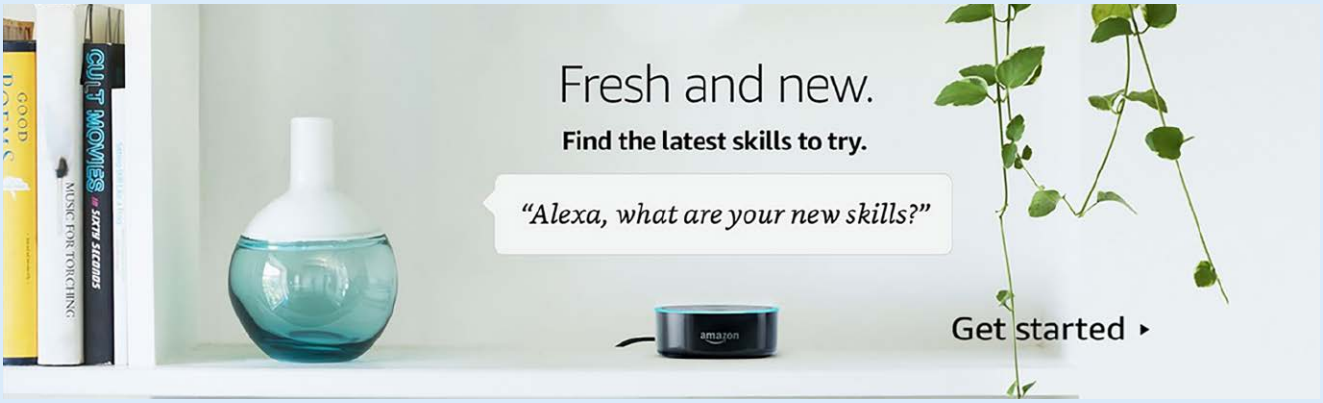


Amazon Alexa (2)

Tworzymy własne umiejętności Skills

W drugiej części kursu tworzenia umiejętności dla asystentki głosowej Alexa opiszemy sposób wykonania oprogramowania odtwarzania plików audio oraz uruchomimy w konfiguratorze wyświetlanie plansz z grafiką ilustrującą wynik losowania kostką.

W artykule omówimy projektowanie ulepszonej wersji umiejętności niewidzialnej kostki do gry. W nowej umiejętności, którą nazwano **Invisible Dice Plus** przed „rzutem" będziemy odtwarzali dźwięk rzucanej kostki. Dodatkowo, wynik losowania będziemy wyświetlali w formie graficznej w oknie konfiguratora. W artykule wielokrotnie będziemy nawiązywali do pierwszej części cyklu opublikowanej w EP 12/2017. Dlatego też przed przystąpieniem do drugiej części kursu warto zapoznać się ze wspomnianym artykułem.

Voice User Interface

W projekcie *Invisible Dice Plus* wykorzystamy interfejs głosowy użytkownika z umiejętności niewidzialnej kostki do gry *Invisible Dice*. Najpierw jednak musimy utworzyć nową umiejętność. Uruchamiamy portal developera ***http://developer.amazon.com*** i tworzymy umiejętność użytkownika *Custom Skill* o nazwie *Invisible Dice Plus*. Postępujemy zgodnie z procedurą opisaną w pierwszej części kursu (rysunki 1…5). Następnie uruchamiamy kreator tworzenia interfejsu głosowego użytkownika *Skill Builder*. Po uruchomieniu kreatora wybieramy zakładkę *Code Editor* i importujemy plik z konfiguracją przygotowaną dla umiejętności niewidzialnej kostki do gry. Plik o nazwie *IntentSchema.json* jest dostępny w folderze projektu (EP 12/2017). Po wgraniu konfiguracji zatwierdzamy wprowadzone ustawienia (**rysunek 1**). Na zakończenie zapisujemy (*Save Model*) oraz kompilujemy (*Build Model*) nowo utworzony interfejs głosowy użytkownika (**rysunek 2**).

Alexa Skill Code

Kod programu dla umiejętności *Invisible Dice Plus* będzie bazował na programie do obsługi umiejętności niewidzialnej kostki do gry *Invisible Dice*.

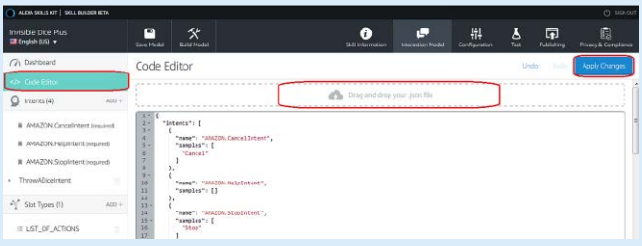
Tworzymy folder nowego projektu *InvisibleDicePlus*. Następnie z projektu niewidzialnej kostki do gry *Invisible Dice* (EP 12/2017) kopiujemy kod programu (plik **index.js**) oraz folder zawierający framework *Alexa Skills Kit SDK* (folder **node_modules**). Skopiowany plik i folder kompresujemy do archiwum w formacie zip. Plik z archiwum użyjemy do konfiguracji funkcji Lambda.

W pierwszej części kursu kod programu modyfikowaliśmy korzystając z darmowego edytora *Atom*. Teraz użyjemy kompilatora, który wbudowano w funkcję Lambda. Jest to nowa funkcjonalność platformy AWS. Kompilator online pozwala na szybkie zmiany w oprogramowaniu bez konieczności wielokrotnego importu archiwum do funkcji Lambda.

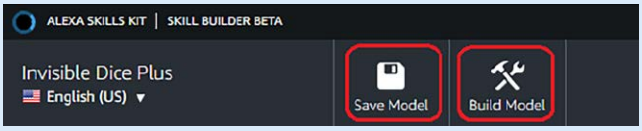
Image on the Home Card

Tworząc umiejętności typu *Custom* (użytkownika) mamy możliwość definiowania karty domowej w konfiguratorze asystentki Alexa (konfigurator www dostępny pod adresem ***http://alexa.amazon.com*** oraz aplikacja na urządzenia mobilne *Alexa app*). Możemy definiować tytuł karty, opis karty oraz wyświetlić grafikę powiązaną z kartą, jak na **rysunku 3**. Obrazy wyświetlane w karcie muszą być w jednym z formatów JPEG, PNG. Każdy obraz musi być przygotowany w dwóch rozdzielczościach: „małym", o wymiarach 720×480 pikseli oraz „dużym" o wymiarach 1200×800 pikseli.

Będziemy wyświetlali wynik rzutu kostką. Korzystając np. z darmowego oprogramowania GIMP (***http://gimp.org***) przygotowałem pliki PNG w obu wymaganych rozdzieczościach. Pliki są dostępne w folderze projektu. Obrazy w karcie domowej aplikacji wyświetlamy korzystając z funkcji ***emit()*** z parametrem *tellWithCard*



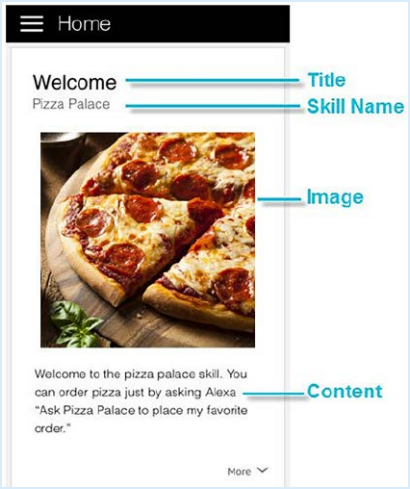
Rysunek 1. Amazon developer. Interfejs głosowy użytkownika. Import konfiguracji



Rysunek 2. Amazon developer. Interfejs głosowy użytkownika. Zapis i kompilacja konfiguracji

Poprzednie części kursu i dodatkowe materiały dostępne są na stronie www.media.avt.pl

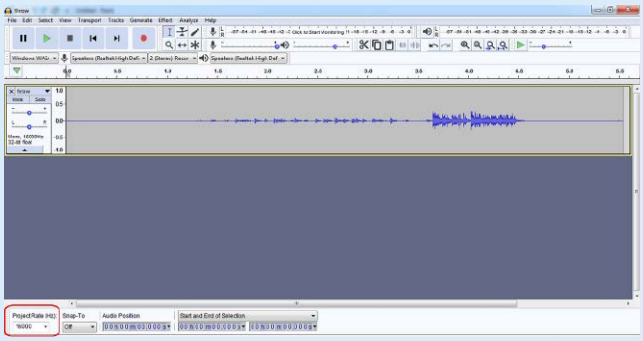
(ewentualnie *askWithCard*). W argumentach funkcji, poza transmitowanym tekstem, przekazujemy nazwę karty, jej opis oraz obiekt z adresami URL grafiki, którą chcemy wyświetlić („mały” i „duży” format). Sposób użycia funkcji *emit* oraz jak hostowania plików z grafiką opiszę w dalszej części artykułu.



Rysunek 3. Konfigurator Alexa. Przykładowa karta

Audio files in a Skill

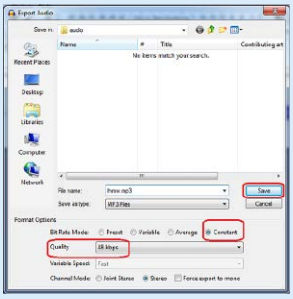
Pliki audio w umiejętnościach dla asystentki Alexa możemy odtwarzać korzystając z wbudowanego odtwarzacza audio lub z mechanizmu SSML (Speech Synthesis Markup Language). Większe możliwości sterowania odtwarzaniem dźwięku daje użycie odtwarzacza audio (funkcje: stop, start, pauza, next, repeat i inne), ale łatwiejszy w użyciu jest syntezator SSML i takie rozwiązanie zastosujemy w naszej umiejętności.



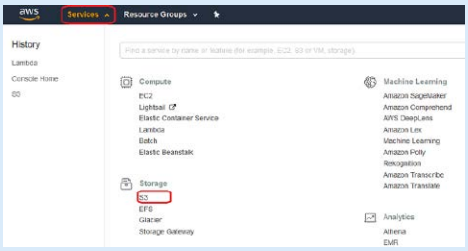
Rysunek 4. Audacity. Ustawienie pasma pliku audio

Syntezator SSML odpowiada za konwersję tekstu na mowę. Dodatkowo ma też możliwość odtwarzania plików audio. Żeby uruchomić odtwarzanie plików wystarczy, że do wypowiedzi dodamy znacznik audio i podamy adres URL pliku (”<audio src = ”https://filesource/file.mp3” />”).

Aby plik audio był poprawnie odtwarzany, musi być zapisany w formacie mp3 (kodek MPEG2), z przepływnością 48 kb/s, przy



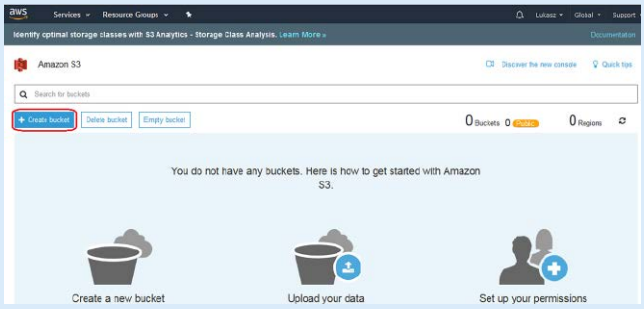
Rysunek 5. Audacity. Eksport pliku audio. Ustawienie przepływności



Rysunek 6. Platforma AWS. Serwis S3. Uruchomienie

szerokości pasma 16 kHz. Nie może przy tym trwać dłużej niż 90 sekund. Do konwersji pliku do formatu akceptowanego przez mechanizm SSML możemy użyć darmowego oprogramowania Audacity. W tym celu logujemy się na stronie projektu ***http://audacityteam.org***, pobieramy i instalujemy oprogramowanie. Następnie ze strony ***http://lame.buanzo.org/#lamewindl*** pobieramy archiwum zip z plikiem zawierającym kodek mp3. Rozpakowujemy archiwum i plik kodeka *lame_enc.dll*, kopiujemy do folderu instalacji oprogramowania Audacity. W następnym kroku uruchamiamy oprogramowania Audacity i otwieramy plik audio, który chcemy przekonwertować (File → Open). W ustawieniach programu (lewy, dolny róg aplikacji) wybieramy przepływność oraz pasmo 16000 Hz, jak na **rysunku 4**. Z menu programu wybieramy opcję eksportu do formatu mp3 (*File → Export → Export as MP3*). Ustawiamy rodzaj próbkowania, przepływność 48 kb/s i zapisujemy plik na dysku komputera (**rysunek 5**).

Przed podaniem wyniku rzutu kostką będziemy odtwarzać odgłos rzucanej kostki. Z dostępnej w sieci bazy darmowych dźwięków pobieramy nagranie odgłosu rzutu kostką. Następnie, korzystając z oprogramowania Audacity, zapisujemy plik audio z parametrami akceptowanymi przez mechanizm SSML. Plik z nagraniem rzutu kostką *throw.mp3* jest dostępny w folderze projektu.



Rysunek 7. Platforma AWS. Serwis S3. Tworzenie magazynu danych

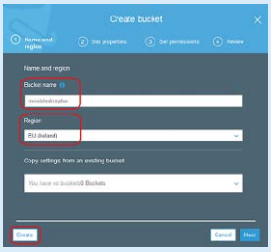
Amazon S3

Aby móc wyświetlić pliki graficzne w karcie konfiguratora oraz odtwarzać pliki audio musimy mieć publiczny dostęp do plików. Serwer hostingowy podczas transmisji danych musi posługiwać się certyfikatem SSL zaufanym przez Amazon oraz wspierać CORS (*Cross-origin resource sharing*).

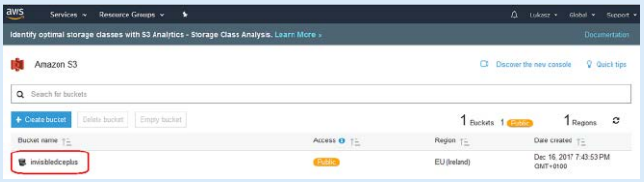
Skorzystamy z rozwiązania oferowanego przez firmę Amazon, jakim jest S3 (*Simple Storage Service*). W celu konfiguracji serwisu S3 logujemy się do platformy AWS (***http://aws.amazon.com***). Następnie uruchamiamy zakładkę *Services* i wybieramy S3, jak na **rysunku 6**. W kolejnym kroku, wybieramy utworzenia nowego magazynu danych (**rysunek 7**). Podajemy nazwę magazynu (invisiblediceplus) oraz region, *w którym* będą umieszczone dane (EU Ireland).

Po wprowadzeniu ustawień zatwierdzamy utworzenie magazynu (**rysunek 8**). Po utworzeniu magazynu wskazujemy jego nazwę i przechodzimy do zawartości (**rysunek 9**). Nowo utworzony magazyn jest pusty. Korzystając z dostępnych opcji możemy zaimportować pliki, utworzyć foldery przechowywania oraz zdefiniować prawa dostępu do zaimportowanych danych (**rysunek 10**).

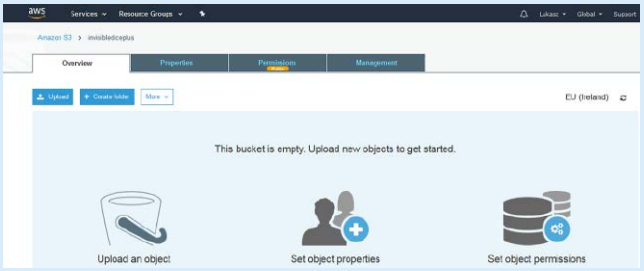
Import plików rozpoczynamy po wybraniu opcji *Upload*. Następnie dodajemy pliki i przyciskiem i zatwierdzamy umieszczenie



Rysunek 8. Platforma AWS. Serwis S3. Tworzenie magazynu danych

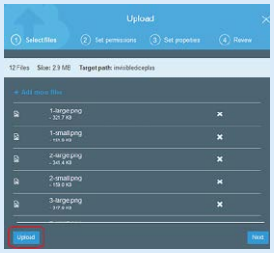


Rysunek 9. Platforma AWS. Serwis S3. Otwarcie magazynu danych



Rysunek 10. Platforma AWS. Serwis S3. Zarządzanie magazynem danych

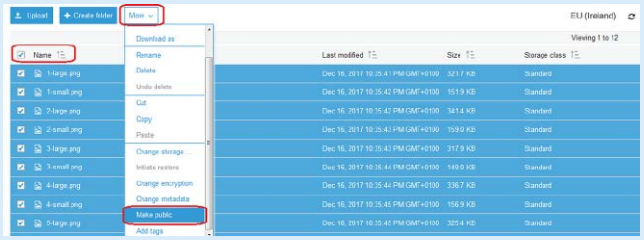
plików w magazynie danych (**rysunek 11**). Zaimportowane pliki nie są dostępne publicznie i oprogramowanie do obsługi umiejętności asystentki Alexa nie ma praw dostępu do plików. Musimy zatem zmienić prawa dostępu do plików i udostępnić je publicznie. W tym celu zaznaczamy wszystkie zaimportowane pliki, otwieramy menu konfiguracyjne i z dostępnych opcji wybieramy *Make public* (**rysunek 12**).



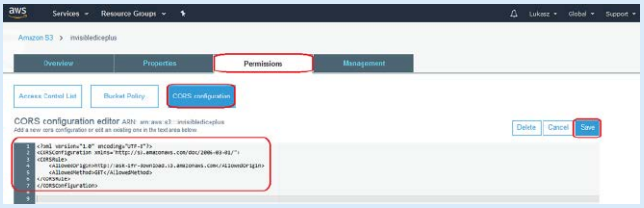
Rysunek 11. Platforma AWS. Serwis S3. Import plików do magazynu danych

Na zakończenie konfiguracji magazynu danych ustawimy politykę dostępu CORS (*Cross-origin resource sharing*). W tym celu wybieramy zakładkę *Permissions*. Następnie uruchamiamy edytor konfiguracji CORS. Wprowadzamy ustawienia i zapisujemy konfigurację, jak na **rysunku 13**. Plik z ustawieniami CORS jest dostępny w folderze projektu.

Po wgraniu i udostępnieniu plików (grafika i audio) musimy skopiować adresy URL plików. Będą one potrzebne przy tworzeniu kodu programu dla umiejętności *Invisible Dice Plus*. W tym celu, z magazynu danych wybieramy kolejno nazwę pliku

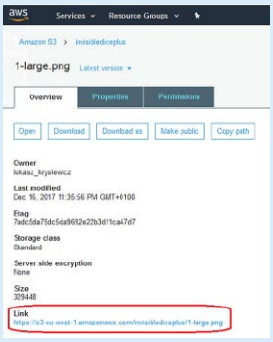


Rysunek 12. Platforma AWS. Serwis S3. Zmiana praw dostępu do plików



Rysunek 13. Platforma AWS. Serwis S3. Konfiguracja CORS

i po uruchomieniu okna z właściwościami pliku kopiujemy jego adres URL (**rysunek 14**).

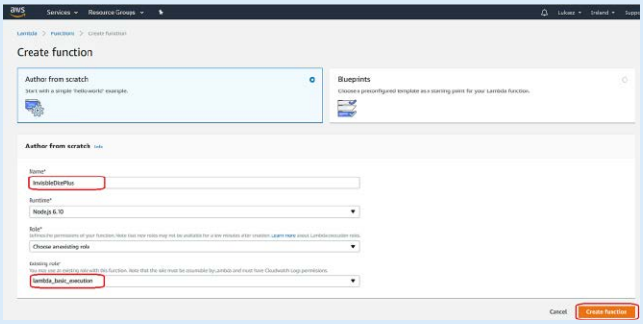


Rysunek 14. Platforma AWS. Serwis S3. Adres URL pliku

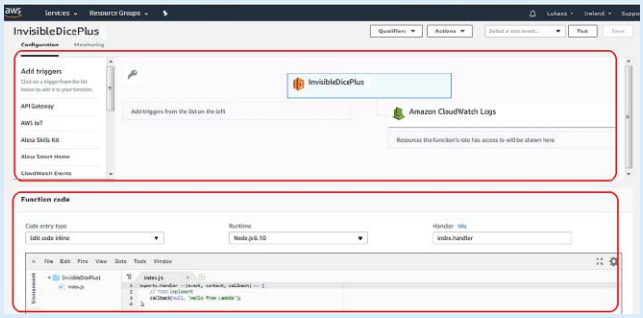
Lambda Function

Po zakończeniukonfigurowania serwisu S3 przechodzimy do panelu zarządzania funkcjami Lambda (*Services* → *Lambda*). Następnie rozpoczynamy tworzenie nowej funkcji o nazwie *InvisibleDicePlus*. Procedura tworzenia funkcji Lambda jest identyczna, jak opisana w pierwszej części kursu, jednak działanie kreatora zostało zmodyfikowane. Żeby zademonstrować działanie kreatora funkcji Lambda po zmianach jeszcze raz zademonstrujemy jego użycie.

Aby rozpocząć tworzenie nowej funkcji *Lambda* w oknie zarządzania funkcjami wybieramy opcję *Create a function*. Powoduje to uruchomienie kreatora tworzenia funkcji *Lambda* z domyślnie zaznaczoną opcją budowy funkcji według projektu użytkownika (*Author from scratch*). Wprowadzamy nazwę funkcji (*InvisibleDicePlus*), definiujemy uprawnienia (rola *lambda_basic_execution*) oraz zatwierdzamy utworzenie nowej funkcji (**rysunek 15**). W systemie zostaje utworzona funkcja *Lambda* oraz jest uruchamiany panel zarządzania funkcją. W panelu zarządzania są dostępne zakładki konfiguracja oraz monitoring. Domyślnie jest uruchomiona zakładka konfiguracji funkcji. Korzystając z dostępnych opcji możemy zdefiniować wyzwalacz funkcji oraz rozpocząć pracę z kodem źródłowym (**rysunek 16**). Teraz przechodzimy do sekcji zarządzania kodem źródłowym funkcji i z listy rozwijalnej

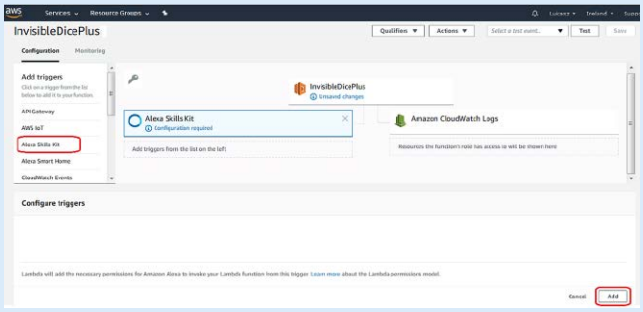


Rysunek 15. Platforma AWS. Funkcja Lambda. Nowa funkcja. Informacje podstawowe

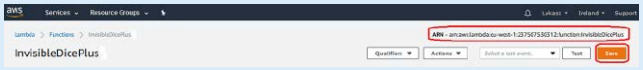


Rysunek 16. Platforma AWS. Funkcja Lambda. Panel sterowania. Zakładka konfiguracja

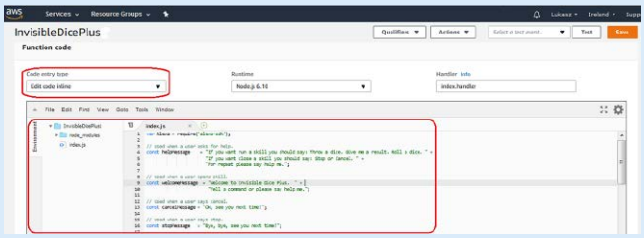
Rysunek 17. Platforma AWS. Funkcja Lambda. Upload archiwum ZIP z kodem funkcji



Rysunek 18. Platforma AWS. Funkcja Lambda. Konfiguracja wyzwalacza funkcji



Rysunek 19. Platforma AWS. Funkcja Lambda. Zakończenie konfiguracji



Rysunek 20. Platforma AWS. Funkcja Lambda. Edycja kodu źródłowego funkcji

Code entry type wybieramy opcję *Upload a .ZIP file*. Następnie, dołączamy archiwum ZIP zawierające kod programu umiejętności *Invisible Dice* z pierwszej części kursu (**rysunek 17**). Po wgraniu archiwum z kodem programu funkcji *Lambda* przechodzimy do okna konfiguracji wyzwalacza funkcji. Z dostępnych opcji wybieramy serwis *Alexa Skills Kit*, a następnie zatwierdzamy wybór (**rysunek 18**). Na zakończenie zapisujemy ustawienia funkcji Lambda oraz kopiujemy identyfikator ARN utworzonej funkcji *InvisibleDicePlus* (**rysunek 19**).

Nasza funkcja zawiera kod programu umiejętności z pierwszej części kursu. Teraz naszym zadaniem będzie zmodyfikowanie kodu źródłowego funkcji i dodanie obsługi wyświetlania obrazów oraz odtwarzania plików audio. W tym celu, ponownie uruchamiamy edytor funkcji *Lambda* (*Services* → *Lambda* → *Functions* → *InvisibleDicePlus*) i przechodzimy do edytora kodu źródłowego funkcji (**rysunek 20**). W kodzie programu podajemy adresy URL plików z grafiką oraz pliku audio. Następnie, w sekcji do obsługi intencji rzutu kostką *ThrowADiceIntent* wykonujemy program obsługi odtwarzania dźwięku (budowa polecenia dla mechanizmu SSML) oraz prezentację karty domowej w konfiguratorze (wywołanie funkcji *emit* z parametrami: *tellWithCard*, tytuł i opis karty, obiekt z adresem pliku graficznego – dwa adresy format „mały” i „duży”). Gotowy kod programu do obsługi umiejętności pokazano na **listingu 1**.

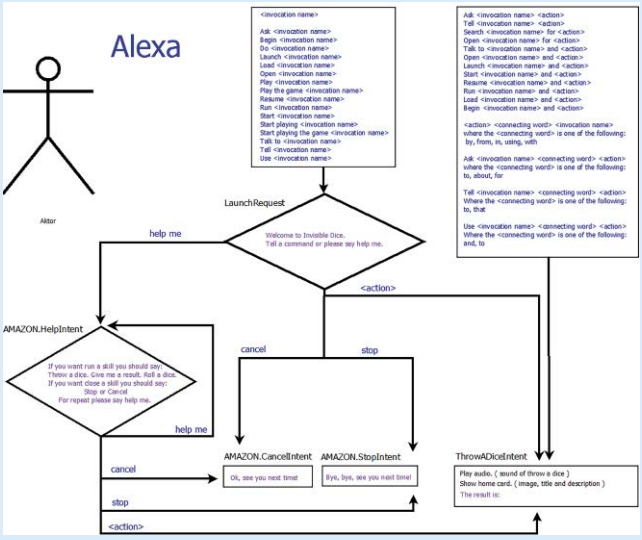
Teraz przed podaniem wyniku rzutu kostką Alexa odtworzy nagranie odgłosu rzutu kostką. Dodatkowo, w karcie domowej konfiguratora zostanie wyświetlona grafika prezentująca wynik rzutu. Diagram działania umiejętności *Invisible Dice Plus* pokazano na **rysunku 21**.

Connect VUI to Code

Mam już stworzony interfejs głosowy użytkownika *VUI (Voice User Interface)* oraz oprogramowaną funkcję *Lambda* do obsługi umiejętności. Żeby połączyć

Listing 1. Kod programu do obsługi umiejętności: Invisible Dice Plus

```
var Alexa = require(‚alexa-sdk’);

// Used when a user asks for help.
const helpMessage = „If you want run a skill you should say: „ +
                    „Throw a dice. Give me a result. Roll a dice. „ +
                    „If you want close a skill you should say: Stop or Cancel.”+
                    „For repeat please say help me.”;

// Used when a user opens skill.
const welcomeMessage = „Welcome to Invisible Dice Plus.  „ +
                       „Tell a command or please say help me.”;

// Used when a user says cancel.
const cancelMessage = „Ok, see you next time!”;

// Used when a user says stop.
const stopMessage   = „Bye, bye, see you next time!”;

// Used to randomise numbers
const t_numbers = [
      „one”,
      „two”,
      „three”,
      „four”,
      „five”,
      „six”];
// Used to show images
  const t_images = [
    { imageSmall: „https://s3-eu-west-1.amazonaws.com/invisiblediceplus/1-small.png”,
      imageLarge: „https://s3-eu-west-1.amazonaws.com/invisiblediceplus/1-large.png” },
    { imageSmall: „https://s3-eu-west-1.amazonaws.com/invisiblediceplus/2-small.png”,
      imageLarge: „https://s3-eu-west-1.amazonaws.com/invisiblediceplus/2-large.png” },
    { imageSmall: „https://s3-eu-west-1.amazonaws.com/invisiblediceplus/3-small.png”,
      imageLarge: „https://s3-eu-west-1.amazonaws.com/invisiblediceplus/3-large.png” },
    { imageSmall: „https://s3-eu-west-1.amazonaws.com/invisiblediceplus/4-small.png”,
      imageLarge: „https://s3-eu-west-1.amazonaws.com/invisiblediceplus/4-large.png” },
    { imageSmall: „https://s3-eu-west-1.amazonaws.com/invisiblediceplus/5-small.png”,
      imageLarge: „https://s3-eu-west-1.amazonaws.com/invisiblediceplus/5-large.png” },
    { imageSmall: „https://s3-eu-west-1.amazonaws.com/invisiblediceplus/6-small.png”,
      imageLarge: „https://s3-eu-west-1.amazonaws.com/invisiblediceplus/6-large.png” },
];
// Used to play audio
  const qMark       = „\””;
  const srcAudio    = „https://s3.eu-west-1.amazonaws.com/invisiblediceplus/throw.mp3”;
  const audioSource = qMark + srcAudio + qMark;

const handlers = {
    ‚LaunchRequest’: function () {
        this.emit(‚:ask’, welcomeMessage, welcomeMessage);
    },

    ‚ThrowADiceIntent’: function () {
        var number = Math.floor(Math.random() * t_numbers.length);

        var sAudio   =”<audio src=” +  audioSource + „ />”;
        var sOutput  = sAudio + „The result is:” + „  „+ t_numbers[number];

        var cardTitle = „Result of the throw”;
        var cardDescription = „Thanks for using.\n” + „The result is: „ + t_numbers[number];
        var imageObj  = {
                    smallImageUrl: t_images[number].imageSmall,
                    largeImageUrl: t_images[number].imageLarge
        };
        this.emit(‚:tellWithCard’, sOutput,cardTitle,cardDescription,imageObj);
    },
    ‚AMAZON.HelpIntent’: function () {
        this.emit(‚:ask’,helpMessage, helpMessage);
    },
    ‚AMAZON.StopIntent’: function () {
        this.emit(‚:tell’, stopMessage, stopMessage);
    },
    ‚AMAZON.CancelIntent’: function () {
        this.emit(‚:tell’, cancelMessage, cancelMessage);
    }
}

exports.handler = function (event, context, callback) {
    var alexa = Alexa.handler(event, context);
    alexa.registerHandlers(handlers);
    alexa.execute();
};
```



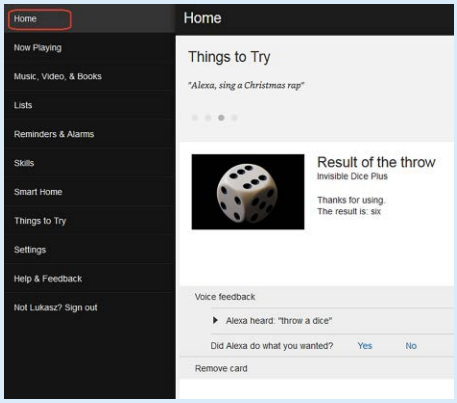
Rysunek 21. Invisible Dice Plus. Diagram działania umiejętności

oba element ze sobą uruchamiamy portalu developera ***http://developer.amazon.com***. Następnie, z listy dostępnych umiejętności wybieramy *InvisibleDicePlus* oraz zaznaczamy edycję umiejętności. W kolejnym kroku uruchamiamy zakładkę *Configuration*. Dane uzupełniamy w identyczny sposób, jak zostało to opisane w pierwszej części kursu (EP 12/2017 – rysunek 24). Jedyną zmianę, którą musimy zrobić, to zmiana identyfikatora ARN (wprowadzamy ARN dla funkcji *Lambda InvisibleDicePlus*, jak na rys. 20.

Testy

Działanie umiejętności testujemy w identyczny sposób, jak to zostało opisane w pierwszej części kursu (EP 12/2017 – rysunek 25). Dla każdej z utworzonych intencji w portalu developera (zakładka *Test*) wprowadzamy zdefiniowane wypowiedzi i weryfikujemy czy odpowiedź asystentki Alexa jest poprawna (interfejs głosowy użytkownika nie był zmieniany, ale modyfikowaliśmy kod programu do obsługi umiejętności, dlatego też należy przeprowadzić testy). Żeby przetestować odtwarzanie audio oraz wyświetlanie obrazów w karcie domowej konfiguratora możemy użyć urządzenia wyposażonego w asystentkę Alexa, a jeśli go nie mamy, to możemy skorzystać w symulatora *Echosim* (***http://echosim.io***).

Na potrzeby artykułu skorzystamy z symulatora *Echosim* oraz z konfiguratora internetowego ***http://alexa.amazon.com***. Uruchamiamy symulator *Echosim* i wydajemy polecenie rzutu kostką (*„Alexa, Open invisible dice plus and throw a dice”*). Sposób korzystania z symulatora pokazaliśmy w pierwszej części kursu (EP 12/2017 – rysunek 30). W trakcie wykonania polecenia nasza umiejętność *Invisible Dice Plus* nie zgłasza błędów. Jest odtwarzany plik audio z dźwiękiem rzucanej kostki, a następnie jest podawany wynik rzutu kostką. Żeby sprawdzić poprawność wyświetlania



Rysunek 22. Invisible Dice Plus. Konfigurator Amazon Alexa. Karta domowa umiejętności

karty domowej logujemy się do konfiguratora ***http://alexa.amazon.com*** i wybieramy zakładkę *Home*. Wygląd karty startowej również jest poprawny (**rysunek 22**). Oznacza to, że wykonana przez nas umiejętność działa poprawnie i możemy przejść do jej publikacji.

Publikacja

Umiejętność niewidzialnej kostki do gry *Invisible Dice Plus* dla asystentki głosowej Alexa jest już gotowa. *Żeby* przekazać umiejętność do certyfikacji oraz do publikacji, postępujemy w identyczny sposób, jak to zostało opisane w pierwszej części kursu (EP 12/2017 – rysunki 26…28). Zmieniamy jedynie nazwę umiejętności, uaktualniamy opis oraz ikony (nowe ikony są dostępne w folderze projektu).

Uruchomienie

Nasza umiejętność została opublikowana i jest dostępna dla wszystkich użytkowników urządzeń z asystentką głosową Alexa! Dla nas (dla właściciela) umiejętność została automatycznie dodana do grupy umiejętności przypisanych do naszego konta. Pozostali użytkownicy, jeśli chcą skorzystać z naszej umiejętności powinni dodać ją do własnego konta w serwisie Alexa. Sposób instalowania umiejętności opisano w pierwszej części kursu (EP 12/2017 – rysunek 29). Po dodaniu umiejętności do konta użytkownika każdy z czytelników może zacząć używanie umiejętności i grać w gry planszowe bez użycia kostki do gry. *Życzę* dobrej zabawy!

Podsumowanie

Dzisiejszym artykułem kończymy kurs tworzenia umiejętności typu *Custom* dla asystentki głosowej Alexa. W kolejnym artykule poświęconym asystentce **głosowej** Alexa opiszemy sposób wykonania umiejętności typu *Smart Home*. Jak sama nazwa wskazuje, **będą to umiejętności** do sterowania urządzeniami w inteligentnym domu.

Łukasz Krysiewicz, EP