JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 12 (2019) 232-239
Wysłane: 2019-06-14
Przyjęte: 2019-06-22

Autorski system inteligentnego budynku w porównaniu z rozwiązaniem otwarto - źródłowym

Cezary Kryczka*
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Artykuł ten jest próbą odpowiedzi na pytanie: czy i w jakich warunkach, opłacalne jest opracowanie autorskiego systemu inteligentnego budynku, kiedy dostępnych jest wiele bezpłatnych systemów o otwartym kodzie źródłowym. Publikacja przedstawia charakterystykę autorskiego systemu automatyki domowej – sHome, a także systemu open-source – Domoticz, w konfiguracji realizującej możliwie najbardziej zbliżoną do funkcjonalności systemu autorskiego. Pracę kończy analiza porównawcza systemów oraz wnioski z przeprowadzonej analizy.

Słowa kluczowe: inteligentny budynek; automatyka domowa; system autorski; system otwarto-źródłowy.

*Autor do korespondencji.

Adres e-mail: cezary@kryczka.eu

Own system of intelligent building in comparison with an open - source solution

Cezary Kryczka*
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. This article is an attempt to answer the question whether and under what conditions it is beneficial to develop an own intelligent building system, when many free open source systems are available. The publication presents the characteristics of author's own home automation system - sHome, as well as the open-source system - Domoticz, in a configuration that is as close to the functionality of the author's system as possible. The work ends with a comparative analysis of the systems and conclusions from the analysis.

Keywords: intelligent building; smart home; home automation; own system; open-source system.

*Corresponding author.

E-mail address: cezary@kryczka.eu

1. Wstęp

Artykuł ten jest próbą odpowiedzi na pytanie: czy i w jakich warunkach, opłacalne jest opracowanie autorskiego systemu inteligentnego budynku, kiedy dostępnych jest wiele bezpłatnych systemów o otwartym kodzie źródłowym, które można wykorzystać do realizacji nawet bardzo rozbudowanych instalacji.

Ponieważ autor artykułu nie odnalazł żadnego rzetelnego opracowania na temat popularności systemów open-source w inteligentnych budynkach, przeprowadził samodzielny przegląd dostępnych w Internecie artykułów dotyczących takich systemów i na tej podstawie określił jakie rozwiązania są popularne oraz który z systemów jest najczęściej opisywany. Metody wybrane przez autora do zbadania powyższych zagadnień to przegląd artykułów oraz eksperyment, polegający na budowie autorskiego systemu i późniejszym porównaniu go z najpopularniejszym rozwiązaniem open-source.

Autor zakłada, że poprzez budowę i analizę nawet prostego systemu automatyzującego sterowanie urządzeniami w domu oraz porównanie go z istniejącymi rozwiązaniami można przeprowadzić szereg analiz i wyciągnąć wnioski z uruchomienia takich systemów.

2. Definicja domu inteligentnego

Określeniem „inteligentny dom” można określić zarówno prosty system monitorujący podstawowe parametry jak temperatura, czy stan czujników oraz pozwalający na zdalną kontrolę odbiorników prądu, ale również rozbudowany system oparty o sztuczną inteligencję i sieci neuronowe, który będzie umożliwiał nie tylko sterowanie za pomocą dedykowanego interfejsu ale również będzie w stanie prowadzić swobodną konwersację w języku naturalnym [1]. Cechą wspólną dla wszystkich systemów automatyki jest to, że każdy z nich do prawidłowego działania potrzebuje czujników i elementów wykonawczych, którymi będzie mógł sterować.

3. Przegląd rozwiązań open-source

Na podstawie dokonanego przeglądu artykułów na stronach internetowych, które określają ogólną popularność systemów automatyki domowej oraz prognoz rynkowych, powołując się na badania firm z tej branży lub branż pokrewnych, autor określił najbardziej popularne systemy inteligentnych budynków oferowane na licencji open-source, przedstawione w tabeli 1 [5-14]:

Tabela 1. Ranking popularności systemów open-source

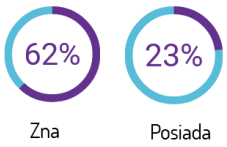
Lp.	Nazwa systemu	Ilość wystąpień
1	Domoticz	10
2	OpenHAB	10
3	Home Assistant	9
4	Calaos	7
5	OpenMotics	6
6	MisterHouse	3
7	ioBroker	2
8	Jeedom	2
9	LinuxMCE	2
10	OpenNetHome	2
11	PiDome	2
12	EventGhost	1
13	Fhem	1
14	FreeDomotic	1
15	Guardian Project	1
16	MajorDoMo	1
17	Mycontroller	1
18	Mycroft	1
19	MyPi	1
20	Node-RED	1
21	OpenRemote	1
22	Pimatic	1
23	Pytomation	1
24	Smarthomatic	1

W analizowanych artykułach wymienione były 24 systemy, z czego dwa, Domoticz i OpenHAB, występowały w każdej publikacji. Home Assistant to nieznacznie mniej popularny system, wystąpił on w 9 na 10 artykułów.

Według danych Komisji Europejskiej z 2018 roku, obecnie w Unii Europejskiej jest około 8,5 mln inteligentnych domów i apartamentów, na rok 2021 szacuje się liczbę 80 mln (Rys. 1.), co oznacza niemal dziesięciokrotny wzrost popularności takich rozwiązań w okresie 3 lat [2]. Trudno przewidzieć jaką część z nich będą stanowiły rozwiązania open-source, szczególnie, że znakomita większość tych rozwiązań dostarcza tylko oprogramowanie, a cześć sprzętową użytkownicy muszą uruchomić samodzielnie korzystając np. z popularnych mikrokomputerów jak Raspberry Pi.

Rys. 1. Prognozy rynkowe [2]

Według badań serwisu Oferteo.pl przeprowadzonych wśród osób, które w 2017 roku budowały dom, rozwiązania inteligentnych budynków są znane 62% ankietowanych. 23% pytanych osób zdecydowało się na wdrożenie takich rozwiązań (Rys. 2) [6].



Rys. 2. Znajomość systemów inteligentnych domów [6]

Badania przeprowadzone dla firmy Somfy pokazują, że 31% respondentów planuje posiadanie urządzeń będących elementem inteligentnych budynków. Eksperci uważają, że wraz z rosnącą satysfakcją z użytkowania takich systemów będzie przybywało jej zwolenników i posiadaczy. Wśród badanych osób tylko 15% użytkowników nie jest zadowolonych z posiadania takich rozwiązań, natomiast 62% jest z nich zadowolonych [8].

W serwisie github.com, w którym przechowywany może być kod źródłowy aplikacji open-source, zapytanie „smart home" zwraca ponad 7000 wyników, a zapytanie „home automation system" ponad 1300 wyników. Świadczy to o tym, że istnieje wiele otwarto źródłowych systemów inteligentnych budynków. Oczywiście duża część wyników dotyczy dodatkowych modułów do istniejących rozwiązań, jednak zakładając, że 10% to unikatowe systemy daje to liczbę około 800 systemów open-source.

4. Kryteria oceny badanych systemów

Aby rzetelnie ocenić, czy w obecnych warunkach opracowywanie autorskiego systemu inteligentnego budynku jest uzasadnione, należy wyznaczyć kryteria oceny, które pozwolą wyciągnąć wnioski, na temat okoliczności w jakich opracowanie takiego systemu jest racjonalne.

4.1. Kryterium czasu opracowania

W tym kryterium analizowany był czas potrzeby do zbudowania i uruchomienia autorskiego projektu, przy założeniu, że autor ma podstawową wiedzę w zakresie posługiwania się przynajmniej jednym językiem programowania. Czas stworzenia systemu autorskiego został porównany z czasem uruchomienia wybranego systemu open-source o podobnej funkcjonalności.

4.2. Kryterium bezpieczeństwa aplikacji i komunikacji

Analiza przeprowadzona na potrzeby niniejszej publikacji ma charakter bardzo uproszczonego badania na wąski zakres podatności. Na potrzeby niniejszego artykułu aplikacje zostały przebadane pod kątem podatności na najpopularniejsze od wielu lat zagrożenie dla aplikacji internetowych tj. SQL Injection [21, 22].

W kontekście bezpieczeństwa komunikacji systemy zostały przebadane pod kątem wykorzystanych mediów i protokołów zabezpieczających transmisję danych wewnątrz

systemu oraz komunikację z użytkownikiem za pośrednictwem dedykowanego interfejsu.

4.3. Kryterium skalowalności

Badanie aplikacji pod względem skalowalności polegało na porównaniu jakie możliwości rozbudowy i dostosowania systemu do innego wdrożenia oferuje rozwiązanie autorskie, a jakie rozwiązanie open-source.

4.4. Kryterium kosztów

Pod względem biznesowego wdrożenia aplikacji, duże znaczenie mają koszty i pracochłonność rozwiązania. W kryterium kosztów analizowana jest opłacalność budowy systemu autorskiego oraz oszacowane zostały koszty wdrożenia wybranego systemu open-source.

4.5. Kryterium wartości dodanych

W tym kryterium analizie zostały poddane umiejętności i kompetencje, które można rozwinąć podczas wdrożenia gotowego rozwiązania open-source, w porównaniu z tymi, jakie można zdobyć lub rozwinąć projektując i wdrażając system autorski.

4.6. Kryterium trwałości w czasie

W tym kryterium aplikacje zostaną poddane analizie pod kątem starzenia się kodu, określone zostaną także nakłady niezbędne do utrzymania dobrej jakości systemu.

5. Systemu autorski – sHome

System opracowany na potrzeby niniejszej publikacji jest prostym systemem inteligentnego domu, którego funkcjonalność sprowadza się do odczytu danych z czujników i sterowania elementami wykonawczymi.

5.1. Wymagania funkcjonalne

Wymagania funkcjonalne określają realizowane przez system funkcje i ich zakres. W przedstawionym systemie autorskim określono następujące wymagania funkcjonalne:

1) Sterowanie oświetleniem: włącz/wyłącz.
2) Sterowanie zasilaniem urządzeń elektrycznych (sterowanie gniazdami elektrycznymi).
3) Monitorowanie stanu sterowanych urządzeń.
4) Automatyczna reakcja na zagrożenia, np. odcięcie dopływu wody za pomocą elektrozaworu w przypadku wykrycia zalania.
5) Zarządzanie użytkownikami systemu.

5.2. Wymagania niefunkcjonalne

Wymagania niefunkcjonalne określają cechy jakimi powinien charakteryzować się system aby był użyteczny i wygodny w obsłudze. System autorski zakłada następujące wymagania w tym zakresie:

1) Dostępność systemu 24 godziny na dobę, przez 7 dni w tygodniu.
2) Prosty, czytelny, atrakcyjny wizualnie i ergonomiczny interfejs.
3) Dostępność również dla osób bez wykształcenia technicznego.
4) Czas ładowania interfejsu nie powinien przekraczać 1,5s.
5) Jak najmniejsza ingerencja w istniejące instalacje.
6) Interfejs www działający na większości popularnych przeglądarek.
7) Elastyczność implementacji – możliwość wdrożenia systemu na etapie projektowania instalacji, a także uruchomienia w budynku z istniejącą instalacją.

5.3. Architektura systemu

Architektura systemu zakłada działanie centralnego urządzenia (serwera) zarządzającego całym system. W systemie sHome rolę tę pełni mikrokomputer Raspberry Pi 3+, pracujący pod kontrolą systemu operacyjnego Raspbian. Do sterowania urządzeniami wykonawczymi oraz do odczytu danych z czujników wykorzystane są wbudowane w Raspberry Pi piny GPIO (General Purpose Input/Output, ang. wejście/wyjście ogólnego przeznaczenia), które stanowią interfejs wejścia i wyjścia. Interfejs użytkownika stanowi strona internetowa napisana w języku PHP, za pomocą której wywoływane są skrypty napisane w języku Python, które odpowiadają za komunikację z elementami peryferyjnymi (czujnikami, przekaźnikami i diodami). Dane są przechowywane w bazie danych MySQL, a komunikacja sieciowa oparta jest na sieci WiFi, z zastosowaniem szyfrowania WPA.

5.4. Platforma sprzętowa i jej parametry techniczne

Istotną częścią systemu automatyki domowej są czujniki. W systemie sHome zastosowane zostały dwa rodzaje czujników. Czujniki temperatury DS18B20, pracujący w zakresie temperatur: -55°C to +125°C, o dokładności ±0.5°C w zakresie -10°C – +85°C, poza tym zakresem, dokładność wynosi ±2°C. Komunikacja odbywa się za pomocą protokołu 1-Wire [20]. Czujnik zalania Yl-83 Rain Detector, charakteryzuje się czułością minimalną na poziomie 0.05cm² (istnieje możliwość regulacji), opóźnienie wyłączenia to około 5min (zależne od czasu schnięcia wody). Posiada on zarówno wyjście cyfrowe (DO, detekcja – stan niski), jak i analogowe (AO) [21]. Badany system korzysta z wyjścia cyfrowego. Elementami wykonawczymi są podwójne moduły przekaźników, które odpowiadają za sterowanie gniazdami elektrycznymi oraz dioda LED, symbolizująca oświetlenie.

5.5. Czas przygotowania systemu

System sHome został napisany przez jedną osobę – autora niniejszej publikacji, który ma niewielkie doświadczenie programistyczne. Czas powstania systemu to około pięć tygodni, czas pracy w tygodniu wyniósł średnio 40 godzin. Zatem łączny czas realizacji projektu zajął około 200 godzin.

5.6. Bezpieczeństwo systemu i komunikacji

Dzięki zastosowaniu sanityzacji i walidacji ciągu znaków wpisywanych do formularzy przez użytkowników, za pomocą funkcji htmlentities() oraz mysqli mysql_real_escape_string(), system sHome jest odporny na ataki typu SQL Injection. Dzięki wykorzystaniu tych funkcji, wprowadzana do formularza treść jest filtrowana i wykonane jest jedynie właściwe zapytanie. Drugą istotną kwestią dotyczącą bezpieczeństwa systemu jest wyłonienie grup użytkowników, którzy mają dostęp tylko do niezbędnych funkcji.

Za zapewnienie bezpieczeństwa komunikacji w systemie odpowiada kilka mechanizmów, w zależności od tego, z jakim elementem systemu zachodzi komunikacja. Bezpieczeństwo dostępu do interfejsu webowego jest zapewnione poprzez protokół SSL, który zapewnia szyfrowanie transmisji pomiędzy serwerem, a przeglądarką internetową. Bezpieczny dostęp do terminala systemu Raspbian odpowiada protokół SSH, który jest jedynym dopuszczonym do komunikacji z konsolą systemową. Bezpieczeństwo transmisji sieciowej jest zapewnione przez protokół WPA, który odpowiada za szyfrowanie danych pomiędzy bezprzewodowym punktem dostępowym i mikrokomputerem Raspberry Pi.

5.7. Skalowalność

Skalowalność systemu sHome jest zapewniana przez konstrukcję interfejsu i bazy danych, które pozwalają na proste dodawanie nowych urządzeń do systemu. Dodanie nowego urządzenia wykonawczego sprowadza się do dodania nowego wpisu do bazy za pomocą interfejsu graficznego. Wybrany model obsługi czujników temperatury pozwala na dodawanie kolejnych termometrów poprzez dołączenie ich do istniejącej szyny 1-Wire oraz odczytanie i konwersję wartości temperatury za pomocą skryptu. Dzięki wbudowanej w Raspberry Pi natywnej obsłudze sieci, można również rozbudować system o bezprzewodowe moduły (np. ESP8266), za pomocą których można zbierać dane z odległych czujników oraz sterować urządzeniami elektrycznymi po połączeniu z czujnikami lub modułami przekaźników. Powyższe czynności mogą być wykonane podczas pracy systemu i nie wymagają jego rekompilacji, dzięki wykorzystaniu skryptowych języków programowania. Możliwe jest również uruchomienie systemu sHome na innej platformie sprzętowej, jednak wymagałoby to opracowania nowych skryptów do obsługi interfejsu wejścia/wyjścia.

5.8. Koszty

Aby określić wysokość kosztów poniesionych na zbudowanie systemu sHome, należy określić co będzie uwzględniało to kryterium. Z pewnością najłatwiej jest obliczyć koszt użytego sprzętu:

- Raspberry Pi 3 B+
- Karta 32GB
- Zasilacz
- Płytka prototypowa
- Moduł Przekaźnika (podwójny)
- Czujnik zalania
- Dioda LED

Trudniej jest określić koszt pracy. Tutaj należałoby zwrócić uwagę na koszt alternatywny, rozumiany przez wartość najlepszej utraconej korzyści, w wyniku poświęcenia czasu na budowę systemu. Zatem określenie kosztów pracy zostało zawężone jedynie do ilości godzin poświęconych na stworzenie systemu. W przypadku systemu autorskiego czas poświęcony na budowę systemu wyniósł około 200 godzin, nie uwzględniono tu jednak czasu przewidywanego na utrzymanie, aktualizację, czy rozbudowę systemu.

5.9. Wartości dodane

Podczas budowania systemu autor poszerzył swoją wiedzę z obszaru programowania w językach PHP, HTML, CSS oraz Python, którą może wykorzystać w innych projektach. Jest to wiedza o rozwiązaniach programistycznych, która jest uniwersalna i może stanowić bazę do jej pogłębiania i tworzenia nowych projektów, jak również podnosi wartość autora na rynku pracy. Zdobyte doświadczenie pozwala również na wybranie specjalizacji, która najbardziej odpowiada preferencjom autora.

5.10. Trwałość w czasie

Specyfika systemów informatycznych jest taka, że zmiany zachodzące w tym środowisku są bardzo szybkie i napisanie systemu raz, jako skończonej całości jest praktycznie niemożliwe. System sHome zbudowany przez jedną osobę, nie jest odporny na porzucenie go przez autora, zatem nie jest on również odporny na zmiany zachodzące w czasie. Dopóki zastosowane rozwiązania będą obowiązującymi, system będzie aktualny.

6. System open-source – Domoticz

System Domoticz pojawiał się we wszystkich analizowanych artykułach dotyczących inteligentnych domów opartych na otwartym kodzie źródłowym [5-14]. Jest on również pierwszy w przedstawionym w tabeli 1 zestawieniu. W związku z tym, system ten został wybrany do porównania go z systemem autorskim sHome. Przeprowadzenie analizy kolejnych systemów może stanowić wartościową kontynuację pracy badawczej autora.

6.1. Opis autorskiego systemu open-source – Domoticz

System Domoticz wspiera wiele rodzajów czujników i urządzeń wykonawczych, można go uruchomić na wielu platformach sprzętowych (procesory ARM32, ARM64, x86) i systemach operacyjnych takich jak: Windows, Mac/OSX, Linux, czy Synology. System ten wspiera także platformę Raspberry Pi oraz współpracuje z różnymi, również komercyjnymi rozwiązaniami automatyki domowej, takimi jak: MySensors, RFXCOM, Z-Wave, RF-Link, P1 Smart Meter, YouLess, Smart Meter, Pulse Counters, EnOcean, iTead Sonoff, Xiaomi. Interfejs webowy Domoticz pozwala na korzystanie z różnych tematów, które ułatwiają

personalizację jego wyglądu. Dostępna jest również aplikacja mobilna, która pozwala na sterowanie systemem oraz na odczyt stanu czujników i systemu.

6.2. Architektura systemu

Kod źródłowy systemu Domoticz napisany jest w języku C/C++. Głównym elementem tego systemu jest centralne urządzenie (serwer) zarządzające całym systemem. W badanej konfiguracji rolę tę pełni mikrokomputer Raspberry Pi 3+. Czujniki i urządzenia wykonawcze komunikują się z mikrokomputerem poprzez piny GPIO. System umożliwia także użycie bezprzewodowych modułów ESP8266, które dzięki wbudowanym własnym pinom GPIO pozwalają na bezprzewodowy odczyt danych z czujników, jak również sterowanie zasilaniem urządzeń elektrycznych za pośrednictwem np. przekaźników. Dla zachowania podobieństwa funkcjonalności do systemu autorskiego, ta funkcjonalność w badanej konfiguracji użyto wyłącznie pinów GPIO. Głównym interfejsem służącym do sterowania, zarządzania i monitorowania systemu jest strona www. Dane są przechowywane w bazie danych SQLite. Istnieje także aplikacja mobilna w bezpłatnej wersji, która umożliwia sterowanie automatyką domową oraz podgląd stanu systemu.

6.3. Czas przygotowania systemu

Na czas potrzebny na uruchomienie systemu Domoticz, składają się: przygotowanie i połączenie elementów sprzętowych, uruchomienie systemu operacyjnego, zapoznanie z dokumentacją, instalacja, uruchomienie i konfiguracja systemu Domoticz. Proces instalacji przebiega automatycznie po uruchomienia skryptu automatycznej instalacji. Łączny czas potrzebny do uruchomienia systemu w badanej konfiguracji wyniósł około 16 godzin.

6.4. Bezpieczeństwo systemu i komunikacji

Bezpieczeństwo systemu Domoticz jest zapewniane przez wiele mechanizmów. Jednym z nich są konta użytkowników i ich podział na grupy, pod względem uprawnień do sterownia i modyfikacji systemu. System Domoticz posiada trzy predefiniowane grupy użytkowników. Oprócz tego, system umożliwienie otwarcia interfejsu bez autoryzacji systemu użytkownikom łączącym się z określonych adresów IP, a także możliwe jest wymuszenie autoryzacji dla czujników zewnętrznych podłączanych poprzez protokoły sieciowe oraz istnieje możliwość zabezpieczenia hasłem dostępu sterowania przełącznikami.

System Domoticz został także zbadany pod kątem podatności na wstrzykiwanie kodu SQL (SQL Injection) [15]. Aby zbadać tę podatność zostały wykonane testy, polegające na wprowadzeniu w pole nazwy użytkownika i hasła, ciągu znaków pozwalających na modyfikację zapytania SQL w taki sposób by uzyskać dostęp do interfejsu systemu bez znajomości danych autoryzacyjnych. System Domoticz przeszedł te testy pomyślnie. Według informacji ze strony vuldb.com [16], system Domoticz, w wersji poniżej 4.10578 może być podatny na atak SQL Injection, poprzez funkcję CWebServer::GetFloorplanImage w pliku WebServer.cpp. Podatność polega na możliwości manipulacji parametrem idx tak by wykonać wstrzyknięcie kodu SQL. Opisana podatność ma wpływ na poufność, spójność i dostępność systemu. Informacja o podatności została opublikowana w dniu 2019-03-31. Identyfikatorem tej podatności jest CVE-2019-10664 [17]. Na dzień pisania artykułu szczegóły techniczne ataku są już znane, jednak pomimo, że możliwe jest zdalne przeprowadzenie ataku, według informacji zawartych na stronie vuldb.com, wiadomo że obecnie nie ma dostępnego exploita wykorzystującego tę podatność. Aktualizacja do wersji 4.10578 eliminuje tę podatność [16]. Wersja 4.10717, która została zainstalowana na potrzeby niniejszej publikacji, jest odporna na to zagrożenie.

6.5. Bezpieczeństwo komunikacji

Za bezpieczeństwo transmisji danych pomiędzy przeglądarką a serwerem systemu odpowiada protokół HTTPS, wykorzystujący protokół SSL do szyfrowania danych. Bezpieczny dostęp do terminala systemu operacyjnego zapewnia protokół SSH. Bezpieczeństwo komunikacji elementów systemu jest zapewniane przez przewodowe połączenie z czujnikami i urządzeniami wykonawczymi, co znacząco utrudnia np. atak "Man in the middle", polegający na modyfikacji danych pomiędzy serwerem, a czujnikami i urządzeniami wykonawczymi. Również atak typu DoS (brak dostępu do usługi ang. Denial of Service) jest znacznie utrudniony. Wymienione wyżej rodzaje ataku były możliwe jedynie poprzez fizyczny dostęp do infrastruktury, a atak typu DoS mógłby zostać wykonany także poprzez wywołanie silnego impulsu elektromagnetycznego. W przypadku, gdyby użytkownik chciał wykorzystać bezprzewodowe czujniki, czy też bezprzewodową komunikację z elementami wykonawczymi za pomocą sieci WIFI, za bezpieczną transmisję danych odpowiada protokół WPA, wspierany na poziomie systemu operacyjnego Raspbian.

6.6. Skalowalność

System Domoticz, poprzez swoją modułową budowę, jak również wykorzystanie możliwości definiowania wirtualnych urządzeń, daje bardzo duże możliwości rozbudowy automatyki domowej. Społeczność, programistów pracujących nad tym projektem, stara się nadążyć za współczesnymi wymaganiami takich systemów i dostarcza nowe rozwiązania. Bardzo duży wpływ na skalowalność tego rozwiązania ma fakt, że użytkownik może samodzielnie pisać własne skrypty, które stworzą bardzo szerokie pole do automatyzacji procesów w inteligentnym domu.

6.7. Koszty

Koszty budowy systemu open-source ograniczają się jedynie do warstwy sprzętowej, która jest tożsama z wykorzystaną w systemie autorskim. W warstwie programowej, wszystkie niezbędne elementy można uzyskać bezpłatnie, dzięki temu jest to projekt o otwartym kodzie. Jedynym kosztem dodatkowym może być płatna wersja aplikacji mobilnej, jednak udostępniana jest również bezpłatna wersja, która dostarcza najważniejsze

funkcjonalności. Ważnym czynnikiem wpływającym na koszty wdrożenia systemu jest czas potrzebny do jego uruchomienia, który, w przypadku badanej konfiguracji wyniósł około 16 godzin roboczych, tj. 2 dni.

6.8. Wartości dodane

W przypadku systemu open-source, bardzo duże znaczenie ma wsparcie społeczności tworzącej system Domoticz. Społeczność szybko reaguje na znalezione luki bezpieczeństwa oraz nieustannie rozbudowuje system. Oprócz tego istnieje możliwość przyłączenia zdalnych bezprzewodowych czujników m.in. za pomocą modułów ESP8266, co zwiększa jeszcze bardziej możliwości rozbudowy systemu. Istotny jest również krótki czas potrzebny do uruchomienia systemu.

6.9. Trwałość w czasie

System Domoticz jest tworzony przez grupę entuzjastów, którzy często są również użytkownikami tego systemu oraz pasjonatami nowoczesnych rozwiązań. Projekt jest na bieżąco aktualizowany i rozbudowywany. Jego kod podlega publicznej weryfikacji i dyskusji, a błędy są na bieżąco naprawiane. Nie zmienia to faktu, że jak każdy rozbudowany system jest podatny na błędy popełniane przez deweloperów. System będzie odporny na starzenie się tak długo, jak będzie istniała aktywnie rozwijająca go społeczność.

7. Analiza porównawcza

Aby porównanie systemów było jak najbardziej rzetelne, obydwa systemy pracowały na identycznym sprzęcie, tak, by różnice wynikały wyłącznie z obszaru oprogramowania i środowiska obydwu aplikacji (tabela 2).

Tabela 2. Porównanie systemów

Cecha systemu	**sHome**	**Domoticz**
Język programowania	PHP, Python	C/C++
Czas przygotowania	200 godz.	16 godz.
Środowisko sprzętowe	Raspberry Pi 3+	Raspberry Pi 3+
System operacyjny	Raspbian	Raspbian
Aplikacja mobilna	Nie	Tak

Zarówno system autorski, jak również system open-source Domoticz, w badanej konfiguracji spełniają podstawowe założenia dotyczące systemów inteligentnych budynków, określone na wstępie, w szczególności dzięki samodzielnemu reagowaniu na sygnały z czujników, czego przykładem jest automatyczne zamykanie zaworów wody w przypadku wykrycia zalania. Pod kątem złożoności kodu i środowiska programistycznego systemy różnią się zasadniczo. System Domoticz napisany został w języku C/C++ i oferuje znacznie więcej funkcjonalności niż system sHome. System autorski jest stworzony głównie w języku PHP z elementami języka Python i jest bardzo uproszczony względem systemu Domoticz, ponieważ zawiera tylko niezbędne funkcje określone w pierwotnych wymaganiach.

7.1. Kryterium czasu opracowania

W analizowanym przypadku, czas wdrożenia systemu open-source wyniósł 16 godzin, natomiast zakres prac niezbędnych do stworzenia autorskiego projektu jest zdecydowanie większy niż przy wdrożeniu gotowego projektu, co miało również znaczący wpływ na czas opracowania systemu. System autorski sHome powstawał przez około 200 godzin.

7.2. Kryterium bezpieczeństwa aplikacji i komunikacji

Ze względu na pierwotne ograniczenia nałożone podczas formułowania kryteriów analizy systemów pod kątem bezpieczeństwa oraz ograniczonej funkcjonalności systemów, zostały one przeanalizowane pod kątem podziału uprawnień użytkowników oraz podatności na najbardziej popularne zagrożenie dla aplikacji internetowych jakim według raportów OWASP (Open Web Application Security Project) jest SQL Injection [15], a także pod kątem wykorzystanych protokołów i mediów transmisyjnych odpowiadających za komunikację.

Oba analizowane systemy posiadają predefiniowane grupy użytkowników, które podnoszą poziom bezpieczeństwa systemów poprzez ograniczenie dostępu tylko do odpowiednich dla danej grupy użytkowników funkcji. Przeprowadzone testy pokazały również, że obydwa systemy są odporne na ataki SQL Injection. Starsze wersje systemu Domoticz były podatne na ten rodzaj ataku, przez błąd w zaimplementowanej funkcji, jednak dzięki szybkiej reakcji społeczności deweloperów zagrożenie to zostało wyeliminowane w kolejnych wersjach systemu. W badanej konfiguracji, w obydwu systemach, za bezpieczeństwo dostępu do konsoli systemu operacyjnego odpowiada protokół SSH, który jest jedynym protokołem komunikacyjnym uruchomionym na potrzeby zdalnego zarządzania systemem operacyjnym. Za bezpieczeństwo komunikacji z interfejsem webowym odpowiada protokół HTTPS, którzy szyfruje przesyłane dane za pomocą protokołu SSL. Czujniki i elementy wykonawcze są połączone za pomocą przewodowych połączeń, dzięki czemu możliwość ingerencji w przesyłane informacje jest znacznie ograniczona, a atak musiałby zakładać fizyczny dostęp do systemu lub wywołanie potężnego impulsu elektromagnetycznego. W przypadku zastosowania bezprzewodowych elementów wejścia/wyjścia, o które mogą być rozbudowane obydwa systemy, za szyfrowaną komunikację bezprzewodową odpowiada protokół WPA.

7.3. Kryterium skalowalności

System sHome, w podstawowym stopniu umożliwia jego rozbudowę. Jednak uproszczony sposób jest dostępny tylko dla elementów wykonawczych i czujników juz zdefiniowanych w systemie. Rozbudowa systemu o obsługę bezprzewodowych modułów ESP8266 wymagała by jednak opracowania kolejnych elementów systemu w postaci nowych skryptów i instalacji dodatkowych pakietów oprogramowania z systemie Raspbian.

System Domoticz zaraz po instalacji udostępnia szeroki wachlarz możliwości przyłączania nowego sprzętu, bez konieczności edycji kodu. Cała konfiguracja jest dostępna z poziomu strony internetowej. Przyłączenie dodatkowych elementów komunikujących się bezprzewodowo wymaga jedynie ich konfiguracji w systemie. Domoticz współpracuje między innymi z bardzo popularnym modułem bezprzewodowym ESP8266. Możliwość wgrania alternatywnego oprogramowania do modułu ESP (np. ESPEasy) [18], umożliwia bardzo proste dodawanie kolejnych elementów systemu.

7.4. Kryterium kosztów

Koszt elementów sprzętowych jest taki sam dla obydwu badanych systemów, ponieważ działają one na tym samym sprzęcie. Niewątpliwie największym kosztem systemu autorskiego jest czas jego projektowania i uruchomienia, który jest 12,5 krotnie większy od czasu wdrożenia systemu open-source.

7.5. Kryterium wartości dodanych

System open-source oferuje możliwość bardzo szybkiego uruchomienia i prostej rozbudowy systemu o kolejne elementy i funkcjonalności. Zbudowanie systemu autorskiego posiada znaczne walory edukacyjne. Napisanie własnego systemu wymaga rozważenia wielu problematycznych kwestii oraz przeglądu technologii, tak by wybrać tę najbardziej adekwatną do zadania. Daje również znacznie większą elastyczność w dopasowaniu rozwiązania do konkretnego wdrożenia.

7.6. Kryterium trwałości w czasie

System Domoticz, który powstał w 2014 roku, jest na bieżąco rozwijany i ulepszany przez społeczność programistów. Dzięki tej aktywności projekt stale ewoluuje i ma zdecydowanie większe prawdopodobieństwo przetrwania niż projekt sHome, wykonany przez jednego autora. Według danych z serwisu github.com (Rys. 3), system Domoticz tworzy grupa ponad 80 współpracowników, którzy tylko w okresie 30.04.2019 – 31.05.2019 dokonali zmian w 923 plikach, na które składa się 44117 zmian i 26847 usunięć w kodzie [19].

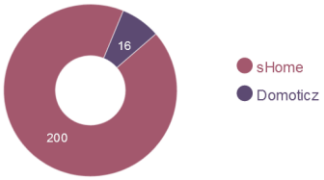
Rys. 3. Wkład społeczności w rozwój projektu Domoticz z wyłączeniem zatwierdzeń scalania (ang. commits) [19]

System sHome od momentu powstania jego finalnej wersji nie zmienił się ani razu.

8. Wnioski

Systemy inteligentnych budynków są coraz bardziej powszechne, budowa i wdrożenie takiego systemu ma wpływ na poprawę komfortu użytkowania domu, czy mieszkania, a także, pozwala na automatyczną reakcję systemu w przypadku wykrycia zagrożenia, bez konieczności wykonywania akcji przez użytkownika. Celem artykułu była próba odpowiedzi na pytanie: czy i w jakich warunkach uzasadnione jest budowanie takiego systemu, mając na względzie mnogość dostępnych bezpłatnie, gotowych rozwiązań o otwartym kodzie źródłowym. Zapewne porównanie kilku lub wszystkich wymienionych w niniejszym artykule systemów (tabela 1) z systemem autorskim znacznie podniosło by wartość merytoryczną publikacji, jednak ograniczenia czasowe i objętościowe dotyczące artykułu pozwoliły autorowi na przeprowadzenie analizy tylko jednego wybranego systemu open-source. Analiza porównawcza pozostałych systemów może stanowić wartościową kontynuację przeprowadzonych badań.

Analiza bezpieczeństwa badanych systemów wykazała, że system Domoticz, poprzez upublicznienie kodu i swoją popularność, jest bardziej niż system sHome narażony na ataki hackerskie, jednak aktywność społeczności podnosi poziom bezpieczeństwa, poprzez szybkie usuwanie odnalezionych luk w kodzie systemu. System autorski, może być również podatny na wiele zagrożeń, jednak tajność kodu i niewielka jego popularność działa na jego korzyść, poprzez zmniejszenie prawdopodobieństwa ataku. W warstwie komunikacyjnej odporność systemów wynika wprost z wykorzystanych, powszechnie znanych i stosowanych mechanizmów bezpieczeństwa.



Rys. 4. Czas uruchomienia systemów w godzinach

Czas potrzebny na opracowanie systemu autorskiego był 12,5-krotnie większy niż czas uruchomienia systemu open-source (Rys. 4). Poddaje to w wątpliwość opłacalność pisania takiego systemu jedynie na potrzeby jednego wdrożenia, szczególnie w przypadku tak prostej i popularnej funkcjonalności jaka jest założona w niniejszym opracowaniu. Aby system autorski, z punktu widzenia opłacalności biznesowej, miał szanse konkurować z systemem open-source, musi oferować wartość, której system open-source nie posiada. Taką wartością może być ekskluzywny charakter takiego systemu, możliwość pełnej kontroli nad jego strukturą i dostosowanie interfejsu dokładnie do wymagań klienta. W przypadku, kiedy wdrażane byłyby specyficzne, niszowe rozwiązania, czas wdrożenia systemu open-source mógłby znacząco się wydłużyć, ponieważ wymagałoby to napisania własnych modułów do obsługi nietypowego sprzętu czy protokołu oraz dogłębnego poznania struktury systemu Domoticz lub przynajmniej jego API (Application Programming Interface).

Można zatem stwierdzić, że w przypadku typowego zastosowania, system open-source będzie rozwiązaniem zdecydowanie szybszym od systemu autorskiego, jednak przy niszowym specyficznym zastosowaniu można rozważyć opcję budowy systemu autorskiego.

System Domoticz oferuje zdecydowanie większe możliwości skalowania systemu niż system sHome. Jednak jeśli system autorski zostałby opracowany pod kątem konkretnego wdrożenia daje o wiele większą elastyczność i prostotę obsługi, poprzez implementację wyłącznie tych funkcji jakie rzeczywiście w danym wdrożeniu są wykorzystywane. Dojrzały system już na etapie projektowania powinien zakładać trwałość i utrzymanie kodu w czasie. Popularny system open-source jest w tym wypadku znacznie bardziej odporny od systemu autorskiego na starzenie się kodu oraz pojawiające się zagrożenia bezpieczeństwa, szczególnie kiedy system autorski jest projektem tworzonym przez niewielki zespół lub jednego autora. System autorski jest na tyle odporny na starzenie się kodu, na ile jego autor będzie zdecydowany kontynuować projekt. Istnieją co najmniej dwie możliwości uniknięcia sytuacji zamknięcia projektu: pierwsza z nich to upublicznienie kodu i próba stworzenia wokół niego społeczności programistów gotowych dalej go rozwijać, drugą możliwością jest komercjalizacja rozwiązania, poprzez zawarcie stałej umowy pomiędzy potencjalnym klientem, a autorem i zapewnienie ten sposób finansowania dalszych prac nad projektem oraz zatrudnienie do jego rozwoju zespołu deweloperów.

Konkluzją wynikającą z analizy badanych systemów jest stwierdzenie, że budowa systemu autorskiego ma uzasadnienie głównie w projektach służących edukacji i nabywaniu praktycznych umiejętności programowania. Drugą przesłanką przemawiającą za budową autorskiego projektu, jest chęć pełnej kontroli nad kodem aplikacji i dostosowania systemu do bardzo specyficznych indywidualnych warunków wdrożenia oraz wymagań użytkownika. Istnieją jednak dość poważne ograniczenia i zagrożenia wypływające z budowy takiego systemu, wynikające przede wszystkim z konieczności stałej kontroli i aktualizacji oprogramowania, a także dużego ryzyka porzucenia projektu. Zdaniem autora, stając przez wyborem uruchomienia systemu dla typowego rozwiązania, autor byłby skłonny wykorzystać dostępny system open-source, oferujący także wsparcie społeczności, pomocne w przypadku problemów z funkcjonowaniem czy wykryciem podatności na ataki hackerskie. System autorski zaprojektowany z poszanowaniem elementarnych zasad bezpieczeństwa budowy oprogramowania, powinien również zapewniać bezpieczeństwo, ze względu na to, że system niszowy obniża prawdopodobieństwo wykrycia podatności. Będzie on również mniej atrakcyjnym celem dla cyberprzestępców, którzy zazwyczaj wybierają rozwiązania popularne, po to by zmaksymalizować zasięg ataku. System autorski zapewnia również większe poczucie prywatności, szczególnie w porównaniu z systemami chmurowymi, które przesyłają uzyskane dane do chmury obliczeniowej, gdzie są analizowane przez algorytmy sztucznej inteligencji, a użytkownik traci kontrolę nad tym, gdzie i w jaki sposób analizowane są dane z systemu zarządzania automatyką w jego domu. Zatem budowa autorskiego systemu inteligentnego budynku jest uzasadniona szczególnie w przypadku zapewnienia prywatności, dostosowania systemu do specyficznych nietypowych zastosowań oraz do zdobycia wiedzy i doświadczenia w kwestii budowy aplikacji.

Literatura

Pomimo usilnych poszukiwań artykułów naukowych w języku polskim i angielskim, dotyczących problematyki systemów inteligentnych budynków udostępnianych na licencji open-source, autor artykułu nie dotarł do żadnych opracowań tym temacie. Dlatego na podstawie przedstawionych w literaturze artykułów przeprowadził samodzielne badania na temat takich rozwiązań oraz ich popularności. Na potrzeby badań autor artykułu korzystał z następujących pozycji internetowych:

[1] https://pl.wikipedia.org/wiki/Inteligentny_budynek [07.04.2019].

[2] https://cyfrowa.rp.pl/technologie/13104-inteligentny-dom-powoli-zyskuje-popularnosci [02.05.2019].

[3] https://analizarynku.eu/125 [02.05.2019]

[4] https://inwestor.newseria.pl/newsy/rynek_inteligentnych,p1432688098 [02.05.2019].

[5] https://smarthomegearguide.com/5-open-source-home-automation-tools/ [03.05.2019].

[6] https://randomnerdtutorials.com/9-home-automation-open-source-platforms-for-your-projects/ [03.05.2019].

[7] http://24-7-home-security.com/open-source-home-automation-software/ [03.05.2019].

[8] https://zwavezone.com/open-source-home-automation-systems/ [03.05.2019].

[9] https://www.smarthomeblog.net/openhab-home-assistant-domoticz/ [03.05.2019].

[10] https://www.quora.com/What-is-the-best-open-source-home-automation-platform-I-e-Plays-nice-with-other-popular-smart-device-ecosystems-can-be-deployed-on-Linux-etc [03.05.2019].

[11] https://opensource.com/tools/home-automation [03.05.2019].

[12] https://medium.com/@ronnymenestrel/seven-open-source-home-automation-tools-d25c27816f8 [03.05.2019].

[13] https://www.electromaker.io/blog/article/9-best-raspberry-pi-smart-home-software-options [03.05.2019].

[14] https://www.techupyourhome.com/home-security/best-open-source-home-automation-tools/ [03.05.2019].

[15] https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

[16] https://vuldb.com/pl/?id.132644 [28.05.2019].

[17] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-10664 [28.05.2019].

[18] https://www.letscontrolit.com/wiki/index.php/ESPEasy

[19] https://github.com/domoticz/domoticz/graphs/contributors [28.05.2019].

[20] https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf [07.04.2019].

[21] https://www.openhacks.com/uploadsproductos/rain_sensor_module.pdf [07.04.2019].