МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАСИЛЯ СТУСА

В. Г. Крижановський, С. П. Сергієнко

До практичної реалізації пристроїв Інтернету речей (IoT)

Навчально-методичний посібник

Вінниця
ДонНУ імені Василя Стуса
2019

УДК 004.77(075.8)
К 822

Рекомендовано до друку вченою радою фізико-технічного факультету (протокол № 2 від 21 жовтня 2019 р.)

Автори:
В. Г. Крижановський, проф. кафедри радіофізики та кібербезпеки;
С. П. Сергієнко, доц. кафедри радіофізики та кібербезпеки

Рецензент: *П. К. Ніколюк,* д-р фіз.-мат. наук, проф., проф. кафедри комп'ютерних технологій.

К 822
До практичної реалізації пристроїв Інтернету речей (IoT): навчально-методичний посібник / В. Г. Крижановський, С. П. Сергієнко. Вінниця: ДонНУ імені Василя Стуса, 2019. 61 с.

Розглядаються особливості побудови реалізованих систем Інтернету речей (IoT), починаючи з простих прикладів, які можливо відтворити навіть у домашніх умовах.

Посібник рекомендовано для студентів вищих навчальних закладів за напрямами «Прикладна фізика. Технології Інтернету речей», «Кібербезпека» та «Ком'пютерні науки та інформаційні технології. Інтелектуальні інформаційні технології».

УДК 004.77(075.8)

ВСТУП

Інтернет речей (Internet of Things, IoT) – концепція, яка зародилася з різних джерел, а зараз широкому загалу найбільш відома у зв'язку з поняттям «Розумний будинок». Фахівці пов'язують виникнення «Інтернету речей» з розвитком сенсорних мереж (мереж давачів) та з різновидами автоматизованих систем управління технологічними процесами. Спільною у цих галузях є потреба забезпечення гнучкого та економічного (з малим споживанням енергії) процесу передачі інформації з невизначеного кола джерел, який до того ж має бути переважно бездротовим. Зрозуміло, що Інтернет як глобальна мережа стає зразком для створення розподіленої, гнучкої, надійної та побудованої з однотипних елементів системи.

1. ЗАГАЛЬНІ ВИМОГИ ДО АПАРАТНОЇ ЧАСТИНИ ІНТЕРНЕТУ РЕЧЕЙ

1.1. Галузі впровадження та оцінки поширення IoT

Інтернет речей (IoT) – це прикладний простір, що швидко розвивається, готовий стати найбільшим ринком електроніки для напівпровідникової галузі. Пристрої IoT орієнтовані на зондування та приведення в дію нашого фізичного середовища та мають майже необмежену сферу використання. Визначено дві загальні проблеми, які існують у цьому просторі: *досягнення працювати за наднизької потужності споживання* та *проектування системи за допомогою модульних, універсальних компонентів* [1, 2].

Пристрої IoT унікальні тим, що вони зосереджені на фізичних інтерфейсах, що дозволяє їм відчувати та актуалізувати навколишній світ, на відміну від попередніх обчислювальних платформ, таких як настільні та портативні пристрої, які в основному зосереджені на людських інтерфейсах. Ці людські інтерфейси вимагають відносно великих пристроїв введення та виведення та потребують розміщення пристроїв разом із своїми користувачами. На відміну від цього, пристрої IoT отримують вхідні дані за допомогою датчиків, які часто значно мініатюризовані за допомогою технології MEMS, і надсилають інформацію через дротовий або бездротовий інтерфейси на мобільні та хмарні комп'ютери. Це дає можливість розміщувати пристрої IoT в нових місцях та додатках, де раніше не було обчислень.

У роботі [2] наведені приклади схем з наднизькими рівнями споживання енергії, які будуть сприяти використанню пристроїв IoT. Завдяки змінам фізичних параметрів транзисторів вдалося зменшити

напругу та струм живлення схеми і зменшити температурну залежність вихідної напруги за струму живлення близько 10 пА.

Іншим важливим елементом у більшості пристроїв IoT є таймери, що постійно вмикаються і повинні мати наднизьку потужність. Розроблено кварцовий генератор зі споживаною потужністю 5,5 нВт, що означає п'ятикратне зниження, порівняно з попередніми конструкціями. Він знижує енергоспоживання за рахунок зменшення амплітуди коливань (зазвичай ~1 В) до лише ~150 мВ. Щоб отримати сильне коливання, вихідний каскад приводиться в дію сигналом розгойдування 750 мВ, який генерується підсиленням вихідної напруги вузла генератора. Додаткове зменшення потужності відбувається за рахунок того, що вихідний каскад активується тільки протягом короткого періоду, коли напруга вводу генератора досягає своїх мінімальних / максимальних значень, тим самим зменшуючи напругу на перемикачі драйвера, а отже, і втрати потужності.

Пристрої IoT часто потребують отримання енергії в різних енергетичних умовах, що вимагає перетворення напруги з рівнями вхідної потужності, котрі значно різняться між собою. Крім того, фактори форми пристроїв IoT дозволяють повністю інтегрувати перетворювач DC-DC на мікросхему. Було розроблено повністю інтегрований подвоювач напруги, який має більш ніж 70 % ефективності в надзвичайно широкому діапазоні струму від 1 nA до 300 uA. Створивши генератор з конденсаторами, що перемикаються, вдалося уникнути спеціального генератора та регуляторів рівня, що підвищує ефективність. Для систем збирання енергії з оточуючого середовища (харвестінг) подвоювач напруги самозапускається від вхідної напруги 140 мВ і може заряджати 4 В-ий акумулятор від сонячної батареї 1 мм^2 за слабкого освітлення в приміщенні.

У зв'язку з великою різноманітністю застосувань IoT, необхідно створити системи з використанням декількох сумісних мікросхем, які з'єднані між собою за допомогою простої шини зв'язку. Існуючі стандарти вимагають або великого розміру чипу через велику кількість проводів (SPI), або великої потужності через використання підтягуючих резисторів (I^{2}C). Отож, було запропоновано новий протокол шини під назвою MBus, який вимагає лише двох односпрямованих проводів, з'єднаних у ланцюгову топологію. Протокол шини знижує активну потужність, усуваючи потребу в тактовому генераторі на кожному підлеглому шарі, і використовує широке регулювання потужності для зменшення споживання енергії у режимі спокою.

Використовуючи вищезазначені методи малої потужності та Mbus-медіа-комунікації, побудували триміліметрові сенсорні вузли

[2]. Датчики засновані на наборі мікросхем з 8 кристалів, які містять вимірювач температури, датчик зображення, датчик тиску, процесор і пам'ять, управління живленням, пристрій прийняття енергії з простору, радіостанції ближнього та дальнього радіуса дії, а також кілька розмірів акумуляторів та сонячних батарей. Кожен датчик використовує підмножину кристалів, з'єднаних стопкою зі спілкуванням через Mbus.

1.2. Сучасні мікросхеми для Інтернету речей

У більшість сучасних пристроїв Інтернету речей до цього моменту впроваджували звичайні чипи SoC (System on Chip, однокристальні мікроЕОМ). Одна така мікросхема здатна самостійно виконувати функції всього пристрою в цілому, без необхідності її підключення до інших мікросхем. Отже, на ній можна розмістити всі необхідні для роботи елементи – ПЗП, ОЗП, блоки ЦАП і АЦП, інтерфейси для USB, Ethernet тощо. Така автономність SoC дуже зручна, а близьке розташування різних елементів також дозволяє ще й прискорити процес обробки даних в системі. Однак така мікросхема має істотний недолік.

Справа в тому, що структура SoC будується за певним шаблоном. Розташовані на ній елементи не можна модернізувати, як не можна і підігнати їхню роботу під певний тип пристроїв. Для пристроїв Інтернету речей це має вирішальне значення, оскільки кожен з них працює по-різному і призначений для різних завдань. Простіше кажучи, до кожного з них потрібен особливий підхід, інакше ефективно здійснювати свої функції вони не зможуть [3].

Вимоги до чипів для IoT-пристроїв

Окрім того, що вже було сказано, нові мікросхеми повинні відповідати ще кільком критеріям. У тому числі вони повинні бути:

- компактними (настільки, щоб забезпечити безперебійну роботу смарт-годинників і браслетів, які за визначенням невеликі, або навіть таких технологій, як електронні татуювання);
- енергоефективними (чим менше їм потрібно енергії для роботи, тим краще);
- безпечними;
- доступними за ціною.

Окрім того, для IoT-мікрочипів важливою є функція безперебійної підтримки бездротових мереж, в ідеалі – декількох різних. Зрештою, саме в тому, щоб підтримувати зв'язок з Інтернетом, і полягає взагалі весь сенс створення розумних пристроїв.

З урахуванням усього перерахованого зараз і ведеться розроблення нових мікрочипів для Інтернету речей.

Останні досягнення

У цілому розробок в цьому напрямі досить багато. Розглянемо деякі з них.

- ARTIK. Система випущена компанією Samsung і являє собою ціле сімейство модулів різного типу. Найбільш компактні моделі мають розмір 15×13 мм^2. Що стосується інших характеристик, то Samsung ARTIK може похвалитися наявністю процесора ARM Cortex-M4 з функцією DSP-обчислень, оперативною пам'яттю на 32 Кб, флеш-пам'яттю на 256 Кб, підтримкою Bluetooth версії 4.2. Схеми сімейства Samsung ARTIK призначені для спортивних mHealth-гаджетів, а також датчиків для різних домашніх і промислових розумних IoT-систем. Що стосується більш потужних систем – Samsung ARTIK 1020, – то вони характеризуються підтримкою ZigBee, Wi-Fi, Bluetooth LE, наявністю процесорів Quad Cortex-A15 + Quad Cortex-A7, 2 Гб оперативної пам'яті і 16 Гб флеш-пам'яті, а також відео-, аудіовходів / виходів.
- LAUNCHXL-CC2650. Розробник – компанія Texas Instruments. Не підтримує Wi-Fi, проте сумісний з мережею 6LoWPAN, а також ZigBee (із ZigBee RF4CE включно) і Bluetooth. За розміром менше, ніж аналоги від Samsung, і випускається в трьох видах: 4×4; 5×5; і 7×7 мм^2. Працює з процесорами ARM Cortex-M3, Cortex-M0 і Sensor Controller, має 128 Кб флеш-пам'яті та 8 Кб оперативної пам'яті.
- Atom E3900. Розробник – Intel. Цей тип мікросхем – останній в асортименті виробника. У Atom E3900 поліпшена ефективність роботи шин, вдосконалена графічна підсистема, додана система для синхронізації пристроїв. Зараз в лінійці представлено 3 моделі з різними показниками частоти роботи, кеш-пам'яті та розрахунковою потужністю.

У Intel, як і у Texas Instruments, існує також ряд інших мікросхем для пристроїв Інтернету речей. Кожна зі своїми особливостями. Також до виробництва чипів для IoT-пристроїв активно підключилися й інші виробники, наприклад: Dialog Semiconductor, Atmel, Espressif Systems, U-Blox та інші.

Але з навчальною метою найчастіше використовується система Arduino – це електронний конструктор і зручна платформа швидкого розроблення електронних пристроїв. Завдяки зручності і простоті мови програмування, а також відкритій архітектурі і програмному коду, платформа користується величезною популярністю в усьому світі.

2. ПРИКЛАДІ РОЗРОБОК ІОТ-ПРИСТРОЇВ НА ОСНОВІ ARDUINO

Подальші приклади взяті з сайту [4].

2.1. Налаштування (конфігурування) ESP8266 для роботи с Arduino

Це інструкція від BreakoutBros.com. Повний посібник дивіться тут[1]. ESP8266, як і Photon, є модулем мікроконтролера з підтримкою Wi-Fi, який можна використовувати як рішення для будь-якого проекту IoT. Однак, якщо Particle Photon була дуже проста у налаштуванні та роботі з усією її підтримкою, то ESP8266 з його варіантами та обмеженнями точно не для початківців. Документацію важко знайти, і багато пошукових запитів Google повертають приклади, які або пропускають кроки, або просто помиляються. В [5] інформація для зручності зібрана в одне місце, щоб простіше налаштувати Arduino і переконатися, що ESP8266 працює.

Компоненти: Arduino UNO and Genuino UNO, Espressif ESP8266 ESP-01.

Рис. 1 – ESP8266 ESP-01

З ESP8266 набагато складніше почати роботу, навіщо тоді навіть використовувати ESP8266? Ну, є дві дійсно великі переваги для ESP8266, які, залежно від проекту, можуть перевершити майже будь-який інший пристрій IoT, такий як Photon.

Ціна. Ціна ESP8266 просто неперевершена, її величина становить 1/4 вартості деяких інших рішень. Якщо Ви маєте невеликий бюджет і Вам потрібно щось зробити, то вивчення ESP8266 цілком варте Вашого часу.

Налаштування. ESP8266 повністю відкритий, на відміну від Photon Particle. Ви не в захваті від користування чужою хмарою або IDE, як Particle, і Вам не потрібна «підтримка» для використання? У Вас є більше свободи в тому, як Ви використовуєте ESP8266, щоб відповідати потребам будь-якого Вашого проекту. Ви можете зробити свій ESP8266 персональним сервером або перевести його в режим AP, щоб він виступав як точка доступу до Wi-Fi. При роботі з ESP8266 Ви

[1] http://www.breakoutbros.com/esp8266-the-ultimate-iot-module/

навчитесь набагато більшому, ніж з Photon, бо багато з того, що Photon робить у фоновому режимі в його інтеграції з Particle, Ви повинні навчитися робити самостійно.

Налаштування обладнання

Як і Photon, ESP8266 – це 3,3 В-ний прилад, тому для його живлення потрібно використовувати 3,3 В. **Не використовуйте** від Arduino живлення 5 V, оскільки це назавжди пошкодить ESP8266. Блок 3,3 В у стандартному Arduino також не подає достатнього струму для живлення ESP8266 під час його пікових перехідних процесів. Використовуйте джерело живлення 3,3 В зі струмом не менше 500 мА. Було використано джерело 3,3 В у комплекті датчиків Elegoo 37. Це було ідеально, оскільки він вписується у стандартну плату, виробляючи на одну лінію 3,3 В, а на іншу – 5,0 В.

Ви можете використовувати ESP8266 у два способи. У [4] продемонстровано, що він функціонує шляхом послідовного дзеркального відбиття послідовного порту Arduino IDE до програмного послідовного порту, підключеного до ESP8266. Це також дозволить нам змінити деякі налаштування та підключити його до домашнього Wi-Fi. Ви також можете придбати послідовний перетворювач USB в TTL і використовувати послідовні програми на основі команд, таких як Putty, щоб надіслати ці команди. Замість послідовного перетворювача USB в TTL, використовували Arduino і Serial, дзеркальні команди. Якщо Ви використовуєте послідовний перетворювач USB в TTL, будь ласка, також використовуйте перемикач рівня, щоб не застосовувати 5 V до ESP8266.

Існує також бібліотека ESP8266wifi та додаток Arduino IDE, які Ви можете додати, що включає бібліотеки для використання ESP8266, які Ви можете використовувати для прямої програми, а не просто відправлення команд із запитом. Зрештою, це буде найпростіший спосіб встановити ESP8266 для більшості проектів IoT, але це заслуговує на цілу окрему статтю.

Схема

Підключіть вивід TX ESP8266 до виводу 2 Arduino, вивід RX – до виводу 3 Arduino та переконайтесь, що лінія GND живлення 3,3 В та лінія GND Arduino з’єднані (рис. 2). Далі подайте на вхід Enable ESP8266 напругу 3,3 В, використовуючи резистор більше 2 кОм. Ви будете живити Arduino через його USB-порт, таким чином Ви будете спілкуватися з ESP8266.

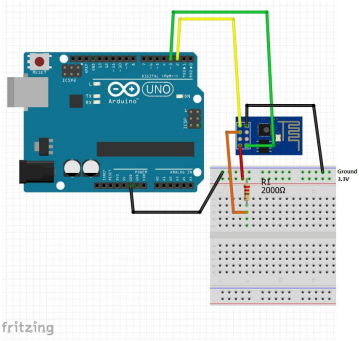

Рис. 2. Схема з’єднання

Код для Arduino

Ідея послідовного дзеркала портів позичена з цього джерела[2]. Перейдіть на цей сайт, щоб отримати код. В основному він просто шукає, коли є серійний доступ, і друкує його до програмного забезпечення ESPSerial, і коли програмне забезпечення ESPSerial доступне, він друкує його до серійного. Спочатку потрібно змінити швидкість передачі даних для форми `Serial.begin` та `ESPSerial.begin` (9600) на (115200). Завантажте це на Arduino, що Ви використовуєте. Після встановлення відкрийте послідовний монітор. Як зазначено в коді, переконайтесь, що обидва NL & CL обрані для перегляду внизу, а 112500 встановлено як швидкість передачі (Рис. 3).

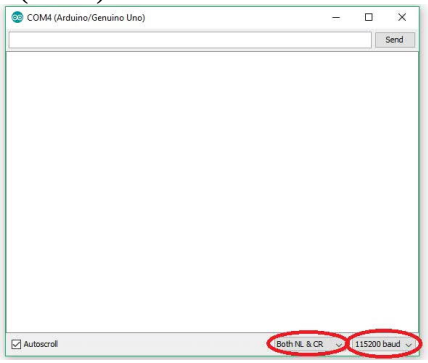

Рис. 3. Скріншот установок послідовного порту

Тепер, коли послідовний монітор запущений, Ви можете увімкнути ESP8266, і Вам потрібно побачити кілька команд завантаження, однак вони будуть ДУЖЕ змішані. Це тому, що для ESP8266 вста-

[2] http://www.martyncurrey.com/arduino-to-esp8266-serial-commincation/

новлено швидкість передачі в боді 115 200, а програмний послідовний порт Arduino просто не може підтримувати цю швидкість.

Введіть «AT» у текстовому полі послідовного порту, і якщо Ви бачите «OK» як відповідь, то Ви правильно підключені. Знову ж таки, це може бути трохи змішане з тарабарщиною. Ви можете бачити це змішування внизу в «redy» та «Box» поруч з AT (рис. 4).

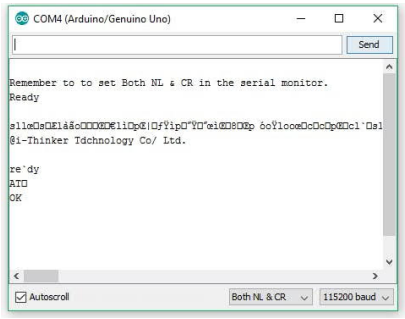

Рис. 4. Скріншот налаштування послідовного порту

Змініть швидкість обміну в бодах ESP8266, набравши текст `AT+UART_DEF=9600,8,1,0,0 .`
Ви повинні натиснути OK.

Тепер, коли Ви вводите, Ви не бачите відповіді. Спочатку вийдіть з програми та змініть Serial.begin (115200) та ESPSerial.begin (115200) на `Serial.begin (9600)` та `ESPSerial.begin (9600)`. Тепер знову відкрийте монітор послідовного порту і переконайтесь, що Ви вибрали 9 600 як швидкість передачі. Тепер введіть команду AT, і Вам потрібно знову натиснути OK (рис. 5).

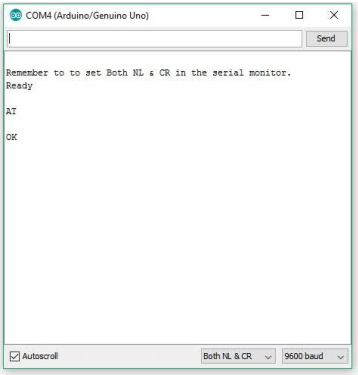

Рис. 5. Правильна робота послідовного порту

Якщо Ви не отримуєте жодної відповіді, ось що спробуйте:

- переконайтеся, що Ви використовуєте джерело живлення 3,3 В для ESP8266, що має мінімум 500 мА;
- переконайтеся, що GND ESP8266 і GND Arduino (загальні шини) підключені;
- переконайтеся, що вивід TX ESP8266 приєднано на вивід 2 Arduino, а вивід RX ESP8266 приєднано до контакту 3;
- переконайтеся, що вивід EN ESP8266 приєднано до напруги 3,3 В з резистором 10 кОм;
- якщо Ви не змінили показник швидкості передачі даних в ESP8266, переконайтеся, що в програмі Arduino для `serial.begin()` записано 115200, а 115200 вибрано як швидкість передачі даних для послідовного монітора Arduino;
- переконайтеся, що вибрано обидва NL & CL;
- якщо у Вас все ще виникають проблеми, може бути ESP8266 з іншою швидкістю передачі, ніж 115200, неправильна проводка або пошкоджений ESP8266.

Команди для підключення ESP8266 до Wi-Fi

Ви можете переглянути всі команди для ESP8266 тут[3]. Ми будемо використовувати лише деякі з них для цієї інструкції. Спочатку переконайтесь, що режим ESP8266 є режимом 1 за допомогою

AT + CWMODE = 1.

Якщо це спрацювало, Ви мали б натиснути OK. Далі подивіться, які мережі доступні, використовуючи:

```
AT + CWLAP.
```

Далі виберіть мережу та введіть:

```
AT + CWJAP = "YOUR_SSID", "YOUR_PASSWORD".
```

- Зверніть увагу, що «YOUR_SSID» – це SSID для Вашої мережі Wi-Fi, а «YOUR_PASSWORD» – це пароль для Вашої мережі Wi-Fi. Ви повинні побачити «Wi-Fi Connected», потім «WiFi Got IP», а потім OK, якщо з'єднання вдале (рис. 6).

Щоб отримати IP-адресу свого ESP8266, Ви вводите (рис. 7):

```
AT + CIFSR.
```

[3] https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/#AT+CWLIF

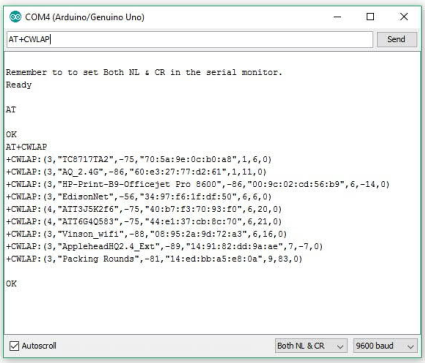

Рис. 6. Робота з Wi-Fi

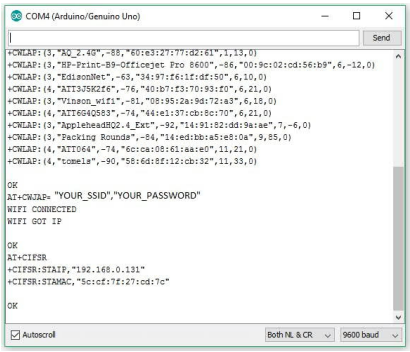

Рис. 7. Отримання IP-адреси

Це буде Ваша локальна IP-адреса. Тепер, якщо Ви хочете побачити щось круте, відкрийте підказку CMD і PING цю IP-адресу (рис. 8):

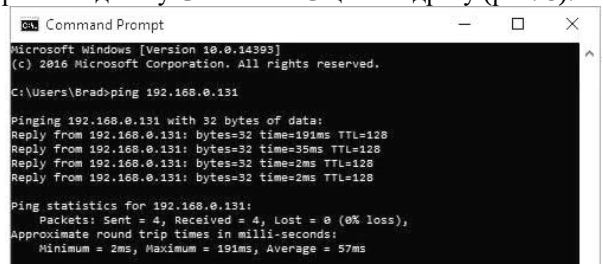

Рис. 8. Застосування функції PING

У ньому 4 успішні пінги[4]! Тепер Ви успішно підключили свій ESP8266 до домашнього Wi-Fi і пінгуєте свій ESP8266 з комп'ютера по Wi-Fi [4].

[4] **Ping** – утиліта для перевірки цілісності і якості з'єднання у мережах на основі TCP / IP, а також повсякденне найменування власне запиту.

2.2. Контроль споживання електроенергії

Підключіть лічильник енергії Modbus до Arduino і відстежте споживання електроенергії за допомогою Home Assistant.

Компоненти
Arduino MKR WiFi 1010
Arduino MKR 485 Shield
USB-A to Micro-USB Cable
USB Power Supply. Any power supply compatible. The MKR485 shield has a 7-24V buck converter built in.

ПРО ЦЕЙ ПРОЕКТ

Якщо Ви дійсно хочете зробити свій будинок розумнішим, Ви, мабуть, захочете почати з щомісячних рахунків (тобто енергії, газу тощо). Як кажуть деякі, добре для планети, гаманця та підсумкових результатів[5]. Обладнання з відкритим кодом – це наш спосіб досягти стійкості в домашніх умовах! Ця ідея запропонувала створити просте і безпечне рішення, яке легко інтегрувати з будь-яким програмним забезпеченням для домашньої автоматизації, оскільки воно відкриває дані через MQTT (у цьому випадку показано, як інтегрувати його в програму Home Assistant).

Огляд

Для вимірювання споживання електричної енергії вирішено використовувати вимірювач енергії Finder, оскільки він розроблений для використання DIN-рейки і ідеально вписується в основний розподільчий щиток нашого будинку. Класна річ у цьому продукті полягає в тому, що він має інтерфейс Modbus RS485, стандартний протокол зв'язку промислового стандарту, який робить розмову з Arduino дуже простою. Фактично, Arduino випустив офіційний щит, MKR485 та дві бібліотеки для розшифровки протоколу. Як основну плату ми вибрали Arduino MKR Wi-Fi 1010, оскільки він поділяє форм-фактор MKR і має підключення Wi-Fi.

Налаштування

На рис. 9 показано зовнішній вигляд вимірювача енергії Finder та плати MKR485.

[5] https://arthamoney.com/people-planet-profit-triple-bottom-line-the-ethical-measure-of-businesses-53750035035f

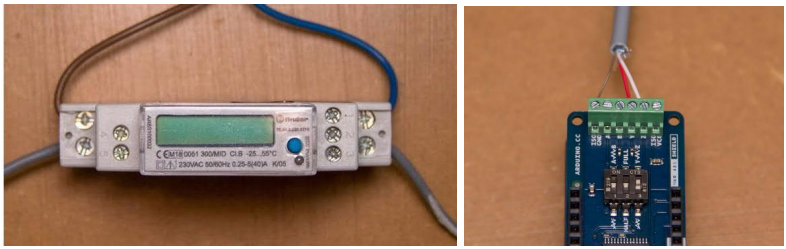

Рис. 9. Підключення до силової мережі, верхні дроти – вхід, нижні – вихід

Увага! Перевірте правила Вашої країни щодо роботи з електричною системою будинку та будьте дуже обережні, оскільки це може бути смертельно небезпечно! Якщо Ви не знаєте як, зателефонуйте до електрика. Перший крок – встановити лічильник у Вашій електричній шафі. Щоб переконатися, що Ви працюєте в безпечному середовищі, вимкніть живлення від електричного терміналу перед системою та двічі перевірте мультиметром, чи немає напруги між клемами.

Зробили? Настав час приєднати з'єднання RS485! Ми будемо використовувати кручений однопарний кабель з екраном, який, як правило, використовується для телефонних ліній. За допомогою цього кабелю Ви можете передавати сигнал на великі відстані (1,2 км). Однак, ми просто використовуємо кабель, який достатньо довгий, щоб вийти з шафи і розмістити Arduino в доступному місці.

Підключення Finder RS485

Інтерфейс RS485 називає свої термінали A, B і COM. Поширеним де-факто стандартом є використання: TX+ / RX+ або D+ як альтернативи B (високий для MARK, тобто простою), TX– / RX– або D– як альтернатива A (низький для MARK, тобто простою).

Оскільки плата MKR підтримує також Full Duplex, є ще два інші термінали, Y і Z. Тут ми збираємося накрутити інший кінець кабелю, оскільки з інструкції відомо, що напівдуплексний зв'язок відбувається лише на Y і Z клеми. Термінал COM повинен бути підключений до ISOGND. Оскільки ми використовуємо напівдуплексне з'єднання і оскільки кабельне з'єднання є одноранговим, ми повинні встановити перемикачі на платі MKR485 так, щоб вони відповідали нашому налаштуванню: ми встановлюємо HALF (від 2 до вимкнення) та навантаження на YZ (від 3 до ВКЛ); перший не має значення. Навантаження –

це опір, що з'єднує два термінали даних, щоб послабити інтерференцію сигналів (завади).

Так і є. Тепер Ви можете закрити шафу і зосередитись на стороні програмного забезпечення!

Програмне забезпечення

Запустіть комп'ютер та відкрийте IDE. Ви можете використовувати Arduino IDE або Arduino Create Editor. Код відповідає таким запитам:

– модульний зв'язок;

– управління Wi-Fi;

– протокол MQTT.

Modbus – це протокол з відкритим кодом для промислових датчиків і машин. Щоб Arduino розмовляв Modbus, ми будемо використовувати бібліотеку Arduino Modbus. Ця бібліотека пакує всі обробники та робить підключення будь-якого пристрою Modbus дуже швидко. Оскільки ми збираємось читати регістри відповідно до таблиць лічильника, ми можемо знайти всю необхідну нам інформацію, як код функцій, адресу регістра та розмір регістра словами. Але щоб зробити це зрозумілішим, пояснимо, як працює Modbus.

Повідомлення Modbus дотримуються простої структури:

```
01 03 04 00 16 00 02 25 C7
```

0x01 – адреса пристрою;

0x03 – це функціональний код, який повідомляє пристрою, якщо ми хочемо читати чи записувати дані (в цьому випадку – читати регістри тримання);

0x04 для Byte Count00 16 – ми надсилаємо 4 байти адреси регістру (00 16), який повідомляє пристрою, що ми хочемо прочитати;

00 02 – це розмір регістру (00 02) у словах (кожне слово – 2 байти);

останні 4 байти – це код CRC[6]. Цей код генерується з математичної функції над попередніми байтами, що забезпечує правильне отримання повідомлення.

В Ардуїно всі ці речі обробляються в умовах if:

```
// Created by Alberto Perro - Officine Innesto
//
```

[6] **Циклі́чнийнадлишко́вий код** (*Cyclic redundancy check*, CRC) – алгоритм обчислення контрольної суми, призначений для перевірки цілісності даних. CRC є практичним додатком завадостійкого кодування, заснованого на певних математичних властивостях циклічного коду.

```
// Read integer from a modbus device holding
registers
float readCurrent(){
    float ampere = 0.;
    //next command format the request and send it
over RS485
    if (!ModbusRTUClient.requestFrom(0x01,
HOLDING_REGISTERS, 0x0016, 2))
    {
    //Error Handling
    Serial.print("failed to read voltage! ");
    Serial.println(ModbusRTUClient.lastError());
    } else {
        uint16_t word1 = ModbusRTUClient.read();
//reading MSW from buffer
        uint16_t word2 = ModbusRTUClient.read();
//reading LSW from buffer
        int32_t milliamp = word1 << 16 | word2;
//join words to retreive int
        ampere = milliamp/1000.0;
//convert to ampere
    }
return ampere;}
```

В іншій частині `else` у нас є обробник відповідей.

Оскільки цей реєстр має два слова, ми мусимо приєднати їх до двійкової математики. Ми читаємо слова з буфера і зберігаємо їх як ціле число 16-бітною довжиною без знаку (2 байти або слово), потім з'єднуємо їх в ціле 32-бітне число, переміщаючи перше слово ліворуч і застосовуємо АБО над другим словом. Таким чином, ми отримуємо виміряний струм в мА, ділимо його на 1 000 і маємо струм в амперах.

Цей процес може бути пристосований до всього, що ми хочемо прочитати. Решта коду присвячена обробці мереж MQTT та WiFi.

Інтеграція домашнього помічника

Додавання лічильника до «Домашнього помічника» є досить простим. Припустимо, що у вас налаштований брокер MQTT[7]. Все, що Вам потрібно зробити, – це додати нові визначення у файл config.yaml.

```
sensor:
    - platform: mqtt
```

[7] https://www.home-assistant.io/integrations/mqtt

```
    name: "Main Voltage"
    state_topic: "energy/main/voltage"
    unit_of_measurement: "V"
```

Тут Ви повинні вказати назву вимірювання, тему MQTT для читання та одиницю вимірювання кількості. Збережіть файл, перевірте конфігурацію та перезавантажте програму «Домашній помічник», тепер вимірювання з'являться на головній сторінці (рис. 10).

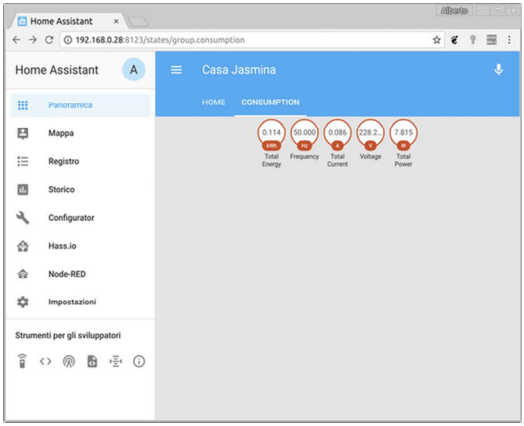

Рис. 10. Панель вимірювання споживання для домашнього помічника, що показує поточні показники

Домашній асистент подбає про створення графіків та автоматизацію процесів, викликаних Вашими показаннями. Поради за цим розділом закінчено, тепер Вам потрібно додати функції та налаштувати його під свої цілі!

КОД

Modbus RTU до моста MQTT для Finder

Цей скетч забезпечує повний міст між лічильником енергії Finder та MQTT. Розроблений для моніторингу витрат та використання електроенергії в Casa Jasmina.

```
/* Modbus RTU to MQTT bridge for Finder Energy Meter
 * ------------------------------------------------
 * This sketch provides a full bridge between the finder
energy meter and mqtt.
 * Developed to monitor electricity costs and usage in
Casa Jasmina.
 * RS485 Half Duplex 9600baud with termination (HALF,
Y\/\/Z)
 * ------------------------
 * ARDUINO        FINDER
 *    Y   ------->   A(-)
```

```
 *    Z   ------->   B(+)
 * ISOGND ------->   COM
 * ------------------------
 * Materials:
 * Arduino MKR WIFI1010
 * Arduino MKR 485 Shield
 * Finder 7E.64.8.230.0210
 * ------------------------
 * Created by Alberto Perro
 * Officine Innesto 2018
*/
//import libraries
#include <ArduinoRS485.h>
#include <ArduinoModbus.h>
#include <WiFiNINA.h>
#include <MQTT.h>
//your wifi credentials
const char ssid[] = "**********";
const char pass[] = "**********";

WiFiClient net;
MQTTClient client;
unsigned long rate = 60000; // default refresh rate in
ms
unsigned long lastMillis = 0;

//connect function
void connect() {
  Serial.print("checking wifi...");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(1000);
  }
  Serial.print("\nconnecting...");
  while (!client.connect("device_name", "user_name",
"user_pw")) { //CHANGE TO MATCH YOUR SETUP
    Serial.print(".");
    delay(1000);
  }
  Serial.println("\nconnected!");
  client.subscribe("energy/main/refreshrate"); //topic
to set refresh rate remotely

}
//mqtt receive callback function
void messageReceived(String &topic, String &payload) {
```

```
  Serial.println("incoming: " + topic + " - " +
payload);
  if(topic == "energy/main/refreshrate"){   //refresh
rate handler
    rate = payload.toInt()*1000;
    Serial.println("new rate "+String(rate));
  }
}

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, pass);
  while (!Serial);
  client.begin("broker_ip", net);   //CHANGE TO MATCH
YOUR SETUP
  client.onMessage(messageReceived);
  // start the Modbus RTU client
  if (!ModbusRTUClient.begin(9600)) {
    Serial.println("Failed to start Modbus RTU
Client!");
    while (1);
  }
}

void loop() {

  client.loop();
  if (!client.connected()) { //check network connection
    connect();
  }
  // publish a message after refresh elapsed (non-
blocking routine)
  if (millis() - lastMillis > rate) {
    lastMillis = millis();
    //make all the read calls
    float volt = readVoltage();
    delay(100);
    float amp = readCurrent();
    delay(100);
    double watt = readPower();
    delay(100);
    float hz = readFreq();
    delay(100);
    double wh = readEnergy();
    //publish results under related topics
```

```
    client.publish("energy/main/voltage",
String(volt,3));
    client.publish("energy/main/current",
String(amp,3));
    client.publish("energy/main/power", String(watt,3));
    client.publish("energy/main/frequency",
String(hz,3));
    client.publish("energy/main/energy", String(wh,3));
    Serial.print(String(volt,3)+"V "+String(amp,3)+"A
"+String(watt,3)+"W ");
    Serial.println(String(hz,3)+"Hz
"+String(wh,3)+"kWh");
    delay(100);
  }
}

/* Functions to read Finder Energy Meter registers
 *
 * Check the modbus protocol manual to understand the
code
 *
https://gfinder.findernet.com/public/attachments/7E/EN/P
RT_Modbus_7E_64_68_78_86EN.pdf
 */

float readVoltage(){
  float volt = 0.;
  if (!ModbusRTUClient.requestFrom(0x01,
HOLDING_REGISTERS, 0x000C, 2)) { //make the call to the
register
    Serial.print("failed to read voltage! ");
    Serial.println(ModbusRTUClient.lastError()); //error
handler
  }else{
    uint16_t word1 = ModbusRTUClient.read();  //read
data from the buffer
    uint16_t word2 = ModbusRTUClient.read();
    uint32_t millivolt = word1 << 16 | word2; //bit math
    volt = millivolt/1000.0;
  }
  return volt;
}
float readCurrent(){
    float ampere = 0.;
  if (!ModbusRTUClient.requestFrom(0x01,
HOLDING_REGISTERS, 0x0016, 2)) {
```

```
    Serial.print("failed to read current! ");
    Serial.println(ModbusRTUClient.lastError());
  }else{
    uint16_t word1 = ModbusRTUClient.read();
    uint16_t word2 = ModbusRTUClient.read();
    int32_t milliamp = word1 << 16 | word2;
    ampere = milliamp/1000.0;
  }
  return ampere;
}

double readPower(){
    double watt = 0.;
  if (!ModbusRTUClient.requestFrom(0x01,
HOLDING_REGISTERS, 0x0025, 3)) {
    Serial.print("failed to read power! ");
    Serial.println(ModbusRTUClient.lastError());
  }else{
    uint16_t word1 = ModbusRTUClient.read();
    uint16_t word2 = ModbusRTUClient.read();
    uint16_t word3 = ModbusRTUClient.read();
    uint64_t milliwatt;
    if(word1 >> 7 == 0){
      milliwatt = word1 << 32 | word2 << 16 | word3;
    }else{
      word1 &= 0b01111111;
      milliwatt = 0b1 << 48 | word1 << 32 | word2 << 16
| word3;
    }
    watt = milliwatt/1000.;
  }
  return watt;
}
float readFreq(){
    float freq = 0.;
  if (!ModbusRTUClient.requestFrom(0x01,
HOLDING_REGISTERS, 0x0040, 2)) {
    Serial.print("failed to read frequency! ");
    Serial.println(ModbusRTUClient.lastError());
  }else{
    uint16_t word1 = ModbusRTUClient.read();
    freq = word1/1000.0;
  }
  return freq;
}
double readEnergy(){
```

```
    double kwh = 0.;
  if (!ModbusRTUClient.requestFrom(0x01,
HOLDING_REGISTERS, 0x0109, 3)) {
    Serial.print("failed to read energy! ");
    Serial.println(ModbusRTUClient.lastError());
  }else{
    uint16_t word1 = ModbusRTUClient.read();
    uint16_t word2 = ModbusRTUClient.read();
    uint16_t word3 = ModbusRTUClient.read();
    uint64_t dwh = word1 << 32 | word2 << 16 | word3;
    kwh = dwh/10000.0;
  }
  return kwh;
}

:Author: me
:Email: {AuthorEmail}
:Date: 19/11/2018
:Revision: version#
:License: Public Domain

= Project: {Project}

Describe your project

== Step 1: Installation
Please describe the steps to install this project.

For example:

1. Open this file
2. Edit as you like
3. Release to the World!

== Step 2: Assemble the circuit

Assemble the circuit following the diagram layout.png
attached to the sketch

== Step 3: Load the code

Upload the code contained in this sketch on to your
board

=== Folder structure
```

```
....
 sketch123                => Arduino sketch folder
   ├── sketch123.ino      => main Arduino file
   ├── schematics.png     => (optional) an image of the
required schematics
   ├── layout.png         => (optional) an image of the
layout
   └── ReadMe.adoc        => this file
....

=== License
This project is released under a {License} License.

=== Contributing
To contribute to this project please contact me
https://id.arduino.cc/me

=== BOM
Add the bill of the materials you need for this project.

|===
| ID | Part name      | Part number | Quantity
| R1 | 10k Resistor   | 1234-abcd   | 10
| L1 | Red LED        | 2345-asdf   | 5
| A1 | Arduino Zero   | ABX00066    | 1
|===
```

=== Help
This document is written in the _AsciiDoc_ format, a markup language to describe documents. If you need help you can search the http://www.methods.co.nz/asciidoc[AsciiDoc homepage]
or consult the http://powerman.name/doc/asciidoc[AsciiDoc cheatsheet]

Подробиці за адресою:
https://create.arduino.cc/projecthub/123325/monitor-your-energy-bill-via-modbus-mkr-wifi-1010-and-rs485-814e5e?ref=tag&ref_id=iot&offset=17#toc-software-2

2.3. Система моніторингу рослин[8]

У цьому прикладі розглядається взаємодія інтелектуального давача вологості з інтернет-сервісами.

Компоненти:

1	Arduino MKR IoT Bundle	1 шт

Додатки і онлайн-сервіси

1	Zapier	
2	Thing Speak API	

Вступ. Як казав колись англійський поет Вільям Вордсворт: «Ваш розум – сад, ваші думки – це насіння, урожай може бути або квітами, або бур'янами». Утримання живих рослин може бути досить складним завданням, оскільки вони дуже погано спілкуються[9].

Один із способів зробити їх щасливими – це взяти з собою Ваші рослини, але, можливо, Ви не хочете обмацуватись з тим великим кактусом або папороттю, що стирчить з кишені зимової куртки. Також більшість рослин не люблять холод.

У цьому параграфі Ви дізнаєтесь, як здійснити віддалений моніторинг своєї рослини, а також продовжите вивчення Arduino.

Контролюючи вологість, температуру та світло, Ви можете переконатися, що Ваші рослини щасливі. Вони можуть щодня надсилати електронні листи і графіки та повідомляти про свої потреби.

Компоненти:

Датчик температури TMP36,
Фототранзистор,
Саморобний датчик вологості.

Цілі навчання

Ознайомлення з Інтернет-сервісом Zapier,
Надсилання HTTP-запитів,
Керування годинником та сигналізацією в режимі реального часу,
Побудова саморобного давача вологості,
Побудова графіків за допомогою thingSpeak #ProTips,
Форматування своєї електронної пошти за допомогою HTML та CSS #ProTips,
Тестування API з PostMan #ProTips.

[8] https://create.arduino.cc/projecthub/arduino/plant-communicator-7ea06f?ref=tag&ref_id=iot&offset=12

[9] Жарт. Цей, та наступний абзац дослівно перекладені для збереження колориту джерела.

Налаштування Zapier

Zapier – це онлайн-інструмент автоматизації, який може легко керувати API інших додатків для нас. Це прекрасний інструмент для комбінування декількох завдань. Також він може використовуватися для додатків, які з різних причин не можуть управляти Arduino.

У нашому випадку ми будемо використовувати його для надсилання електронного листа із даними, отриманими з Arduino. Можна ознайомитися з сайтом Zapier.com та зареєструватися на ньому і створити новий акаунт (рис. 11).

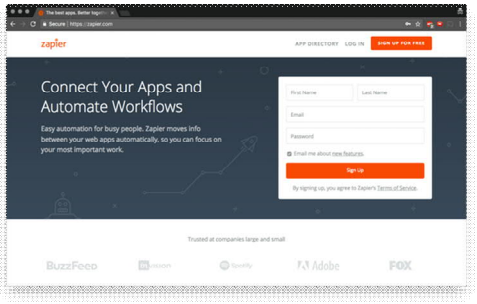

Рис. 11. Скріншот з онлайн-сервісу Zapier.com

Виконайте ці кілька простих кроків, щоб створити свій власний зап!

Для того, щоб продовжити, нам потрібно протестувати «*гачок*». Для цього ми копіюємо вказану URL-адресу та додаємо ці параметри:

```
given_URL?temperature=0&moisture=0&light=0&warning=0
```

Тепер просто скопіюйте та вставте цю URL-адресу на нову сторінку браузера.

Ви повинні побачити таку відповідь:

```
{"status": "success", "attempt": "5a1c833c-2932-4808-aae7-d3c352d98e4", "id": "fd2c7c2d-eef-43b8-b8eb-df3c2a78027a", "request_id": "5a1c83c-2932-4808-aae7-d3c352d98e4"}
```

Вітаємо! Ви щойно надіслали запит на http!

Це саме те, що ми зробимо з нашою панеллю управління (board) Arduino. Але замість того, щоб вставити цю URL-адресу на сторінку браузера, ми надішлемо її безпосередньо на сервер Zapier. Також ми будемо змінювати значення параметрів, які тепер встановлені на нуль за допомогою наших датчиків.

Для більш досконалого способу тестування та роботи з API подивіться #ProTip на сайті [x].

Після тестування Вашого запиту Ви можете приступити до створення Zap (рис. 12).

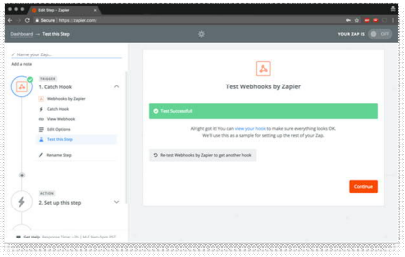

Рис. 12. Створення сторінки на службі Zapier.com

Створить панель управління (board).

Для реалізації всіх функцій ми будемо використовувати такі бібліотеки:

- Wi-Fi101 // для підключення до Інтернету та сканування мереж;
- RTCZero // для управління подіями, викликаними часом.

Ви можете завантажити їх у менеджера бібліотеки, як пояснено в посібнику по Ардуіно[10]. Для використання API Zapier спочатку потрібно завантажити сертифікати на MKR1000. Це стосується більшості онлайн-сервісів та API!

Завантажте приклад оновлення прошивки з бібліотеки Wi-Fi101 та додайте hooks.zapier.com у домени.

```
> examples > WiFi101 > FirmwareUpdater
```

Якщо Ви використовуєте веб-редактор Arduino, ця функція ще не реалізована, і Вам потрібно це зробити через додаток Arduino Desktop[11] (Рис. 13).

[10] https://www.arduino.cc/en/Guide/Libraries
[11] https://www.arduino.cc/en/Main/Software

Рис. 13. Скріншот з Arduino Desktop

Надсилання HTTP-запиту

Зараз ми готові надіслати HTTP-запит з дошки Arduino. Ми встановимо довільне значення для нашого параметра, яке згодом буде замінено реальними значеннями датчика.

```
float temperature = 22;
 int moisture = 150;
 int light = 40;
 String warning = "This is a warning message";
 warning.replace(" ", "%20");
```

Зауважте, що рядок має бути закодовано, оскільки порожні пробіли не можуть надсилатися в HTTP-запиті. Усі порожні пробіли замінені кодованим еквівалентом «%20».

Після встановлення наших параметрів ми викличемо функцію send_email (), яка перетворить їх у Strings і відновить ту саму URL-адресу, яку ми використовували раніше, а потім опублікуємо її на сервер Zapier.

Заповніть цей скетч (ескіз) своїм обліковим записом Wi-Fi та частиною URL-адреси, яку Ви отримали від Zapier, та завантажте.

```
#include <WiFi101.h>
#include<WiFiSSLClient.h>
const char* ssid = SECRET_SSID;    //  your
network SSID (name)
const char* password = SECRET_PSWD;  // your
network password
String httpsRequest = SECRET_REQUEST; // your
Zapier URL
const char* host = "hooks.zapier.com";
WiFiSSLClient client;
void setup() {
 Serial.begin(9600);
```

```
 while (!Serial);
 delay(2000);
 Serial.print("Connecting Wifi: ");
 Serial.println(ssid);
 while (WiFi.begin(ssid, password) !=
WL_CONNECTED) {
   Serial.print(".");
   delay(500);
 }
 Serial.println("");
 Serial.println("WiFi connected");
}
void loop() {
 float temperature = 22;
 int moisture = 150;
 int light = 40;
 String warning = "This is a warning message";
 warning.replace(" ", "%20");
 send_email(temperature, moistue, light, warning
);
 delay(20000)
}
void send_email(float temperature, int moisture,
int light, String warning) {
 // convert values to String
 String _temperature = String(temperature);
 String _moisture = String(moisture);
 String _light = String(light);
 String _warning = warning;
 if (client.connect(host, 443)) {
   client.println("POST " + httpsRequest +
"temperature=" + _temperature + "&moisture=" +
_moisture + "&light=" + _light + "&warning=" +
_warning + " HTTP/1.1");
   client.println("Host: " + String(host));
   client.println("Connection: close");
   client.println();
   delay(1000);
   while (client.available()) { // Print on the
console the answer of the server
     char c = client.read();
```

```
      Serial.write(c);
    }
    client.stop();  // Disconnect from the server
  }
  else {
    Serial.println("Failed to connect to client");
  }
}
```

Він надсилатиме електронний лист на вибрану Вами адресу кожні 20 секунд.

Будьте уважні до запитів, Zapier дозволяє вам лише 100 безкоштовних запитів на місяць.

RTC та сигналізації

Ми можемо використовувати годинник в режимі реального часу MKR1000, щоб надсилати електронне повідомлення щодня в певну годину. Цей скетч встановлює час та дату о 16:00 4 грудня 2017 р., а потім спрацьовує тривога щодня о 16:01.

Зауважте, що оскільки сигнал тривоги приєднаний до функції переривання, ми не можемо включати жодних затримок, але ми можемо використовувати булеву змінну для запуску дій у циклі (`loop()`)

```
#include <RTCZero.h>
RTCZero rtc; // create RTC object
/* Change these values to set the current initial
time */
const byte seconds = 0;
const byte minutes = 0;
const byte hours = 16;
/* Change these values to set the current initial
date */
const byte day = 4;
const byte month = 12;
const byte year = 17;
bool email_already_sent = false;
void setup() {
 Serial.begin(9600);
 while (!Serial);
 delay(2000);
 rtc.begin(); // initialize RTC 24H format
 rtc.setTime(hours, minutes, seconds);
 rtc.setDate(day, month, year);
```

```
 rtc.setAlarmTime(16, 1, 0);  // Set the time for
the Arduino to send the email
 rtc.enableAlarm(rtc.MATCH_HHMMSS);
 rtc.attachInterrupt(alarmMatch);
}
void loop() {
 if (!email_already_sent) {
   // send_email();
   email_already_sent = true;
 }
}
void alarmMatch() { // triggered when the alarm
goes on
 Serial.println("Alarm Match!");
 email_already_sent = false;
}
```

Саморобний датчик зволоження ґрунту

Два дроти, розміщені в горщику із ґрунтом, утворюють змінний резистор, опір якого змінюється залежно від вологості ґрунту. Цей змінний резистор підключений у конфігурації дільника напруги, і на Arduino надходить напруга, пропорційна опору між двома проводами. Це означає, що чим більш вологий ґрунт, тим менше напруга буде вимірюватися пристроєм Ардуїно. За допомогою резистора 1 МОм та двох проводів ми можемо створити наш власний датчик зволоження ґрунту! (рис. 14).

Завантаживши наступний скетч, Ви можете розпочати зчитування значень датчика, радимо спершу розпочати його тестування в сухому ґрунті і зафіксувати прочитане значення. Це значення буде використано для встановлення порогу, щоб Arduino дізнався, коли Вашим рослинам потрібна вода, і надішле вам електронний лист.

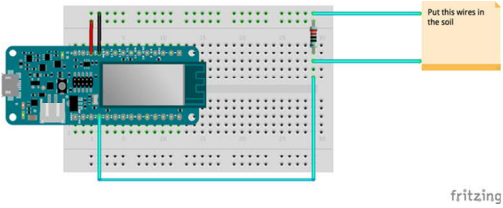

Рис. 14

```
int moisturePin = A2;
// Set this threeshold accordingly to the
resistance you used
```

```
// The easiest way to calibrate this value is to
test the sensor in both dry and wet soil
int threeshold = 800;
void setup() {
 Serial.begin(9600);
 while (!Serial);
 delay(2000);
}
void loop() {
 Serial.println(get_average_moisture());
 delay(5000);
}
int get_average_moisture() { // make an average of
10 values to be more accurate
 int tempValue = 0; // variable to temporarly
store moisture value
 for (int a = 0; a < 10; a++) {
   tempValue += analogRead(moisturePin);
   delay(100);
 }
 return tempValue / 10;
}
```

Додати Датчик температури та світла

Дивіться схему нижче, щоб підключити два датчики. Ми будемо використовувати ці дві функції для зчитування значень з датчиків:

```
float get_temperature(){
 int reading = analogRead(A1);
 float voltage = reading * 3.3;
 voltage /= 1024.0;
// Print tempeature in Celsius
float temperatureC = (voltage - 0.5) * 100 ;
//converting from 10 mv per degree wit 500 mV
offset
// Convert to Fahrenheit
float temperatureF = (temperatureC * 9.0 / 5.0) +
32.0;
return temperatureC;
}
int get_light(){
 int light_value=analogRead(A0);
```

```
return light_value;
```

Зауважте, що Ви можете використовувати одиниці Фаренгейта, повертаючи `temperatureF` замість `temperatureC`

#ProTip: Побудова графіку за допомогою ThingSpeak

Для того, хто є прибічником контролю, наводимо інструкцію для побудови всіх даних, зібраних в графіку (рис. 15):

Рис. 15

Ми будемо використовувати платформу ThingSpeak (рис. 16) для побудови графіків, виконайте ці кроки, щоб розпочати роботу.

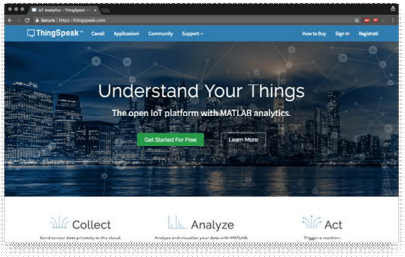

Рис. 16. Скріншот з ThingSpeak

Завантажте бібліотеку ThingSpeak і почнемо роботу. Цей скетч нижче буде щомісяця завантажувати значення датчиків у хмару, просто заповнюйте Ваші облікові дані Wi-Fi та ключі API Вашого каналу та завантажуйте.

```
#include <WiFi101.h>
#include<WiFiSSLClient.h>
#include "ThingSpeak.h"
const char* ssid = SECRET_SSID;    //  your
network SSID (name)
const char* password = SECRET_PSWD;  // your
network password
WiFiClient  ThingSpeakClient;
```

```
unsigned long myChannelNumber = 356392;
const char * myWriteAPIKey = SECRET_WRITE_API;
int lightPin = A0; //the analog pin the light
sensor is connected to
int tempPin = A1; //the analog pin the TMP36's
Vout (sense) pin is connected to
int moisturePin = A2;
void setup() {
 Serial.begin(9600);
 while (!Serial);
 delay(2000);
 Serial.print("Connecting Wifi: ");
 Serial.println(ssid);
 while (WiFi.begin(ssid, password) !=
WL_CONNECTED) {
   Serial.print(".");
   delay(500);
 }
 Serial.println("");
 Serial.println("WiFi connected");
}
void loop() {
 ThingSpeak.setField(1, get_light());
 ThingSpeak.setField(2, get_temperature());
 ThingSpeak.setField(3, get_average_moisture());
 ThingSpeak.writeFields(myChannelNumber,
myWriteAPIKey);
 Serial.println("message sent to cloud");
 delay(60000); // send values every 1 minute
}
float get_temperature() {
 int reading = analogRead(tempPin);
 float voltage = reading * 3.3;
 voltage /= 1024.0;
 // Print tempeature in Celsius
 float temperatureC = (voltage - 0.5) * 100 ;
//converting from 10 mv per degree wit 500 mV
offset
 // Convert to Fahrenheit
 float temperatureF = (temperatureC * 9.0 / 5.0) +
32.0;
```

```
 return temperatureC;
}
int get_average_moisture() { // make an average of
10 values to be more accurate
 int tempValue = 0; // variable to temporarly
store moisture value
 for (int a = 0; a < 10; a++) {
   tempValue += analogRead(moisturePin);
   delay(10);
 }
 return tempValue / 10;
}
int get_light() {
 int light_value = analogRead(A0);
 return light_value;
}
```

У повному скетчі Pro нижче Ви можете побачити, як прикріпити це завантаження до сигналу RTC, який спрацьовує щохвилини. На жаль, Gmail не дозволяє нам вставляти графіки та рамки кадрів в основу електронної пошти, але ми можемо надіслати посилання електронною поштою за допомогою спеціальної кнопки, див. підказку нижче.

Ми вибрали ThingSpeak, але є багато альтернатив! Погляньте, наприклад, на Dweet.io та freeboard.io.

#ProTip: Форматування електронної пошти

Zapier дозволяє нам вставити деякі HTML і css-код в тілі електронної пошти. Ми можемо використовувати це для надсилання досить стилізованого електронного листа у спосіб, наведений на рис. 17.

Для досягнення цього результату просто змініть тип тіла на HTML на інтерфейсі Zapier та додайте свій власний HTML та CSS (рис. 18).

Або Ви можете просто скопіювати та вставити цей шаблон:

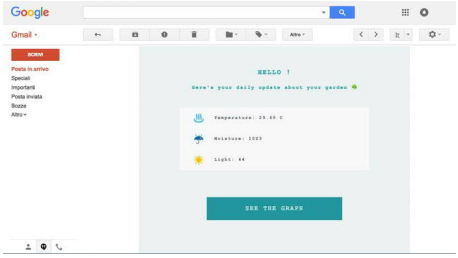

Рис. 17. Надсилання електронного листа

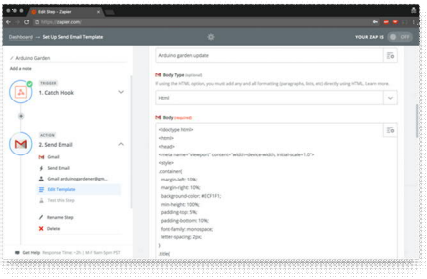

Рис. 18. Форматування електронного листа

```
<!doctype html>
<html>
<head>
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<style>
.container{
 margin-left: 10%;
 margin-right: 10%;
 background-color: #ECF1F1;
 min-height: 100%;
 padding-top: 5%;
 padding-bottom: 10%;
 font-family: monospace;
 letter-spacing: 2px;
}
.title{
 text-align: center;
 color: #0CA1A6;
 font-size: 1.5em;
 padding-top: 0.2vh;
}
.subtitle{
 text-align: center;
 color: #0CA1A6;
 font-size: 1.1em;
 padding-bottom: 4%;
 padding-left: 4%;
 padding-right: 4%;
}
.data{
 padding-bottom: 2%;
```

```
 padding-top: 2%;
 padding-left: 5%;
 margin-left: 15%;
 margin-right: 15%;
 background-color: #F7F9F9;
 text-align: left;
 color: #4E5B61;
 font-size: 1em;
 font-weight: bold;
}
.bttn{
 padding-bottom: 3%;
 padding-top: 3%;
 margin-left: 25%;
 margin-right: 25%;
 margin-top: 10%;
 background-color: #00979D;
 text-align: center;
 color: #F7F9F9;
 font-size: 1.4em;
}
.disclaimer{
text-align:center;
color:#DA5B4A;
font-size: 1.1em;
font-weight: bold;
margin-top: 10%;
}
.bttn:hover{
background-color: #008184;
}
a{
 text-decoration: none;
}
</style>
</head>
<body>
<div class="container">
<h2 class="title">HELLO !</h2>
<h4 class="subtitle">Here's your daily update
about your garden &#9752;</h4>
```

```
<div class="data">  &#9832;    Temperature:
{{querystring__temperature}} C </div>
<div id="humidity" class="data"> &#9748;
Moisture: {{_querystring__moisture}}</div>
<div  class="data"> &#9728;    Light:
{{querystring__light}}</div>
<div
class="disclaimer">{{querystring__warning}}</div>
<a href="your_link_to_thingSpeak">
<div class="bttn">SEE THE GRAPH</div>
</a>
</div>
</body>
</html>
```

#ProTip: Тестовий API з PostMan

Робота з HTTP-запитом може бути важкою, на щастя, існує безліч інструментів, які можуть допомогти нам створити потрібну нам URL-адресу.

Листоноша – прекрасний приклад (рис. 19):

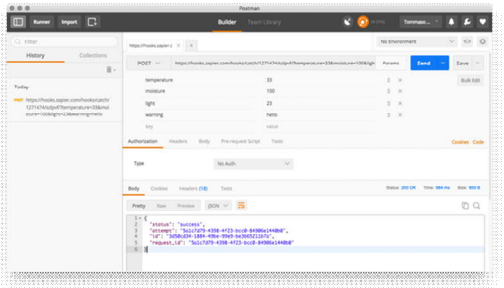

Рис. 19. Скріншот сервісу «Листоноша»

Просто вставте початкову URL-адресу, надану Zapier, додайте параметри та надішліть її. Він роздрукує відповідь із сервера та складе URL для Вас.

Повні тексти коду доступні на сайті https://create.arduino.cc/projecthub/arduino/plant-communicator-7ea06f?ref=tag&ref_id=iot&offset=12

3. ІНТЕЛЕКТУАЛЬНА СИСТЕМА РОЗУМНОГО БУДИНКУ

Будь-який будинок – адміністративна, виробнича чи житлова будівля – складається з певного набору різних систем, які відповідають за виконання певних завдань та вирішення завдань, що виникають протягом життя цієї будівлі чи будинку [5]. Чим більше підсистем і завдань вони виконують, тим складніше керувати ними. Крім того, зростають і витрати на технічне обслуговування, ремонт та персонал. Такі проблеми виникли з появою великих промислових, громадських та адміністративних комплексів. У нинішніх умовах такі будівлі можуть функціонувати повністю автоматично, як місто. У ньому вже є всі необхідні послуги та обладнання для цього. Для правильної та своєчасної роботи будівлі потрібна велика адміністративна служба, необхідно тримати цілу команду, яка займається такими питаннями. Незважаючи на те, що в сучасному світі більшістю послуг, таких як вентиляція, освітлення, пожежна сигналізація, безпека, відеомоніторинг тощо, можна керувати віддалено, усі ці послуги перебувають у процесі постійного моніторингу та управління кваліфікованим персоналом. Однак іноді трапляються випадки, коли навіть навчений персонал не може ефективно виконувати свої обов'язки. Ці ситуації пов'язані з глобальною загрозою будівлі та людям, які там перебувають – пожежа, землетрус та інші стихійні лиха. У таких ситуаціях люди можуть зазнати панічних атак, а швидкість та якість реагування погіршаться через страх та розгубленість. Це одна з переваг автоматизації такого типу завдань, оскільки на комп'ютерні системи такі фактори не впливають. Вартість експлуатації таких систем надзвичайно висока, тому що необхідно утримувати персонал кваліфікованого обслуговуючого персоналу та контролювати всі елементи. Таким чином, вже давно йдеться про створення незалежної інтелектуальної системи, яка зможе керувати всіма її компонентами незалежно або за заздалегідь визначеними схемами. Також у випадку приватних будинків та квартир така система стане засобом централізації та надання загального інтерфейсу для доступу та контролю для всіх розміщених пристроїв, датчиків тощо. Будинки чи квартири, де реалізовані такі системи, називаються розумними.

Таким чином, виникає потреба створити систему, яка, з одного боку, підтримувала б якомога більше пристроїв для розумних будинків, доступних на ринку, які б не залежали від їхніх виробників, а з іншого боку мали б зручний інтерфейс та широкі варіанти налаштування, автоматизації та управління.

СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТ ДОСЛІДЖЕННЯ

Для досягнення мети необхідно виконати ряд підзадач (рис. 20). Для того, щоб зрозуміти архітектуру сервісу, зв'язки між класами та

інтерфейсами, а також процеси їхньої взаємодії, необхідно побудувати схему класів служби [5]. Багато з цих класів відповідають за безпеку системи, адже це одне з найважливіших завдань, оскільки втрата контролю над службами, які керують подіями у всьому будинку, можуть бути надзвичайно небезпечними. Наступним кроком є розгляд можливої взаємодії користувачів із системою. Ця функціональність міститься у діаграмі використання (рис. 21), де описано можливе використання системи (як чорна скриня) з точки зору користувача. Як видно з діаграми, користувачеві доступні безліч різних функціональних варіантів роботи з системою. Такими варіантами можуть бути або Реєстрація, або Авторизація в системі, або Команди викликів, Додавання нових пристроїв або Перегляд відеоданих, збережених відеослужбою.

Рис. 20. Дерево цілей

Рис. 21. Діаграма використання

АНАЛІЗ ЗРАЗКА КОНТРОЛЮ

Наприклад, план будинку доступний у системі (рис. 22), за допомогою якого можна розділити пристрої в будинку на кімнати та ке-

рувати ними не окремо, а всередині приміщення. Наприклад, увімкніть або вимкніть усе освітлення. Після авторизації користувач зможе отримати доступ до головної сторінки (рис. 23) з усіма необхідними елементами управління – управління пристроєм, перегляд даних із сенсорів та лічильників, перегляд відео з відеопослуг тощо. На головній сторінці інтерфейс користувача – це панель управління пристроєм.

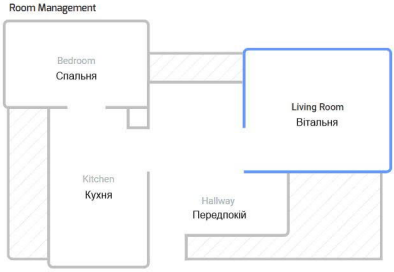

Рис. 22. План будинку

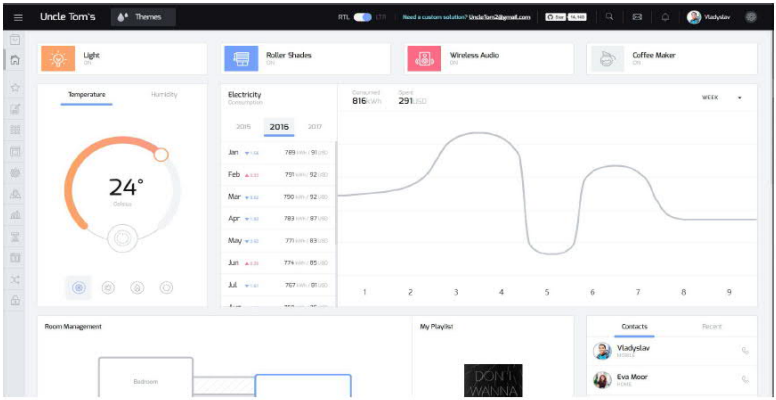

Рис. 23. Головна сторінка системи

За допомогою цієї панелі користувач може вмикати / вимикати пристрої, переглядати їхній стан, змінювати його та отримувати доступ до всіх підключених пристроїв: окремі параметри різних датчиків (рис. 23), наприклад, температура повітря або вологість в межах розумного будинку. Інтерфейс дозволяє переглядати показники та встановлювати потрібні параметри за допомогою однієї і тієї ж панелі слайдів. Коли користувач змінить поточне значення, кондиціонер, нагрівач або зволожувач повітря, який встановить температуру або рівень вологості за вказаним користувачем значенням, автоматично буде активовано.

Система також дозволяє підключатися до розумних лічильників багатьом виробникам та отримувати дані від них. Для зручності користувача ці дані відбиті у вигляді діаграми, яка визначає період та обсяг споживання певного виду послуги. Крім того, інтерфейс також забезпечує можливість перегляду відео в Інтернеті з будь-якої камери, підключеної до системи. Також користувач може переглядати журнали подій, які були зроблені камерами, або переглядати записи, зроблені раніше. Система також підтримує мобільний інтерфейс управління як мобільний додаток (рис. 24).

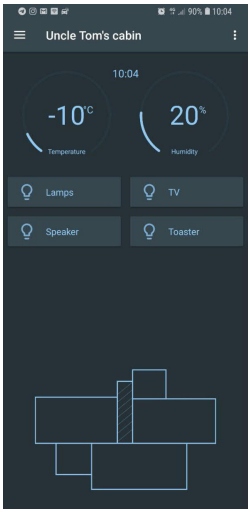

Рис. 24. Мобільний додаток

4. Висновки

У посібнику розглянуто дужу невелику кількість прикладів побудови пристроїв Інтернету речей та напрями розвитку цієї перспективної та багатогранної галузі техніки. З огляду на великий ринок пристроїв та послуг, зрозуміло, чому цій галузі приділяється велика увага, але це приводить до високої конкуренції на цьому ринку, тому попитом будуть користуватись пристрої з максимальною економічністю та зручні у використанні. Отже, тут є велике поле для удосконалення як апаратних, так і програмних засобів.

Додаток А

Інтернет одноразових речей: Датчики, що виробляються з паперу та пластику, підключать повсякденні речі[12]

Рік 2028 р. 8 P.M. В ніч на середу. Ви голодний. Ви тужливо вдивляєтесь у єдиний предмет, що залишився у Вашому холодильнику: пакет ковбас із неприваблива сіруватим відтінком. Тьфу. Вони завжди виглядали так? Вони все ще безпечні для їжі? У 2018 р. Вам доведеться покладатися на свій нюх і грати в азартні ігри. Але в 2028 р. Ви можете просто помахати смартфоном над пакетом. Той спитає вбудований датчик, який вимірює концентрацію газів, пов'язаних з розпадом м'яса, і надішле повідомлення «Безпечно їсти протягом наступних 20 годин», а потім запропонує список рецептів приготування ковбаси. Занадто голодний, щоб турбуватися з рецептами, Ви розриваєте упаковку, кидаєте ковбаси на сковороду і відкидаєте пакунок разом із його сенсорною технологією.

Ця уявна сцена порятунку смартфоном – лише один з багатьох очікуваних додатків Internet of Things. IoT, зараз можливий через зближення недорогих, малопотужних компонентів, зокрема мікропроцесорів, стільникових радіо, радіостанцій Wi-Fi та датчиків MEMS. IHS Markit, дослідницька фірма, яка відслідковує та аналізує електроніку, прогнозує, що глобальний обсяг пристроїв IoT збільшиться з 27 мільярдів підключених пристроїв у 2017 р. до 125 мільярдів у 2030 р.

Сказати, що напівпровідникова індустрія радісно схвильована, було б величезним заниженням. В основі цього хвилювання лежить головне припущення – що всі ці компоненти будуть виготовлені з використанням кремнію, виробляючи існуючі фабрики чіпів на мільярд доларів США, відомі як fabs. Але це припущення неправильне. Одноразові датчики, які визначають безпеку майбутньої ковбаси, не будуть виготовлені на кремнію. Натомість вони будуть надруковані безпосередньо на паперовій або пластиковій упаковці.

Датчики, доступні сьогодні для додатків IoT, базуються на MEMS (мікроелектромеханічних системах). Створені на кремнієвих пластинах з використанням методів, розроблених спочатку для виготовлення напівпровідникових IC, датчики MEMS використовують мікроскопічні механічні структури для виявлення руху, звуку, тиску, світла або навіть видів газу. Крихітні діафрагми, консолі, вібраційні маси чи

[12] Alissa M. Fitzgerald. The Internet of Disposable Things. Spectrum, 2018 INT, No. 12. P. 30–35. spectrum.ieee.org

контурні канали, як правило, менше 100 мікрометрів (діаметр людського волосся), реагують на фізичний стимул і потім перетворюють його на електронний сигнал. У поєднанні з радіо та антеною датчики MEMS можуть бездротово передавати свої електронні дані в Інтернет.

Мікросхеми MEMS зазвичай мають розміри лише 1 або 2 міліметри, а це означає, що на кремнієвих пластинах діаметром 200 мм можна економно виготовити десятки тисяч. Сьогоднішня ціна на масові MEMS-датчики, такі як мікрофони та гіроскопи, становить лише від 0,10 до 0,50 долара США за датчик. Це робить їх придатними для використання в смартфонах, фітнес-трекерах та інших продуктах побутової електроніки, які коштують близько 100 доларів США.

Для разового використання в упаковці харчових продуктів, медичних тестах або розумному одязі, де продукт може коштувати менше 10 доларів США, економіка вимагатиме, щоб датчики коштували ледь копійки. Іншими словами, ціни на датчики MEMS повинні бути не більш ніж одна десята від ціни, яку вони мають сьогодні, щоб використовуватися в таких дешевих IoT-продуктах.

Незважаючи на те, що для виробництва кремнієвих MEMS потрібно ще більше витрат, потенційна економія в кінцевому рахунку обмежується витратами на власне кремній. На сьогодні датчик MEMS розміром 1 мм на 1 мм, виготовлений на пластині діаметром 200 мм, містить кремній приблизно на 0,002 дол. США.

Іншими словами, щоб зробити датчик MEMS, який коштує копійки, кремній повинен був би коштувати п’яту частину того, що коштує сьогодні. Дивлячись на історичні тенденції цін на кремній, здається малоймовірним, що ціни могли колись так впасти. (Насправді ціна кремнію виросла вдесятеро, порівняно з сьогоднішньою ціною, під час буму фотоелектричної індустрії в 2006–2010 рр.). Скорочення мікросхеми MEMS для досягнення цільової ціни в 1 цент також не вийде. Будучи залежними від механічних властивостей кремнію, рідин та газів, більшість пристроїв MEMS не працюватимуть, якщо Ви їх різко змен-

шите. Натомість нам доведеться робити датчики з матеріалів, які значно дешевші, ніж кремній.

У всіх галузях промисловості ринкові сили завжди мотивували виробників переходити до менш затратних матеріалів. Зараз навіть датчики MEMS рухаються до пластика та паперу. Хоча виготовлення електронних пристроїв на цих простих підкладках може здатися новою радикальною ідеєю, це не так. Прецедент триває майже 40 років, коли Т. Пітер Броді та Деррік П. Пейдж у Westinghouse вперше виготовили тонкоплівкові транзистори (TFT) на паперовій підкладці.

Дослідники роками працювали над паперовими та пластиковими датчиками, і зниження вартості сенсорів не стало їхньою єдиною мотивацією. Конкуренція за федеральні фонди змусила дослідників шукати способи зробити мікротехнологію без чистих приміщень та інших екзотичних інструментів. Тоді з'являється все більший інтерес до створення датчиків з матеріалів, що біологічно розкладаються, та більш гнучких, ніж кремній, для використання в інвазивних медичних додатках.

Датчик IoT має два важливі компоненти: власне датчик, який виявляє конкретну фізичну чи хімічну подію, і телеметричний блок, який передає дані датчика в Інтернет. Телеметричному пристрою може не знадобитися акумулятор, якщо він буде отримувати живлення від зчитувача зв'язку у ближньому полі (NFC), чи від зчитувача RFID. Або пристрій може мати акумулятор, радіо та антену, щоб активно передавати дані на частотах мобільного телефону, Wi-Fi або Bluetooth.

Датчик IoT може навіть мати певну кількість інтелекту або можливостей інтерпретації даних. Для цього досліджуються паперові інтегральні схеми, виготовлені з друкованих ліній металевих та напівпровідних чорнил. У 2011 р. інженери бельгійської дослідницької фірми Imec побудували 8-бітний мікропроцесор з гнучкого пластику, використовуючи тонкоплівкові транзистори, виготовлені з органічних напівпровідників. Вони пройшли за ним минулого року за допомогою друкованої пластикової мікросхеми NFC, виготовленої з використанням IGZO – напівпровідника з оксидом металу, який є сумішшю індію, галію, цинку та кисню. Хоча ці конструкції все ще досить примітивні та повільні, порівняно зі своїми кремнієвими двоюрідними братами та не можуть значно просунутися далі через обмеження цих матеріалів, вони поки що зарекомендували себе досить добре, щоб виконати роботу за потрібною ціною.

Але більшість датчиків IoT, особливо дешеві та одноразові, будуть покладатися на зовнішню потужність та інтелект. Обидва широко доступні. Наприклад, усі нові смартфони здатні здійснювати зв'язок у

близькому полі, який використовується в першу чергу для безконтактних електронних платіжних систем, таких як Apple Pay. Це означає, що Ваш телефон, ймовірно, вже обладнаний для живлення та спілкування з майбутнім датчиком свіжості ковбаси.

Один із перших імплантованих датчиків тиску MEMS, затверджений американською адміністрацією з питань харчування та лікарських засобів, створений CardioMEMS (тепер це частина St. Jude Medical), являв собою жорсткий датчик, виготовлений з кварцу, призначений для імплантації поруч із стентом, щоб контролювати ефективність роботи стента, який виконує свою роботу з утримання артерії відкритою. Датчик включає антену, розміщену всередині кварцу, і тонку мембрану з кварцу, яка закриває порожнину, утворюючи таким чином конденсатор. Коли тиск крові потрапляє на мембрану, вона змінює ємність і, отже, резонансну частоту ланки датчика. Зміна частоти може пасивно вимірюватися зовнішнім зчитувачем, який запитує датчик з радіочастотним сигналом відомої частоти, а потім порівнює його зі зміненою частотою зворотного сигналу.

Проблема з кварцовим датчиком – і з кремнієм – полягає в тому, що вони відносно жорсткі і досить крихкі. Один із співзасновників КардіоМЕМС, Марк Аллен з Університету Пенсільванії, згодом перейшов від жорстких датчиків кремнію та кварцу до створення гнучких та таких, що біологічно розкладаються, датчиків. Аспіранти групи Аллена тепер отримують інженерні датчики та навіть акумулятори з полімерів, таких як полікислота (молочна когліколева, PLGA), полівініловий спирт (PVA) та полікапролактон (PCL) – ті самі матеріали, що використовуються в хірургічних швах, що розчиняються. Міжз'єднання та антени утворюються з металів, що розкладаються біологічно, такі як магній або цинк. За допомогою методів мікрофабрикації MEMS, таких як гололітографія та гальванічне покриття, студенти продемонстрували датчики тиску, які біологічно розкладаються, розміром в міліметр з можливостями бездротової передачі даних.

Родоначальник датчика нюхання ковбаси вже продемонстрований. Група С. Андрееску з Університету Кларксона в Потсдамі, штат Нью-Йорк, створила паперовий датчик, який може виявити псування їжі. Наноструктури, виготовлені з неорганічних хімічних речовин, надруковані на папері, реагують з газами, що виділяються при розкладанні; потім наноструктури змінюють колір пропорційно концентрації газу. Зараз результат датчика повинен бути прочитаний візуально. Для нашого уявного додатка для смартфонів додавання засобу електронного зчитування та передачі даних зробить сенсор IoT готовим.

Група Джорджа Уайтсайда з Гарвардського університету робить саме це, адаптуючи на папері медичні діагностичні датчики, які змінюють колір і читаються візуально, як домашня тест-смужка на вагітність, до методу зчитування, заснованого на ВЧ. Використовуючи такі технології, як трафаретний друк та струменевий друк для осадження воску та струмопровідних фарб, дослідники Whitesides змогли створити паперові мікрофлюїдні датчики, хімічні датчики, датчики вологості та навіть датчики сили. Ви можете зігнути та скласти ці пристрої відповідно до мети, і коли вони виконали свій обов'язок, можете підпалити їх сірником або просто дозволити їм природним чином розкладатися. Оскільки ми розглядаємо мільярди датчиків та продуктів IoT, їхня здатність руйнуватися та повертатися до природи може запобігти переповненню сміттєзвалищ пристроями IoT.

Внаслідок великої гнучкості паперу та пластику утруднюється їхнє використання в датчиках, які зазвичай вимагають жорсткого контролю за розмірами. Наприклад, пластмаси розтягуються та деформуються за найменших змін сили чи температури. У напівпровідниковій галузі ми навчились контролювати розміри функцій до кількох нанометрів; розмір пластикової деталі може змінюватися на десятки мікрометрів лише через підвищення температури на кілька градусів. Нам знадобляться нові та різноманітні набори дизайнерських інструментів та виробничого обладнання, щоб розпочати масове виробництво датчиків із пластику та паперу.

Однак технології обробки паперу та пластику добре впроваджені десятиліттями, навіть століттями, в інших сферах застосування: друкарському верстаті та машині для оброблення паперу для виготовлення газет і книг; трафаретний друк та струменеві форсунки для точного друку; виготовлення рулонних пластиків для виготовлення стрічок і перев'язок; великоформатна літографія та тонкоплівкове осадження для виготовлення широкоформатних телевізійних панелей.

Коли у виробництві використовуються прості матеріали та методи, не потрібно працювати в приміщенні, майже такому ж складному, як і фабрика; могла бути фабрика в кожному місті чи навіть у кожному домогосподарстві. Це розподілена модель виготовлення. Уявити собі, що впродовж 20 років передові 3D-принтери також можуть виготовляти гнучкі датчики та електроніку вдома.

Звичайно, не всі сенсорні технології могли перейти на пластик або на папір. Високоефективні датчики, такі як мікрофони та гіроскопи, повинні залишатися на кремнію, оскільки їхні фізичні показники залежать від властивостей кремнію або допусків менше мікрометра, досягнутих лише за допомогою традиційної технології кремнію. Однак

більш прості пристрої, необхідні для споживачів IoT з великим об'ємом, такі як датчики температури, світла, газу або тиску, будуть переходити з кремнію на пластик або навіть папір.

Література

1. В. Петин. Создание умного дома на базе ARDUINO. М.: ДМК Пресс, 2018. 180 с.
2. Blaauw D., Sylvester D., Dutta P., Lee Y. at al. IoT Design Space Challenges: Circuits and Systems University of Michigan, Ann Arbor, MI, USA. URL: https://patpannuto.com/pubs/blaauw14iot.pdf 11.10.2019
3. https://iotconf.ru/ru/article/sovremennie-mikrochipi-dlya-interneta-veshchey-kakie-oni-65909
4. https://create.arduino.cc/projecthub/projects/tags/iot?page=2
5. Lytvyn V., Vysotska V., Peleshchak I., Peleshchak R., Kohut I. and Mykhailyshyn V. Intelligent System of a Smart House. 3rd IEEE International Conference on Advanced Information and Communication Technologies (AICT), 2019.

ЗМІСТ

Навчальне видання

Крижановський Володимир Григорович
Сергієнко Сергій Петрович

До практичної реалізації пристроїв Інтернету речей (IoT)

Навчально-методичний посібник

Редактор Д. В. Чернов
Технічний редактор Цяпало А.О.

Підписано до друку 21.11.2019
Формат 60 х 84/16. Папір офсетний.
Друк – цифровий. Умовн. друк. арк. 1,95
Тираж 10 прим. Зам.

Донецький національний університет імені Василя Стуса
21021, м. Вінниця, 600-річчя, 21
Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру серія ДК № 5945 від 15.01.2018