручник для ВНЗ/В.І. Мілих, О.О. Шавьолкін; за ред. В.І. Мілих. - К.: Каравела, 2007. - 688 с.
4. Синдеев, Ю.Г. Электротехника с основами электроники: учебное пособие.; Ростов-на-Дону: Феникс, 2011. - 407 с.
5. Arduino. [Електронний ресурс]. – Режим доступу: <https://www.arduino.cc/>
6. Autodesk 123D Circuit. [Електронний ресурс]. – Режим доступу: <https://123d.circuits.io/>
7. Atmel Studio. [Електронний ресурс]. Microchip Atmel. – Режим доступу: <http://www.atmel.com/tools/atmelstudio.aspx>

Лабораторна робота №2. Простий пристрій моніторингу температури

Мета роботи: Оволодіти основними принципами роботи з Arduino та підключенням різних тепературних датчиків до плати, ознайомитись з різновидами функцій та команд для створення програми для зняття даних про температуру.

1. Теоретичні відомості

1.1 Послідовний інтерфейс UART

В перекладі це універсальний асинхронний приймач. Дані UART передаються послідовним кодом в наступному форматі.



Рис 1.1 Схема формату передачі даних

Кожен біт передається за рівні проміжки часу. Час передачі одного біта визначається швидкістю передачі. Швидкість передачі вказується в бодах (біт в секунду). Крім бітів даних інтерфейс UART вставляє в потік біти синхронізації: стартовий і стоповий. Таким чином, для передачі байта інформації потрібно 10 бітів, а не 8. Похибка тимчасових інтервалів передачі бітів повинна бути не більше 5% (рекомендується не більше 1,5%).

Існують варіанти з різною кількістю бітів даних, бітів синхронізації, може бути доданий біт контролю парності і т.п. Але ці формати використовуються рідко. Головне запам'ятати:

- в неактивному режимі вихід UART знаходиться в високому стані;
- передача байта починається зі стартового біта (низького рівня);
- передача байта закінчується стоповим бітом (високого рівня);
- дані передаються молодшим бітом вперед;
- для передачі байта потрібно 10 бітів;

Час передачі одного байта розраховується виходячи з швидкості передачі і кількості бітів (10).

Часто використовуються такі стандартні швидкості передачі інтерфейсу UART.

Швидкість передачі, бод	Час передачі одного біту, мкс	Час передачі байту, мкс
4800	208	2083
9600	104	1042
19200	52	521
38400	26	260
57600	17	174
115200	8,7	87

Обмін інформацією через UART відбувається в дуплексному режимі, тобто передача даних може відбуватися одночасно з прийомом. Для цього в інтерфейсі UART є два сигнали:

TX - вихід для передачі даних;

RX - вхід для прийому даних.

При з'єднанні двох UART пристроїв вихід TX одного пристрою з'єднується з входом RX іншого. А сигнал TX другого UART підключається до входу RX першого.

1.2 Протокол 1-Wire

Протокол 1-Wire - це протокол, який використовується для управління пристроями, які виробляються компанією Dallas Semiconductor (нині Maxim). Хоча 1-Wire є торговою маркою Dallas, програмісти, які використовують драйвери 1-Wire, не зобов'язані робити за це ніяких виплат.

Мережа 1 Wire, яку Даллас називає MicroLan (торгова марка), складається з одного ведучого пристрою, до якого за допомогою єдиної лінії передачі даних з'єднання з одним або декілька ведених пристроїв. Ця лінія, крім іншого, можна використовувати для електроживлення ведених пристроїв (ситуація, коли пристрої підживлюються через шину 1-Wire, називають «паразитне живлення»).

Серед інших пристроїв, що працюють через протокол 1-Wire, особливо популярні температурні датчики - вони недорогі, прості у використанні і дозволяють безпосередньо зчитувати калібровані цифрові температурні дані. Крім того, вони терпимі до довгих проводах, якими часто доводиться користуватися при створенні ланцюга з Arduino. Один із прикладів нижче демонструє, як працювати з 1-Wire на прикладі цифрового термометра DS18S20. Багато Чіпи 1 Wire можуть працювати і через паразитне, і через нормальне живлення.

1.3 USB

Універсальна послідовна шина, призначила для з'єднання периферійніе Пристрій обчислювальної техніки.

USB роз'єм типу А — найпоширеніший і найвідоміший із нинішніх. Більшість пристроїв, що підключаються через USB, мають саме його. Комп'ютерна миша, USB флеш-накопичувач, клавіатура, фото- й відеокамера та багато інших речей оснащені USB типу А, який бере свій початок ще з 90-х. Одна з найголовніших переваг цього порту — надійність. Він може пережити досить велику кількість підключень, не розвалюється й дійсно заслужив стати найпоширенішим засобом підключення всього, чого тільки можна. Однак для портативних пристроїв він не підходить, тому що має досить великі габарити, що врешті-решт призвело до появи модифікацій із меншими розмірами — Mini та Micro.

USB роз'єм типу В найчастіше використовується для приєднання до комп'ютера принтерів і сканерів, зрідка — інших пристроїв.

Розроблені також типи Mini-AB та Micro-AB для з'єднання через конектори відповідного розміру як типу А, так і типу В.

В даний час існує декілька видів роз'ємів USB, які бувають трьох версій - USB v1.1, USB v2.0 і USB v3.0. Принципова різниця між ними , тільки в швидкості передачі даних яка в USB v3.0 найшвидша 500 Мбайт/сек , USB v2.0 48 Мбайт/сек і USB v1.1 12 Мбайт/сек .

1.3 Датчик температури DS18B20

DS18B20 має різні форм-фактори. Доступно три варіанти: 8-Pin SO (150 mils), 8-Pin μ SOP, і 3-Pin TO-92. Серфінг по eBay або Aliexpress показує, що китайці пропонують версію TO-92 у вологозахищеному корпусі. Тобто, можна сміливо занурювати подібне диво в воду,

використовувати під дощем і т.д. і т.п. Ці сенсори виготовляються з трьома вихідними контактами (чорний - GND, червоний - Vdd і білий - Data).

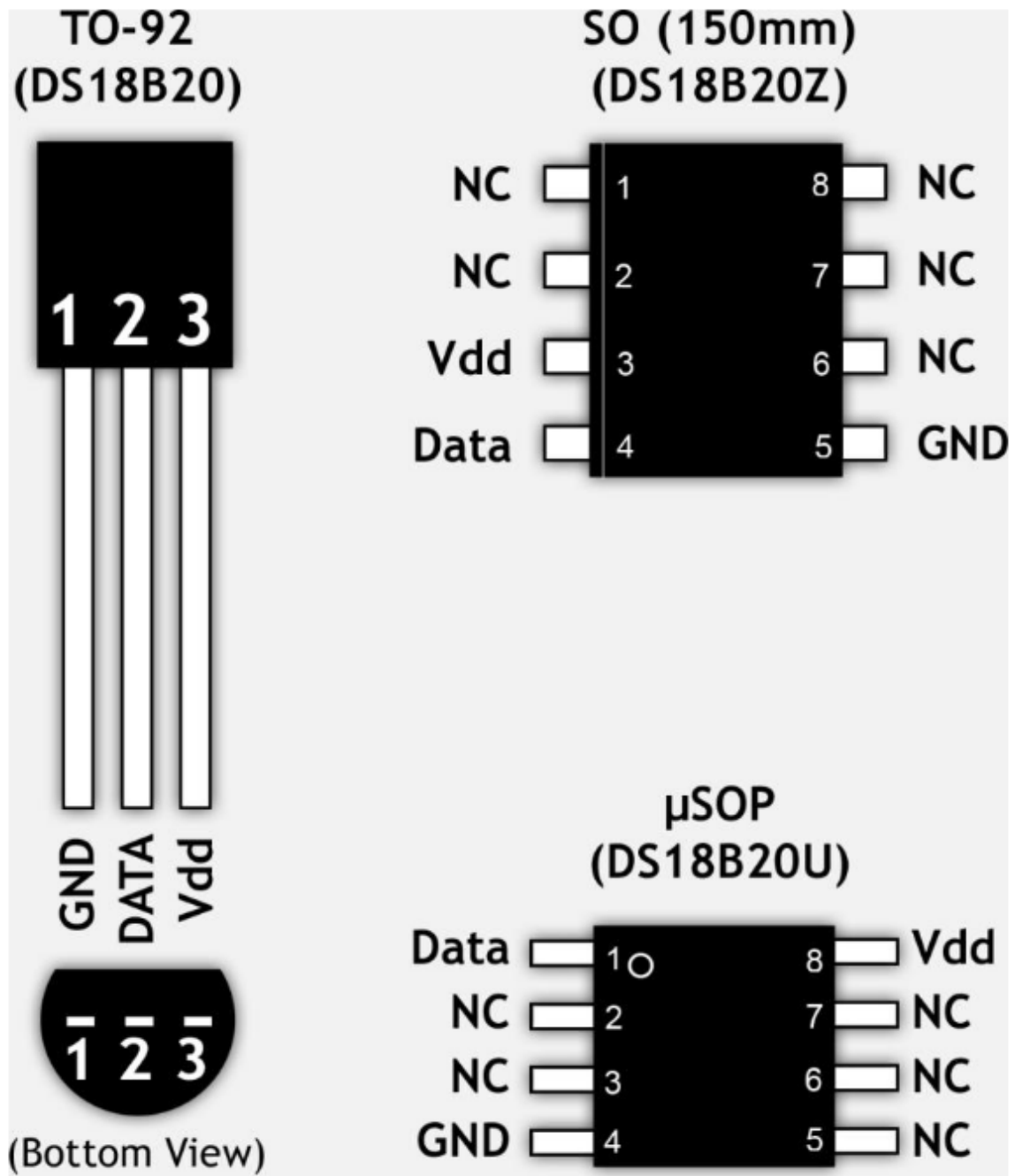


Рис 1.2 Схема датчика температури

DS18B20 зручний у використанні. Живити його можна через контакт data (в такому випадку використовується всього два контакти з трьох для підключення). Сенсор працює в діапазоні напруг від 3.0 В до 5.5 В і вимірює температуру в діапазоні від -55 ° С до + 125 ° С (від -67 ° F до + 257 ° F) з точністю ± 0.5 ° С (від -10 ° С до + 85 ° С).

Також можна підключити паралельно до 127 датчиків і прочитувати свідчення температури з кожного окремо. Не зовсім зрозуміло, в якому проекті подібне може знадобитися, але підключити два сенсора і контролювати температуру в холодильнику і морозильній камері можна. При цьому останеться вільними купу пинов на Arduino.

2. Завдання до лабораторної роботи

2.1 Підключіть датчик до плати Arduino згідно зі схемою, яка зображена на рисунку 2.1.

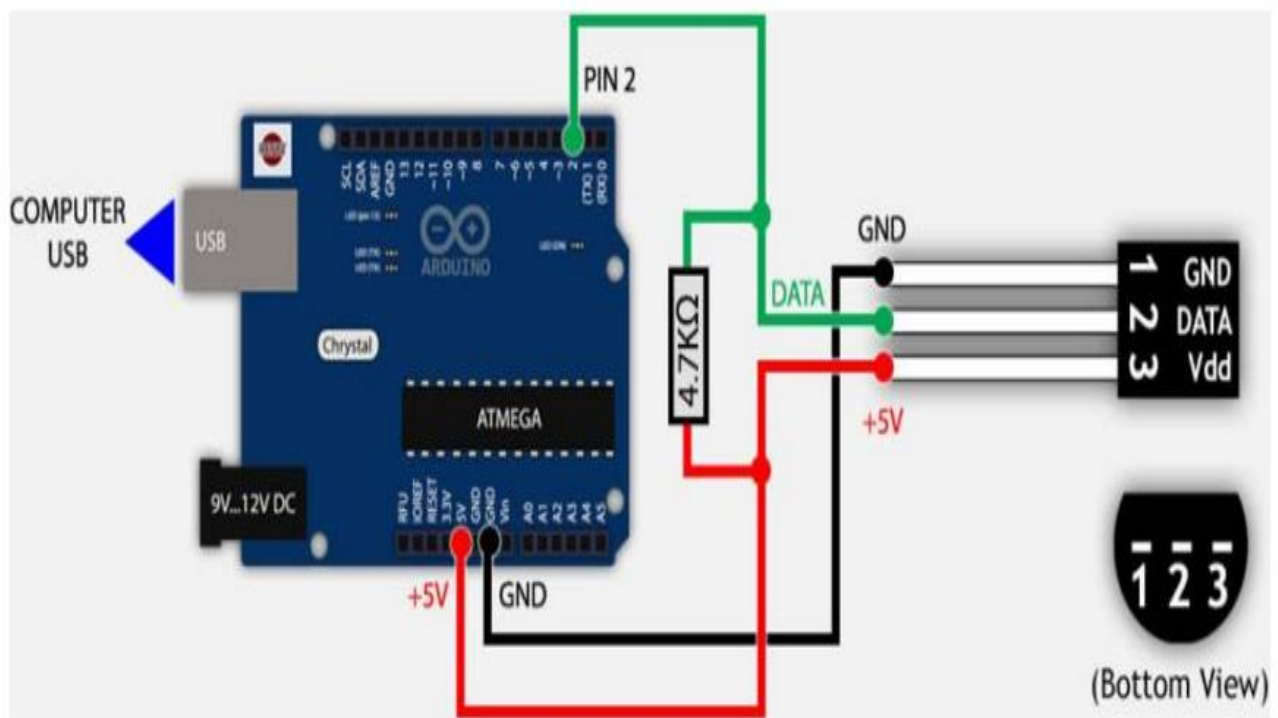


Рисунок 2.1 Схема підключення датчика температури

2.2 Запустіть на комп'ютері середовища розробки Arduino.

2.3 Підключіть плати Arduino через USB до ПК.

2.4 Написати код програми за таким алгоритмом.



2.5 Натисніть кнопку Verify и переконайтесь, що у нижній частині вікна з'явився надпис Done Compiling. Це означає, що у написаній програмі не знайдено помилок.

2.6 Виберіть у Tools->Board ваш тип плати. Перевірте, чи правильно обрано USB-порт в Tools->Serial port. Після натисніть на кнопку Upload.

Якщо внизу з'явився надпис “Done uploading” – процес запису прошив успішно.

3. Зміст звіту

- 3.1. Тема та мета роботи.
- 3.2. Схема електрична-принципова плати Arduino.
- 3.3. Код програми.
- 3.4. Висновки з роботи.

4. Контрольні запитання

- 1. Що таке послідовний інтерфейс UART ?
- 2. Як відбувається передача даних в UART?
- 3. Що таке протокол 1-Wire і навіщо він потрібний ?
- 4. Скільки існує видів USB роз'ємів і чим вони відрізняються ?
- 5. Скільки контактів має датчик температури і що кожний з них означає?

5. Список рекомендованої літератури

- 1. Шегедин, О.І. Теоретичні основи електротехніки : навч.посіб. / О.І. Шегедин, В.С. Маляр.; Львів : Магнолія 2006, 2012. – 167с.

2. Коруд, В.І. Електротехніка : підручник для ВНЗ / В.І.Коруд, О.Є. Гамола.; за заг. ред. В.І. Коруда.- 3-тє вид., перероб. і доп.- Львів:Магнолія, 2007.- 447с.
3. Мілих, В.І. Електротехніка,електроніка та мікропроцесорна техніка:підручник для ВНЗ/В.І. Мілих, О.О.Шавьолкін;за ред.В.І.Мілих.-К.: Каравела, 2007.-688с.
4. Синдеев, Ю.Г. Электротехника с основами электроники: учебное пособие.; Ростов-на-Дону: Феникс, 2011. -407 с.
5. Arduino. [Електронний ресурс]. – Режим доступу: <https://www.arduino.cc/>
6. Autodesk 123D Circuit. [Електронний ресурс]. – Режим доступу: <https://123d.circuits.io/>
7. Atmel Studio. [Електронний ресурс]. Microchip Atmel. – Режим доступу: <http://www.atmel.com/tools/atmelstudio.aspx>

Скетч

```
//підключити бібліотеки OneWire і DallasTemperature
#include <OneWire.h>

#include <DallasTemperature.h>

// вказати до якого порту ми підключили контакт по передачі данихв
нашому випадку до 2

#define ONE_WIRE_BUS 2

#define TEMPERATURE_PRECISION 9

//налаштування для oneWire для спілкування з будь-якими
пристроями-Wire

OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature sensors(&oneWire);

// масиви для зберігання адрес пристроїв

DeviceAddress insideThermometer, outsideThermometer;
```

```

void setup(void)
{
  // почати послідовного порту
  Serial.begin(9600);

  Serial.println("Dallas Temperature IC Control Library Demo");

  // Запуск бібліотеки
  sensors.begin();

  // знайти пристрої на шині
  Serial.print("Locating devices...");
  Serial.print("Found ");
  Serial.print(sensors.getDeviceCount(), DEC);
  Serial.println(" devices.");

  // вказуємо вимоги до паразитного живлення
  Serial.print("Parasite power is: ");
  if (sensors.isParasitePowerMode()) Serial.println("ON");
  else Serial.println("OFF");

  // алгоритм пошуку виглядає так . Повертає 1, якщо нова адреса була
знайдена
  // нуль може означати, що шина замкнута, немає пристроїв,
  // або вже витягнуто всі з них
  // перевірка CRC, щоб переконатися, що ми не отримуємо сміття

```

```

// привласнює першу адресу, вказану в insideThermometer

// якщо Serial.println ( "Не вдалося знайти адресу для
insideThermometer") (oneWire.search (insideThermometer)!);

// привласнює секундної адресу, вказану в outsideThermometer

// якщо Serial.println ( "Не вдалося знайти адресу для
outsideThermometer") (oneWire.search (outsideThermometer)!);

if (!sensors.getAddress(insideThermometer, 0)) Serial.println("Unable to
find address for Device 0");

if (!sensors.getAddress(outsideThermometer, 1)) Serial.println("Unable to
find address for Device 1");

// показати адреси які ми знайшли на шині

Serial.print("Device 0 Address: ");

printAddress(insideThermometer);

Serial.println();

Serial.print("Device 1 Address: ");

printAddress(outsideThermometer);

Serial.println();

// встановити дозвіл на 9 біт

sensors.setResolution(insideThermometer, TEMPERATURE_PRECISION);

sensors.setResolution(outsideThermometer, TEMPERATURE_PRECISION);

Serial.print("Device 0 Resolution: ");

```



```

Serial.print(sensors.getResolution(insideThermometer), DEC);
Serial.println();

Serial.print("Device 1 Resolution: ");
Serial.print(sensors.getResolution(outsideThermometer), DEC);
Serial.println();
}

// Функція для друку адресу пристрою
void printAddress(DeviceAddress deviceAddress)
{
  for (uint8_t i = 0; i < 8; i++)
  {

    if (deviceAddress[i] < 16) Serial.print("0");
    Serial.print(deviceAddress[i], HEX);
  }
}

// Функція для друку температури для пристрою
void printTemperature(DeviceAddress deviceAddress)
{
  float tempC = sensors.getTempC(deviceAddress);
  Serial.print("Temp C: ");

```

```

Serial.print(tempC);
Serial.print(" Temp F: ");
Serial.print(DallasTemperature::toFahrenheit(tempC));
}

// Функція для друку дозволу пристрою
void printResolution(DeviceAddress deviceAddress)
{
    Serial.print("Resolution: ");
    Serial.print(sensors.getResolution(deviceAddress));
    Serial.println();
}

// головна функція для друку інформації про пристрій
void printData(DeviceAddress deviceAddress)
{
    Serial.print("Device Address: ");
    printAddress(deviceAddress);
    Serial.print(" ");
    printTemperature(deviceAddress);
    Serial.println();
}

void loop(void)

```

```
{  
  // sensors.requestTemperatures виклик для видачі глобальної температури  
  // запит на всі пристрої на шині  
  
  Serial.print("Requesting temperatures...");  
  sensors.requestTemperatures();  
  Serial.println("DONE");  
  
  // друк інформації про пристрій  
  printData(insideThermometer);  
  printData(outsideThermometer);  
}
```

Лабораторна робота №3. Послідовний інтерфейс, РК-дисплей, інтерфейс 1-wire

Вступ

У техніці часто потрібно розробляти пристрої, які не повинні бути пов'язані з комп'ютером. Частіше за все потрібно робити зчитування з датчиків будь-яких значень, виконувати їх обробку і виводити інформації не через послідовний порт в комп'ютер, а, наприклад, на ЖК-дисплей. Також можлива ситуація, коли необхідно отримати деяку реакцію на зовнішній подразник. Яскравим прикладом може послужити система підігріву, яка автоматично включається / вимикається при певних значеннях температури. Також таким прикладом може бути звичайна сигналізація.

Мета роботи:

Вивчити основні принципи організації обміну даними між контролером Arduino і комп'ютером через віртуальний COM-порт. Навчитися робити індикацію на LCD-дисплеї. Вивчити основні принципи прийому даних з датчика температури DALLAS 18B20, їх обробки і індикації. Вивчити і перевірити роботу датчика.

Теоретичні відомості:

1.1. Послідовний порт

Послідовний порт або інтерфейс стандарту RS-232 - це одне з багатьох засобів передачі інформації між пристроями. Все COM-порти мають декількома властивостями:

1) Повнодуплексний обмін даними. Чи означає, що можна одночасно передавати і приймати потік даних. існують два апаратно і програмно незалежних каналу передачі даних. один канал для передачі даних, інший канал для прийому даних. причому COM-портам байдуже, чим зайнятий

процесор в цей час, у них присутні власні буфери прийому і передачі даних. У цих буферах дані шикуватися в чергу на передачу і черга на прочитання даних процесором. Будь-яка програма може звернутися до СОМ-порту і отримати дані з його буфера, тим самим очистивши його.

2) Набір сервісних сигналів. Сервісні сигнали, передбачені стандартом RS-232с, дозволяють організувати обмін даними між двома пристроями одночасно в обох напрямках. Сервісні сигнали представлені окремими цифровими входами і виходами з пам'яттю.

3) Програмна незалежність. Для реалізації цієї незалежності використовується UART (Universal Asynchronous Receiver-Transmitter). UART-вузол обчислювальних пристроїв, призначений для зв'язку з іншими

цифровими пристроями. Він перетворює заданий набір даних в послідовний вид так, щоб було можливо передати їх по однопровідною цифровий лінії іншому аналогічному влаштуванню. UART повністю реалізований апаратно і не залежить від програмного забезпечення і ОС.

4) Асинхронна передача даних по каналу зв'язку. Чи означає те, що РС може послати дані на кінцеве пристрій, не піклуючись про синхронність їх надходження. Кінцеве пристрій сам підлаштовується під отримані дані. У синхронних протоколах для цього служить спеціальний сигнал, передається по окремому проводу.

1.2. Передача даних через послідовний порт

Плата Arduino не вдалося підключитися через реальні СОМ-порти. вона підключається через інтерфейс USB, але для зв'язку з комп'ютером все одно використовує віртуальний СОМ-порт. Для обміну даними з пристроями, які підтримують послідовний інтерфейс використовується набір функцій Serial. Для обміну даними Serial використовують цифрові порти введення / виводу 0 (RX) і 1 (TX), а також USB порт. Важливо враховувати, що при використанні функцій Serial не можна одночасно використовувати порти 0 і 1 для інших цілей.

Середовище розробки Arduino має вбудований монітор послідовного інтерфейсу (Serial monitor), який можна викликати командою Сервіс -> Монітор порту або натисканням комбінації клавіш Ctrl + Shif + M. Для

початку обміну даними необхідно запустити монітор натисканням кнопки Serial monitor і виставити ту ж швидкість зв'язку (зазвичай вона дорівнює 9600), з якої викликана функція begin ().

Нижче в таблиці 1.1 представлений набір основних функцій Serial.

Функція	Опис

Відео:

<https://www.youtube.com/watch?v=BhpBw0DzBzU>

<https://www.youtube.com/watch?v=JDoTn1sLxWQ>

Датчик Dallas18B20

Датчик Dallas18b20 дозволяє виміряти температуру з більшою точністю. Він працює по шині One-Wire. Для пристроїв, що працюють по цій шині, в середовищі Arduino використовується бібліотека OneWire.

Характеристики:

Показатель DS18B20

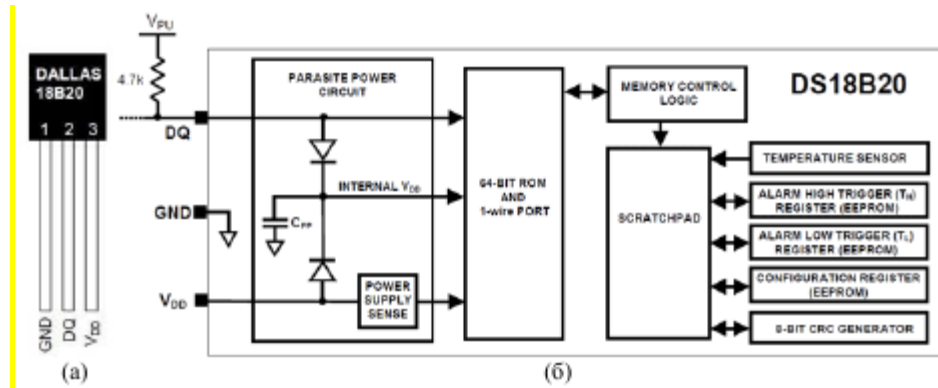
Допустимий діапазон t, °C -55..+125

Погрешність вимірювання t, min ±0.5°C при t=10..+85°C

Погрешність вимірювання t, max ±2°C

Разрешение шкалы t, °C 0.5 / 0.25 / 0.125 / 0.0625

Датчик может находиться в металлическом корпусе, который заизолирован термоусадочной трубкой. Поэтому его можно погружать в воду. На металле трубка прилегает очень плотно, на кабеле слабее, поэтому есть вероятность того, что внутрь может просочиться вода. С целью избегания данной ситуации желательно не погружать датчик в воду целиком. Сам датчик изображен на рисунке 1.



Датчик содержит уникальный 64-битный ROM код, состоящий из 8 битов, определяющих код серии, 48 бит уникального номера и 8 бит помехоустойчивого CRC кода.

Информация об измеренной температуре хранится в оперативной памяти датчика, которая состоит из 9 байт:

1 и 2 байты хранят информацию о температуре.

3 и 4 байты хранят соответственно верхний и нижний пределы температуры.

5 и 6 байты зарезервированы.

7 и 8 байты используются для сверхточного измерения температуры.

9 байт хранит помехоустойчивый CRC код предыдущих 8 байт.

Подключение датчика к плате имеет свои особенности. Он использует так называемое «паразитное питание». Паразитное питание – это питание электронного устройства напряжением каких-либо сигналов без использования специально выделенной шины питания. Сигнальная шина данных, с которой берется паразитное питание, может быть специально предназначена для такого режима. Но возможен также случай, когда сигнальная шина не предназначена для предоставления паразитного питания, при этом отбираемая мощность должна быть достаточно малой, чтобы не нарушить работу сигнальной линии по ее прямому назначению.

Логическая линия может быть использована как источник питания, если на ней гарантированно присутствует сигнал высокого уровня в течение достаточно длительного времени. Устройство, получающее паразитное питание, может некоторое время заряжаться от линии, а затем передать по этой же линии некоторую информацию.

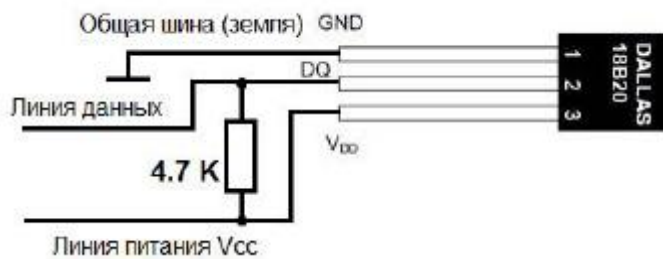
Интерфейс 1-Wire использует паразитное питание.

1-Wire – двунаправленная шина связи для устройств с низкоскоростной передачей данных (обычно 15,4 Кбит/с, максимум 125 Кбит/с), в которой данные передаются по цепи питания (то есть всего используются два провода – один для заземления, а второй для питания и данных; в некоторых случаях используют и отдельный провод питания).

Обычно интерфейс 1-Wire используется для того, чтобы связываться с недорогими простыми устройствами, такими как, например, цифровые термометры и измерители параметров внешней среды.

Собственно, схема подключения датчика изображена на рисунке 1.2.

Рисунок 1.2 – Схема подключения датчика



Пример. В данном примере используется один светодиод и датчик Dallas 18b20. Яркость светодиода зависит от значения температуры, которое снимается с датчика. Значения преобразований сигналов подобраны таким образом, что если вы возьмёте в кулак датчик температуры и подержите его около 1-2 минут, то светодиод начнет плавно включаться и увеличивать яркость свечения.

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
const int analogOutPin = 9; // Аналоговый выход, к которому подключен светодиод
// На аналоговые выходы подаются значения от 0 до 255. 0 соответствует
// отсутствию напряжения, 255 – максимальному напряжению 5В. Данная функция
// преобразует значение val из одного диапазона (from; to) пропорционально в
// значение другого диапазона (rFrom; rTo)
float ConvertMap(float val, float from, float to, float rFrom, float rTo){
    float d1, d2, vval;
    float res;
    d1 = to - from;
    d2 = rTo - rFrom;
    vval = val - from;
    res = vval / d1 * d2 + rFrom;
    if (res < rFrom){
        return rFrom;
    }
    else if (res > rTo){
        return rTo;
    }
    else{
        return res;
    }
}

void setup(void)
{
    // start serial port
    Serial.begin(9600);
    Serial.println("Dallas Temperature IC Control Library Demo");

    // Start up the library
```



```

sensors.begin();
}
void loop(void)
{
// call sensors.requestTemperatures() to issue a global temperature
// request to all devices on the bus
Serial.print(" Requesting temperatures... \n");
sensors.requestTemperatures(); // Send the command to get temperatures
Serial.print("Temperature for Device is: ");
float t = sensors.getTempCByIndex(0);
Serial.print(sensors.getTempCByIndex(0));
//Преобразуем полученные с датчика данные с диапазона от 30 до 80 градусов
//по Цельсию в диапазон значений аналогового выхода (0; 255)
int outputValue = ConvertMap(t, 30, 80, 0, 255);
Serial.print("\n");
Serial.print(outputValue);
analogWrite(analogOutPin, outputValue);
delay(10);
}

```

Схема подключения для данной программы:

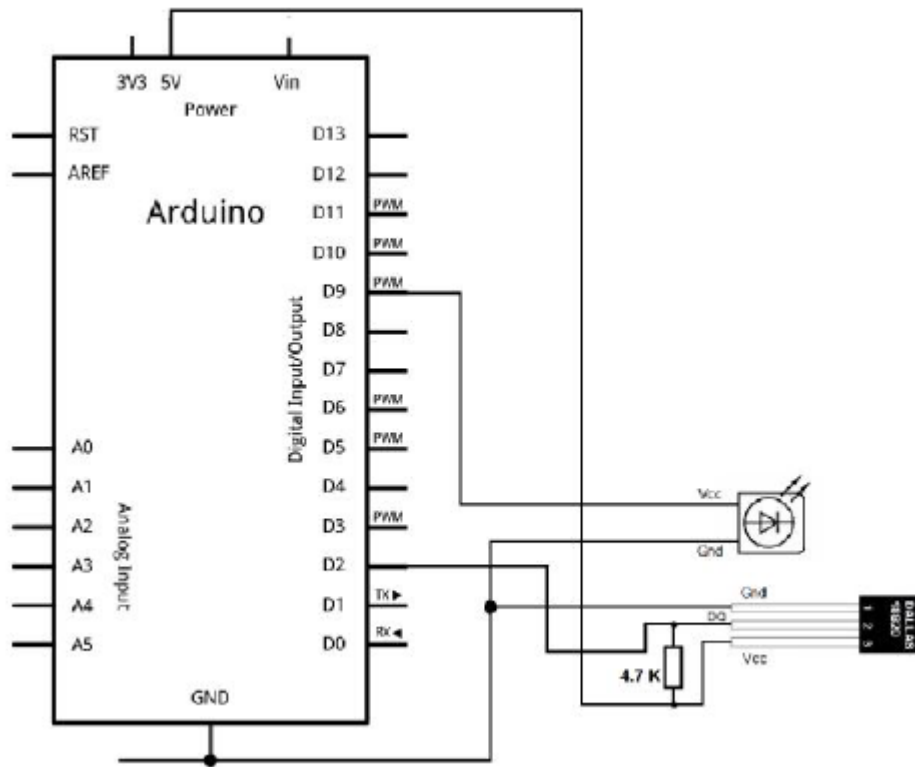


Рисунок 1.3 – Схема подключения к примеру

Важно! К земле подключается короткая ножка светодиода. Также следует следить за полярностью подключения датчиков и правильностью подключения резисторов. Неправильное подключение может повредить элементы схемы.

Яркость светодиода в основном определяется небольшим приростом напряжения от нуля до 1.5-3В, в зависимости от типа применяемого светодиода. Дальнейшее увеличение яркости слабозаметное.

Лабораторна робота №4. Простий радіо-зв'язок точка-точка

Arduino - торгова марка апаратно-програмних засобів для побудови простих систем автоматики і робототехніки, орієнтована на непрофесійних користувачів. **Програмна** частина складається з безкоштовної програмної оболонки(IDE)для написання програм, їх компіляції та програмування апаратури. **Апаратна** частина являє собою набір змонтованих друкованих плат, що продаються як офіційним виробником, так і сторонніми виробниками. Повністю відкрита архітектура системи дозволяє вільно копіювати або доповнювати лінійку продукції Arduino.

Arduino може використовуватися як для створення автономних об'єктів автоматики, так і підключатися до програмного забезпечення на комп'ютері через стандартні дротові і бездротові інтерфейси.

Мета і основні завдання роботи

В цій лабораторній ми навчимося з'єднувати дві arduino по радіоканалу ISM діапазону, використовуючи радіо модуль nRF24L01 / +, на відстані до 100 м. Якщо використовувати радіо модулі NRF24L01 + PA + LNA, то відстань між arduino можна збільшити до 1 км, не змінюючи код скетчу.

Теоретичні відомості

При створенні деяких проектів, потрібно розділити виконувані завдання між декількома arduino.

Наявність двох або більше Arduino які з'єднані один з одним

радіосигналом на відстані відкриває безліч можливостей:

- Дистанційні датчики температури, тиску, сигналізації, і багато іншого
- Управління роботом і контроль з відстані від 50 футів до 2000 футів
- Дистанційне керування і моніторинг сусідніх будівель
- Автономні транспортні засоби всіх видів

Недоліки:

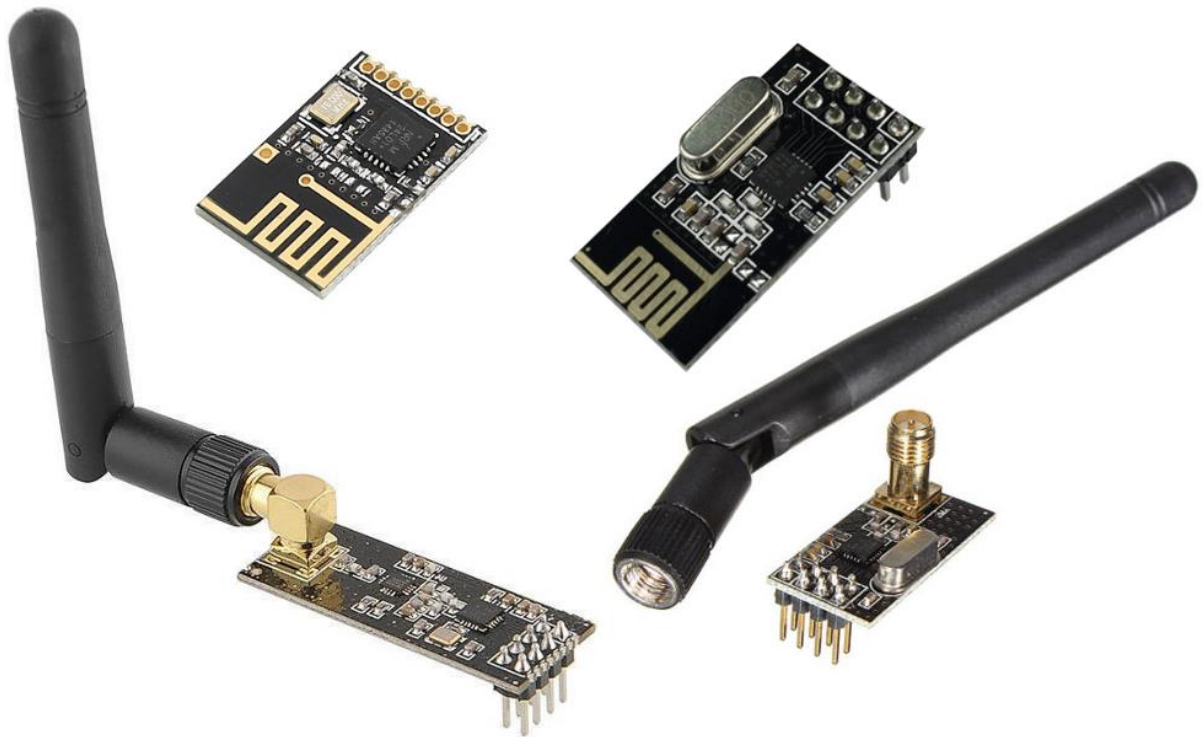
- Модулі nRF24L01 + працюють в діапазоні радіохвиль ISM (Industrial, Scientific, Medical) 2,4 ГГц, на якому працюють WiFi, Bluetooth і інші пристрої, наприклад радіо телефони і навіть СВЧ печі. Ці пристрої можуть «глушити» деякі канали даного діапазону. Тому поблизу таких пристроїв дальність зв'язку між модулями, на деяких каналах, різко зменшується. Збільшити дальність можна змінивши канал зв'язку на будь-який з 128 доступних модулів nRF24L01 +.
- При виборі швидкості 2 Мб / с, задіюються відразу два канали (обраний і наступний за ним).
- Модулі живляться від напруги 3,3 В постійного струму. Але їх можна живити від 5 В через адаптер nRF24L01 +.

Особливості контролерів Arduino:

- До складу входить мікроконтролер Atmel.
- Програмуються зручним способом - через USB, завдяки мікросхемі USB-to-Serial.
- Допускається використання багатьох I / O висновків у зовнішніх схемах.

Обладнання, пристрої та матеріали

Радіомодуль nRF24L01 / +



Напруга живлення - від 1,9 до 3,6v, рекомендований 3,3v.

Входи толерантні до напруги до 5v, тобто можна підключати до мікроконтролера, що працює від 5v.

Взаємодіє з мікро контролером по шині SPI.

Можливі швидкості обміну даними - 250kbps, 1Mbps і 2Mbps.

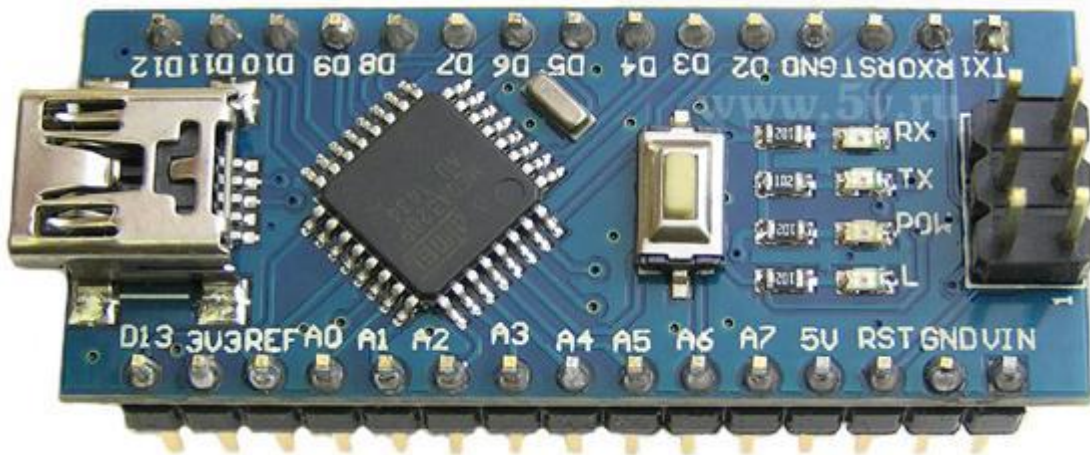
Кількість каналів - 126, з кроком 1МГц. Якщо встановлена швидкість передачі 2MBPS, то використовується ширина двох каналів.

Енергоспоживання близько 0,9мкА в режимі power-down, до 11,3мА при передачі, і до 13,5мА при прийомі.

«Адаптер» зі стабілізатором на 3.3v.



Arduino Nano v3.0



Опис:

Мініатюрний модуль на основі популярного мікроконтролера ATMEGA328P.

Оптимально підходить для макетування із застосуванням безпечно-макетних плат, бо всі контакти виведені на дві лінійки по краях плати, крок виводів 2,54мм, відстань між лінійками 15мм.

Вбудований bootloader і перетворювач USB \leftrightarrow COM на базі мікросхеми CH340, дозволяє оновлювати прошивку без використання програматора, єдиним натисканням кнопки на комп'ютері. Однак, при необхідності, може бути "прошитий" і будь-яким стандартним програматором зі

стандартним 6-вивідним інтерфейсом ISP.

Nano 3.0 (CH340G) є аналогом поширених модулів Nano 3.0, і відрізняється від них лише переробленою схемою перетворювача USB \leftrightarrow COM, інтегрованого на плату. Замість мікросхеми FT232RL виробництва FTDI, в цій версії модуля застосована мікросхема CH340G, виробництва WCH.

З точки зору використання, модуль нічим не відрізняється від поширених модулів Nano 3.0, тому що перетворювач USB \leftrightarrow COM, з точки зору програмування прозорий, і всього-лише забезпечує з'єднання з комп'ютером за допомогою додавання ще одного COM-порту.

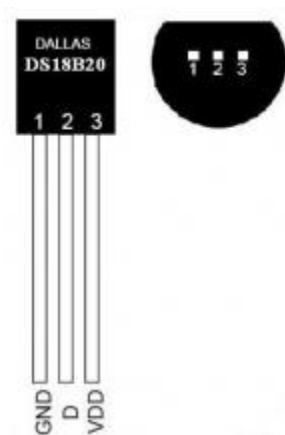
Специфікація:

- Мікроконтролер: ATmega328P
- Напруга ядра (логічні рівні): +5 вольт.
- Напруга живлення по входу VIN (рекомендований): +7 ... + 12 вольт.
- Гранично допустимий потенціал напруги живлення по входу VIN: +6 ... + 20 вольт.
- Цифрових входів / виходів: 14 (6 з яких дозволяють апаратно виводити PWM / ШІМ).
- Светодіоди: Живлення, Rx (прийом), Tx (Передача).
- Входи АЦП: 8.
- Максимально допустимий впадає / витікаючий струм висновків: 40 мА.
- ППЗУ (Flash Memory): 32 кбайт, з яких 2 кБ використовується під завантажувач (bootloader).
- ОЗУ (SRAM): 2 кБайт (вбудовано в ATmega328P)
- EEPROM: 1 Кб (вбудовано в ATmega328P)
- Тактова частота 16 МГц стабілізована кварцом.
- Готовий модуль.

- Мініатюрні розміри: 45 * 18 мм.
- Вага модуля 6 гр.

Драйвера

Цифровий датчик температури DS18B20



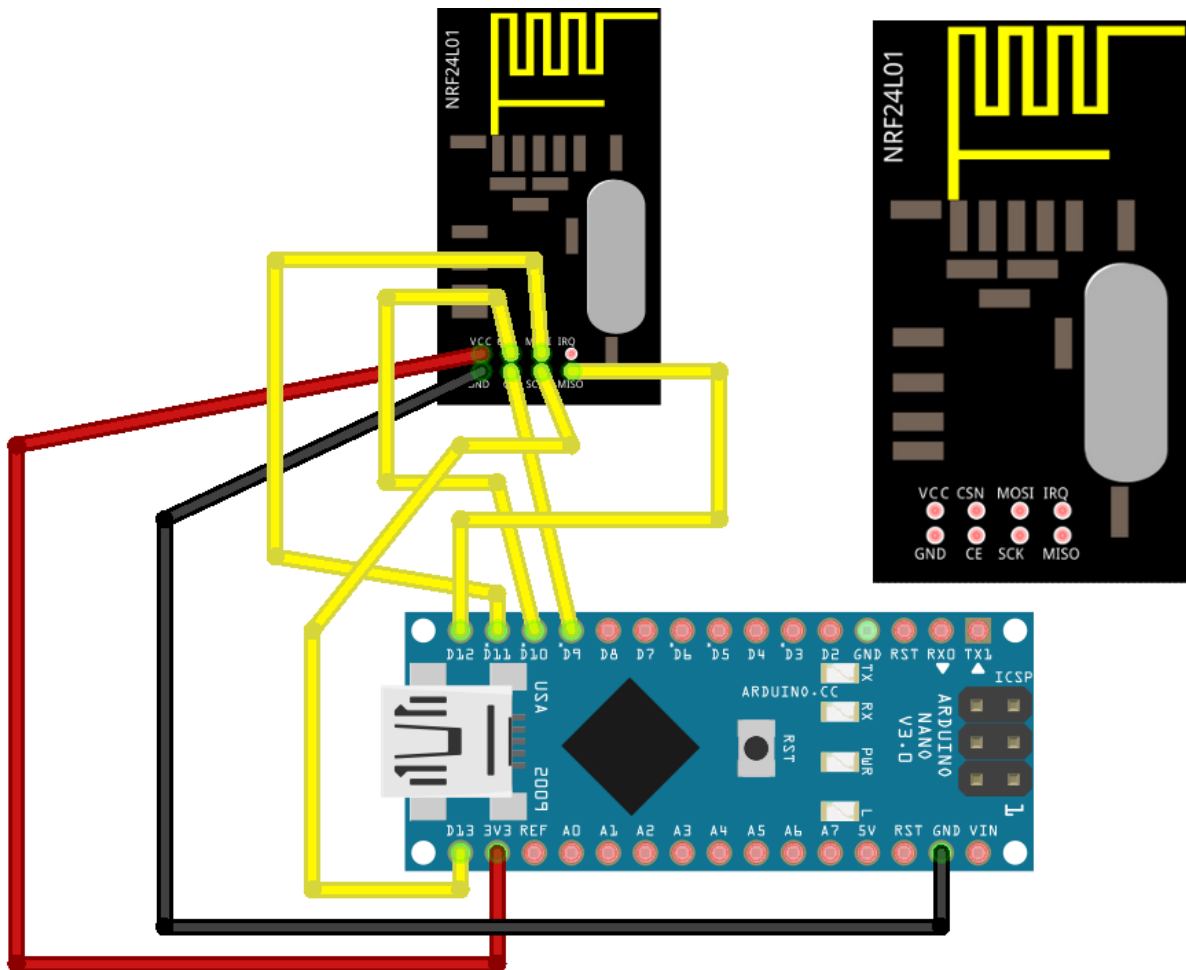
Діапазон вимірюваних температур від -55°C до $+125^{\circ}\text{C}$. зчитувати з приладу цифровий код є прямим безпосереднім кодом вимірюного значення температури і не потребує додаткових перетвореннях. Програмована користувачем роздільна здатність вбудованого АЦП може бути змінена в діапазоні від 9 до 12 розрядів вихідного коду. Абсолютна похибка перетворення менше $0,5^{\circ}\text{C}$ в діапазоні контрольованих температур -10°C до $+85^{\circ}\text{C}$. Максимальний час повного 12-ти розрядного перетворення $\sim 750\text{мс}$ (при дозволі 12 розрядів). Для підключення потрібно резистор 4.7кОм (див. Другий малюнок).

Внутрішня енергонезалежна пам'ять температурних установок забезпечує запис довільних значень верхньої та нижньої межі установок. Крім того, мікросхема містить вбудований логічний механізм пріоритетною сигналізації в лінію про факт виходу температури за один з обраних порогів. Вузол 1-Wire-інтерфейсу приладу організований таким чином, що існує теоретична можливість адресації необмеженої кількості подібних пристроїв на однопровідною лінії.

Термометр має індивідуальний 64-розрядний реєстраційний номер (груповий код 028H) і забезпечує можливість роботи без зовнішнього джерела живлення, тільки за рахунок паразитного живлення однопроводної лінії. Живлення приладу через окремий зовнішній висновок виробляється напругою від 3.0В до 5.5В.

Порядок і рекомендації до виконання роботи і обробки результатів

1. Підключення виробляти за схемою, наведеною на малюнку 5



SCK - D13

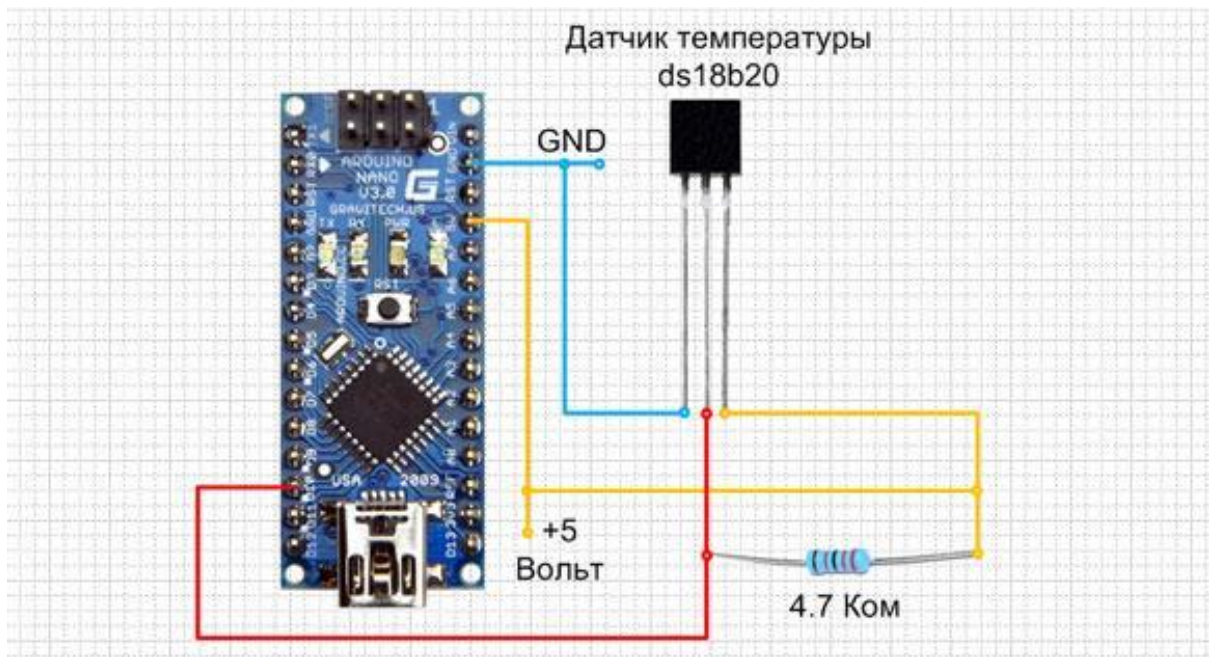
MOSI - D11

MISO - D12

CSN - D10

CE / SS - D9

Цифровий датчик підключати за схемою, наведеною на рис 6



2. Підключення датчика DS18B20 до Arduino здійснюється нормальним способом через резистор номіналом 4.7 кОм.

3. Скетч

```

1.
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
RF24 radio(9, 10);
const uint64_t pipe = 0xE8E8F0F0E1LL;

2.
void setup(void) {
  Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(1, pipe);
  radio.startListening();
  Serial.begin(9600);
  Serial.println("Temperature Starting.....");
  delay(1000);
}

3.
void loop(void) {
  if (radio.available()) {
    float temperature = 0;
    if (!radio.read(&temperature, sizeof(float)))
  }

Serial.println("ACK not received by client.");
}
Serial.print("Temperature : ");
Serial.println(temperature);
delay(1000);
}

4.
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 8

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

RF24 radio(9, 10);
const uint64_t pipe = 0xE8E8F0F0E1LL;

5.
void setup(void) {

Serial.begin(9600);
sensors.begin();
radio.begin();
radio.openWritingPipe(pipe);
}

6.
void loop(void) {
  sensors.requestTemperatures();
  float temperature = sensors.getTempCByIndex(0);
  radio.write(&temperature, sizeof(float));
  delay(1000);
}

```

З даних блоків скласти 2 скетча - відправка і зчитування температурних параметрів.

4. Обробка даних полягає в перевірці надходження даних з температурного датчика на відповідний com-порт в який підключена приймаюча arduino.

Оформлення звіту

Звіт повинен містити отримані дані, лістинг програми відправки та отримання даних.

Контрольні питання

1. Навіщо в даній лабораторній потрібен перехідник на 3.3V?
2. У чому особливість контролерів Arduino?
3. Опишіть призначення кожного виведення.
4. Опишіть принцип роботи температурного датчика.
5. Навіщо в схемі потрібен резистор на 4.7кОм?

Список рекомендованої літератури

1. NRF24L01-Arduino [Електронний ресурс].- Режим доступу:
<https://istarik.ru/blog/arduino/40.html>
2. Iarduino.ru Все для радіоаматорів[Електронний ресурс]. - Режим доступу: <https://lesson.iarduino.ru/page/urok-26-4-soedinyаем-dve-arduino-po-radiokanalu-cherez-nrf24l01/>
3. Learn! Do! Arduino-info![Електронний ресурс]. - Режим доступу:
<https://arduino-info.wikispaces.com/Nrf24L01-2.4GHz-HowTo>

Лабораторна робота №5.

Однокристальний мікроконтролер ESP8266 з WiFi та протокол MQTT

Мета роботи: ознайомитись з Wi-fi модулем ESP8266. Зрозуміти основні принципи роботи інтернету речей. Оволодіти навичками програмування модулю, та освоїти програмне забезпечення для обміну даними з модулем за допомогою MQTT протоколу.

Завдання на лабораторну роботу: зібрати схему і написати програму для зчитування температури з датчика DS18B20. Значення температури передаються у хмару Wi-Fi радіоканалом, що побудовано з використанням модуля Wi-Fi вузла *ESP8266*.

1. Теоретичні відомості

Інтернет речей (IP) (англ. Internet of Things, IoT) — це мережа, що складається із взаємозв'язаних фізичних об'єктів (речей) або пристроїв, які мають вбудовані датчики, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом і комп'ютерними системами, за допомогою використання стандартних протоколів зв'язку. Крім датчиків, мережа може мати виконавчі пристрої, вбудовані у фізичні об'єкти і пов'язані між собою через дротові і бездротові мережі. Ці взаємопов'язані об'єкти (речі) мають можливість зчитування та приведення в дію, функцію програмування та ідентифікації, а також дозволяють виключити необхідність участі людини, за рахунок використання інтелектуальних інтерфейсів.



Рис.1. Схематичне зображення принципу Інтернету речей

В даній лабораторній роботі використовується модуль ESP8266.

ESP8266 - мікроконтролер китайського виробника Espressif з інтерфейсом Wi-Fi. Крім Wi-Fi мікроконтролер відрізняється можливістю виконувати програми з зовнішньої флеш-пам'яті з інтерфейсом SPI.

- 80 MHz 32-bit процесор Tensilica Xtensa L106. Можливий негарантований розгін до 160 МГц.
- IEEE 802.11 b / g / n Wi-Fi. Підтримується WEP і WPA / WPA2.
- 14 портів введення-виведення (з них можливо використовувати 11), SPI, I²C, I²S, UART, 10-bit АЦП.
- Живлення 2,2 ... 3,6 В. Споживання до 200 мА в режимі передачі, 60 мА в режимі прийому, 40 мА в режимі очікування. Режим зниженого споживання зі збереженням з'єднання з точкою доступу ~ 1 мА, режим глибокого сну 0.1 мкА.

Мікроконтролер не має на кристалі користувальницької незалежної пам'яті. Виконання програми ведеться з зовнішньої SPI ПЗУ шляхом динамічного підвантаження необхідних ділянок програми в кеш інструкції. Підвантаження йде апаратно, прозора для програміста. Підтримується до 16 МБ зовнішньої пам'яті програм. Можливий Standard, Dual або Quad SPI інтерфейс.

Для обміну даними з хмарою пропонуємо скористатись протоколом MQTT. MQTT (Message Queue Telemetry Transport) - спрощений мережевий протокол, який працює поверх TCP / IP. Використовується для обміну повідомленнями між пристроями за принципом видавець-підписчик.

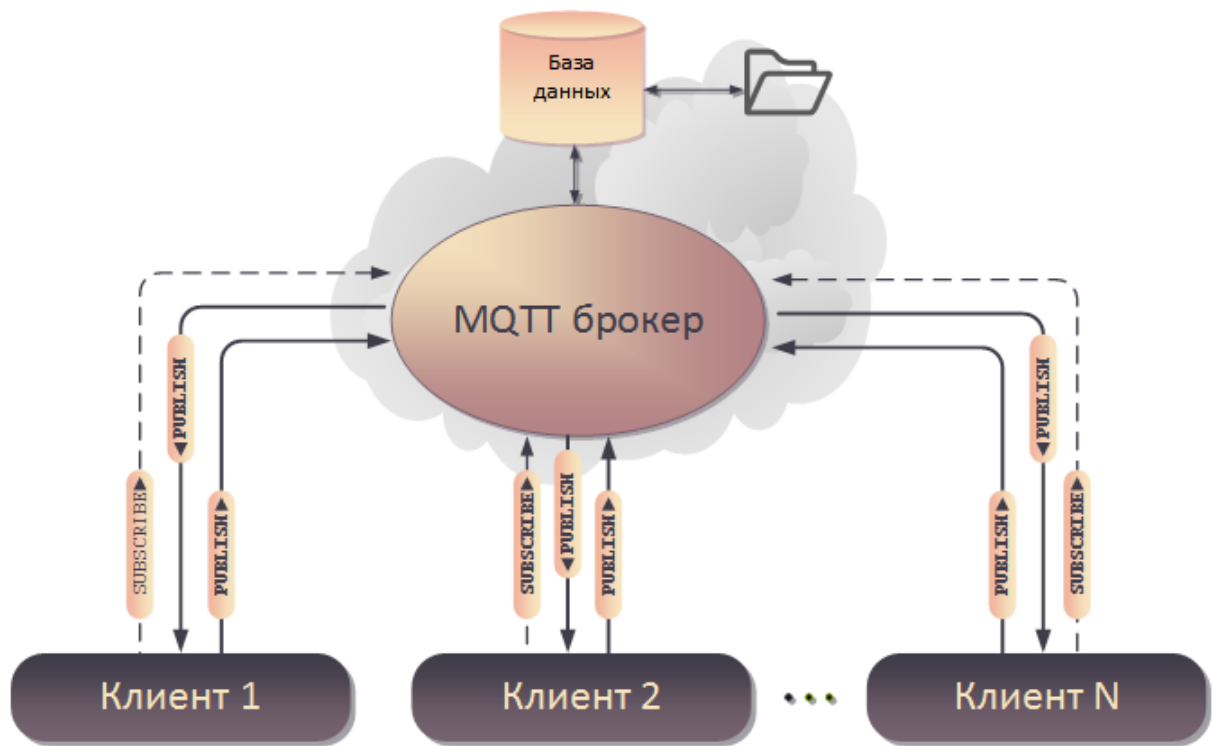


Рис.2. Структурна схема роботи протоколу MQTT.

MQTT дуже примітивний: з коротким заголовком, без контролю цілісності, що не накладає ніяких обмежень на структуру, кодування або схему даних. Єдина вимога до даних в кожному пакеті - вони повинні супроводжуватися ідентифікатором інформаційного каналу. Цей ідентифікатор в специфікації називається Topic Name або простіше топік. Дані передаються пакетами поверх протоколу TCP. Кількість даних в пакеті може бути від одного байта до 268 435 455 байт. Хоча публічні хмарні сервіси вводять тут більш жорсткі обмеження, до декількох кілобайт.

Протокол MQTT вимагає обов'язкової наявності брокера даних. Це центральна ідея технології. Всі пристрої посилають дані тільки брокеру і приймають дані теж тільки від нього. Брокер - це програма, яка виконує функції TCP сервера з динамічною базою даних.

База даних брокера зокрема містить таблицю з усіма отриманими пакетами з індексацією по топікам цих пакетів. Отримавши пакет, брокер надсилає його усім пристроям в мережі згідно їх підписці. Щоб пристрій щось отримав від брокера він повинен підписатися на топік. Топіки виникають динамічно за фактом підписки або за фактом приходу пакету з даним топіком. Від підписки на топік можна і відмовитися. Таким чином топіки є зручним механізмом організації зв'язків різних видів: один до

багатьох, багато до одного і багато до багатьох. Якщо у пакета немає підписчика, то він відкидається. Якщо підписчика немає на зв'язку, то пакет або відразу стирається в базі брокера, або чекає підключення підписчика деякий заданий в конфігурації час. Варіант поведінки визначається атрибутом QoS пакету.

Можливості протоколу:

- Простий у використанні. Протокол є програмним блоком без зайвої функціональності, що може бути легко вбудований в будь-яку складну систему;
- Зручний для більшості рішень з датчиками. Дає можливість пристроям виходити на зв'язок і публікувати повідомлення, які не були заздалегідь відомі або визначені;
- Легкий у адмініструванні;
- Низьке навантаження на канал зв'язку;
- Робота в умовах постійної втрати зв'язку або інших проблем на лінії;
- Немає обмежень на формат переданого контенту.

Методи MQTT:

MQTT визначає методи (так звані «дієслова»), щоб вказати бажану дію, яка повинна виконуватися на ідентифікованому ресурсі. Чим є цей ресурс, будь то вже існуючі дані або дані, що генеруються динамічно, залежить від реалізації сервера. Часто ресурс відповідає файлу або результату виконання якогось файлу, розміщеного на сервері.

- **Connect.** З'єднати: Чекає установки з'єднання з сервером.
- **Disconnect.** Роз'єднати: Чекає доки клієнт MQTT закінчить будь-яку роботу, що має зробити, і доки роз'єднається TCP/IP сесія.
- **Subscribe.** Підписатися: Чекає на завершення методу Subscribe чи UnSubscribe.
- **UnSubscribe.** Відписатися: просить сервер відписати клієнта від одного або кількох тем.
- **Publish.** Публікувати: негайно повертається в потік додатку після того, як передасть запит клієнту MQTT.

Що ще важливо:

- Пристрої першими встановлюють зв'язок з брокером. Тобто пристрої можуть перебувати за трансляторами мережевих адрес (NAT-ами) і не мати статичних IP-адрес і це не завадить зв'язку.

- Може бути застосований протокол SSL для шифрування трафіку. Але можна працювати і без SSL навіть з сервісами IBM, що полегшує налагодження.

- В виключно важких випадках MQTT брокери дозволяють підключатися до них через протокол WebSocket на 80-й порт.

- Також може бути довільно змінений порт штатного протоколу MQTT.

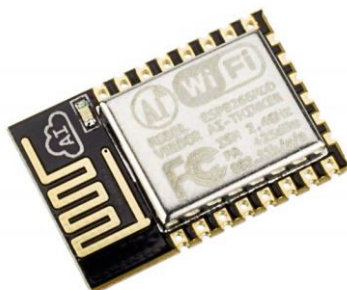
- І клієнт/клієнти з графічним користувацьким інтерфейсом і брокер можуть знаходитися на одному комп'ютері. Тобто рішення може бути і абсолютно локальним і масштабуватись в глобальне одним кліком.

- Різні брокери можуть з'єднуватися між собою підписуючись на повідомлення один у одного.

- Концепція топіків добре лягає на технологію NoSQL баз даних. З тією ж метою хмарні сервіси схиляють користувачів використовувати JSON кодування даних.

2. Перелік використаного обладнання

- Модуль ESP8266.



- Адаптер для Wifi модулів ESP8266

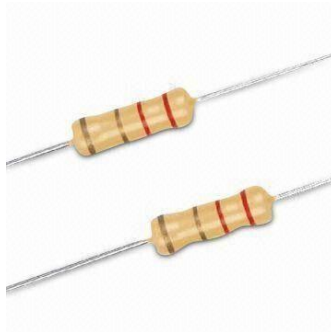


мікроАмпер
uAmper.com

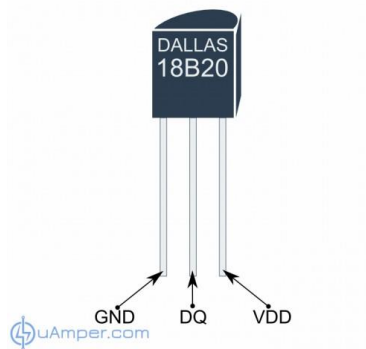
- USB-UART перетворювач



- Резистори: 1кОм, 4.7кОм



- Температурний датчик DS18B20



- Перемички папа-мама, папа-папа, мама-мама.

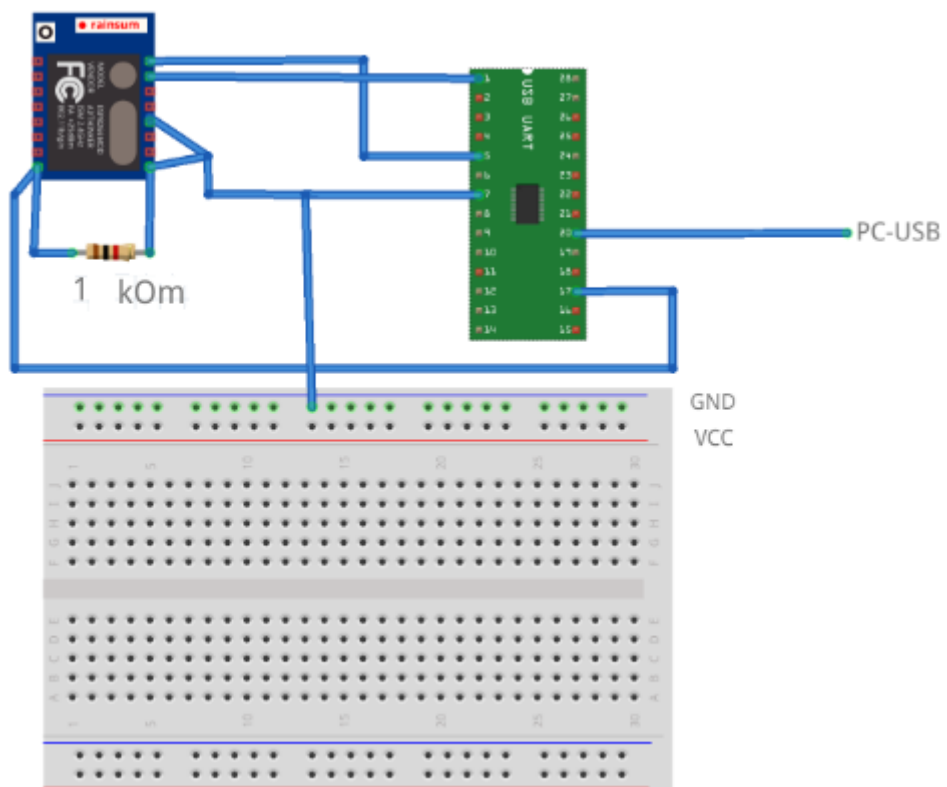


- Живлення для макетної плати 5В/3.3В



3. Порядок виконання роботи

- 1) Перевірте наявність всіх необхідних елементів макету.
- 2) Під'єднайте елементи у схему як зображено на рис. 3.



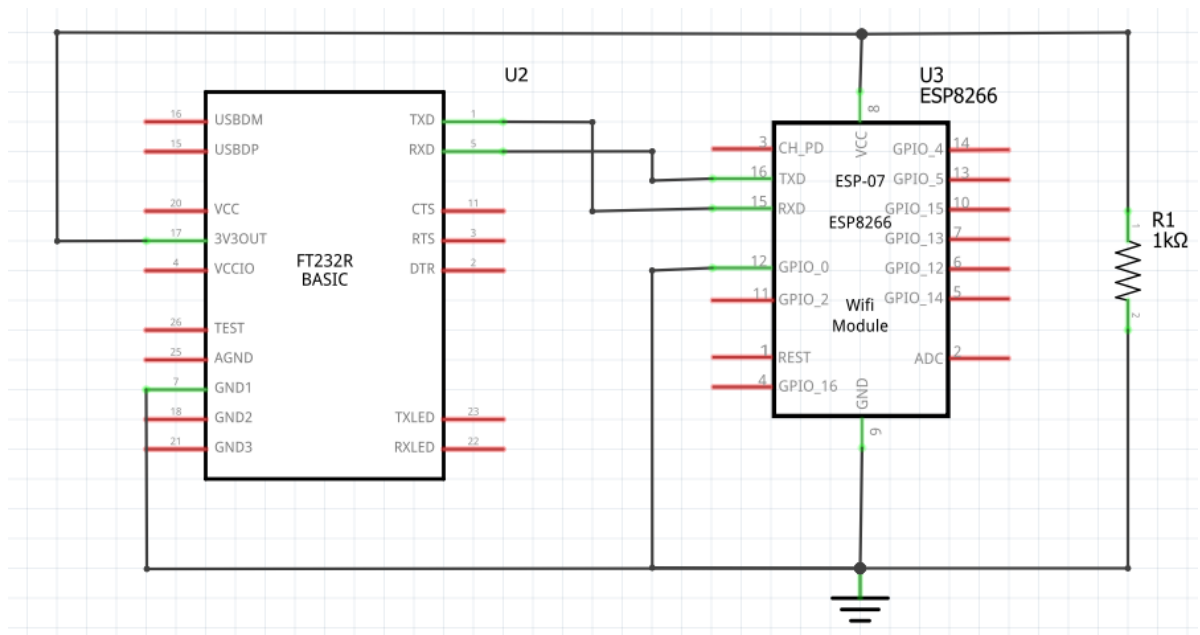


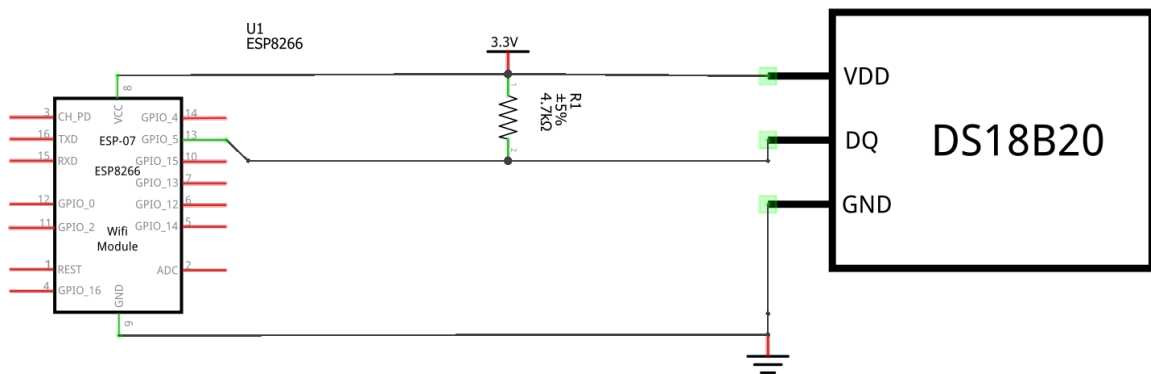
Рис. 3. Схема підключення ESP8266 до USB-UART перетворювача при прошивці.

Під час виконання даної лабораторної роботи можна використовувати будь-який USB-UART перетворювач, головне правильно під'єднати модуль, як зображено на рис. 3.

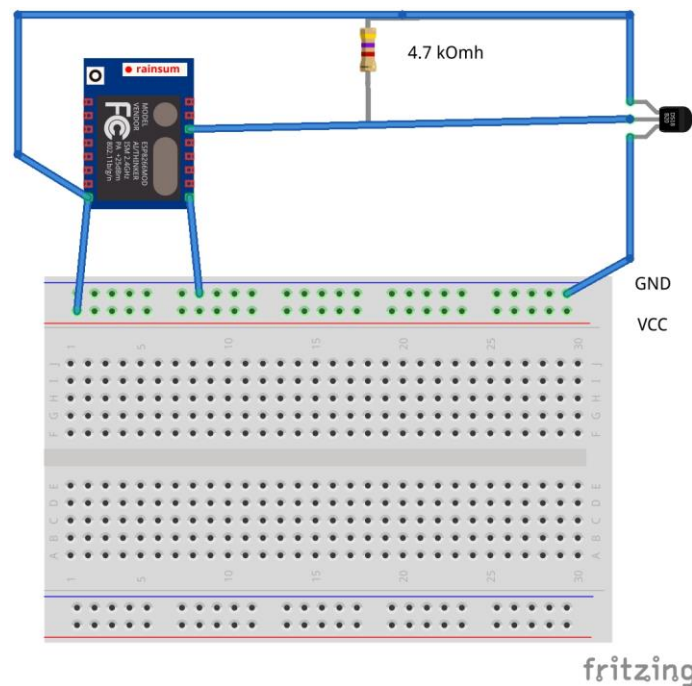
3) Тепер під'єднайте зібрану схему до комп'ютера і живлення. Запустіть Arduino IDE і відкрийте потрібний скетч (Додаток 1). Після того як переконались, що всі дані у кодї ввели правильно, завантажуйте скетч на контролер.

4) Перевірте налаштування MQTT-брокера Mosquitto і OpenHAB (як в лабораторній роботі №6).

5) Складіть схему як показано на рис. 4.



fritzing



fritzing

Рис. 4. Схема підключення датчика температури DS18B20.

б) Запустіть брокер і OpenHab і під'єднайте схему до живлення. У вікні інтерфейсу OpenHab можете спостерігати значення температури навколишнього середовища. Проекспериментуйте, зажміть датчик пальцями, і потримайте декілька секунд. Можна спостерігати зміну температури.

4. Зміст звіту

1. Мета та завдання роботи.
2. Схема електрична-принципова прошивки контролера і безпосереднього підключення датчика до модуля.
3. Код програми.

4. Висновки з роботи і аналіз результатів.

5. Контрольні запитання

1. Що таке Інтернет речей?
2. Яке значення напруги живлення в ESP-8266?
3. Розшифруйте аббревіатуру MQTT?
4. Опишіть принцип роботи протоколу MQTT.
5. Перерахуйте методи MQTT.

6. Список рекомендованої літератури

1. Arduino. [Електронний ресурс]. – Режим доступу: <https://www.arduino.cc/>
2. MQTT. Wikipedia. [Електронний ресурс].- Режим доступу: <https://uk.wikipedia.org/wiki/MQTT>
3. Протокол MQTT і відкритий проект клієнта MQTT на Delphi. Geektimes. [Електронний ресурс].- Режим доступу: <https://geektimes.ru/post/268018/>
4. esp8266 - Спільнота розробників. [Електронний ресурс].- Режим доступу: <https://esp8266.ru/>

Додаток 1.

Скетч для даної роботи можна розділити на декілька блоків:

- **Блок ініціалізації** – призначений для підключення бібліотек і введення основних параметрів точки доступу і серверу MQTT.

```
#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <OneWire.h>

#include <DallasTemperature.h>

#define ONE_WIRE_BUS 5
```

```

const char* ssid = "Zgurovsky"; // вводимо ssid точки доступу
const char* password = "james12345"; // вводимо пароль точки доступу
const char* mqtt_server = "192.168.1.181"; //вводимо ip-адресу серверу
MQTT

char msg[50];

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);
char temperatureCString[6];

WiFiClient espClient;
PubSubClient client(espClient);

long lastMsg = 0;
int value = 0;

```

- **Блок підключення до Wi-fi** – призначений для підключення до мережі.

```

void setup_wifi() {

delay(10);

Serial.println();

Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}

Serial.println("");
Serial.println("WiFi connected");

```

```

Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

```

- **Блок установки** – призначений для виклику необхідних функцій та присвоєння бодрейту і порту сервера.

```

void setup() {
    DS18B20.begin();
    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
}

```

- **Блок зворотного зв'язку** – призначений для отримання даних з серверу.

```

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic); //Виводимо в монітор порту назву топика
    Serial.print("] ");
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]); //Виводимо в монітор порту
        //значення отриманих даних
    }
    Serial.println();
}

```

- **Блок моніторингу підключення** – призначений для перевірки на відображення підключення до мережі.

```

void reconnect() {
    while (!client.connected()) {

```

```

Serial.print("Attempting MQTT connection...");
if (client.connect("ESP8266Client")) {
    Serial.println("connected");
    client.publish("test", "OK");
    client.subscribe("inTopic");
} else {
    Serial.print("failed, rc=");
    Serial.print(client.state());
    Serial.println(" try again in 5 seconds");
    delay(5000);
}
}
}

```

- **Блок основної функції програми** – призначений для зчитування даних температури з датчика і передання їх на сервер.

```

void loop() {

    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    float tempC;

    DS18B20.requestTemperatures();

    tempC = DS18B20.getTempCByIndex(0); // присвоюємо змінній значення
    температури з датчика

    dtostrf(tempC , 2, 2, msg); // переписуємо отримане значення в char
    client.publish("home/temperature", msg ); // передаємо повідомлення
    delay (2000);
}

```


Лабораторна робота №6. **Робота з OpenHab та підключення MQTT broker**

Тема: Робота з OpenHab та підключення MQTT broker.

Мета: Навчитися встановлювати OpenHab та налаштовувати його для роботи з MQTT broker.

Короткі теоретичні відомості

Напрямок розумних будинків розвивається вже давно. За цей час з'явилося безліч різних постачальників «розумного» обладнання (освітлення, RGB підсвічування, термостати, приводи для штор / жалюзів, різноманітні датчики і т.д.) Всі вони працюють на різних протоколах.

В результаті, користувач, охочий автоматизувати своє житло, повинен вибирати між продуктами від різних виробників. В кінцевому підсумку йому доводиться користуватися різними системами (пульти, додатки) для контролю різними пристроями. До того ж, завдання створення повноцінного інтернету речей стає дуже складною.

OpenHAB реалізує єдину шину, тобто дозволяє об'єднати всі пристрої з різними протоколами в єдину мережу. Таким чином, можна користуватися єдиним засобом управління (скажімо, додатком на смартфоні) і реалізувати як завгодно складну логіку взаємозв'язку між пристроями.

Що відрізняє проект openHAB від аналогів?

По-перше, він розвивається вже досить давно і вже зараз готовий до повноцінного використання. Зараз розробники працюють вже над другим поколінням платформи, заснованої на спеціалізованому фреймворку Eclipse SmartHome.

По-друге, це проект з відкритим вихідним кодом. А це значить, що весь код створюється програмістами з усього світу, які зацікавлені в темі інтернету речей і створення єдиної системи розумного будинку. Цей код доступний в репозиторії GitHub.

По-третє, в результаті відкритості вже зараз openHAB підтримує близько 50 різних протоколів «розумних» пристроїв. Серед них ZWave, KNX, EnOcean, системи мультимедії типу Sonos, кінотеатр XBMC, Samsung SmartTV і багато-багато інших. Це, в свою чергу, дає можливість користувачеві вибирати пристрої для свого будинку по найрізноманітнішим параметрами, практично не обмежуючись можливостями всієї платформи.

OpenHAB - це спеціальний сервер, який може працювати на будь-якому комп'ютері під керуванням будь-якої ОС (ви навіть можете використовувати RaspberryPi). Вся установка полягає в розпакуванні дистрибутива сервера та встановлення Java машини. Далі починається процес налаштування і творення.

OpenHAB - це дуже гнучкий конструктор. Заснований на технології OSGi, він дозволяє конфігурувати кожен окремий плагін (Binding) «нальоту», без перезавантаження всього серверу.

Налаштування item-ів і binding-ів

Кожен «розумний» пристрій в будинку повинний бути налаштований в openHAB як один або кілька item-ів. Вони описуються в спеціальному файлі в директорії configuration / items. Ось приклад одного такого item-а:

```
Dimmer Light_GF_Living_Table "Table" (GF_Living, Lights)
```

Тут описаний один item з ім'ям Light_GF_Living_Table і типом Dimmer. Тобто це якийсь пристрій, який може бути включено, виключено або змінити своє значення в діапазоні від 0 до 100). Як правило, Dimmer є різні світлові прилади, але також це може бути Dimmer розетка, або що-небудь ще.

Важливо розуміти, що item в конфігурації не зобов'язаний бути окремим пристроєм. Скажімо, я міг би конфігурувати Dimmer для керування тільки однією функцією будь-якого більш складного пристрою, наприклад, гучності домашнього кінотеатру або потужністю роботи конвекторів. Ось приклад використання підсвічування для управління гучністю:

```
Dimmer Volume "Volume [%.1f %%]"
```

Як бачите, openHAB дає можливість використовувати різні item-и як елементи керування для всього, що підходить під будь-які критерії (в даному випадку, щось повинно вміти включатися, вимикатися і приймати значення від 0 до 100).

Ім'я item-а дає можливість прив'язувати до нього елементи UI в конфігурації користувальницького інтерфейсу (sitemap) і звертатися до нього в коді ваших скриптів і правил.

У загальному випадку, синтаксис опису item-а такий

```
itemtype itemname ["labeltext"] [] [(group1, group2, ...)]  
[{{bindingconfig}}
```

Як бачите, спершу йде тип (Switch, Dimmer, Color, String, Number, Rollershutter, DateTime, Contact, Group), потім унікальне в рамках файлу ім'я, опціональні label (те, що буде відображено в UI), іконка (з набору openHAB), одна або кілька груп (для об'єднання пристроїв за змістом або розташуванням) і власне конфігурація самого Біндинга - в фігурних дужках.

Binding - або як зв'язати item з пристроєм

Біндинг пов'язує item з якимось конкретним пристроєм (або його функцією). Це потрібно для того, щоб openHAB перетворював абстрактні команди типу On / Off в конкретні повідомлення потрібного протоколу, а також правильно інтерпретував повідомлення від пристрою. У кожного Біндинга свій синтаксис конфігурації. У openHAB це один рядок в фігурних дужках, в якій має бути вказано все що потрібно для зв'язку з пристроєм по конкретному протоколу.

Також не забувайте, що у кожного Біндинга є набір загальних параметрів, які потрібно вказати в файлі configuratnios/openhab.cfg.

Користувальницький інтерфейс

Тут openHAB пропонує досить цікавий спосіб побудови UI "на льоту" за допомогою файлів опису sitemap. В директорії configuration/sitemaps лежать файли, в яких потрібно описати те, як повинен виглядати користувальницький інтерфейс (а точніше, його layout) для керування тими пристроями, які ми описали в items. Ось простий приклад

```
sitemap demo label="Main Menu"
{
  Frame label="Percent-based Widgets" {
    Slider item=DimmedLight switchSupport
    Colorpicker item=RGBLight icon="slider"
    Switch item=DemoShutter
    Slider item=DemoBlinds
  }
}
```

Тут UI складається тільки з одного "фрейму" (не плутати з фреймами в термінах HTML), в якому є 4 елементи (два слайдера для управління диммерами, вибір кольору для підсвічування, простий перемикач). Список всіх можливих елементів інтерфейса можна знайти на сторінці Sitemaps.

MQTT (Message Queue Telemetry Transport) — спрощений мережевий протокол, що працює на TCP/IP. Використовується для обміну повідомленнями між пристроями за принципом видавець-передплатник.

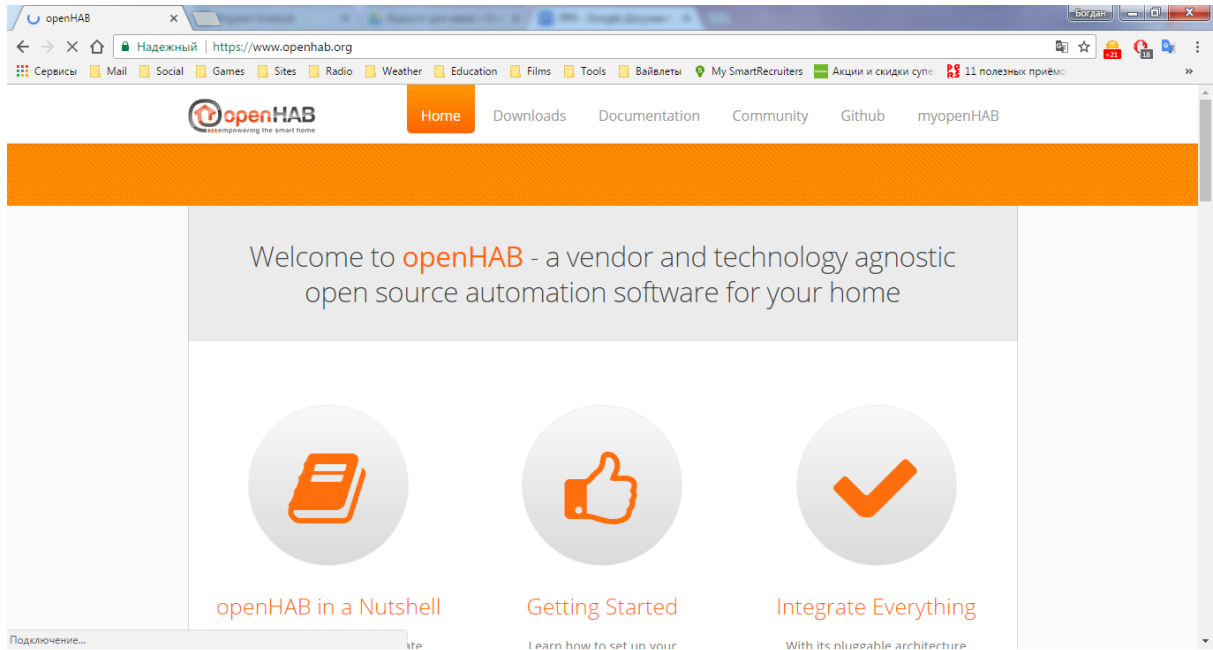
Можливості:

- Простий у використанні. Протокол є програмним блоком без зайвої функціональності, що може бути легко вбудований в будь-яку складну систему;
- Зручний для більшості рішень з датчиками. Дає можливість пристроям виходити на зв'язок і публікувати повідомлення, які не були заздалегідь відомі або визначені;
- Легкий у адмініструванні;
- Низьке навантаження на канал зв'язку;
- Робота в умовах постійної втрати зв'язку або інших проблем на лінії;

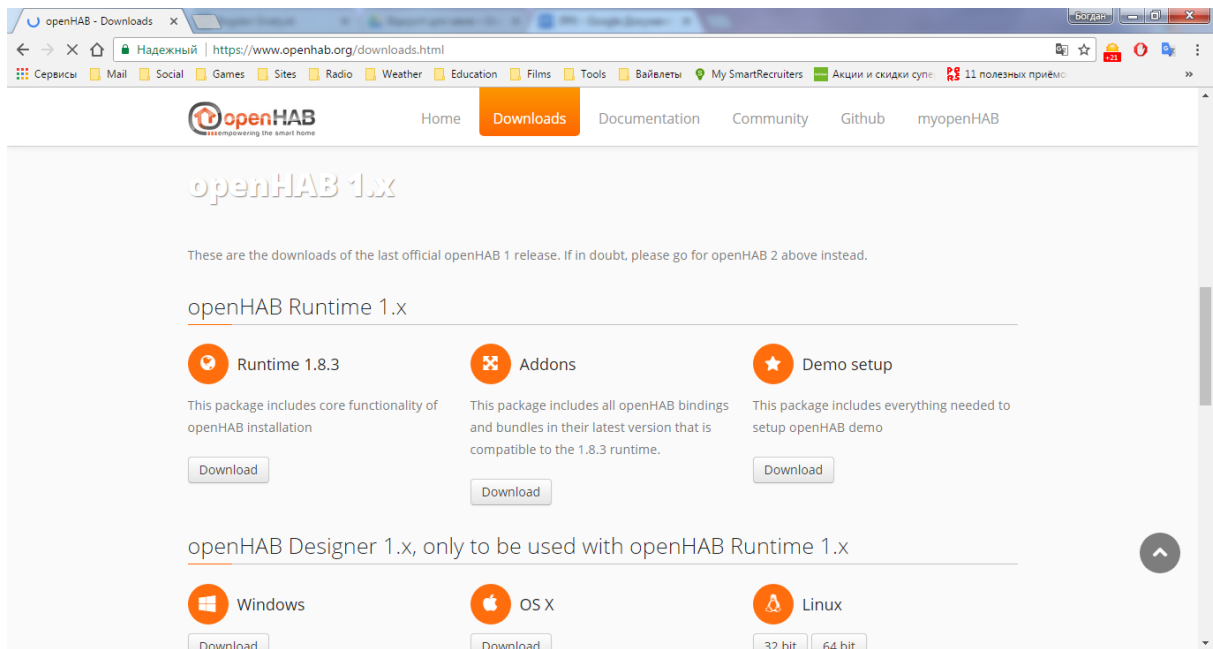
- Немає обмежень на формат переданого контенту.

I. Встановлення OpenHAB версії 1.8.3.

1. Перейдіть по силці: <https://www.openhab.org/>


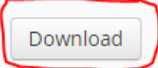

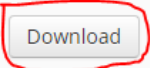


2. Відкрийте вкладку Download.

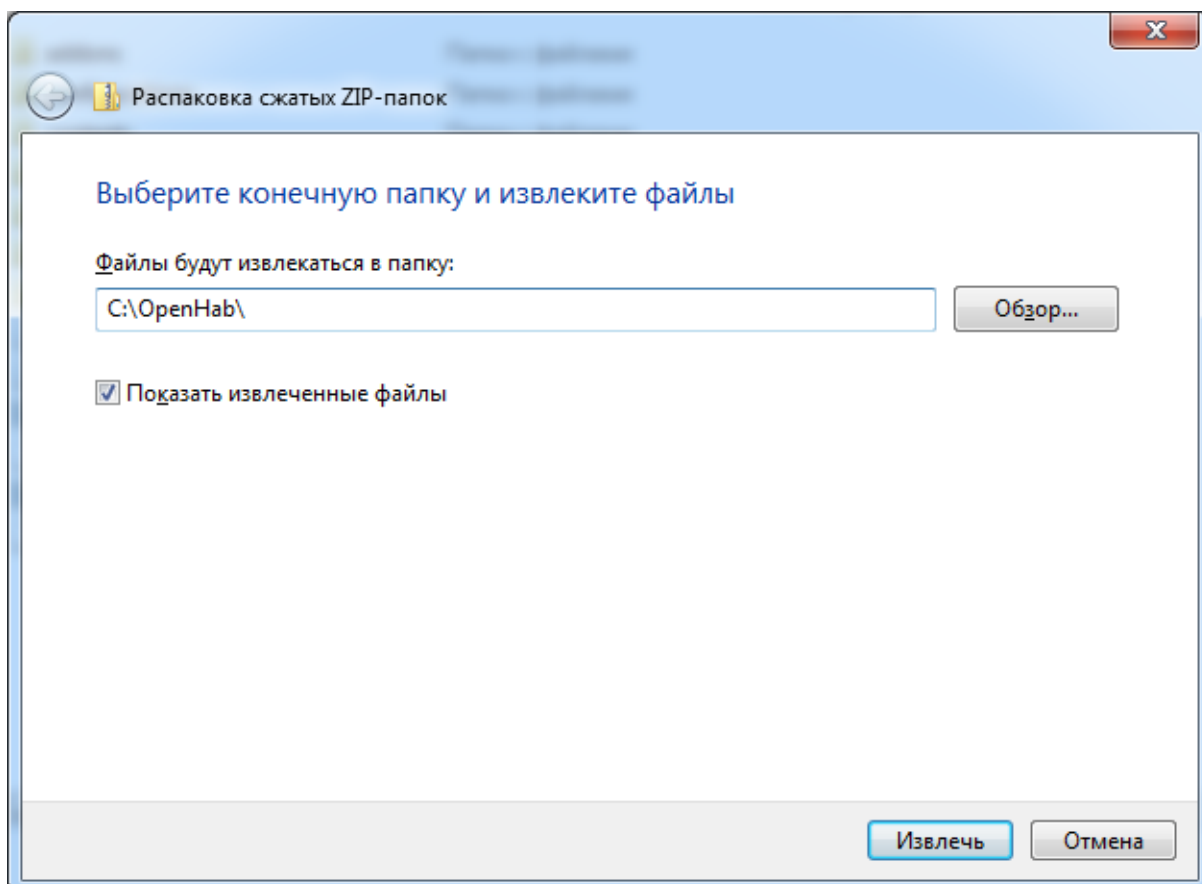


3. Скачайте **Runtime 1.8.3, Addons**.

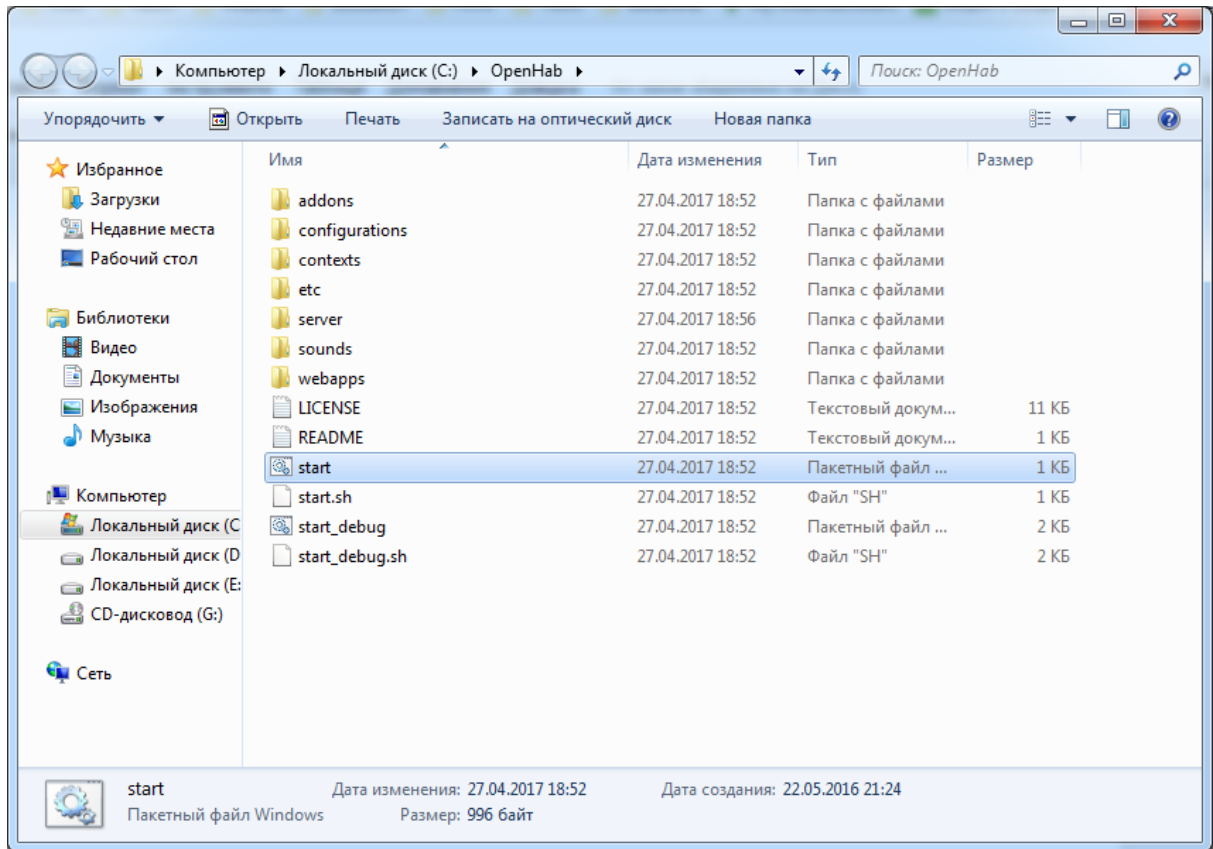
openHAB Runtime 1.x

 Runtime 1.8.3 This package includes core functionality of openHAB installation 	 Addons This package includes all openHAB bindings and bundles in their latest version that is compatible to the 1.8.3 runtime. 
---	---

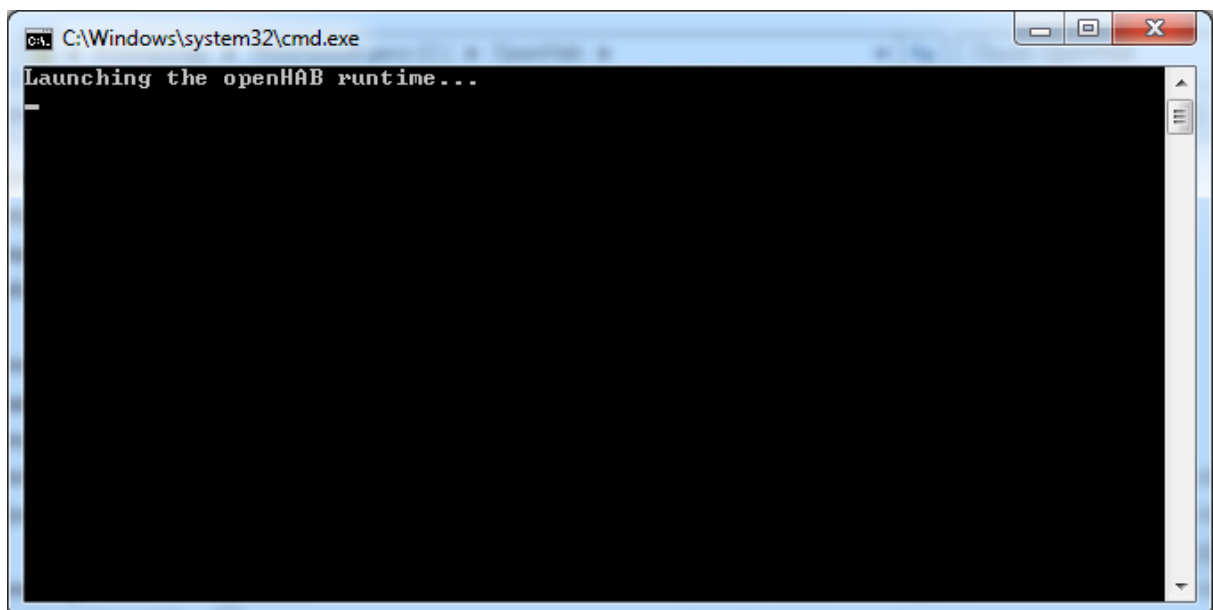
4. Розархівуйте файл **Runtime 1.8.3** в корінь системного диску.



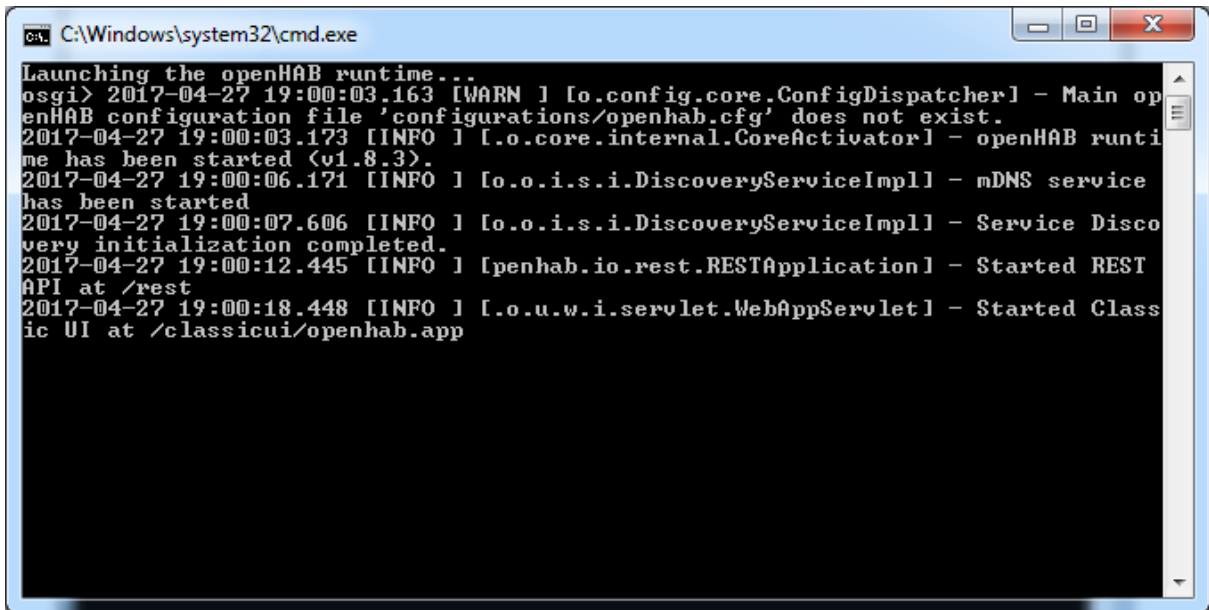
5. Відкрийте папку, в яку ви розархівували **Runtime 1.8.3** та запустіть **start.bat**.



Ви маєте побачити наступне:

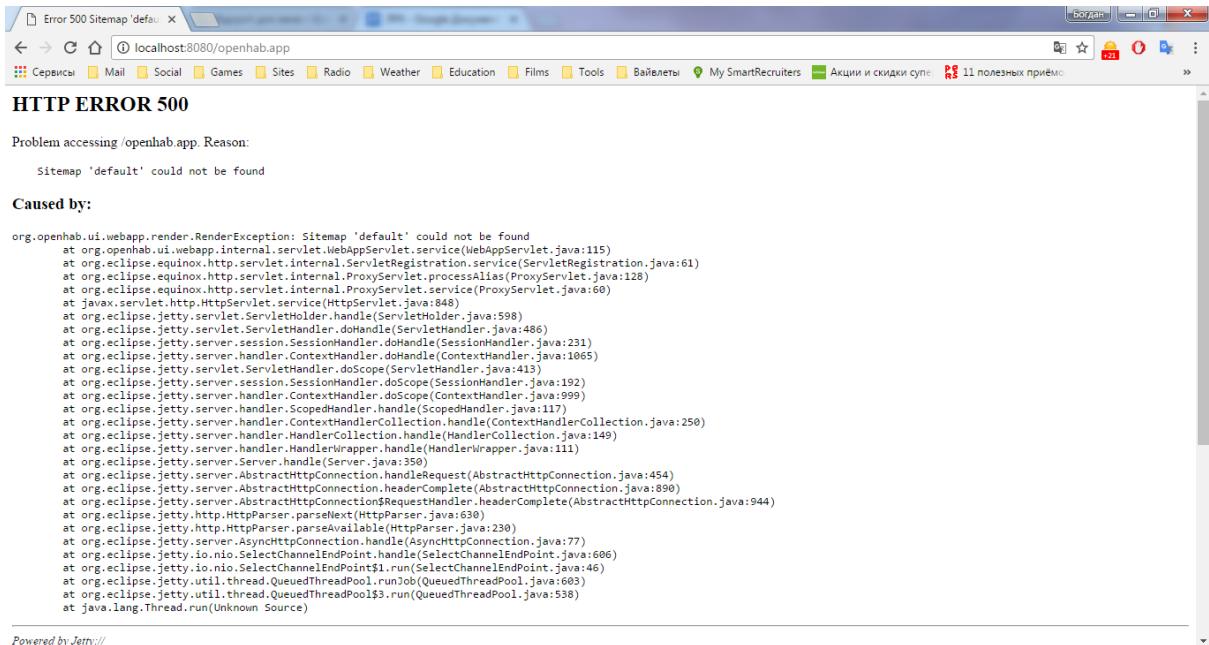


Коли запуститься OpenHab консоль матиме такий вигляд:



```
C:\Windows\system32\cmd.exe
Launching the openHAB runtime...
osgi> 2017-04-27 19:00:03.163 [WARN ] [o.config.core.ConfigDispatcher] - Main op
enHAB configuration file 'configurations/openhab.cfg' does not exist.
2017-04-27 19:00:03.173 [INFO ] [o.core.internal.CoreActivator] - openHAB runti
me has been started (v1.8.3).
2017-04-27 19:00:06.171 [INFO ] [o.o.i.s.i.DiscoveryServiceImpl] - mDNS service
has been started
2017-04-27 19:00:07.606 [INFO ] [o.o.i.s.i.DiscoveryServiceImpl] - Service Disco
very initialization completed.
2017-04-27 19:00:12.445 [INFO ] [penhab.io.rest.RESTApplication] - Started REST
API at /rest
2017-04-27 19:00:18.448 [INFO ] [o.u.w.i.servlet.WebAppServlet] - Started Class
ic UI at /classicui/openhab.app
```

6. Щоб перевірити роботу OpenHab відкрийте будь-який браузер та перейдіть за посиланням **localhost:8080**

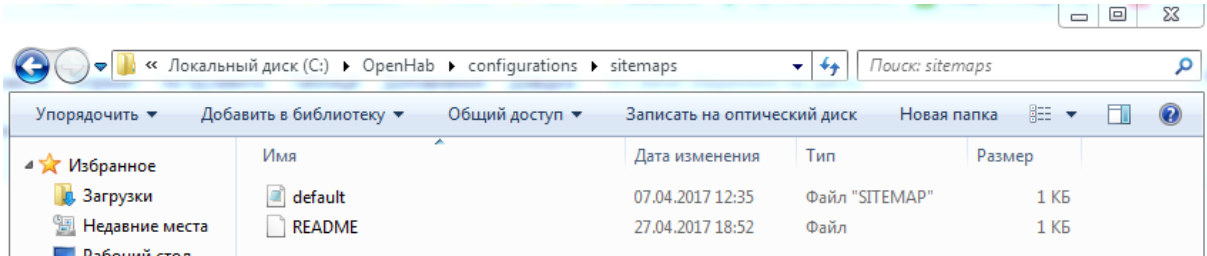


```
Error 500 Sitemap 'default'
localhost:8080/openhab.app
Сервисы Mail Social Games Sites Radio Weather Education Films Tools Вайфай My SmartRecruiters Акции и скидки супер 11 полезных приемс
HTTP ERROR 500
Problem accessing /openhab.app. Reason:
Sitemap 'default' could not be found
Caused by:
org.openhab.ui.webapp.render.RenderException: Sitemap 'default' could not be found
at org.openhab.ui.webapp.internal.servlet.WebAppServlet.service(WebAppServlet.java:115)
at org.eclipse.equinox.http.servlet.internal.ServletRegistration.service(ServletRegistration.java:61)
at org.eclipse.equinox.http.servlet.internal.ProxyServlet.processAlias(ProxyServlet.java:128)
at org.eclipse.equinox.http.servlet.internal.ProxyServlet.service(ProxyServlet.java:68)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:848)
at org.eclipse.jetty.servlet.ServletHolder.handle(ServletHolder.java:598)
at org.eclipse.jetty.servlet.ServletHandler.doHandle(ServletHandler.java:486)
at org.eclipse.jetty.server.session.SessionHandler.doHandle(SessionHandler.java:231)
at org.eclipse.jetty.server.handler.ContextHandler.doHandle(ContextHandler.java:1065)
at org.eclipse.jetty.servlet.ServletHandler.doScope(ServletHandler.java:413)
at org.eclipse.jetty.server.session.SessionHandler.doScope(SessionHandler.java:192)
at org.eclipse.jetty.server.handler.ContextHandler.doScope(ContextHandler.java:999)
at org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:117)
at org.eclipse.jetty.server.handler.ContextHandlerCollection.handle(ContextHandlerCollection.java:250)
at org.eclipse.jetty.server.handler.HandlerCollection.handle(HandlerCollection.java:149)
at org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java:111)
at org.eclipse.jetty.server.Server.handle(Server.java:350)
at org.eclipse.jetty.server.AbstractHttpConnection.handleRequest(AbstractHttpConnection.java:454)
at org.eclipse.jetty.server.AbstractHttpConnection.headerComplete(AbstractHttpConnection.java:890)
at org.eclipse.jetty.server.AbstractHttpConnection$RequestHandler.headerComplete(AbstractHttpConnection.java:944)
at org.eclipse.jetty.http.HttpParser.parseNext(HttpParser.java:630)
at org.eclipse.jetty.http.HttpParser.parseAvailable(HttpParser.java:230)
at org.eclipse.jetty.server.AsyncHttpConnection.handle(AsyncHttpConnection.java:77)
at org.eclipse.jetty.io.nio.SelectChannelEndPoint.handle(SelectChannelEndPoint.java:686)
at org.eclipse.jetty.io.nio.SelectChannelEndPoint$1.run(SelectChannelEndPoint.java:46)
at org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.java:683)
at org.eclipse.jetty.util.thread.QueuedThreadPool$3.run(QueuedThreadPool.java:538)
at java.lang.Thread.run(Unknown Source)
Powered by Jetty://
```

Наш OpenHab встановлено, але для коректної роботи нам потрібно його налаштувати.

7. Створення Sitemap.

Перейдіть в папку з встановленим OpenHab, далі в configurations->sitemaps та створіть файл txt з назвою default.sitemap.

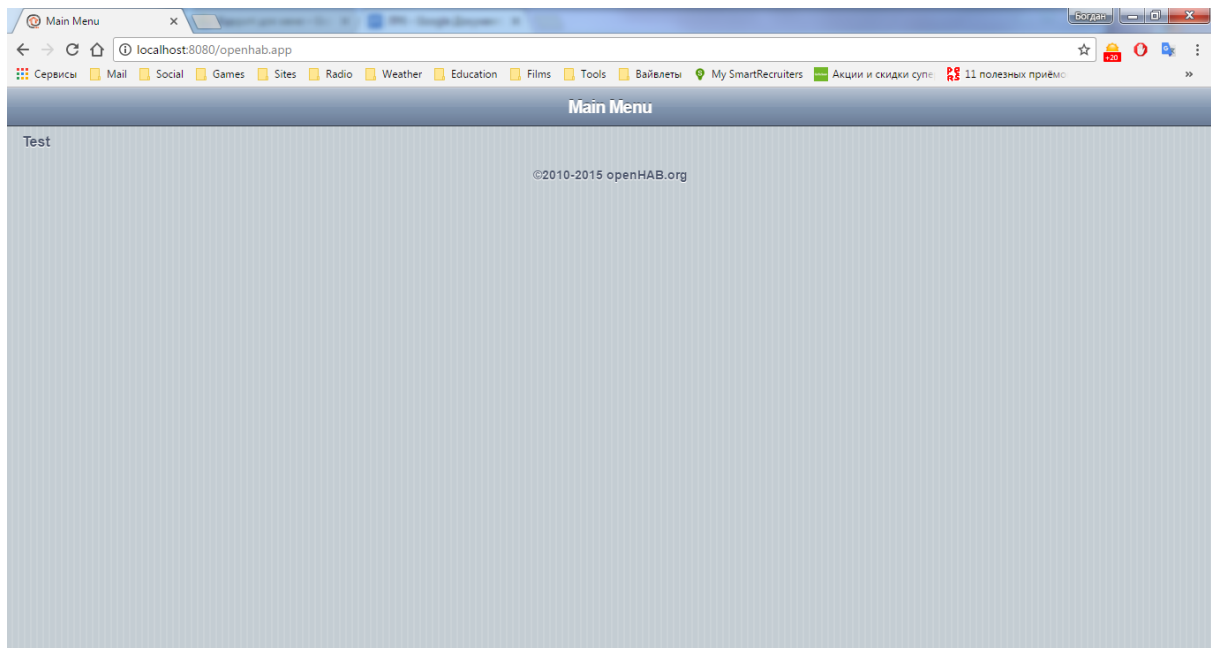


Увага: sitemap це розширення файлу.

Відкрийте створений файл за допомогою блокноту та напишіть наступне:

```
sitemap                default                label="Main                Menu"
{
    Frame                label="MQTT                Test"                {
        Text                item=TestTemperature
    }
}
```

Збережіть файл та закрийте його. Після цього знову запустіть OpenHab як описано в пункті 5-6. Коли ви перейдете за посиланням **localhost:8080** маєте побачити наступне:



II. Встановлення MQTT broker **mosquitto**.

1. Для цього переходимо на сайт <https://mosquitto.org/> та відкриваємо вкладку **Downloads**.

2. Скачуємо файл **mosquitto-1.4.11-install-win32.exe (~200 kB) (Native build, Windows Vista and up, built with Visual Studio Community 2013)**

Binary Installation

Windows

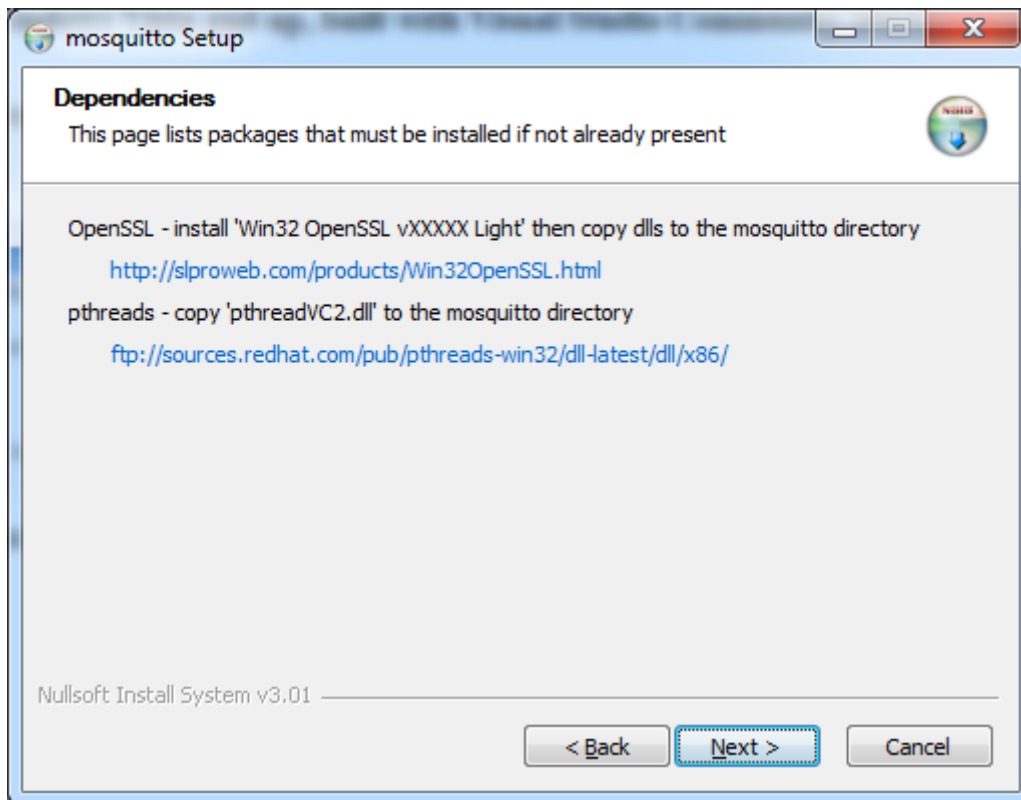
- **mosquitto-1.4.11-install-win32.exe (~200 kB) (Native build, Windows Vista and up, built with Visual Studio Community 2013)**
- [mosquitto-1.4.11-install-cygwin.exe \(~200 kB\)](#) (Cygwin build, Windows XP and up)

See the `readme-windows.txt` after installing for Windows specific details and dependencies.

Mac

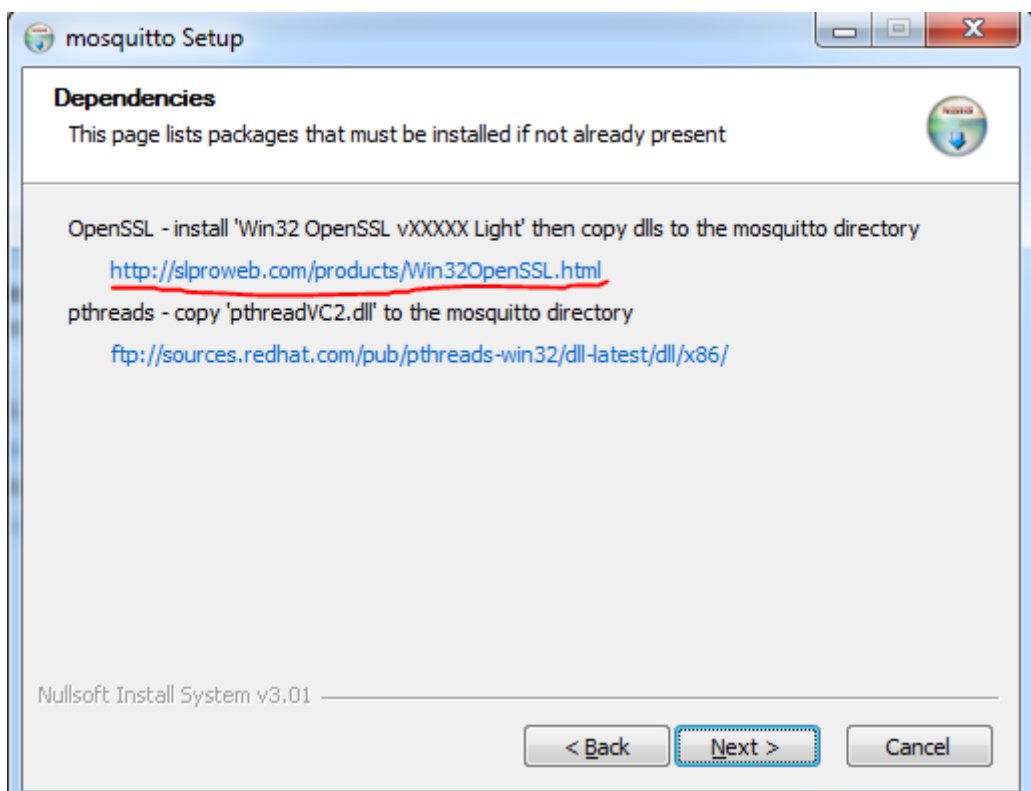
Mosquitto can be installed from the homebrew project. See <http://brew.sh/> and then use "brew install mosquitto"

3. Запускаємо скачений нами інсталтор, натискаємо Next->.



Для правильної роботи **mosquitto** потрібно скачати додатково OpenSSL що покане в вікні інсталлятора.

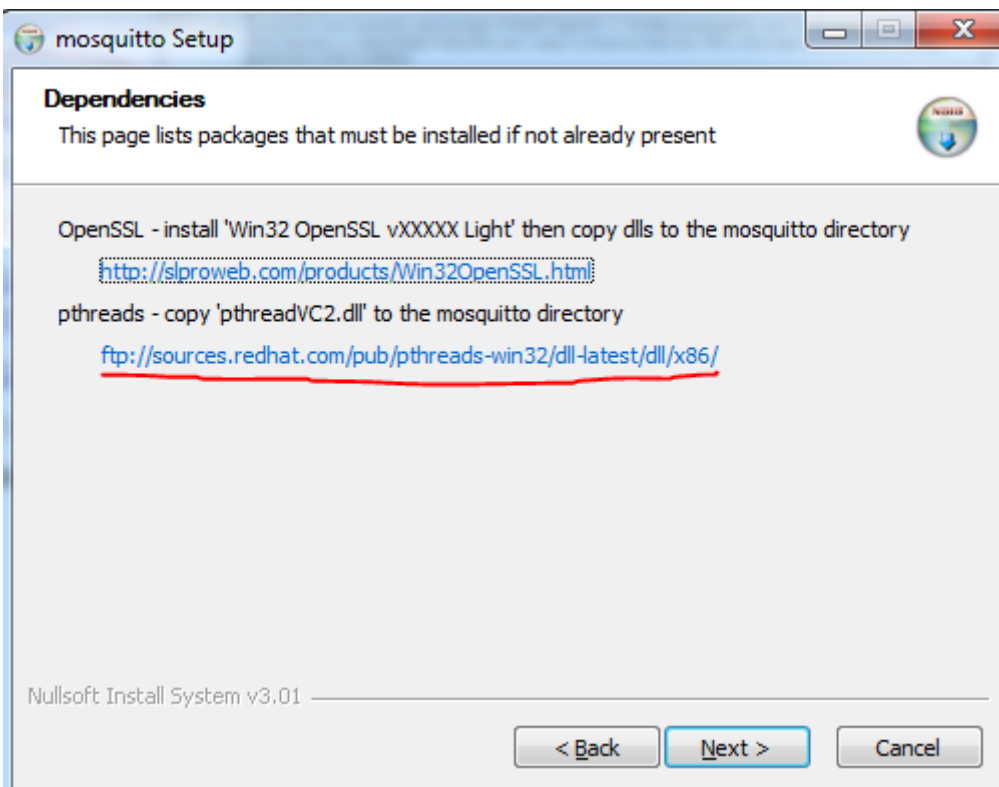
Для завантаження перейдіть по ссилці:



Та скачайте файл Win32 OpenSSL v1.1.0e.

Download Win32 OpenSSL		
File	Type	Description
Win32 OpenSSL v1.1.0e Light	3MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v1.1.0e (Recommended for users by the creators of OpenSSL). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.1.0e	30MB Installer	Installs Win32 OpenSSL v1.1.0e (Recommended for software developers by the creators of OpenSSL). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.1.0e Light	3MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v1.1.0e (Only install this if you need 64-bit OpenSSL for Windows. Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.1.0e	33MB Installer	Installs Win64 OpenSSL v1.1.0e (Only install this if you are a software developer needing 64-bit OpenSSL for Windows. Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.0.2k Light	2MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v1.0.2k (Recommended for users by the creators of OpenSSL). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.0.2k	20MB Installer	Installs Win32 OpenSSL v1.0.2k (Recommended for software developers by the creators of OpenSSL). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.0.2k Light	3MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v1.0.2k (Only install this if you need 64-bit OpenSSL for Windows. Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.0.2k	23MB Installer	Installs Win64 OpenSSL v1.0.2k (Only install this if you are a software developer needing 64-bit OpenSSL for Windows. Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

Після цього перейдіть по другій посилці в вікні інсталятора **mosquitto**.

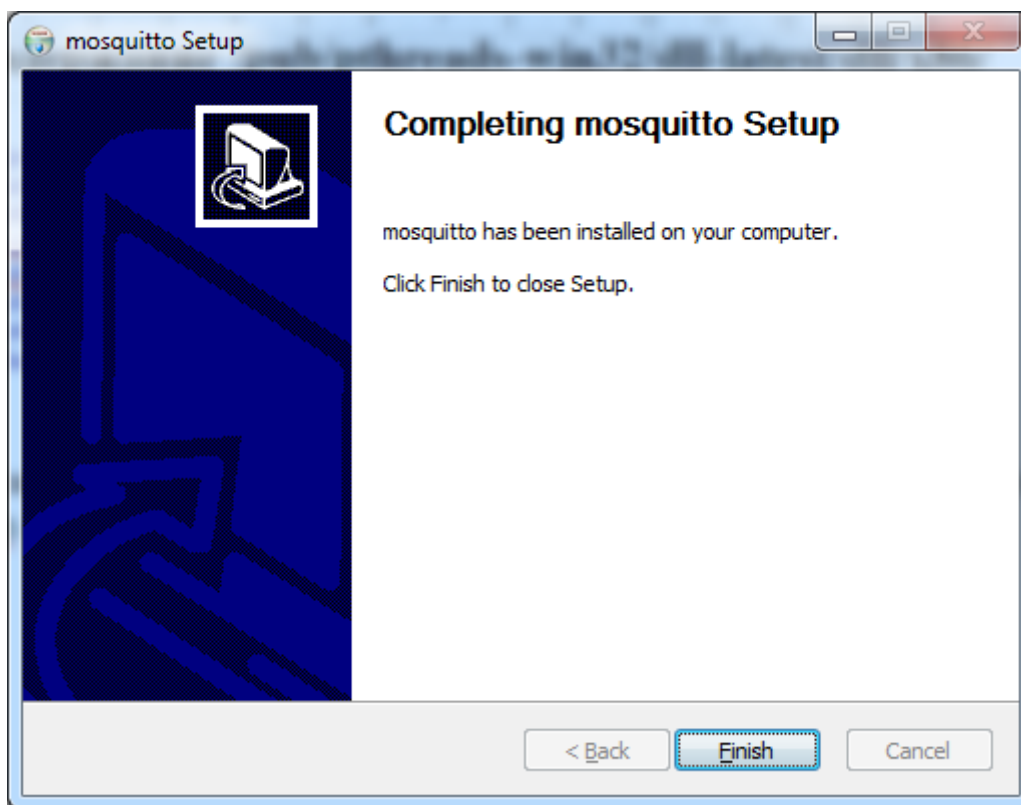


Та скачайте файл pthreadVC2.dll просто натиснувши на нього.

Содержание /pub/pthreads-win32/dll-latest/dll/x86/

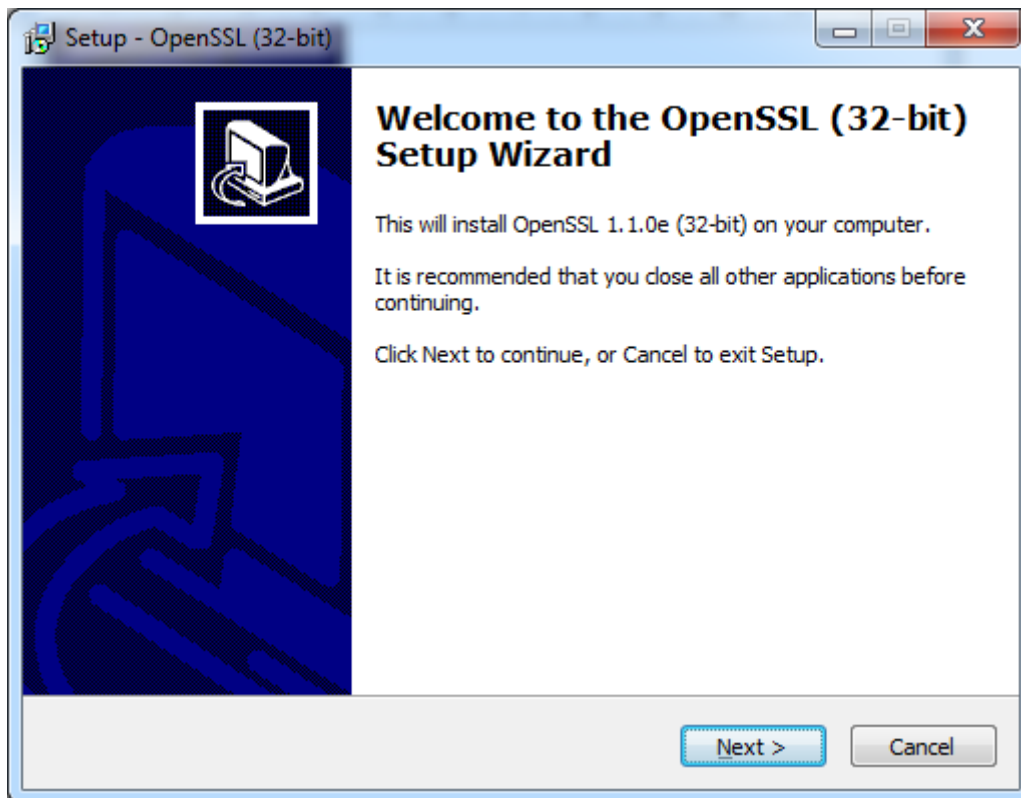
Название	Размер	Последнее изменение
[родительский каталог]		
md5.sum	293 B	05.02.2015, 2:00:00
pthreadGC2.dll	117 kB	27.05.2012, 3:00:00
pthreadGCE2.dll	119 kB	27.05.2012, 3:00:00
pthreadVC2.dll	54.5 kB	27.05.2012, 3:00:00
pthreadVCE2.dll	60.5 kB	27.05.2012, 3:00:00
pthreadVSE2.dll	56.0 kB	27.05.2012, 3:00:00
sha512.sum	866 B	05.02.2015, 2:00:00

Після завантаження цих файлів в вікні інсталятора **mosquitto** натисніть Next-
>. нічого не змінюючи продовжуємо інсталяцію до самого кінця.

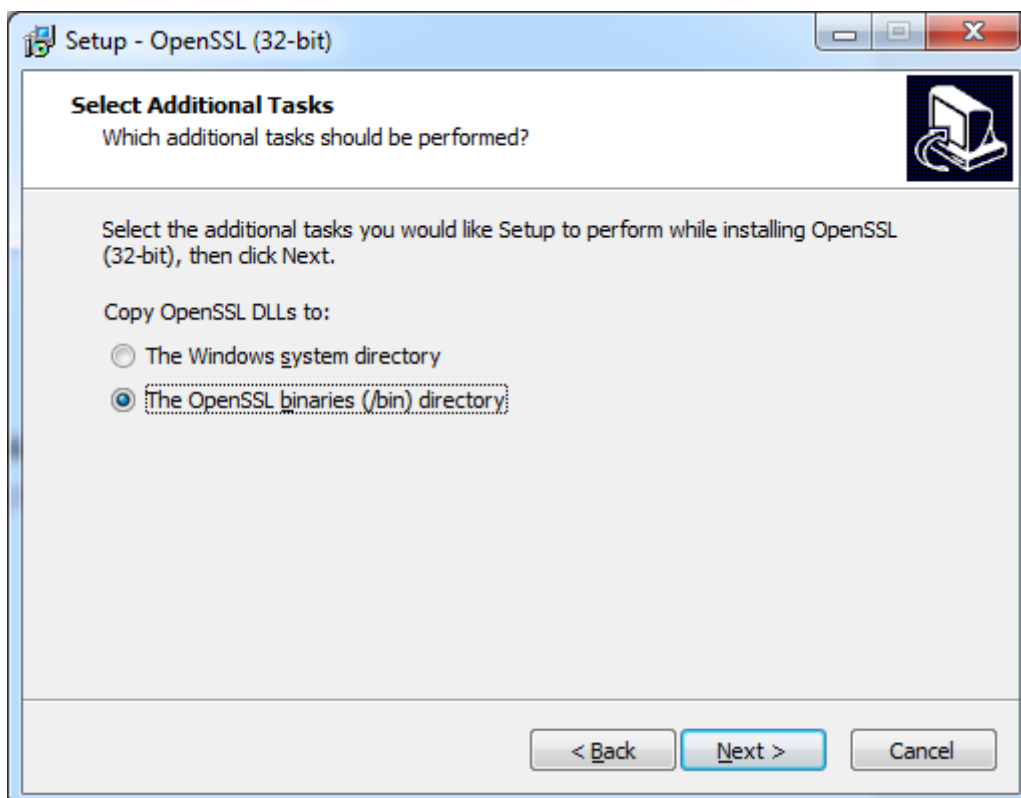


4. Встановлення OpenSSL.

Для цього відкрийте раніше завантажений інсталятор OpenSSL.



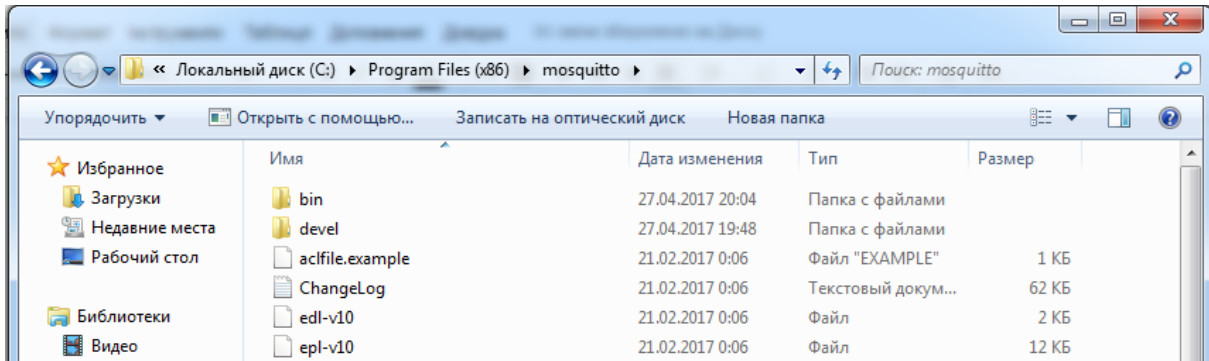
В вкладці інсталятора **Select Additional Tasks** оберіть **The OpenSSL binaries (/bin) directory**, як показано нижче.



Та продовжте інсталяцію до кінця.

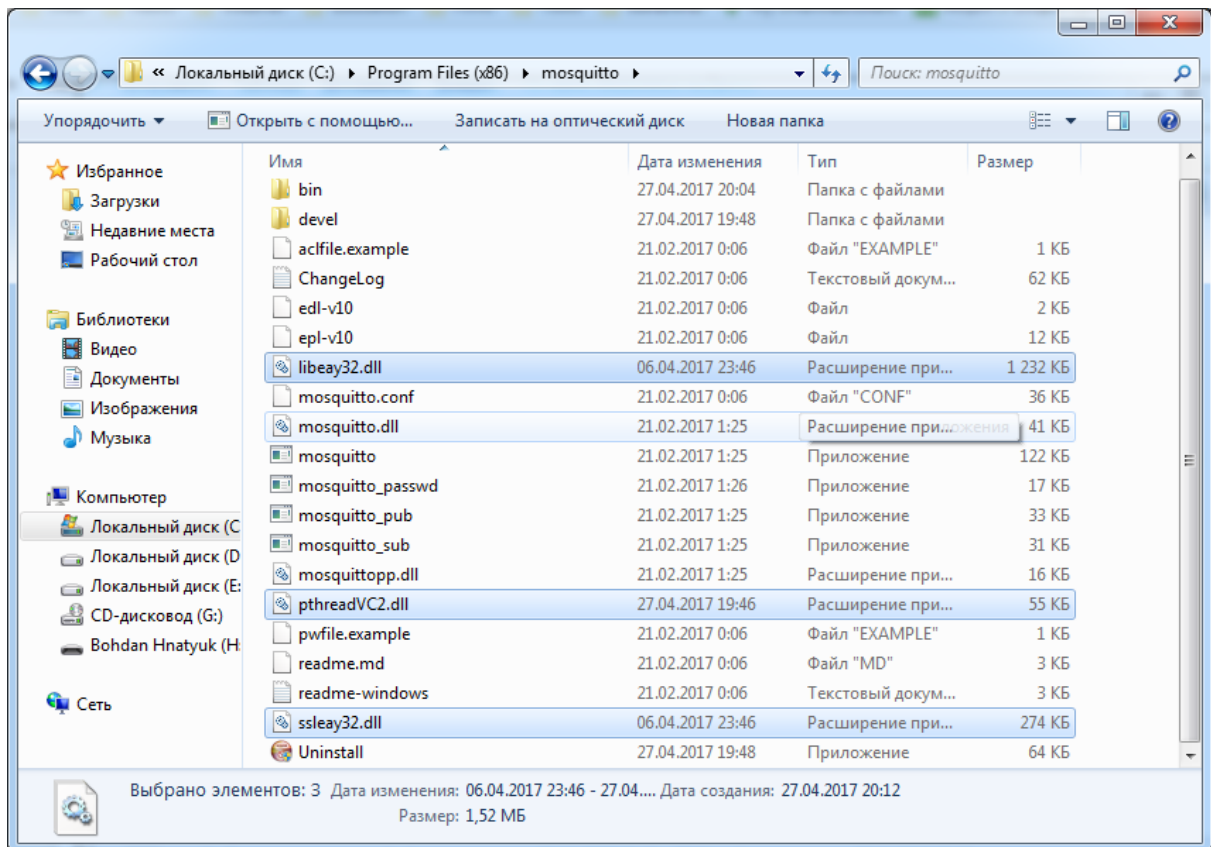
5. Компановка бібліотек.

Відкриваємо папку з встановленим OpenSSL (C:\OpenSSL-Win32) та копіюємо папку bin в папку з встановленим mosquitto (C:\Program Files (x86)\mosquitto)



\Скачуємо бібліотеки libeay32.dll, ssleay32.dll з сайту <https://ru.dll-files.com/>.

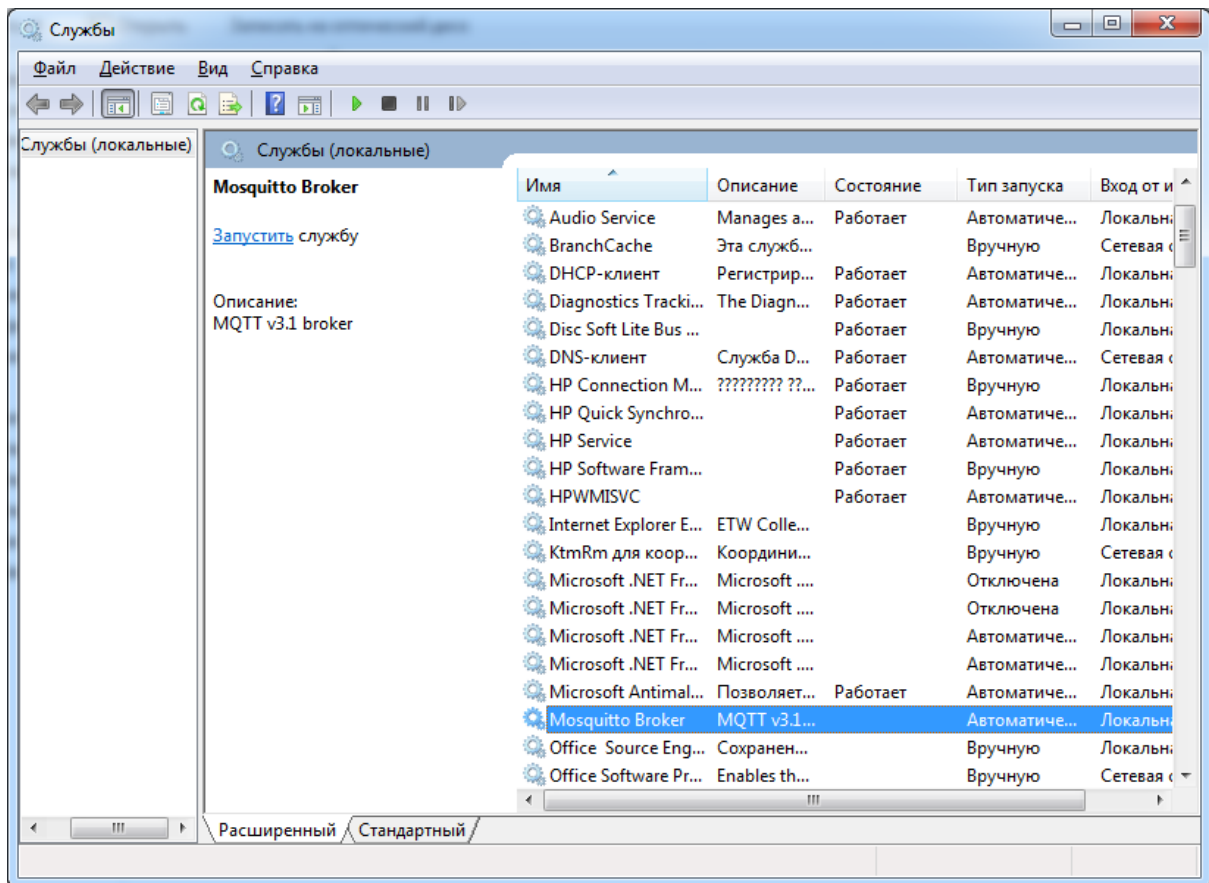
Та переміщуємо всі скачані бібліотеки(libeay32.dll, ssleay32.dll, pthreadVC2.dll) в корінь папки mosquitto (C:\Program Files (x86)\mosquitto) .



Перевстановлюємо **mosquitto**, в вікні інсталятора **mosquitto** натисніть Next->. нічого не змінюючи продовжуємо інсталяцію до самого кінця.

6. Запуск mosquitto.

Заходимо в панель керування->Адміністрування->Служби. Знаходимо службу **Mosquitto Broker** та запускаємо її.



III. Налаштування OpenHab для роботи з Mosquitto Broker.

1. Розархівуємо файл **openhav-1.9.0-addons**. Знаходимо там файл **org.openhab.binding.mqtt-1.9.0** та копіюємо його в папку **C:\OpenHab\addons**.

2. Налаштування OpenHab для роботи з MQTT.

Відкриваємо в блокноті файл **openhav_default** який знаходиться в папці **configurations** за адресою **C:\OpenHab\configurations**.

Шукаємо заголовок MQTT Transport та нижче вставляємо наступні команди:

mqtt:mymosquitto.url=tcp://localhost:1883

mqtt:mymosquitto.retain=true

Зберігаємо та закриваємо.

3. Заходимо в папку **C:\OpenHab\configurations\items** та створюємо default.items за аналогією default.sitemap. Відкриваємо файл в блокноті та вставляємо наступне:

Group All

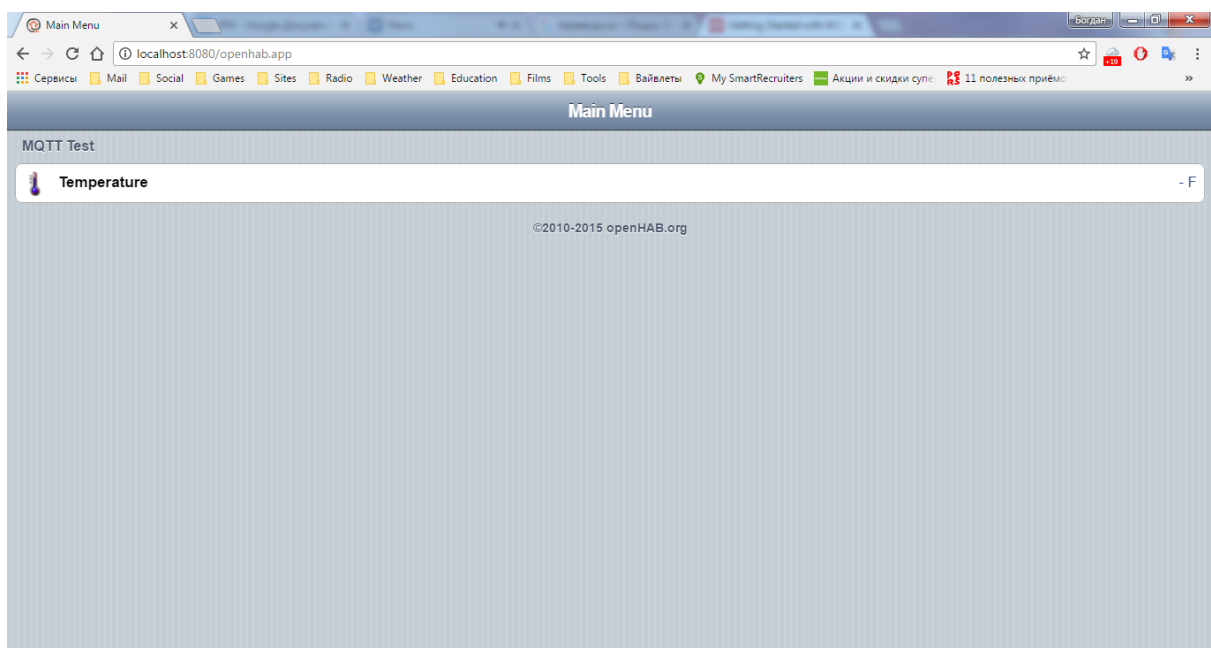
Group gGroundFloor (All)

Group GF_Living "Living Room" <video> (gGroundFloor)

Number TestTemperature "Temperature [%0.1f F]" <temperature> (GF_Living) {mqtt="<[mymosquitto:home/temperature:state:default]"}

4. Запускаємо OpenHab.

Вікно має виглядати як на картинці нижче



5. Надсилання команди через MQTT

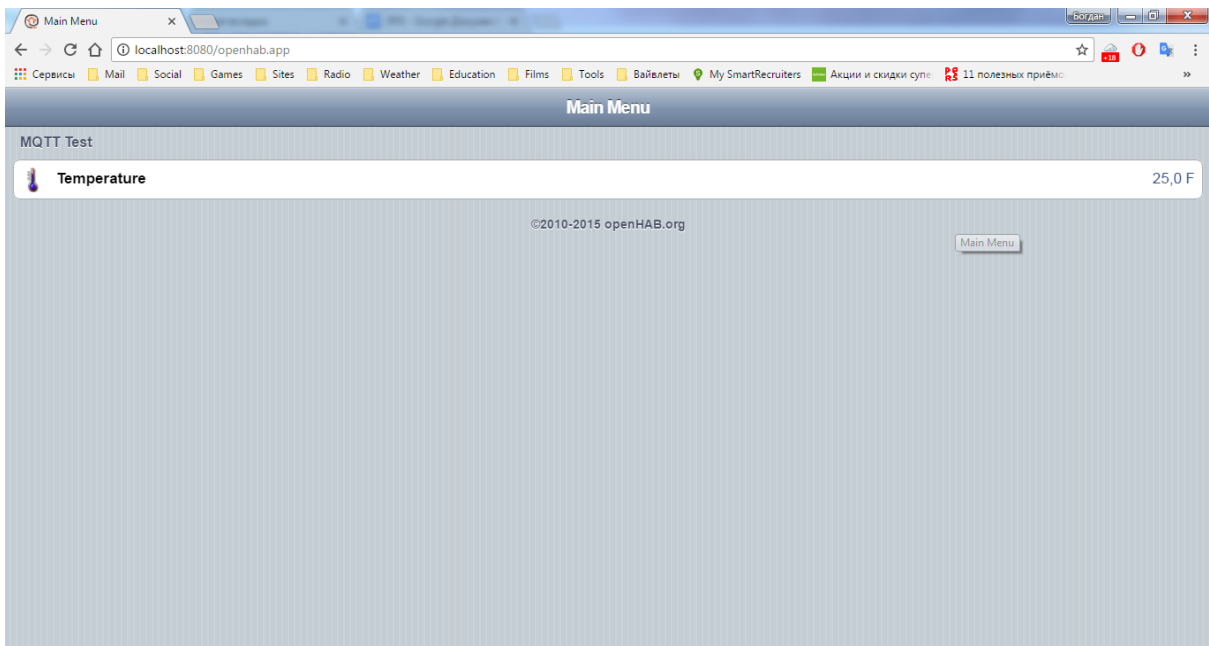
Відкрийте “Командну строку” та введіть наступну команду:

cd C:\Program Files (x86)\mosquitto

```
Администратор: Командная строка
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.
C:\Windows\system32>cd C:\Program Files (x86)\mosquitto
C:\Program Files (x86)\mosquitto>_
```

Для відправки повідомлень потрібно ввести наступну команду:

mosquitto_pub -t "home/temperature" -m "25" -q 1 -r



Контрольні запитання

1. Для чого потрібен OpenHAB?
2. Що таке sitemap, item?
3. Що таке binding?

4. Для чого потрібен MQTT broker та як його налаштувати на прикладі Mosquitto?
5. Назвіть основні можливості MQTT broker-ів.
6. Створіть новий sitemap, на якому буде мінімум два фрейма з мінімум двома різними елементами та створіть до них відповідні Item-ми.

Список рекомендованої літератури

1. Протокол MQTT і відкритий проект клієнта MQTT на Delphi. Geektimes. [Електронний ресурс].- Режим доступу: <https://geektimes.ru/post/268018/>
2. OpenHAB — стань программістом собственного жилища [Електронний ресурс].- Режим доступу: <https://habrahabr.ru/post/232969/>
3. OpenHAB "A vendor and technology agnostic open source automation software for your home" [Електронний ресурс].- Режим доступу: <http://www.homeautomationforgeeks.com/project/openhab.shtml>
4. Datasheet ESP-8266 [Електронний ресурс].- Режим доступу: <http://download.arduino.org/products/UNOWIFI/0A-ESP8266-Datasheet-EN-v4.3.pdf>
5. OpenHab [Електронний ресурс].- Режим доступу:<https://www.openhab.org/>