Programowanie Intel Edison

Masz już platformę Intel Edison? Zatem do dzieła. Gdy platforma stoi na naszym biurku nadchodzi moment, w którym możemy zająć się już pisaniem programu. Nie jest to jednak takie oczywiste, bo do wyboru mamy kilka opcji. Dobra wiadomość dla programistów jest taka, że na stronie Intela znaleźć możemy informacje przygotowane dla użytkowników Windowsa, Mac OS oraz Linuxa

Podczas testów Edisona korzystałem z Windowsa i tego systemu dotyczy dalsza część tekstu. Jednak kolejność postępowania jest dla różnych systemów jest podobna i nie powinna sprawiać nikomu większych problemów. Aplikacje możemy tworzyć za pomocą 3 kompilatorów:

1. Arduino.
2. Intel XDK IoT Edition.
3. Eclipse.

Ja skorzystałem z programowania Edisona wykorzystując Arduino. Dlaczego? Arduino samo w sobie jest przyjazd dla użytkownika, więc podświadomie liczyłem na **podobnie łatwy start z Edisonem.** Zanim zabrałem się jednak za miganie diodą, musiałem zrobić kilka wstępnych rzeczy.

Aktualizacja systemu, ustawienia

Po wgraniu paczki programów (do pobrania ze strony Intela – *https://goo.gl/k12Y2q*) dołączyłem płytkę do komputera za pomocą dwóch kabli USB. Zgodnie z informacją na stronie producenta zewnętrzny zasilacz jest wskazany, ale nie jest konieczny. Więc oczywiście go nie podłączyłem, co okazało się błędem.

Płytka bez dodatkowego zasilania nie chciała w pełni wystartować, nie była poprawnie wykrywana. Oczywiście najpierw przeinstalowałem sterowniki, wymieniłem kable na krótsze i przeczytałem trochę materiałów w sieci. Ostatecznie **podłączyłem zasilacz i system ruszył w 100%.** Komputer pokazał wyczekiwany przeze mnie nowy dysk.

Następnie zaktualizowałem system operacyjny (Yocto Linux). Odbyło się to stosunkowo łatwo. W dużym skrócie: podłączamy układ do komputera, usuwamy stary system, ściągamy odpowiednie archiwum ze strony producenta, wgrywamy na powyższy dysk, łączymy się z modułem przez Putty, wydajemy komendę: „reboot ota".

Po chwili **ukazuje się nowy system**. Następnym krokiem była **konfiguracja WiFi** – wszystko robimy z poziomu konsoli wydając polecenie: „configure_edison –wifi".

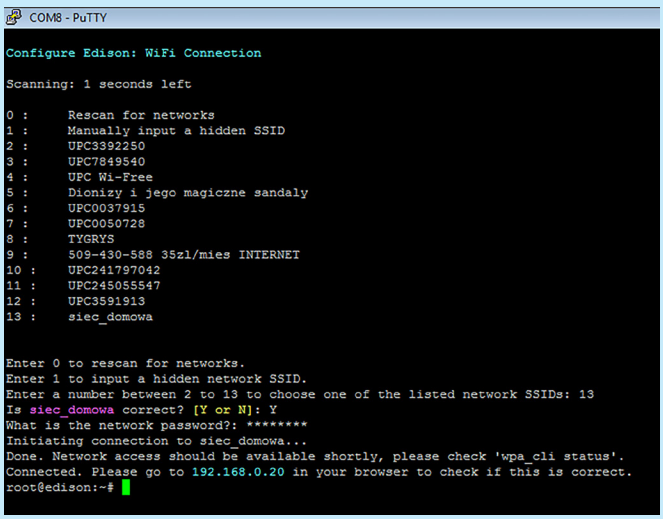
Proces konfiguracji WiFi wygląda jak na **rysunku 1**. Od tej pory możemy połączyć się z platformą przez podany adres IP, co udowadnia zrzut pokazany na **rysunku 2**. Na ten moment pozostawiamy konsolę i przechodzimy do Arduino, które zostało wgrane z całą paczką programów od Intela. Jest to wersja kompilatora, która do długiej listy płytek Arduino ma dodane **Intel Galileo** oraz **Intel Edison**.

Po wybraniu platformy pozostaje wskazanie portu COM, tutaj wybieramy „ten drugi", ponieważ nie wgrywamy programów przez port, który używaliśmy wcześniej do komunikacji z Linuksem.

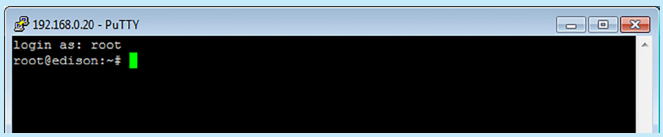
Na początek wgrałem standardowy kody migania diodą: Wstyd jednak wykorzystywać tak rozbudowaną platformę do migania 1 diodą (**rysunek 3**). Dlatego zabrałem się za stworzenie kolejnego programu. Miała to być prosta *stacja pogodowa*, która zbierze następujące informacje z czujników podłączonych do Edisona:

1. Temperatura otoczenia
2. Temperatura pierwszego rdzenia Edisona
3. Natężenie oświetlenia

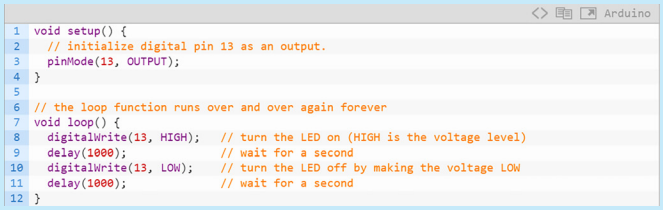
W związku z powyższym przygotowałem, na płytce stykowej, bardzo prosty obwód (**fotografia 4**). Zgodnie z poniższym schematem. W roli czujnika oświetlenia wykorzystałem zwyczajny **fotorezystor**, a rolę termometru powierzyłem czujnikowi analogowemu **LM35**. Potencjometr widoczny na schemacie służył regulacji rezystancji w dzielniku napięcia utworzonym wraz z fototranzystorem. Dzięki temu mogłem łatwo regulować wskazania tego czujnika (**fotografia 5**). Następnie, korzystając z przykładów stworzyłem prosty program, który zbierał informacje



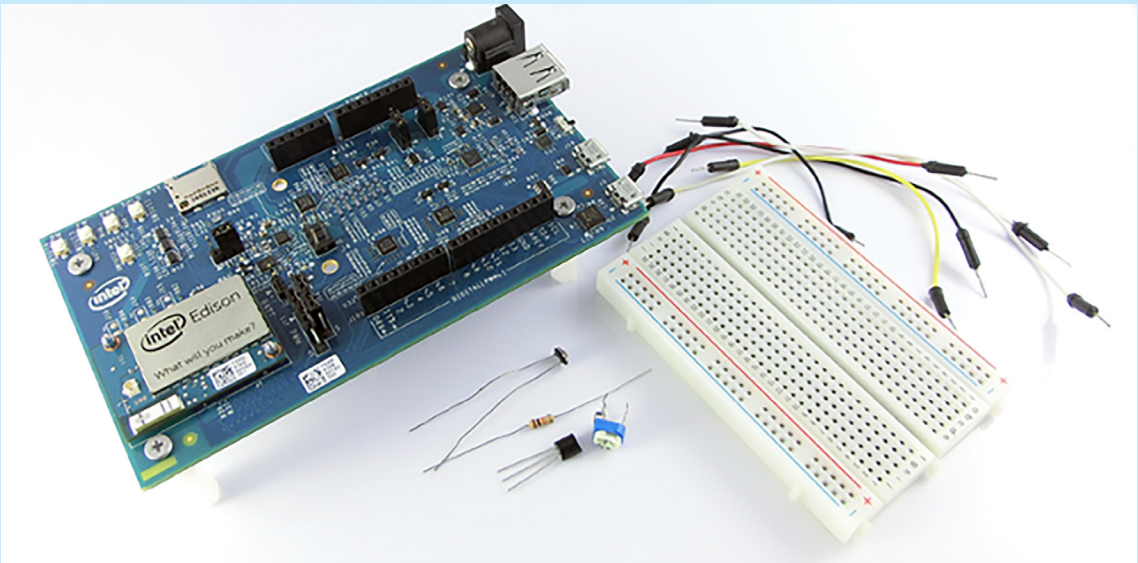
Rysunek 1. Konfigurowanie WiFi na Intel Edison



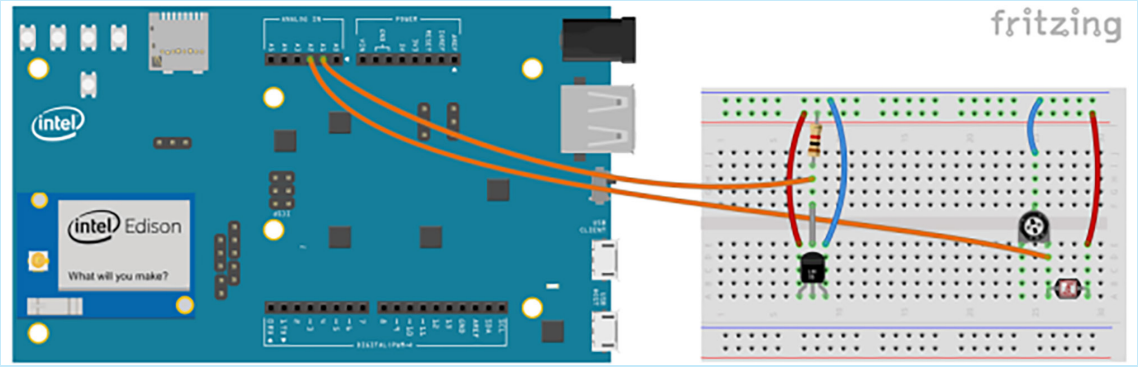
Rysunek 2. Połączenie z Edisonem



Rysunek 3. Standardowy kod migania diodą



Fotografia 4. Elementy potrzebne w przykładzie



Fotografia 5. Schemat do pierwszego przykładu z Intel Edison

z tych czujników. Zbierał, to dużo powiedziane – po prostu je odczytywał i nic z nimi nie robił. Całość wyglądała jak na **rysunku 6**.

Informacje zebrane. Pora na poznanie chmury Intel IoT Analytics.

Pierwsze starcie z chmurą

W celu rozpoczęcia przygody z IoT konieczne jest odwiedzenie strony projektu *Intel IoT Analytics* (*https://dashboard.us.enableiot.com/ui/auth#/login*). Na stronie znajdziemy formularz logowania i rejestracji.

Po zalogowaniu do systemu przechodzimy w ustawienia konta, będziemy musieli pobrać nasz kod aktywacyjny **ważny przez najbliższą godzinę.**

Teraz wracamy do naszego Edisona i łączymy się z nim za pomocą WiFi (przez Putty). Tak jak wspomniałem na wstępie opis ten ma jedynie przedstawić ścieżkę, którą trzeba podążać, dlatego nie będę teraz dokładnie opisywał wszystkich czynności.

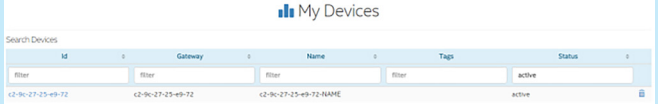
Konieczne jest zainstalowanie agenta *iotkit-admin*, który pozwoli nam na komunikację z chmurą. Następnie wykorzystując podany kod i aktywujemy Edisona. Po kilku sekundach będzie on widoczny w naszych urządzeniach, co pokazano na **rysunku 7**.

Rejestracja komponentów systemu

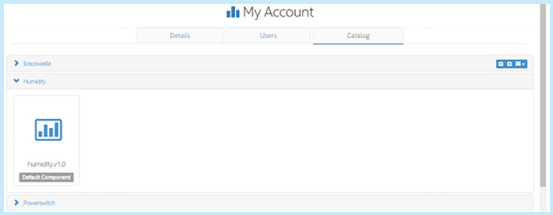
Teraz musimy **zarejestrować komponenty**, które będzie wykorzystywał nasz system. Mówiąc bardziej zrozumiale musimy zadeklarować, jakie informacje będziemy wysyłać

```
Arduino
1  #include <IoTkit.h>    // include IoTkit.h to use the Intel IoT Kit
2  #include <Ethernet.h>  // must be included to use IoTkit
3  #include <aJSON.h>
4
5  IoTkit iotkit;
6  char* therm_file = "/sys/devices/virtual/thermal/thermal_zone3/temp";
7
8  int odczytSwiatla = 0;
9  int tempCore = 0;
10 int tempOut = 0;
11
12 void setup() {
13   pinMode(13, OUTPUT);
14 }
15
16 void loop() {
17   tempCore = getTemp();
18   tempOut = (analogRead(A1) * 5.0) / 1024.0 * 100;
19   odczytSwiatla = analogRead(A2);
20 }
21
22 int getTemp() {
23   bool successful = true;
24
25   int socTemp;
26   char rawTemp[6];
27   FILE *fp_temp;
28   fp_temp = fopen(therm_file, "r");
29   if(fp_temp != NULL) {
30     fgets(rawTemp, 6, fp_temp);
31     fclose(fp_temp);
32   } else {
33     Serial.println("Cannot open file for reading.");
34     Serial.println(therm_file);
35     Serial.println("Try another sensors readings in this directory");
36     successful = false;
37   }
38
39   if(successful) {
40     socTemp = atoi(rawTemp)/1000;
41     return socTemp;
42   }
43   return 0;
44 }
```

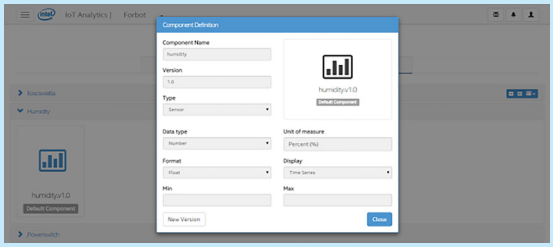
Rysunek 6. Funkcja odczytująca temperaturę rdzenia



Rysunek 7. Urządzenia podłączone do naszego konta



Rysunek 8. Katalog



Rysunek 9. Szczegóły komponentu w chmurze Intel IoT

z Edisona oraz, co ma się z nimi później dziać. W tym celu korzystamy z czujników oraz elementów wykonawczych, które są dostępne w katalogu, na przykład tak, jak na **rysunku 8**.

Widok szczegółowy dostatecznie wyjaśnia, czym są komponenty (**rysunek 9**). Teraz pora na właściwą rejestrację komponentów, którą wykonujemy z konsoli Edisona. Każdy czujnik należy dodać osobno poleceniem zbliżonym do poniższego: *iotkit-admin register tempOut temperaturę.v1.0*.

Każdy element musi mieć przypisany swój **unikalny alias**, w tym wypadku jest to *tempOut* oraz typ – tutaj było to wskazanie temperatury. Wybranie odpowiedniego typu sprawia, że na wykresie zobaczymy później **odpowiednią skalę oraz jednostkę. Po dodaniu komponentów konieczne jest zrestartowanie klienta iotkit-admin!** Teraz możemy wrócić do Arduino i zacząć wysyłać dane.

Intel Edison – wysyłanie danych do chmury

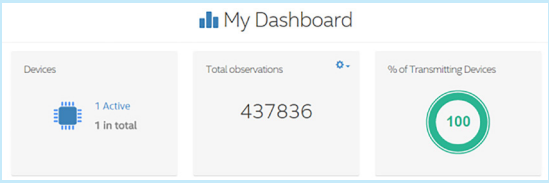
Teraz pora wzbogacić nasz poprzedni program o funkcje wysyłania informacji do chmury. Jest to bardzo łatwe. Wystarczy wywoływać polecenia zbliżone do poniższego *iotkit.send(„tempOut”, tempOut);*. Komunikacja z chmurą nie jest dużo trudniejsza od obsługi UARTa z Arduino. Wywołanie tego polecenia sprawia, że **wartość zmiennej** *tempOut* podanej jako drugi argument zostaje przypisana w chmurze do komponentu zarejestrowanego pod nazwą *tempOut*. Zbieżność nazw tych dwóch wartości jest oczywiście przypadkowa.

Nie musimy martwić się żadnym adresem (ID/IP/MAC) Edisona – wszystko zostało już przypisane i zapamiętane na naszym koncie podczas aktywacji z użyciem kodu.

Cały program wygląda tak, jak pokazano na **rysunku 10**. Po wgraniu programu dane zaczną być regularnie wysyłane do chmury. Będzie to widoczne zaraz po zalogowaniu, na stronie głównej. Znajdziemy tam informacje o **liczbie urządzeń podłączonych do naszego konta**, które wysyłają dane oraz **ilości próbek**, które jakie zgromadzono (**rysunek 11**).

W zakładce chart znajdziemy natomiast, to co najciekawsze, czyli wykresy. Na początku musimy wskazać urządzenie oraz informacje, które nas interesują. Poniżej wyświetla się wykres. Możemy wybrać jak często ma się on odświeżać. Dzięki temu możemy uzyskać

Rysunek 10. Zrzut całego programu



Rysunek 11. Informacja o aktywnych urządzeniach

Rysunek 12. Automatyczna aktualizacja wykresu

wykres zmieniający się *na żywo* (**rysunek 12**). Co ciekawe, chmura pozwala również na definiowanie pewnych reguł wyzwalających pewne akcje. Przykładowo możemy otrzymać e-mail, gdy wartość z czujnika spadnie poniżej zadanej wartości. Opcji jest więcej, jednak na ten moment nie będę ich opisywał.

Podsumowanie przygody z Intel Edison

Mimo braku dużego doświadczenia z IoT muszę przyznać, że wdrożenie się w ten świat było stosunkowo proste. Jak sami widzicie napisanie pierwszego programu korzystającego z GPIO, WiFi oraz chmury nie jest trudne. Zachęcam zatem do podjęcia tego wyzwania.

Damian Szymański