|  |  |  |
| --- | --- | --- |
| wordml://75.png |  | wordml://76.png |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | |  | | --- | | Imię i nazwisko studenta: Jan Nielek | | | |  | | --- | | Nr albumu: 139643 | | | |  | | --- | | Studia pierwszego stopnia | | | |  | | --- | | Forma studiów: niestacjonarne | | | |  | | --- | | Kierunek studiów: Informatyka | | | Specjalność/profil: - | |
|  |
|  |
|  |
| |  | | --- | | **PROJEKT DYPLOMOWY INŻYNIERSKI** | |
| |  | | --- | | Tytuł projektu w języku polskim: System monitorowania domu z wykorzystaniem układów ESP8266 i platformy dla systemów wbudowanych Raspberry Pi | | Tytuł projektu w języku angielskim: Home monitoring system using ESP8266 and Raspberry Pi platform | |  | |
| |  |  |  |  | | --- | --- | --- | --- | |  | |  | | --- | | Potwierdzenie przyjęcia projektu | | | |  | |  | | --- | | Opiekun projektu |   *podpis* | |  | | --- | | Kierownik Katedry/Zakładu (pozostawić właściwe) |   *podpis* | |  | dr inż. Krzysztof Bikonis |  | |
| |  | | --- | | Data oddania projektu do dziekanatu: | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  |
|  | **OŚWIADCZENIE** | | |
|  | |  |  | | --- | --- | | |  | | --- | | Imię i nazwisko: Jan Nielek  Data i miejsce urodzenia: 30.07.1992, Człuchów  Nr albumu: 139643 | | | |  | | --- | | Wydział: Wydział Elektroniki, Telekomunikacji i Informatyki  Kierunek: informatyka | | | |  | | --- | | Poziom studiów: pierwszy  Forma studiów: niestacjonarne | | | | |
|  | |  | | --- | | Ja, niżej podpisany(a), wyrażam zgodę/nie wyrażam zgody\* na korzystanie z mojego projektu dyplomowego zatytułowanego: System monitorowania domu z wykorzystaniem układów ESP8266 i platformy dla systemów wbudowanych Raspberry Pi  do celów naukowych lub dydaktycznych.1 | | | |
| |  |  | | --- | --- | | Gdańsk, dnia .................................. | .....................................................  *podpis studenta* | | | | |
|  | |  | | --- | | Świadomy(a) odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. z 2016 r., poz. 666 z późn. zm.) i konsekwencji dyscyplinarnych określonych w ustawie Prawo o szkolnictwie wyższym (Dz. U. z 2012 r., poz. 572 z późn. zm.),2 a także odpowiedzialności cywilno-prawnej oświadczam, że przedkładany projekt dyplomowy został opracowany przeze mnie samodzielnie.  Niniejszy projekt dyplomowy nie był wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem tytułu zawodowego.  Wszystkie informacje umieszczone w ww. projekcie dyplomowym, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.  Potwierdzam zgodność niniejszej wersji projektu dyplomowego z załączoną wersją elektroniczną. | | | |
| |  |  | | --- | --- | | Gdańsk, dnia .................................. | .....................................................  *podpis studenta* | | | | |
|  | Upoważniam Politechnikę Gdańską do umieszczenia ww. projektu dyplomowego w wersji elektronicznej w otwartym, cyfrowym repozytorium instytucjonalnym Politechniki Gdańskiej oraz poddawania jego procesom weryfikacji i ochrony przed przywłaszczaniem jego autorstwa. | | |
| |  |  | | --- | --- | | Gdańsk, dnia ................................. | .....................................................  *podpis studenta* | | | | |
|  | |  | | --- | | \*) niepotrzebne skreślić | | | |
| |  | | --- | |  | | | | |
| |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | 1 | | |  | | --- | | Zarządzenie Rektora Politechniki Gdańskiej nr 34/2009 z 9 listopada 2009 r., załącznik nr 8 do instrukcji archiwalnej PG. | | | |  | | --- | | 2 | | |  | | --- | | Ustawa z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym: | | |  | |  | | --- | | Art. 214 ustęp 4. W razie podejrzenia popełnienia przez studenta czynu podlegającego na przypisaniu sobie autorstwa istotnego fragmentu lub innych elementów cudzego utworu rektor niezwłocznie poleca przeprowadzenie postępowania wyjaśniającego. | | |  | |  | | --- | | Art. 214 ustęp 6. Jeżeli w wyniku postępowania wyjaśniającego zebrany materiał potwierdza popełnienie czynu, o którym mowa w ust. 4, rektor wstrzymuje postępowanie o nadanie tytułu zawodowego do czasu wydania orzeczenia przez komisję dyscyplinarną oraz składa zawiadomienie o popełnieniu przestępstwa. | | | | | |

**Streszczenie**

Internet Reczy jest coraz powszechniejszy w dzisiejszym świecie. W ramach projektu inżynierskiego został zaprojektowany system wpasowujący się w podstawowe założenia IoT. [TODO]

**Słowa kluczowe:** Internet Rzeczy, systemy wbudowane, [TODO]

**Dziedziny nauki i techniki zgodnie z wymogami OECD:** [TODO]

**Abstract**

Internet of Things is more and more prevailing in the current world. As part of the project, a home monitoring system was designed, that fits basic concepts of IoT. [TODO]

**Keywords:** Internet of Things, embedded systems [TODO]

Spis treści

[Spis treści 5](#_Toc504156976)

[1. Wykaz ważniejszych skrótów i oznaczeń 7](#_Toc504156977)

[2. Wstęp i cel pracy 8](#_Toc504156978)

[3. Analiza projektowa w odniesieniu do stanu wiedzy 9](#_Toc504156979)

[3.1. Medium transmisyjne sensor – serwer 9](#_Toc504156980)

[3.1.1. Kryterium poboru mocy 9](#_Toc504156981)

[3.1.2. Kryterium powszechności użycia 9](#_Toc504156982)

[3.1.3. Kryterium kosztów implementacji 9](#_Toc504156983)

[3.2. Wybór mikrokontrolera 10](#_Toc504156984)

[3.3. Protokół transmisji danych pomiędzy sensorami a serwerem 10](#_Toc504156985)

[3.3.1. Własny protokół w oparciu o TCP/IP 10](#_Toc504156986)

[3.3.2. Interfejs REST 10](#_Toc504156987)

[3.3.3. MQTT 10](#_Toc504156988)

[3.4. Architektura aplikacji serwerowej 11](#_Toc504156989)

[3.5. Platforma sprzętowa serwera 11](#_Toc504156990)

[4. Implementacja 12](#_Toc504156991)

[4.1. Sensor temperatury i wilgotności 12](#_Toc504156992)

[4.1.1. Schemat blokowy urządzenia 12](#_Toc504156993)

[4.1.2. Projekt obwodu drukowanego 12](#_Toc504156994)

[4.1.3. Produkcja prototypu 13](#_Toc504157002)

[4.1.4. Oprogramowanie ESP8266 13](#_Toc504157003)

[4.2. Serwer 16](#_Toc504157004)

[5. Wyniki testowania 17](#_Toc504157005)

[6. Podsumowanie 18](#_Toc504157006)

[Bibliografia 19](#_Toc504157007)

[Spis rysunków 20](#_Toc504157008)

[Dodatek A: Schemat ideowy sensora temperatury 21](#_Toc504157009)

[21](#_Toc504157010)

[Dodatek B: 22](#_Toc504157011)

1. Wykaz ważniejszych skrótów i oznaczeń

IoT – ang. *Internet of Things* Internet Rzeczy;  
WiFi – ang. *Wireless Fidelity* sieć bezprzewodowa

[TODO]

1. Wstęp i cel pracy

Internet Rzeczy jest jedną z najszybciej rozwijających się gałęzi sektora systemów wbudowanych. Wydatną jego częścią są tzw. Inteligentne Domy. Pod tym pojęciem mieści się szeroka gama urządzeń monitorujących wiele parametrów takich jak temperatura, wilgotność czy też nawet nasłonecznienie. Korzystając z tych danych, system może odpowiednio sterować ogrzewaniem tak aby poprawić komfort mieszkańców przy równoczesnym zmniejszeniu zużycia energii.

W poniższej pracy skupiono się na monitorowaniu parametrów przy użyciu zasilanych bateryjnie mikrokontrolerów. Zaprojektowany został przykładowy moduł współpracujący z serwerem. Komunikacja odbywała się z wykorzystaniem sieci Wi-Fi. Pomiary były zapisywane do bazy danych w celu późniejszego zaprezentowania ich użytkownikowi poprzez stronę WWW.

1. Analiza projektowa w odniesieniu do stanu wiedzy

Systemy automatyki domowej są coraz popularniejsze. Są wykorzystywane do zwiększenia komfortu 2i poprawy bezpieczeństwa użytkowników. Inteligentny Dom jest w stanie ułatwić życie ludziom starszym lub z niepełnosprawnością i zmniejszyć nakład pracy poprzez zautomatyzowanie pewnych zadań. System może być zaprojektowany z wykorzystaniem centralnego kontrolera, który steruje i monitoruje wiele niezależnych urządzeń takich jak gniazda zasilania, oświetlenie, czujniki temperatury i wilgotności, czujniki dymu, gazu i ognia czy też urządzeń alarmowych. Jedną z największych zalet automatyki domowej jest to, że można nią łatwo sterować za pomocą całej gamy urządzeń poczynając od telefonów, tabletów czy też komputerów PC. Szybki rozwój technologii bezprzewodowych pozwala na zdalne zarządzanie systemem. W podrozdziałach przedstawiono selekcję najbardziej odpowiednich technologii i rozwiązań wykorzystanych w projekcie. Selekcji dokonano na podstawie aktualnego stanu wiedzy.

* 1. Medium transmisyjne sensor – serwer

Podstawowym problemem, z którym zmierzyć musi się projektant sieci sensorów jest sposób komunikacji pomiędzy składowymi systemu. Zasadniczego podziału można dokonać na komunikację przewodową i bezprzewodową. Ze względu na mobilność i łatwość w realizacji w projekcie wykorzystano wariant bezprzewodowy. Na rynku dominują trzy standardy: Bluetooth, WiFi (ang. *WirelessFidelity*) i ZigBee. Selekcji najodpowiedniejszego z nich do projektu dokonano wg. trzech kryteriów: poboru mocy, powszechności użycia i kosztów implementacji.

* + 1. Kryterium poboru mocy

Wprowadzony na rynek standard Bluetooth Low Energy pozwala na zastosowanie urządzeń zasilanych z baterii pastylkowych które będą działać nawet ponad 10 lat [1]. ZigBee także oferuje niskie zużycie energii, na poziomie 2 lat zasilania z wykorzystaniem dwóch baterii AA. WiFi nie jest energooszczędne, jednak przy zastosowaniu odpowiednich trybów uśpienia i oszczędzania energii można osiągnąć półroczny czas pracy bez ładowania.

* + 1. Kryterium powszechności użycia

Obecnie, każde gospodarstwo domowe podłączone do sieci Internet posiada punkt dostępu do sieci WiFi. Dzięki powszechności tego rozwiązania dodanie nowych urządzeń do systemu jest stosunkowo łatwe. Bluetooth i ZigBee wymagają utworzenia nowej sieci i podłączenia do niej serwera a także z racji innych protokołów tunelowania do sieci IP.

* + 1. Kryterium kosztów implementacji

Obecnie na rynku dostępne są zarówno tanie mikrokontrolery wspierające sieć WiFi (np. ESP8266 firmy Espressif) jak i mikrokontrolery wspierające sieć Bluetooth (nRF52 firmy Nordic Semiconductors). Rozwiązania wspierające sieć ZigBee są **na ogół droższe**, co przesądziło o odrzuceniu tego standardu w poniższym projekcie. Na rzecz WiFi przemawia możliwość wykorzystania istniejącej infrastruktury.

W odniesieniu do powyższych podpunktów, zadecydowano o wykorzystaniu WiFi jako medium transmisyjnego pomiędzy sensorami a serwerem.

* 1. Wybór mikrokontrolera

Obecnie większość z producentów ma w ofercie mikrokontroler wspierający sieć WiFi. Są to na przykład Particle Photon (rdzeń ARM Cortex-M3), WiFiMCU (Cortex-M4) czy też układ firmy Espressif ESP8266 (Tensilica L106) [2]. Spośród wymienionych, ten ostatni jest charakteryzuje się najniższą ceną. Do innych jego zalet należy wsparcie dla biblioteki Arduino [3], stosunkowo niewielka ilość zewnętrznych elementów wymaganych do uruchomienia [4] czy też wsparcie dla trybu głębokiego uśpienia (*Deep Sleep,* [2, p. 19]). Z tych powodów, został on wybrany jako główny układ sterujący dla sensorów.

* 1. Protokół transmisji danych pomiędzy sensorami a serwerem

Ze względu na wybór sieci WiFi jako medium transmisyjnego, komunikacja pomiędzy składowymi systemu może odbywać się w oparciu o warstwę sieciową modelu OSI (warstwy internetu w modelu TCP/IP). W takim przypadku można wykorzystać unikalny adres MAC każdego z sensorów do jego identyfikacji. Obecnie w przemyśle nie istnieje jeden standard dla komunikacji, wielu producentów korzysta z opracowanych już protokołów takich jak np. MQTT (ang. *Message Queuing Telemetry Transport*, transport telemetrii w oparciu o kolejkowanie wiadomości), organizacje takie jak Open Mobile Alliance próbują wdrożyć własne standardy a jeszcze inni producenci wdrażają własnościowe, zamknięte protokoły [5].

* + 1. Własny protokół w oparciu o TCP/IP

Pomimo wielu korzyści własnej implementacji takich jak pełna kontrola nad ilością danych, ścisłe dopasowanie do sprzętu czy też możliwe najniższe zużycie energii, takie rozwiązanie posiada wiele wad. Wśród nich wyróżnić można: czas pracy potrzebny na prawidłowe zaprojektowanie i przetestowanie takiego protokołu czy też potencjalne problemy w przyszłości przy zmianie architektury systemu. Z tego względu to rozwiązanie zostało odrzucone na etapie planowania projektu.

* + 1. Interfejs REST

Model REST (ang. *Representational State Transfer*, zmiana stanu poprzez reprezentację) to sposób na zapewnienie bezstanowej komunikacji za pomocą zorganizowanego i jawnie określonego zbioru operacji [6]. Znajduje on wykorzystanie głównie w aplikacjach webowych. Ze względu na to, że komunikacja odbywa się przy pomocy protokołu HTTP (ang. *Hypertext Transfer Protocol*, protokół transmisji hipertekstu) świetnie sprawdza się przy interakcji na stronach WWW. Niestety, wykorzystanie HTTP wiąże się z zwiększeniem ilości wysyłanych danych ze względu na dodatkowe nagłówki [7, pp. 30-32]. Z tego powodu, nie jest on idealny dla systemów wbudowanych.

* + 1. MQTT

MQTT to standard bardzo prostego protokołu transmisji danych [8]. Został on zaprojektowany specjalnie dla sieci M2M (ang. *Machine to machine*, interfejs maszyna-maszyna) czy też IoT. Charakteryzuje się niskim zapotrzebowaniem na energię, prostotą tworzenia i odbierania wiadomości.  
Na jego niekorzyść świadczy to, że wymaga do poprawnego działania zewnętrznego brokera wiadomości.

Ze względu na istniejące już biblioteki wspierające protokół MQTT na mikrokontroler ESP8266,   
a także łatwość implementacji jego obsługi po stronie serwera, został on wykorzystany w projekcie

* 1. Architektura aplikacji serwerowej

W założeniach projektowych, serwer powinien obsługiwać trzy funkcjonalności, mianowicie: odbiór danych z sensorów, zapis i odczyt z bazy danych a także prezentację odczytów użytkownikowi. W celu możliwie największej przenośności na inne architektury sprzętowe założono wykorzystanie języka Python w wersji 3.6. Umożliwia to zarówno uruchomienie aplikacji serwerowej na systemach Windows, czy też Linux z procesorami z rodzin ARM i x86. Python oferuje wiele platform programistycznych do wdrażania aplikacji webowych, jedną z nich jest Flask [9]. Flask jest określany przez twórców jako mikroframework,   
i nie wymaga dużych zasobów sprzętowych jak i nakładu pracy, aby uruchomić na nim prostą stronę WWW. Na jego korzyść świadczy także duża ilość dodatkowych bibliotek rozszerzających jego funkcjonalność. Temat konfiguracji, uruchomienia i zewnętrznych bibliotek dla serwera został szerzej opisany w rozdziale Implementacja.

Z racji wykorzystania w projekcie protokołu MQTT, potrzebny był broker wiadomości. W sieci dostępnych jest wiele darmowych brokerów [10], jednakże z racji ograniczenia się do jednej sieci LAN (ang. *Local Area Network*, lokalna sieć komputerowa) uruchomiono własnego brokera: mosquitto [11], równolegle do działającego serwera głównego aplikacji. Pozwoliło to na zwiększenie bezpieczeństwa jak i kontroli nad całą siecią sensorów, a także ułatwiło diagnostykę poprzez dostęp do pełnych logów serwera.

Jako serwer bazy danych wykorzystano popularny silnik MySQL [12],

* 1. Platforma sprzętowa serwera

Zgodnie z założeniami projektowymi jako serwer służyć może dowolna współczesna architektura sprzętowa która wspiera systemy Windows lub Linux. Z tego względu jako platforma deweloperska wykorzystany został system Windows (procesor x86), co pozwoliło na szybkie prototypowanie i testowanie kodu lokalnie. Jako serwer produkcyjny wykorzystany został mikrokomputer Raspberry Pi (procesor ARM) z systemem Raspian (dystrybucja Linuksa oparta o Debiana). Pozwoliło to znacząco zaoszczędzić koszty utrzymania całego systemu ze względu wielokrotnie mniejsze zużycie energii względem komputera PC.

1. Implementacja

Projekt można podzielić na dwie części – sensory i serwer. Całość została zaprojektowana na przełomie czerwca i lipca 2017, po czym dopisywano na bieżąco co było potrzebne.

* 1. Sensor temperatury i wilgotności

Jako przykładowy układ, zaprojektowany został zasilany bateryjnie sensor temperatury i wilgotności wykorzystujący układ Si7021 [13], komunikujący się z mikrokontrolerem ESP8266 za pomocą interfejsu I2C. Sensor został także wyposażony w układ ładowania ogniw litowo-polimerowych lub litowo-jonowych [14]. Ładowanie odbywa się z wykorzystaniem złącza microUSB. Program kontrolujący pracę mikrokontrolera został napisany w języku C++, z wykorzystaniem biblioteki Arduino.

* + 1. Schemat blokowy urządzenia

Pierwsze prototypy sensora powstały z wykorzystaniem płytki stykowej, jednakże nie było to rozwiązanie praktyczne, szczególnie przy długotrwałych pomiarach np. rozładowywania akumulatora. W tym celu został zaprojektowany schemat blokowy, który został potem wykorzystany do stworzenia płytki drukowanej. Schemat został umieszczony jako Dodatek A.

W celu łatwego prototypowania, wyprowadzone zostały następujące interfejsy: UART, I2C i SPI. Ponadto, aby ułatwić programowanie, wyprowadzony został IO0 jako specjalny pin, za pomocą którego bootloader decyduje o tym w jakim trybie ma zostać uruchomiony ESP8266 po restarcie. W przypadku, kiedy GPIO0 będzie w stanie niskim (zero logiczne), bootloader będzie oczekiwał na aktualizację oprogramowania przez połączenie szeregowe, w przeciwnym przypadku uruchomi się normalnie. Na szczególną uwagę zasługuje także podłączenie pinu IO16 do RST. ESP8266 oferuje tryb tak zwanego głębokiego uśpienia, w którym drastycznie ogranicza zużycie prądu. Podłączenie pinu IO16 do RST pozwala na wybudzenie mikrokontrolera z tego trybu. Jedyne wejście analogowe dostępne w ESP8266, czyli pin ADC pozostawiono niepodłączone. Było to działanie celowe, ze względu na możliwość pomiaru napięcia zasilania poprzez wewnętrzne podłączenie tego pinu do zasilania układu [2, p. 17].

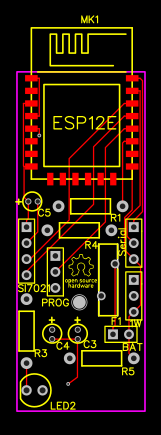
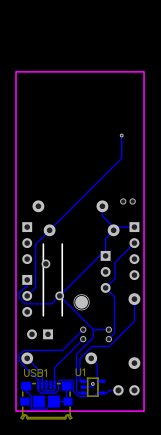
Na schemacie blokowym znajduje się także układ scalony MCP73831 firmy Microchip. Zastosowano standardową aplikację, proponowaną w dokumentacji układu. Pozwala ona na ładowanie podłączonego akumulatora bezpośrednio poprzez zasilanie z portu USB. W celu zabezpieczenia baterii przed zwarciem i w konsekwencji niebezpieczeństwem wystąpienia pożaru, zastosowano bezpiecznik polimerowy.

Celowo zrezygnowano z zastosowania stabilizatora napięcia zasilania, po tym jak eksperymentalnie dowiedziono, że zarówno ESP8266 jak i Si7021 są w stanie w pełni poprawnie pracować nawet przy napięciu zasilania wynoszącym 4.2 V (szczytowe napięcie ładowania akumulatorów litowo jonowych [14]).

* + 1. Projekt obwodu drukowanego

Kolejnym krokiem po wykonaniu schematu blokowego było zaprojektowanie prototypu PCB (ang. *printed circut board*, płytka drukowana). Powstały dwie wersje, różniące się rozmiarem i zastosowaną technologią montażu. Ze względu na małoseryjną produkcję i brak dostatecznej umiejętności lutowania w technologii SMT (ang. *surface-mount technology*, montaż powierzchniowy*)*, ostatecznie wyprodukowano prototyp o większej powierzchni, z większością elementów montowaną w technologii THT. Niestety, nie wszystkie elementy posiadały wariant przystosowany do montażu THT. Mikrokontroler ESP8266, układ MCP73831 i gniazdo micro USB zostały zamontowane w technologii SMT.

PCB jest dwustronne, raster wyprowadzeń to 2.54 mm (Rys. 4.1). Wymiary płytki z zamontowanymi elementami (20 mm na 60 mm) odpowiadają wymiarom koszyka na jedno ogniwo litowo jonowe, co pozwala na wygodne ukrycie płytki za baterią.



B)

A)

Rys. 4.1 Projekt PCB: A) strona górna B) strona dolna

4. 1. 3. Produkcja prototypu

W celu zapewnienia jak najlepszej jakości wykonania zwrócono się do zewnętrznej firmy zajmującej się produkcją obwodów drukowanych. Dzięki temu, możliwe stało się uzyskanie takich detali jak druk dwustronny, metalizacja otworów czy warstwa opisowa, co było by bardzo trudne przy samodzielnej produkcji amatorskimi metodami.

* + 1. Oprogramowanie ESP8266

Program kontrolujący pracę serwera nie jest skomplikowany, w dużej mierze dzięki zastosowaniu zewnętrznych bibliotek odpowiedzialnych za obsługę WiFi (ESP8266WiFi.h), komunikację z czujnikiem Si7021 (Adafruit\_Si7021.h), protokół MQTT (PubSubClient.h) czy też formatowanie wiadomości do JSON (ang. *JavaScript object notation*, lekki format wymiany danych komputerowych, bazujący na podzbiorze języka JavaScript) (ArduinoJson.h). Uproszczony schemat działania programu znajduje się na rys.4.2. Z racji wykorzystania trybu DeepSleep, program nie wykonuje się w nieskończonej pętli jak jest to zazwyczaj implementowane w mikrokontrolerach, tylko nieustannie resetuje się pod koniec wykonania. Spowodowane jest to implementacją tego trybu uśpienia w ESP8266. Przy wywołaniu metody deepSleep mikrokontroler wyłącza większość wewnętrznych modułów, pozostawiając tylko uruchomiony timer, który po określonym czasie zresetuje mikrokontroler poprzez ustawienie pinu GPIO0 w stan wysoki. Do poprawnego działania tego mechanizmu wymagane jest zewnętrzne podłączenie pinu GPIO0 do pinu RST mikrokontrolera.

W celu utrzymania jednolitego kodu zarówno do długotrwałych testów pracy na baterii jak i testów poprawnego wykonania kodu zdecydowano się na zastosowanie makr pozwalających na usunięcie z kodu niepotrzebnych fragmentów na etapie preprocesora. W ten sposób, gdy nie zostanie zdefiniowane marko DEBUG, preprocesor podmieni wywołania metod wypisujących dane diagnostyczne poprzez połączenie szeregowe na puste instrukcje. Pozwala to na zmniejszenie zużycia energii, gdy nie jest to konieczne już na etapie kompilacji programu.

Procesor odczytuje temperaturę i wilgotność z czujnika Si7021 przy pomocy protokołu I2C. Obsługa protokołu jak i samego odczytu wiadomości jest już zaimplementowana w bibliotekach zewnętrznych. Wskazania czujnika można odczytać przy pomocy dwóch metod readTemperature i readHumidity.

W ramach projektu odczytywana jest także podstawowa telemetria układu. Wykorzystano możliwość wewnętrznego pomiaru napięcia zasilania poprzez jedyny dostępny ADC (ang. *analog to digital converter*, przetwornik analogowo cyfrowy). Aby zainicjalizować przetwornik, wymagane jest zastosowanie makra ADC\_MODE w którym należy zadeklarować w jakim trybie będzie on pracował. W celu wykorzystania wewnętrznego połączenia zasilania z ADC, należy skorzystać z trybu ADC\_VCC. Odczyt zasilania wykonuje funkcja getVcc. Poza napięciem zasilania, odczytywana jest także ilość pozostałej pamięci. W przypadku tego sensora nie jest to bardzo istotna informacja, jako że układ przechodzi nieustanne resetowanie pamięci w związku z trybem uśpienia. Jednakże, gdyby zaszła potrzeba zastosowania czujnika pracującego w sposób ciągły, odczyt mógłby wskazać wycieki pamięci jeszcze na etapie testowania programu, przed ewentualnym zatrzymaniem działania procesora w wyniku wyczerpania się dostępnej pamięci. Odczyt tego parametru odbywa się w funkcji getFreeHeap.

Jako unikalny identyfikator dla każdego z sensorów wykorzystano jego adres MAC (ang. *medium access control address*). Aby umożliwić generyczne programowanie wielu czujników tym samym oprogramowaniem, skorzystano z odczytu adresu MAC w locie. Zapewnione jest to przez funkcję macAddress, która zwraca adres w formie tablicy sześciu bajtów. W celu późniejszego wykorzystania, adres jest konwertowany do tablicy znaków. Całość odbywa się w funkcji setup\_wifi.

Według założeń projektowych, wszystkie sensory powinny znajdować się w jednej sieci WiFi. Z tego względu zadecydowano się na umieszczenie danych dostępowych do sieci na stałe w kodzie programu. Jednak ze względów bezpieczeństwa nie jest to optymalne rozwiązanie. Kwestia bezpieczeństwa została szerzej poruszona w rozdziale Podsumowanie.

reset

Zainicjalizuj połączenie wifi

WiFi  
OK?

Zainicjalizuj połączenie mqtt

MQTT OK?

Odczytaj parametry

Utwórz wiadomość

Wyślij dane

Wejdź w stan uśpienia

Czekaj 0.5 s.

Czekaj 1 s.

nie

tak

nie

tak

Rys. 4.2 Uproszczony schemat działania programu na mikrokontrolerze

* 1. Aplikacje wspierające

Na aplikację serwerową zasadniczo składają się trzy osobne aplikacje: baza danych MySQL, broker wiadomości MQTT i właściwy serwer obsługujący całość, opisany w podrozdziale 4.2

* + 1. Baza danych

W każdej aplikacji zajmującej się przetwarzaniem danych baza danych jest podstawowym komponentem. Ze względu na obiektowy charakter aplikacji głównej, zadecydowano o zastosowaniu ORM (ang. *Object-Relational Mapping*, mapowanie obiektowo-relacyjne). Użyte zostało narzędzie o nazwie SQLAlchemy, które udostępnia mechanizmy pozwalające na mapowanie obiektów do relacji, a także szereg innych operacji ułatwiających korzystanie z języka SQL (ang. *Structured Query Language*, strukturalny język zapytań). Aby w pełni wykorzystać możliwości ORM w aplikacji opartej o framework Flask, użyto rozszerzenia Flask-SqlAlchemy [15]

Na początku prac nad projektem zastosowano bazę danych SQLite, ze względu na łatwość jej użycia i brak zewnętrznych komponentów. Niestety, wraz ze wzrostem ilości danych wymaganych do przetworzenia, wydajność SQLite stała się niewystarczająca. Dzięki wykorzystaniu ORM, zmiana silnika bazy danych na MySQL ograniczyła się jedynie do konfiguracji i migracji danych na nowy system, bez zmian w samym kodzie aplikacji.

* + 1. Broker MQTT

Jako broker wiadomości MQTT, wykorzystana została aplikacja Mosquitto [11]. Wykorzystanie lokalnego serwera miało wiele korzyści, szczególnie na początku rozwoju projektu, gdy dostęp do logów aplikacji pozwalał na łatwe zdiagnozowanie wszelkich błędów. Razem z aplikacją główną dostarczany jest szereg narzędzi ułatwiających jego wykorzystanie. Szczególnie pomocne okazały się następujące programy: mosquitto\_pub i mosquitto\_sub które pozwalały na testowanie i podglądanie w locie danych wysyłanych przez urządzenia.

Mosquitto pozwala także na szyfrowanie wiadomości z wykorzystaniem protokołu TLS (ang. *transport layer security*, zabezpieczenie warstwy transportowej). Wykorzystanie tej funkcjonalności byłoby konieczne w przypadku rzeczywistego systemu dostępnego na rynku. W projekcie zrezygnowano z implementacji.

* 1. Serwer

Główna aplikacja jest najbardziej rozbudowaną częścią całego projektu. W ramach jednego programu połączona została obsługa nadchodzących wiadomości poprzez sieć MQTT, prezentacja danych pobranych z bazy, kontrola dostępu użytkowników jak i panel administracyjny. Kod poszczególnych funkcjonalności podzielono na osobne moduły, o ile było to możliwe. Z racji wykorzystania frameworku Flask, strony WWW są renderowane dynamicznie, przy pomocy silnika szablonów Jinja2. W celu ułatwienia analizy pracy aplikacji i ewentualnego znajdywania błędów zastosowano moduł logów dostępny w bibliotece standardowej języka Python.

* + 1. Komunikacja z sensorami

No przez mqtt nie

* + 1. Wyświetlanie danych

No przez dżawaskript

* + 1. Kontrola dostępu

Chuj nie kontrola

* + 1. Panel administracyjny

Chuj nie admin

1. Wyniki testowania

Działa kurwa elegancko

1. Podsumowanie

Nigdy więcej prac pisemnych

Bibliografia

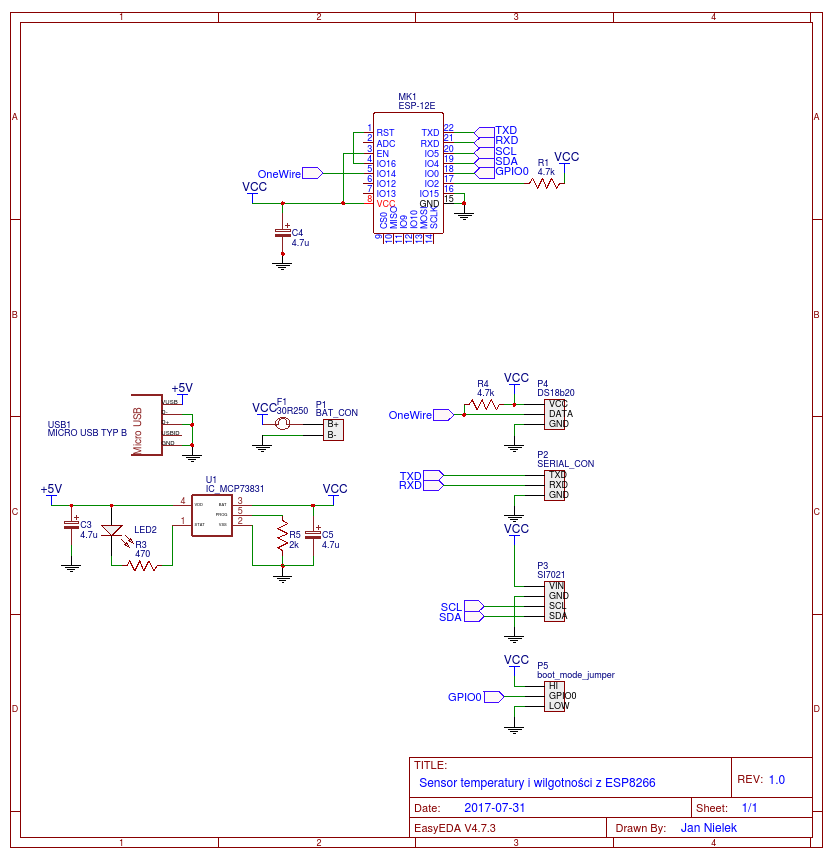
|  |  |
| --- | --- |
| [1] | \*. J. O. 2. Carles Gomez 1, „Overview and Evaluation of Bluetooth Low Energy:An Emerging Low-Power Wireless Technology,” *Sensors,* pp. 8-9, 2012. |
| [2] | Espressif, „ESP8266EX Datasheet,” 2017. [Online]. Available: http://espressif.com/sites/default/files/documentation/0a-esp8266ex\_datasheet\_en.pdf. [Data uzyskania dostępu: 15 01 2018]. |
| [3] | I. Grokhotkov, 2017. [Online]. Available: http://arduino-esp8266.readthedocs.io/en/latest/reference.html. [Data uzyskania dostępu: 15 01 2018]. |
| [4] | I. Grokhotkov, „Minimal ESP8266 hardware setup,” 2107. [Online]. Available: http://arduino-esp8266.readthedocs.io/en/latest/boards.html#minimal-hardware-setup-for-bootloading-and-usage. [Data uzyskania dostępu: 15 1 2018]. |
| [5] | Postscapes, 2017. [Online]. Available: https://www.postscapes.com/internet-of-things-protocols/. [Data uzyskania dostępu: 15 01 2018]. |
| [6] | R. N. T. Roy T. Fielding, „Principled Design of the Modern,” *ACM Transactions on Internet Technology,* tom 2, nr 2, pp. 115-150, 2002. |
| [7] | F. e. al., „Hypertext Transfer Protocol -- HTTP/1.1,” *Standards Track,* 1999. |
| [8] | OASIS, „Oasis Open,” 10 12 2015. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf. [Data uzyskania dostępu: 15 01 2018]. |
| [9] | A. Ronacher, „Flask,” 2018. [Online]. Available: http://flask.pocoo.org/. [Data uzyskania dostępu: 15 01 2018]. |
| [10] | D. Plentz, „MQTT Brokers,” [Online]. Available: https://github.com/mqtt/mqtt.github.io/wiki/servers. [Data uzyskania dostępu: 15 01 2018]. |
| [11] | Eclipse, „Mosquitto,” 2018. [Online]. Available: https://mosquitto.org/. [Data uzyskania dostępu: 15 1 2018]. |
| [12] | Oracle, „MySQL,” 2018. [Online]. Available: https://www.mysql.com/. [Data uzyskania dostępu: 15 01 2018]. |
| [13] | Silicon Labs, „Si7021-A20,” 8 2016. [Online]. Available: https://www.silabs.com/documents/public/data-sheets/Si7021-A20.pdf. [Data uzyskania dostępu: 15 01 2018]. |
| [14] | Mircochip, „MCP73831,” 2008. [Online]. Available: https://www.sparkfun.com/datasheets/Prototyping/Batteries/MCP73831T.pdf. [Data uzyskania dostępu: 15 01 2018]. |

Spis rysunków

[Rys. 4.1 Projekt PCB: A) strona górna B) strona dolna 12](file:///C:\Users\yahu\Documents\Visual%20Studio%202017\Projects\FUSS\docs\projekt.docx#_Toc504157015)

[Rys. 4.2 Uproszczony schemat działania programu na mikrokontrolerze 14](file:///C:\Users\yahu\Documents\Visual%20Studio%202017\Projects\FUSS\docs\projekt.docx#_Toc504157016)

Dodatek A: Schemat ideowy sensora temperatury



Dodatek B: