**DOI: 10.5604/20830157.1194288**

# INTERACTIVE, MULTIFUNCTIONAL MIRROR – PART OF A SMART HOME

**Szczepan Paszkiel, Robert Kania**
Opole University of Technology, Institute of Control & Computer Engineering

***Abstract**. This paper presents a possible implementation of a personal assistant and control interface for a smart home. A prototype is presented, featuring functions such as appliance and light control, a map system based on Google Maps and various informational data such as calendar entries and news. The device takes the form of a voice controlled mirror, allowing for integration in existing systems as a replacement for a common household item, without the need for additional space.*

**Keywords**: control interface, smart home, mirror, Google Maps

## INTERAKTYWNE, WIELOFUNKCYJNE LUSTRO JAKO ELEMENT INTELIGENTNEGO BUDYNKU

***Streszczenie.** W artykule przedstawiono możliwą implementację osobistego asystenta w postaci interaktywnego lustra wchodzącego w skład instalacji inteligentnego domu. Opracowany autorski prototyp urządzenia wyposażono w funkcje, takie jak: kontrolę światła w pomieszczeniu, system wizualizacji map opartych o Google Maps oraz możliwość prezentacji wielu danych informacyjnych, takich jak: kalendarz, bieżące wiadomości etc. Urządzenie ma postać głosowo sterowanego lustra opracowanego według autorskiego pomysłu z możliwością integracji z istniejącymi na rynku systemami operacyjnymi urządzeń mobilnych, jako zamiennik dla wspólnego elementu gospodarstwa domowego jakim jest standardowe lustro, bez konieczności wykorzystywania dodatkowej przestrzeni w budynku.*

**Słowa kluczowe**: interfejs, dom inteligentny, interaktywne lustro, mapy Google

## Introduction

In an age where one can share what he just ate using his fridge, or accept calls using one's watch, it has become a challenge to find ordinary items which may yet be modernized [1]. The current approach to smart home solutions is the addition of a central control panel, commonly a touch device, accompanied by a commercial handheld device with an appropriate app installed [2].

This paper presents an alternative approach, where a common household item – the mirror – is equipped with a control panel's functionalities without impeding its inherent use.

This is done by creating a voice controlled interface, thus making the interaction appear more natural and eliminating the need for physical interaction and cleaning. Additionally features of a personal assistant are added, allowing the user to instantly get any required information by quickly glancing at a part of the mirror while getting ready, or specifically asking for it [3, 5].

## 1. Setup

### 1.1. Hardware

The mirror is composed of a NTT Corrino 617 SU laptop running Ubuntu 14.04, with a Iiyama LED 22" E2283 as external screen and an external microphone. Additionally, a 4 millimeter one-way mirror is put in front of the screen for the desired effect as seen in figure 1. The Iiyama E2283 was chosen as screen due to its built-in speakers and connector layout – they are on the side, allowing for more compact cable management when connecting to the laptop, or any future processing unit.

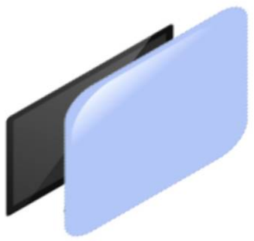


*Fig. 1. Reference drawing of the screen*

Several ESP8266 microcontrollers have been added, connected to, depending on the intended use: DS18B20 digital thermometer for temperature sensors or a 230 V AC relay for controlling lights and other AC appliances.

The ESP8266 has been chosen due to its low price, power usage, processing power and most importantly Wi-Fi capabilities. For coordination of the above mentioned devices, a TP-Link 1043ND router running OpenWRT is used.

### 1.2. Software

The interface consists of a web page displayed by Google Chrome. This approach allows rapid development and sets a foundation for easy extensibility of functionalities. For voice recognition and speech synthesis capabilities, the Web Speech API is used. With its' specification published on 19 October 2012, it is one of the newer additions to web technologies [8]. Although not a W3C standard, parts of the API are being incorporated into popular browsers. Interfacing with additional devices driven by the ESP8266 is done with the MQTT protocol – a machine-to-machine (M2M)/"Internet of Things" connectivity protocol [7]. Due to its minimalistic nature it is well suited for use in microcontrollers. Information in the MQTT protocol is divided into topics, which any device can either subscribe or post to. For example, the topic /lights/kitchen/currentStatus could be subscribed to, in order to always be aware of the light's status, whereas the microcontroller responsible can post messages whenever the state is switched.

In order to provide personal assistant functionalities, a popular calendar application is used – Google Calendar API, which has been chosen mainly due to its free of cost nature. Nevertheless, interfacing with other, paid alternatives such as Apple iCloud is also possible.

Additionally, a Node.JS HTTP server is introduced to overcome the browser's sandboxed nature. Normally, HTML websites can be served locally using the file:// protocol. This approach could not be used for the following reasons: file:// is not seen as a secure connection by Chrome, and therefore, in accordance with the Web Speech API, prompts for user input every time a web page requests access to the microphone; the requests to the Calendar and Translate APIs could not be made due to Cross-site request forgery prevention measures; communication with MQTT is not possible using a Web browser alone.

## 2. Interface and implementation

The mirror provides some features by default, without the need for widgets or modifications. These include displaying the date and time, upcoming calendar events, local weather, post it functionality and maps. The interface itself is designed in an elegant, minimalistic manner, featuring soft fonts such as *Helvetica Neue* and light colors. Additionally, any of the base elements can be hidden by issuing the command *hide* (pl. *schowaj)* <component name>. A combination of these elements can be seen in figure 2.

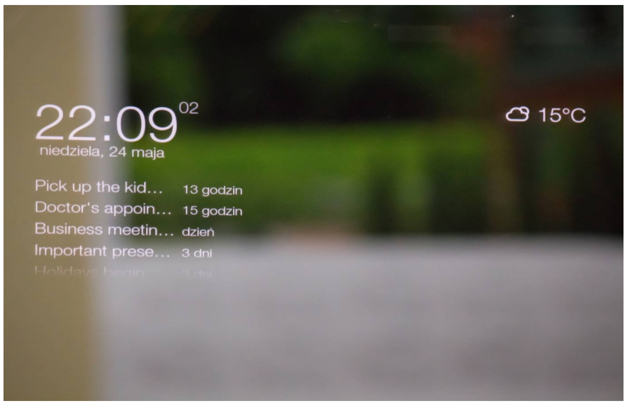


*Fig. 2. Upper part of mirror, depicts date, time, weather and calendar*

The weather is provided using the simpleWeather jQuery plugin, and shown using the weathericons font. Any interaction with the Web Speech API is handled by the annyang! library. Any commands are easily added by issuing a command with a dict as argument, such as annyang.addCommands({'schowaj *term':voice.hide});, where the desired term is the key, and the function to be called is the value of the argument given, making maintenance and easy addition of commands possible.

The map functionality is provided using Geolocation and a custom styled map provided by the Google Maps. The user may ask the mirror for any location – the map in figure 3 was produced by asking: *how to find to Mikolajczyka street in Opole* (pl. *jak dojadę do Mikołajczyka Opole)* – with a correct result. Alternatively, the from location can be set in the settings, and the style of the map can be changed to the default one provided by Google Maps. The map can be dismissed by issuing the command: *delete map* (pl. *usuń mapę)*.

*Fig. 3. The entire interface is overlaid with the requested map*

## 3. Extensibility

Since not all possible hardware configurations and software functionalities can be predicted, a simple Widget API has been implemented to allow for custom extensions to be added [4]. In order to add a new widget, these steps must be followed: The widget must be placed as its own directory inside the /js directory; its own directory must contain a file called script.js; it must be listed inside the config.js file as a plug-in to be loaded.

As an example, a widget for studying the Chinese language, specifically its characters, has been implemented, with a file structure as seen below:

```
chinese/
    lists/
                HSK1.txt
                canvas.curve.min.js
    draw.js
    script.js
    style.css
```

The script.js file is loaded once the $(document).ready() event fires. The file later runs the following snippets:

```
    var js          = document.createElement("script");
    js.type         = "text/javascript";
    js.src          = "/js/chinese/draw.js";
    js.onload       = chinese.init;
    document.body.appendChild(js);
    [...]
```

In order to ensure a proper loading order, so that no script is ran before its dependencies are available, the onload attribute is used. Later, the init function counts the amount of times it has been called, and once a certain threshold is reached, the actual initialization is ran.

```
document.body.innerHTML += canvasHtml;
annyang.addCommands({"nowy znak":chinese.newChar});
```

As seen in this example, widgets are able to easily do the following things: inject its' own dependencies, add HTML to the DOM and add new voice commands.

## 4. Interaction with hardware using ESP8266 and MQTT

The MQTT API has been designed to be as transparent as possible, as to facilitate further additions both in hardware and in software plug-ins. The MQTT Node.js library has been used [9], and three methods have been made available: /mqtt/subscribe?topic=name to subscribe to a topic, /mqtt/publish?topic=name&message=value to publish a message to a topic, and finally /mqtt/get?topic=name to receive the latest publication, along with the date received. All ESP8266 are written in Lua, under the *NodeMCU* environment. Additionally, the scripts are designed around the state machine as seen in figure 4.

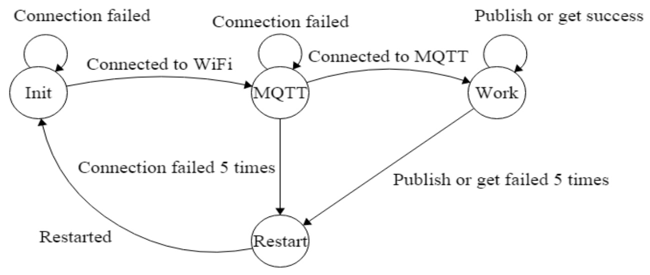


*Fig. 4. ESP8266 finite state machine*

As an example, a lighting module using WS2812 has been implemented. The WS2812 is a RGB LED controlled by PWM, for which *NodeMCU* has a built-in driver starting with the 0.9.6 version (as of 14th June 2015 still in development). Since the WS2812 uses 5 V as its' native voltage, a logic level shifter has been used in order to drive it from the 3.3 V rated ESP8266.

The following code snippets drive the LEDs inside the ESP8266:

```
m:connect( broker , mqttport, 0,
    function(conn)
m:subscribe("diode/color" , 0, function(conn)
print("Subscribing topic:  diode/color")
                end)
                mqtt_state = 20 -- Go to work state
    end)

m:on("message", function(conn, topic, data)
    print(topic .. ":" )
    if (data ~= nil ) then
                r,g,b = string.match(data, "(%d+),(%d+),(%d+)")
                if (r ~= nil and g ~= nil and b ~= nil ) then
                print (r .. ":" .. g .. ":" .. b)
                diode(r,g,b)
                end
    end
end )
function diode(r,g,b)
    ws2812.writergb(6, string.char(g, r, b):rep(2))
end
```

The code subscribes to the *diode/color* topic. Later, when a message is received, formatted like *128,128,128* it is split into three separate integers, each containing the intensity of one light channel on a scale from 0 to 255. Finally *ws2812.writergb* is called.

The first argument is the GPIO port to which the diodes are connected. As WS2812 may be chained (DO to DI connection), settings to several LEDs can be sent at once using the :rep(n) operator, with no additional GPIO ports required.

Finally, a simple plug-in on the interface side is written:

```
ws2812.set = function(r,g,b){
    $.get(
"/mqtt/publish?topic=diode/color&message="+r+","+g+","+b,
function( data ) {
                console.log("Light set
to:"+"/light?r="+r+"&g="+g+"&b="+b);
    })
};
ws2812.speech = function(word){
    switch(word){
                case "czerwone":
                ws2812.set(255,0,0);
                break;
                [...]
    }
};
$(document).ready(function(){
    annyang.addCommands({"ustaw *term
światło":ws2812.speech});
});
```

The code allows the WS2812 to be controlled using voice commands. Stating *set the light red* (pl. *ustaw czerwone światło)* sets all LEDs to red, for example. Various combinations can be implemented, or entirely different use scenarios (for example, a relay could be connected, and a WS2812 could be used as state indicator).

## 5. Summary

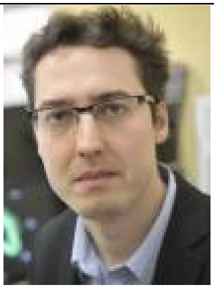
The mirror in its current state is ready for usage by technology enthusiasts. The next step in development would be creation of a voice controlled setup, including initial configuration and addition of plug-ins or hardware. Additionally, parts of the application could be ported to lower level languages, specifically the Web Speech API, as to allow usage of budget hardware for the mirror, eg. the Raspberry Pi 2 (RPi2). Alternatively, other parts of the mirror could be perfected until Windows 10 and Google Chrome is fully compatible with the RPi2.

## References

[1] Alam M., Reaz M., Ali M.: A review of smart homes – past, present, and future. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 42(6), 2012, 1190–1203.
[2] Brezovan M., Badica C.: A review on vision surveillance techniques in smart home environments. Proceedings of the 19th International Conference on Control Systems and Computer Science: (CSCS19), vol. 2, 2013, 471–478.
[3] Green W., Gyi D., Kalawsky R., Atkins D.: Capturing user requirements for an integrated home environment. Proceedings of the Third Nordic Conference on Human-Computer interaction, Tampere, Finland. ACM Press, New York, NY, vol. 82, 2004, 255–258, [DOI: 10.1145/1028014.1028053].
[4] Kim S. W., Park S. H., Lee J., Jin Y. K., Park H., Chung A., Choi S., Choi W. S.: Sensible appliances: applying context-awareness to appliance design. Personal Ubiquitous Comput. 8, 2004, 184–191, [DOI: 10.1007/s00779-004-0276-9].
[5] Lê Q., Nguyen H. B., Barnett T.: Smart Homes for Older People: Positive Aging in a Digital World, Future Internet, 4, 2012, 607–617, [DOI:10.3390/fi4020607].
[6] Morris M. E., et al.: Smart-Home Technologies to Assist Older People to Live Well at Home, Journal of Aging Science, 2013, [DOI:10.4172/jasc.1000101].
[7] http://mqtt.org/ [13.06.2015].
[8] https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html [13.06.2015].
[9] https://www.npmjs.com/package/mqtt [15.06.2015].

**Ph.D. Eng. Szczepan Paszkiel**
e-mail: s.paszkiel@po.opole.pl

Szczepan Paszkiel currently working in the Institute of Control and Computer Engineering on the Faculty of Electrical Control and Computer Engineering at Opole University of Technology. He is a graduate of Informatics and Management and Production Engineering at Opole University of Technology. He is a grant holder and winner of many contests for young scientists. He has been conducting research on processing the EEG signal. He is an author and co-author of many scientific publications.



**Eng. Robert Kania**
e-mail: art@robus.info

Robert Kania, graduated with a Engineer degree of Computer Engineering from Opole University of Technology, Opole, Poland. Currently applying at the Opole University of Technology, for a Master in Computer Engineering.

*otrzymano/received: 03.09.2015* *przyjęto do druku/accepted: 15.01.2016*