

**AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE**

Wydział Informatyki, Elektroniki i Telekomunikacji
Katedra Informatyki



PRACA DYPLOMOWA

**WYSOKOPOZIOMOWY INTERFEJS DO
ZARZĄDZANIA URZĄDZENIAMI
INTERNETU RZECZY**

WOJCIECH BASZCZYK

PROMOTOR:
dr inż. Włodzimierz Funika

Kraków 2016

OŚWIADCZENIE AUTORA PRACY

OŚWIADCZAM, ŚWIADOMY(-A) ODPOWIEDZIALNOŚCI KARNEJ ZA POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZY PROJEKT WYKONAŁEM(-AM) OSOBIŚCIE I SAMODZIELNIE W ZAKRESIE OPISANYM W DALSZEJ CZĘŚCI DOKUMENTU I ŻE NIE KORZYSTAŁEM(-AM) ZE ŹRÓDEŁ INNYCH NIŻ WYMIENIONE W DALSZEJ CZĘŚCI DOKUMENTU.

.....

PODPIS

Contents

1	Cel prac i wizja produktu	4
1.1	Charakterystyka problemu	4
1.2	Przegląd istniejących rozwiązań	4
1.3	Wizja produktu	6
1.4	Analiza zagrożeń	6
A	Załączniki - KU KDM '16	10
A.1	Abstrakt	10
A.2	Poster	11

1. Cel prac i wizja produktu

Praca polega na stworzeniu infrastruktury na którą składają się urządzenia **ESP8266** ze szczególnym uwzględnieniem **topologii**, **niezawodności** jak i **uniwersalności** systemu. W pracy zbudowany został rozproszony system bazujący na układach ESP8266, realizujący komunikację z **mobilnym serwerem za pomocą zdefiniowanego interfejsu**. Zweryfikowana zostanie hipoteza o możliwości integracji modułów ESP8266 z koncepcją internetu rzeczy i możliwością dynamicznego rozszerzenia stworzonego systemu.

1.1. Charakterystyka problemu

Współcześnie istnieje bardzo wiele urządzeń komunikujących się ze sobą, tworzących swoistą sieć, nazwaną Internet rzeczy[1]. Jest to koncepcja, wedle której jednoznacznie identyfikowalne przedmioty mogą pośrednio lub bezpośrednio gromadzić, przetwarzać lub wymieniać dane za pośrednictwem sieci komputerowej. Do tego typu przedmiotów zaliczają się między innymi urządzenia gospodarstwa domowego, artykuły oświetleniowe i grzewcze[4].

Na rynku istnieje kilka rozwiązań pozwalających na zarządzanie tymi urządzeniami. Są to między innymi opisane szerzej w rozdziale 1.2 Contiki OS, Windows 10 IoT Core, Raspberry Pi czy Zetta.

Celem projektu było stworzenie systemu o podobnej funkcjonalności, pozwalającej użytkownikom zarządzać, zbierać dane, sterować różnymi urządzeniami. Użytkownicy mają mieć wygodny i prosty w obsłudze interfejs.

Przyczyna powstania pomysłu, oraz problem i cel projektu, mając na uwadze przedstawione powyżej **tezy**, można formalnie przedstawić następująco:

- **Przyczyna:** Diametralnie zwiększająca się ilość urządzeń wchodzących w skład Internetu rzeczy[1].
- **Problem:** Stworzenie uniwersalnego interfejsu do efektywnego zarządzania tymi urządzeniami.
- **Rozwiązanie:** Dostarczenie modelu działającego w uproszczonym środowisku, dostarczającego wysokopoziomowy interfejs.

1.2. Przegląd istniejących rozwiązań

Możliwość zdalnego konfigurowania urządzeń wchodzących w skład Internetu rzeczy przez użytkownika końcowego, bądź też samodzielnego działania układu w koncepcji IoT posiada niezliczoną ilość zastosowań w wielu obszarach, takich jak:

- Mieszkania,
- Miasta,
- Przemysł,
- Transport

Istnieje kilka projektów adresujących ten problem. Przykładowe z nich to Contiki OS, Windows 10 IoT Core, Raspberry Pi oraz Zetta. Zapoznanie się z nimi, z rozwiązaniami w nich stosowanymi oraz z oferowanymi przez nie interfejsami pozwoliło uniknąć wielu błędów i problemów występujących podczas projektowania własnego systemu.

Contiki OS [3] (The Open Source OS for the Internet of Things) **Contiki** łączy małe, tanie i energooszczędne mikrokontrolery z internetem. Contiki wspiera w pełni standard **IPv6** i **IPv4**, wraz z energooszczędnymi bezprzewodowymi standardami: **6lowpan**, **RPL**, **CoAP**.

Contiki oferuje łatwe i szybkie tworzenie oprogramowania, aplikacje są pisane w czystym C, wraz z symulatorem **Cooja** systemy te mogą być emulowane przed wrzuceniem ich na hardware. Środowisko jest dostępne i całkowicie darmowe, co jest dużym plusem oprogramowania Contiki.

Contiki można uruchomić na wielu energooszczędnych, bezprzewodowych urządzeniach, takich jak[6]: **CC2538**, **Sensortag**, **CC2650**

Windows 10 IoT Core [7] to również system operacyjny, jest to wersja Windows 10, która została zoptymalizowana dla małych urządzeń bez wyświetlacza, oraz które są uruchamiane na **Raspberry Pi 2 i 3**, **Arrow DragonBoard 410c** i **MinnowBoard MAX**. Windows 10 IoT Core wykorzystuje bogate, uniwersalne **API - Universal Windows Platform (UWP)**. UWP API pozwala na tworzenie aplikacji, która może być używana na wielu urządzeniach, telefonach lub komputerach i udostępnia dostęp do tysięcy urządzeń Windows, które można wykorzystywać w projekcie.

Windows 10 IoT Core wspiera łatwe w użyciu Arduino Wiring API używane w Arduino, oraz biblioteki do bezpośredniego dostępu do urządzeń. Aby tworzyć aplikacje można również używać narzędzia Visual Studio.

Raspberry Pi [5] w odróżnieniu od prezentowanych tutaj rozwiązań jest to platforma komputerowa, która może posiadać zainstalowany system Linux, jak i Windows 10 IoT Core.

Zetta [8] (An API-First Internet of Things Platform) jest to platforma open source zbudowana na Node.js, aby zapewnić możliwość tworzenia serwerów Internetu Rzeczy, które są uruchomione na rozproszonych komputerach. Zetta łączy REST API, WebSockets i reaktywne programowanie - idealne dla łączenia wielu urządzeń w data-intensive, aplikacje czasu rzeczywistego.

Podsumowanie: Pomijając aspekty techniczne, przedstawione systemy mają kilka ważnych, łatwo widocznych cech wspólnych, które powinien posiadać również system implementowany w ramach projektu:

- efektywna komunikacja np. standard Wi-Fi
- energooszczędność urządzeń, mimo ciągłej aktywności
- responsywność oraz stabilność systemu
- niezawodność komunikacji

1.3. Wizja produktu

Finalnym produktem powinien być stabilny prototyp urządzenia, połączony z modulem ESP8266, posiadający przynajmniej jeden czujnik. Użytkownik powinien mieć możliwość zdalnego sterowania tym urządzeniem za pomocą mobilnej aplikacji. System powinien składać się przynajmniej z trzech urządzeń, każde z nich powinno automatycznie łączyć się do sieci Wi-Fi po uruchomieniu.

Dodatkowo, system powinien oferować wygodny interfejs dostępny dla użytkownika końcowego. Udostępnienie API poprzez sterowanie urządzeniami za pomocą telnetu pozwoli na programowy dostęp do sterowania urządzeń z poziomu dowolnego języka programowania. Powinny również zostać zaimplementowane mechanizmy autentykacji zleczanych operacji. Oprócz dostępu programowego, planowana jest implementacja graficznego interfejsu mobilnego, przeznaczonego dla użytkowników[2].

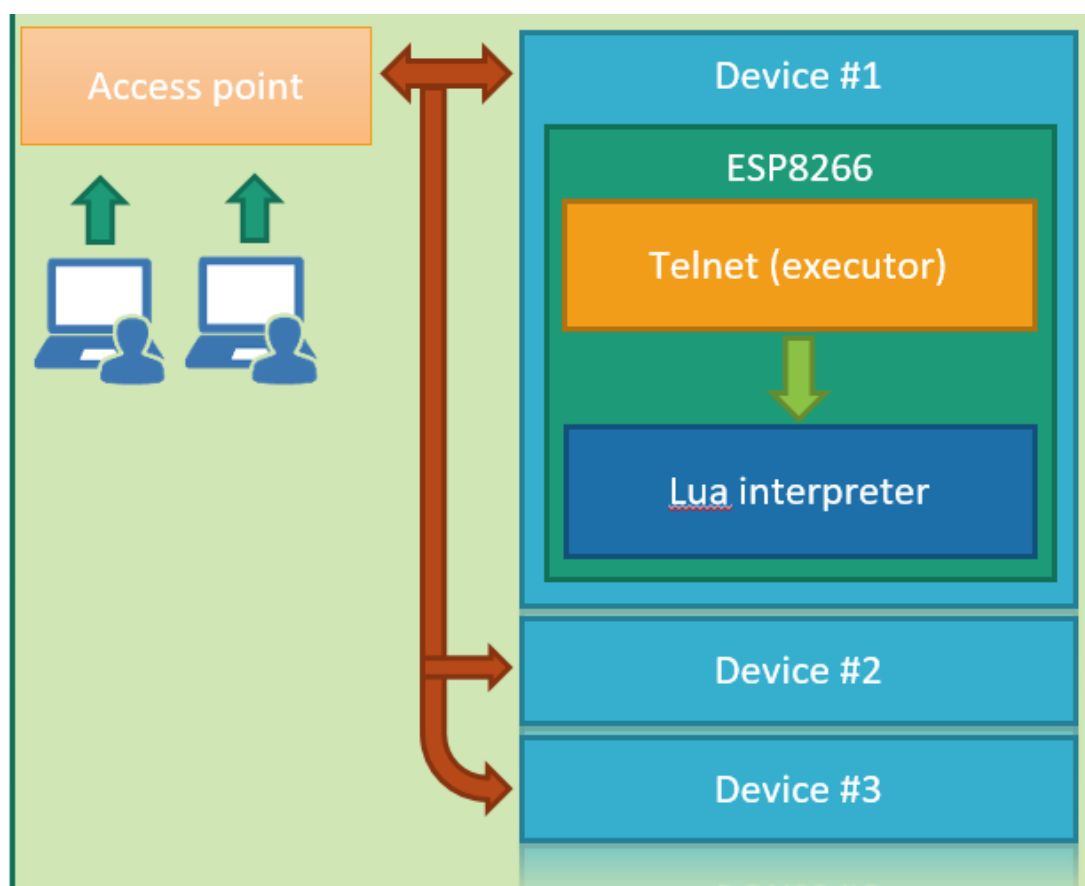


Figure 1: Wizja miejsca użytkownika, budowy urządzenia, oraz sposobu komunikacji

Wizję budowy urządzenia przedstawia Rys. 1.

1.4. Analiza zagrożeń

Wykonalność projektu jest obarczona pewnym ryzykiem. Jest ono głównie związane z możliwą nieznaną jakością obranych w późniejszym czasie technologii oraz małym doświadczeniem w dziedzinie Internetu rzeczy.

Kolejnym zagrożeniem jest ograniczony czas. Proponowane rozwiązanie zakłada stworzenie prototypu urządzenia, a następnie aplikacji, więc stworzenie układu będzie konieczne do uruchomienia i przetestowania całości.

References

- [1] ARM. White paper what the internet of things (iot) needs to become a reality. global strategy and business development. In *Freescale and Emerging Technologies*, June 2014.
- [2] W. Baszczyk and W. Funika. Building a high-level interface with esp8266 for management of devices in iot. In *Proceedings of Cracow Grid Workshop*. ACC CYFRONET AGH, March 2016.
- [3] Contiki. The open source os for the internet of things. <http://www.contiki-os.org/>.
- [4] Kevin Ashton. That 'internet of things' thing. <http://www.rfidjournal.com/articles/view?4986>.
- [5] Raspberry Pi Foundation. Raspberry pi. <https://www.raspberrypi.org/>.
- [6] Texas Instruments. Contiki hardware. <http://www.contiki-os.org/hardware.html>.
- [7] Windows. Windows 10 iot core. <https://developer.microsoft.com/en-us/windows/iot>.
- [8] Zetta. An api-first internet of things platform. <http://www.zettajs.org/>.

List of Figures

1	Wizja architektury systemu.	6
2	Abstrakt przesłany na KU KDM '16. Strona 1.	10
3	Abstrakt przesłany na KU KDM '16. Strona 2.	11
4	Poster zaprezentowany na KU KDM '16.	12

A. Załączniki - KU KDM '16

A.1. Abstrakt

Building a high-level interface with ESP8266 for management of devices in IoT

Wojciech Baszczyk¹, Włodzimierz Funika¹

¹ AGH, Faculty of Computer Science, Electronics and Telecommunication, Dept. of Computer Science, al. Mickiewicza 30, 30-059, Kraków, Poland,

wbaszczy@student.agh.edu.pl, funika@agh.edu.pl

Keywords: Internet of Things, ESP8266, network, communicating machines, circuit board

1. Introduction

We see the Internet of Things [IoT] as billions of smart, connected “things” that will encompass every aspect of our lives [1]. The IoT is comprised of smart machines interacting and communicating with other machines, objects, environments and infrastructures. The IoT is defined in many different ways, and it includes many aspects of life - from connected homes and cities to connected cars and roads to devices that track an individual’s behavior and use the data collected.

In this paper we present the results of an in-depth research into IoT issues. We aimed to develop a prototype of service that provides a high-level interface to a set of IoT devices using ESP8266, which could manage devices of IoT. Our goal was to create a circuit board integrated with ESP8266 and software that ensures management of a set of devices communicating over Wi-Fi network.

ESP8266 is a highly integrated chip. It offers a complete and self-contained Wi-Fi networking solution, allowing it to either host the application or to offload all Wi-Fi networking functions from another application processor. We have chosen ESP8266 due to its powerful on-board processing and storage capabilities that allow it to be integrated with sensors and other application-specific devices through its GPIOs.

2. Description of a problem solution

Our prototype system consists of a number of elements (see Fig. 1, where the communication interface is shown). It represents a circuit board (Device) communication with various types of clients, connected to the same network, like android applications or telnet clients running on a given operating system.

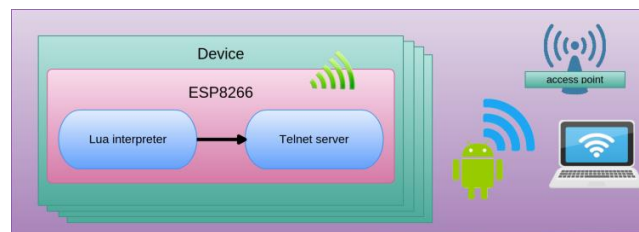


Fig. 1 Circuit board communication interface

Lua [2] is a lightweight programming language designed primarily for embedded systems and clients. It is running on ESP8266 processor. There are some important functions defined and loaded on startup. Basically this platform has been selected because it is easy to control the system just by interpreting commands.

Telnet provides an interactive text-oriented communication facility using a virtual terminal connection, which features great flexibility. There are commands defined, each command can control a different functionality of the system, each of these commands is interpreted by Lua and executed. This solution makes it easy to define an own system, which can provide a user friendly interface for the management of devices in IoT.

3. Results

As a proof-of-concept we have created an android application that implements a circuit board interface. It provides a view for connecting to devices. The devices are configured to control LED brightness by pulse-width modulation (PWM) also as a proximity detector located on circuit boards. The application sends commands to telnet clients and steers each device separately. The functions used by the telnet client are written in Lua and located on ESP8266 microcontroller, at startup each device loads appropriate procedures.

The main aspect to test was the reliability provided by our modules the system is composed of. The testing procedure relies on user activities. At first the user connects to different circuit board modules. Then there are various type of views that provide user friendly interfaces, e.g., setting LED brightness (see Fig. 2). Within the tests performed, the system proved its responsiveness and proper functioning.

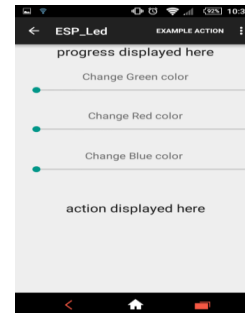


Fig. 2 LED setting GUI

4. Concluding remarks

There are many systems that provide similar solutions for management issues of the Internet of Things. For example, Contiki [3] is an operating system for the Internet of Things, it connects tiny low-cost, low-power microcontrollers. By contrast, we aimed to develop a solution with convenient circuit boards and interfaces. The test results are very promising - the system is responsive and stable. Future work aims to extend the current API and allow the user to develop their own modules by involving various sensors, e.g., for temperature, humidity, etc. The management of devices requires some changes, with special focus on the configurability of the environment. We are going to make the circuit board more adjustable, so that the user could connect and manage various devices of their choice.

Acknowledgements. This research is partly supported by AGH grant no. 11.11.230.124.

References

1. White Paper What the Internet of Things (IoT) Needs to Become a Reality. Global Strategy and Business Development, Freescale and Emerging Technologies, ARM, 2014.
2. What is Lua?. (2016, February 4). <http://www.lua.org/about.html>
3. Why Choose Contiki?. Contiki: The Open Source OS for the Internet of Things. <http://contiki-os.org/index.html#why>

Figure 3: Abstrakt przesłany na KU KDM '16. Strona 2.

A.2. Poster

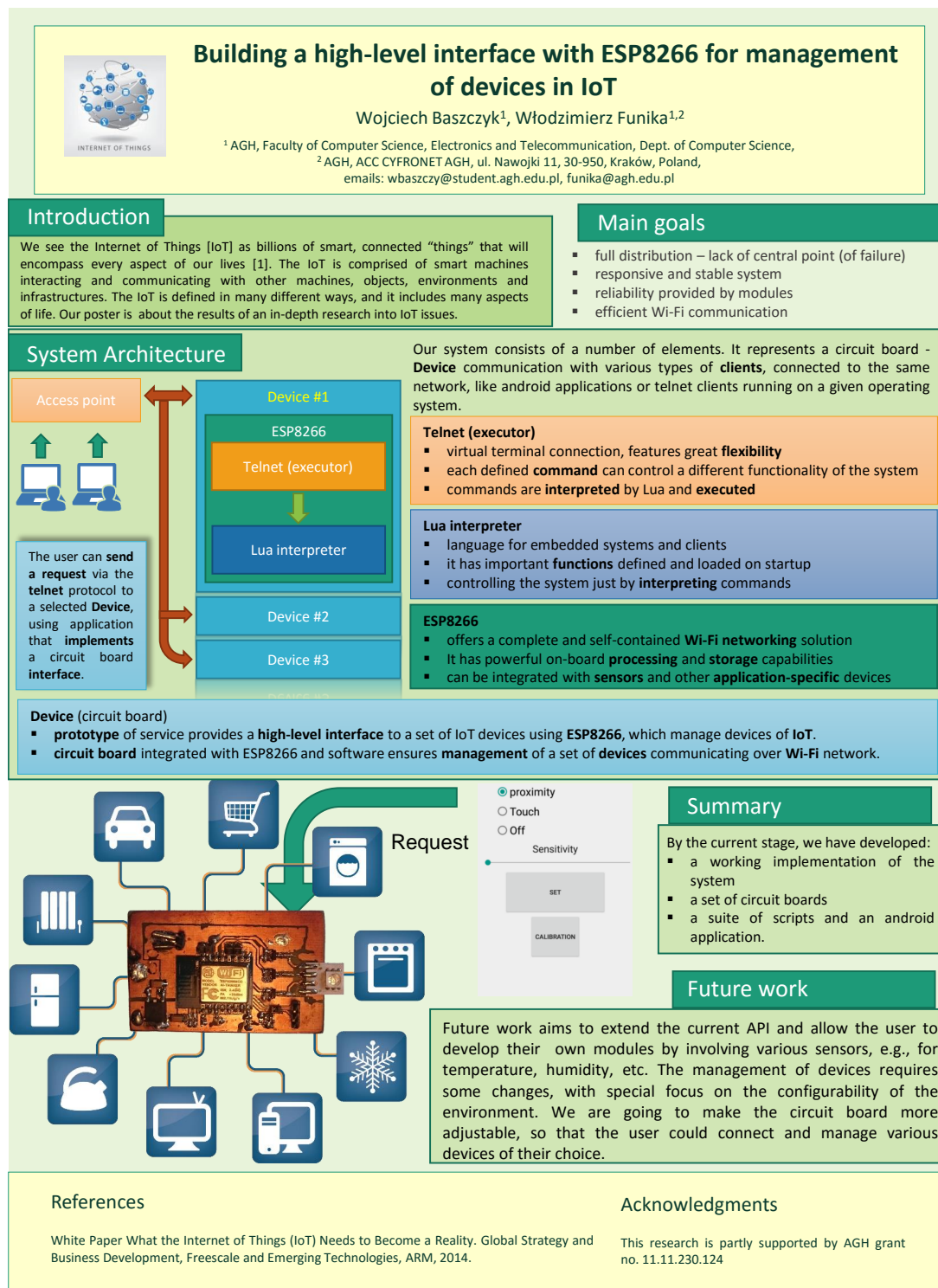


Figure 4: Poster zaprezentowany na KU KDM '16.