Sterownik taśm LED RGB+W zgodny z HomeKit

Szukając rozwiązań odpowiednich dla urządzeń z kategorii smarthome, trafiłem na bibliotekę, która pozwala uruchomić serwer HomeKit w urządzeniu. Biblioteka nazywa się esp-homekit a jej twórcą jest użytkownik o nicku maximkulkin. Dodatkowo biblioteka umożliwia generowanie kodu QR, za pomocą którego dodajemy urządzenie do aplikacji. Postanowiłem użyć biblioteki do budowy sterownika do taśm LED.

Pierwszy artykuł z kategorii smarthome mojego autorstwa, opublikowany w EP 09/19, dotyczył zdalnego sterowania oświetlenia na 230 V. W tamtej wersji sterownika wymagane było użycie mostka na Raspberry Pi, który umożliwiał kontrolowanie urządzenia poprzez aplikację Home firmy Apple. Po sterowniku oświetlenia konstruowałem kolejne sterowniki i uciążliwe stało się utrzymywanie instancji homebridge na Rasperry Pi oraz dodawanie do pliku konfiguracyjnego kolejnych akcesoriów.

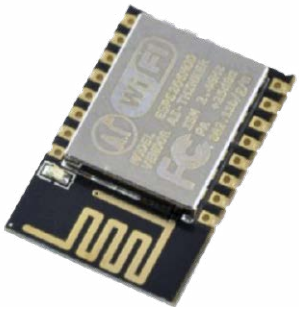
Postanowiłem zagłębić się bardziej w protokół HAP. Okazało się, że Apple udostępniło instrukcję budowania aplikacji przy użyciu tego protokołu. Dodatkowo dostępne są specjalne mikrochipy, dzięki którym można uzyskać certyfikat zgodności z aplikacją. Niestety, żeby mieć dostęp do takich informacji, trzeba być deweloperem homekit, a w tym celu należy mieć firmę oraz specjalną licencję. Wyrobienie licencji wiąże się ze znacznymi kosztami, dlatego zrezygnowałem z tego.

Do budowy sterownika do taśm LED postanowiłem użyć wspomnianej wcześniej, biblioteki *esp-homekit* ***https://bit.ly/3g8y-PyC***. Głównym wymaganiem stawianym sterownikowi taśm LED była uniwersalność konstrukcji, dlatego urządzenie może sterować taśmami LED jednego koloru, RGB oraz, coraz częściej stosowanymi, RGB+W. Dostosowanie sterownika do typu taśmy odbywa się za pomocą odpowiedniej wersji programu.

Budowa i działanie

Jednostką sterującą urządzeniem jest moduł ESP8266 12F (**fotografia 1**), który wybrałem bez wahania. Zbudowałem już kilka różnych urządzeń na tym chipie i zawsze spełniał moje oczekiwania.

Ważnym elementem układu była przetwornica i tranzystory MOSFET do sterowania taśmą. Z racji tego, że na rynku dostępne są taśmy z różną wielkością diod



Fotografia 1. Moduł ESP8266 12F

Dodatkowe materiały do pobrania ze strony www.media.avt.pl

W ofercie AVT* AVT-5778

Podstawowe parametry:
- mały, podłużny kształt, wymiary gotowego sterownika w obudowie: 26×48×16 mm,
- podłączany szeregowo między taśmę a zasilacz,
- zgodność z HomeKit,
- maksymalne obciążenie wyjść: 5 A dla każdego koloru,
- zasilanie 12 V.

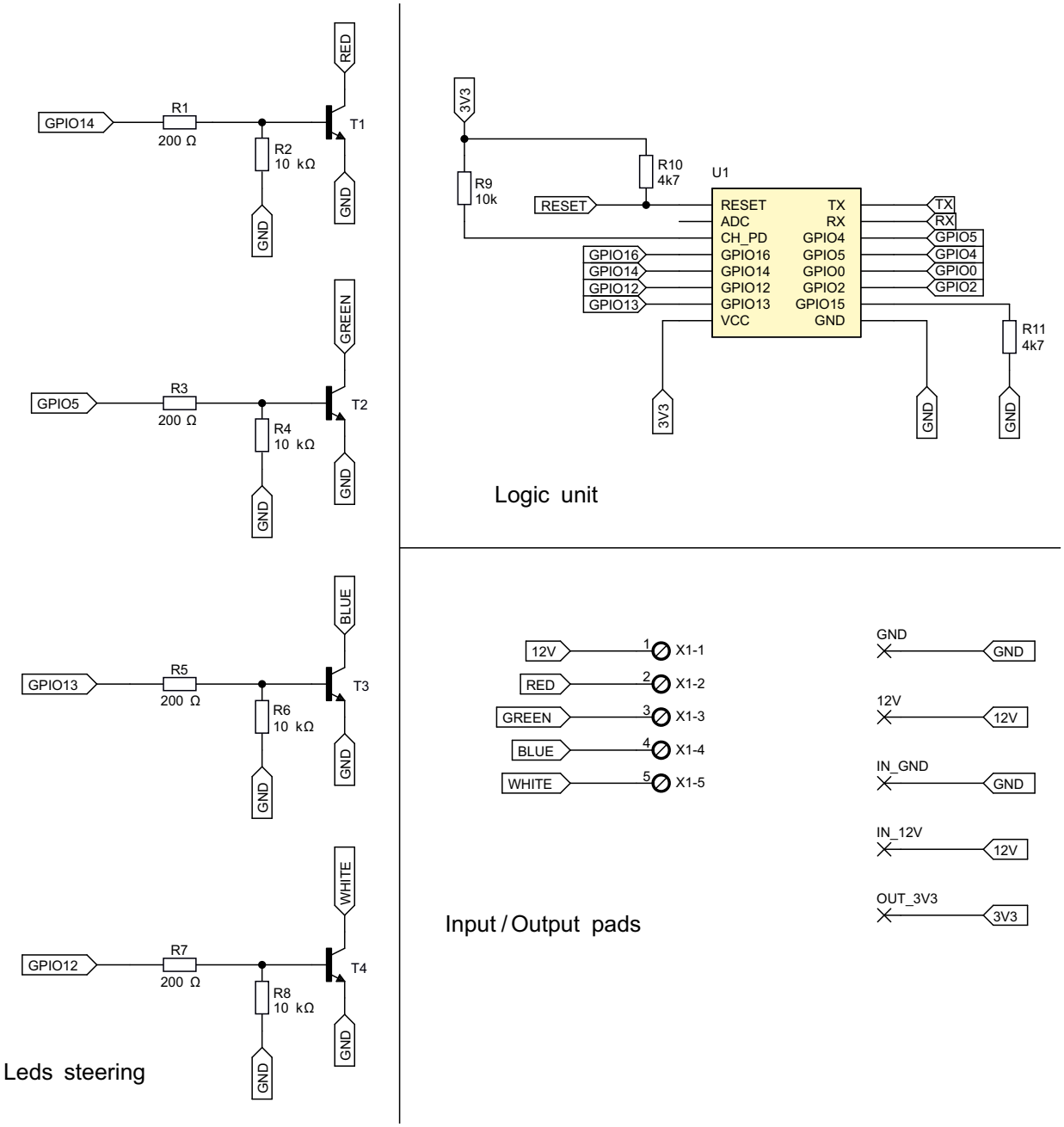
Projekty pokrewne na www.media.avt.pl:

AVT-5768	Zasilacz diod power LED 3,5 W (EP 5/2020)
AVT-5733	Programowany sterownik LED dużej mocy (EP 12/2019)
AVT-5706	Sterownik płynnego rozjaśniania i wygaszania oświetlenia LED (EP 8/2019)
Projekt 237	Wyłącznik taśmy LED – bariera podczerwieni (EP 12/2018)
AVT-1996	Bedlight – sterownik oświetlenia nocnego z czujką ruchu (EP 8/2018)
AVT-1975	Powolny rozjaśniacz do taśm LED 12 V (EP 7/2017)
AVT-5561	Efektowny sterownik oświetlenia (EP 12/2016)
AVT-1918	Oświetlacz pierścieniowy LED (EP 8/2016)
AVT-1912	Miniaturowy sterownik taśmy LED RGB (EP 7/2016)
AVT-5536	Sterownik taśmy LED ze zdalnym sterowaniem (EP 4/2016)

* **Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!**
Podstawową wersją zestawu jest wersja [**B**] nazywana potocznie KIT-em (z ang. zestaw). Zestaw w wersji [**B**] zawiera elementy elektroniczne (w tym [**UK**] – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu.
Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:
- wersja [**C**] – zmontowany, uruchomiony i przetestowany zestaw [**B**] (elementy wlutowane w płytkę PCB)
- wersja [**A**] – płytka drukowana bez elementów i dokumentacji

Kity w których występuje układ scalony wymagający zaprogramowania, mają następujące dodatkowe wersje:
- wersja [**A+**] – płytka drukowana [**A**] + zaprogramowany układ [**UK**] i dokumentacja
- wersja [**UK**] – zaprogramowany układ

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz!
http://sklep.avt.pl. W przypadku braku dostępności na ***http://sklep.avt.pl***, osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt via e-mail: kity@avt.pl.



Rysunek 1. Schemat elektryczny urządzenia

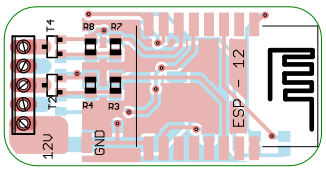
LED, postawiłem wybrać N-MOSFET typu AO3400A. Prąd drenu wynosi aż 5 A, a obudowa to niewielkich rozmiarów SOT23. Po zagłębieniu się w temat przetwornic, doszedłem do wniosku, że nie jest łatwo samemu zaprojektować przetwornicę, gdyż istotne są takie parametry, jak np. grubości ścieżek. Udało mi się znaleźć gotowy moduł regulowanej przetwornicy odpowiedniej dla prezentowanego sterownika. Przetwornica miała pady do montażu goldpinów, ale postanowiłem, że będzie montowana powierzchniowo. Wymagało to zaprojektowania na płytce padów, które będą idealnie pasowały do tych z przetwornicy.

Po wybraniu wszystkich kluczowych komponentów przystąpiłem do projektowania schematu oraz płytki PCB w programie Eagle. Schemat elektryczny jest pokazany na **rysunku 1** a schemat płytki pokazują **rysunki 2** i **3.** Płytka jest dwustronna, ponieważ podczas projektowania chciałem osiągnąć jak najmniejszy rozmiar oraz aby urządzenie było podłużne do wpięcia szeregowego między taśmę a zasilacz.

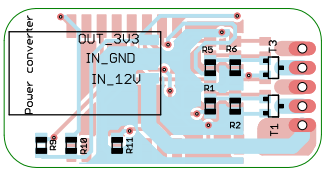
Wykaz elementów:
Rezystory: (SMD 0805)
R1, R3, R5, R7: 200 Ω
R2, R4, R6, R8, R9: 10 kΩ
R10: 4,7 kΩ

Półprzewodniki:
T1...T4: AO3400A N-MOSFET SOT23
U1: ESP8266 12F
przetwornica: DC/DC regulowana, oznaczenie 9168 (https://abc-rc.pl/)

Pozostałe:
gniazdo zasilające DC 2.1/5.5 z przewodem
przewód ze złączem do taśm LED
obudowa



Rysunek 2. Schemat płytki PCB, strona TOP



Rysunek 3. Schemat płytki PCB, strona BOTTOM

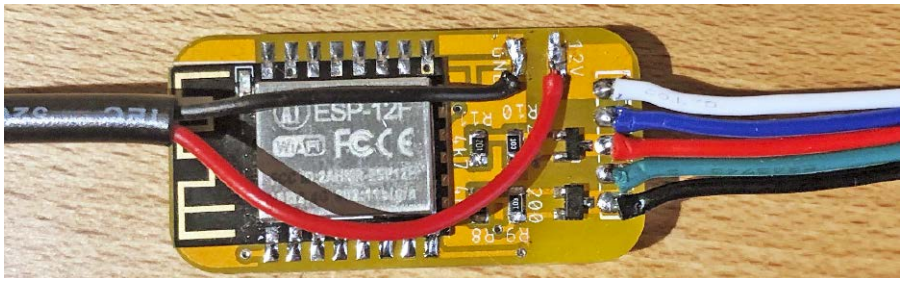
Montaż

Na płytce znajduje się niewiele elementów i nawet mniej doświadczone osoby poradzą sobie z montażem. Punkty dołączenia taśmy LED to pady pod złącze ARK z rastrem 2,54 mm, ale bez problemu można tam przylutować kable ze złączkami taśm

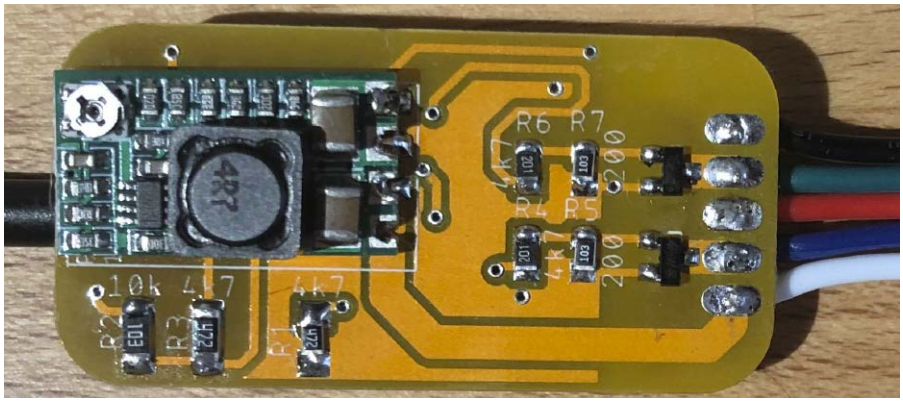
LED. Zasilanie dostarczane jest za pomocą gniazda DC 2,1/5,5, które jest podłączone przewodami do płytki PCB. Zlutowaną płytkę pokazuje **fotografia 2 i fotografia 3.**

Dodatkowo postanowiłem zaprojektować obudowę do urządzenia. Program Eagle zawiera do niektórych elementów biblioteki 3D, a do innych należy je doinstalować samemu. Całą płytkę PCB wyeksportowałem do programu Fusion 360. Nie zwracałem uwagi na brak biblioteki do przetwornicy napięcia, gdyż na płytce były zaznaczone jej wymiary, a wysokość zmierzyłem suwmiarką. Wizualizację pokazuje **rysunek 4**.

Obudowę wydrukowałem z materiału PETG. Zamykana jest na „klik", wystarczy wcisnąć wieczko w obudowę. W środku postanowiłem zaprojektować dwa „klipsy", które trzymają całą płytkę PCB. Wymiary gotowego sterownika w obudowie to 26×48×16 mm.



Fotografia 2. Zmontowany sterownik, strona TOP



Fotografia 3. Zmontowany sterownik, strona BOTTOM

Rysunek 4. Wizualizacja obudowy

Oprogramowanie

Opisane zostaną trzy warianty oprogramowania sterującego pracą urządzenia. Kod bazuje na darmowej bibliotece ESP-Homekit od użytkownika MaximKulkin. Biblioteka została opracowana do frameworka ESP-OPEN-RTOS (***https://bit.ly/2NCBJiX***). Język programowania to dalej C++. MaximKulkin zamieścił też przykładowe projekty na swoim githubie (***https://bit.ly/3g8yPyC***).

Sterowanie taśmą z jednym kolorem

Program składa się z minimum dwóch plików: *Makefile* oraz *main.c*. Zawartość pliku *Makefile* pokazuje **listing 1**. Na samym początku musimy zadeklarować nazwę programu (PROGRAM = LED_strip). Następnym krokiem jest dołączenie zewnętrznych bibliotek (EXTRA_COMPONENTS). Komponent Wolfssl zapewnia nam szyfrowaną komunikację.

Następnie deklarujemy piny, do których jest dołączona taśma LED (w tej wersji jest to tylko jeden pin) oraz deklarujemy rozmiar pamięci flash, w RTOS jest pomnożona przez 8, więc gdy w esp mamy 4 MB, to wpisujemy 32 (4×8). Ostatnim krokiem jest załączenie sdk (include $(SDK_PATH)/common.mk).

Plik *main.c* można podzielić na bloki realizujące określone zadania:

Listing 1. Zawartość pliku Makefile

```
PROGRAM = led_strip
EXTRA_COMPONENTS = \
    extras/http-parser \
    extras/dhcpserver \
    extras/pwm \
    $(abspath ../../components/esp-8266/wifi_config) \
    $(abspath ../../components/esp-8266/cJSON) \
    $(abspath ../../components/common/wolfssl) \
    $(abspath ../../components/common/homekit)
WHITE_PIN ?= 12
FLASH_SIZE ?= 32
EXTRA_CFLAGS += -I../.. -DHOMEKIT_SHORT_APPLE_UUIDS -DWHITE_PIN=$(WHITE_PIN)
include $(SDK_PATH)/common.mk
```

Listing 3. Łączenie z siecią Wi-Fi

```
void on_wifi_event(wifi_config_event_t event) {
    if (event == WIFI_CONFIG_CONNECTED) {
        printf("Connected to WiFi\n");
    } else if (event == WIFI_CONFIG_DISCONNECTED) {
        printf("Disconnected from WiFi\n");
    }
}
```

Listing 2. Import bibliotek i definicja pinów

```
#include <stdio.h>
#include <esp/uart.h>
#include <esp8266.h>
#include <FreeRTOS.h>
// bibilioteka umożliwiająca stworzenie taska
#include <task.h>
// główna biblioteka homekit
#include <homekit/homekit.h>
// biblioteka pozwalająca na korzystanie z charakterystyk dla urzadzeń homekit
#include <homekit/characteristics.h>
// biblioteka umożliwiająca konfiguracje sieci wi-fi
#include <wifi_config.h>
// biblioteka umożliwiajaca korzystanie z pwm na pinach
#include <pwm.h>

#ifndef WHITE_PIN
#error WHITE_PIN is not specified
#endif
```

Listing 4. Zmienna config

```
homekit_server_config_t config = {
    .accessories = accessories,
    .password = "248-12-524",
    .setupId="1A20"
};
```

Listing 5. Inicjalizacja pwm

```
void init_pwm(){
    uint8_t pins[1];
    pins[0] = WHITE_PIN;
    pwm_init(1, pins, false);
    pwm_set_freq(1000);
    pwm_start();
}
```

Listing 6. Metoda indentyfikacyjna urządzenia

```
void led_identify_task(void *_args) {
    pwm_set_duty(UINT16_MAX);
    vTaskDelay(1000 / portTICK_PERIOD_MS);
    pwm_set_duty(0);
    vTaskDelay(1000 / portTICK_PERIOD_MS);
    target = 100;
    led_write(led_on);
    vTaskDelete(NULL);
}

void led_identify(homekit_value_t _value) {
    xTaskCreate(led_identify_task, "LED identifying", 128, NULL, 2, NULL);
}
```

Listing 7. Gettery oraz Settery

```
homekit_value_t led_on_get() {
    return HOMEKIT_BOOL(led_on);
}

void led_on_set(homekit_value_t value) {
    led_on = value.bool_value;
    led_write(led_on);
}

homekit_value_t led_brightness_get() {
    return HOMEKIT_INT(led_brightness);
}

void led_brightness_set(homekit_value_t value) {
    target = value.int_value;
}
```

- Import bibliotek i definicja pinów (**listing 2**). Na samym początku dodajemy standardowe biblioteki, których będziemy używać, a następnie sprawdzamy, czy pin został zadeklarowany, jeśli nie, to pojawi się błąd podczas kompilacji.
- Łączenie z siecią Wi-Fi (**listing 3**). Ten blok odpowiada za tak zwany event. W momencie, gdy połączymy się z siecią Wi-Fi, w terminalu zostanie wyświetlony komunikat. Podobnie stanie się również podczas rozłączenia z siecią. Wykorzystałem to w celach informacyjnych podczas uruchamiania programu, by sprawdzić, czy cała płytka działa prawidłowo, więc esp było podpięte do komputera w trybie monitorowania. Poniższe „teksty" można zastąpić na przykład mignięciem diod przy połączeniu oraz podwójnym mignięciem na rozłączenie.
- Zmienna *config* (**listing 4**). Odpowiada za podstawowe informacje na temat serwera *homekit*. Pierwszy parametr to opis akcesoriów (w dalszej części artykułu zostanie opisany sposób budowy tego fragmentu), drugi to hasło dostępu, a trzeci to id urządzenia potrzebny do wygenerowania kodu QR.
- Inicjalizacja pwm (**listing 5**). Odpowiada za ustawienie pinu do obsługi PWM (o tym, jak stworzyć konfigurację akcesorium, będzie dalej, gdyż będziemy potrzebowali informacji, jak sterujemy taśmą LED). Najpierw tworzymy tablicę o rozmiarze *n* elementów. Liczba *n* określa liczbę pinów, które mają być sterowane pwm. Następnie wywoływana jest metoda *pwm_init* z biblioteki *pwm.h*. Pierwszy argument to nasze *n*, czyli liczba pinów, drugi to tablica z numerami pinów, trzeci to flaga *reverse*. Następnym krokiem jest ustawienie częstotliwości pwm. Standardowo jest to 1 kHz. Ostatni etap to wystartowanie PWM.
- Metoda indentyfikacyjna urządzenia (**listing 6**). Kolejnym punktem na liście jest przygotowanie metody odpowiadającej za identyfikację urządzenia. Identyfikacja jest wykonywana podczas dodawania urządzenia do aplikacji. Moją propozycją identyfikacji urządzenia jest zapalenie diod LED na sekundę, zgaszenie a następnie zapalenie ze stopniowym rozjaśnianiem. Oczywiście każdy może napisać swoją metodę. Gdy mamy metodę, musimy utworzyć z niej tzw. *task*, który będzie uruchomiony. W tym celu zapisujemy metodę *LED_identify*.
- Gettery oraz Settery (**listing 7**). Aplikacja musi mieć możliwość pobrania oraz ustawienia kluczowych danych dla sterownika. Jest to między innymi stan urządzenia (on/off), aktualny poziom jasności, włączenie/wyłączenie taśmy oraz ustawienie poziomu jasności. Oczywiście do metod typu *set* zostanie przesłana wartość przez aplikację.
- Sterowanie taśmą (**listing 8**). Jest to główna metoda sterownika, odpowiada za włączenie i wyłączenie taśmy. W mojej wersji podczas ustawiania poziomu jasności będzie to robione stopniowo ze skokiem 2% co 30 milisekund, dzięki czemu czas zapalenia całej taśmy to 1,5 sekundy. Oczywiście można

Listing 8. Sterowanie taśmą

```
void led_write(bool on) {
    // sprawdzamy, czy taśma led ma być włączona
    if(on){
        // Jeśli tak, to sprawdzamy, czy aktualny poziom jasności jest
        // mniejszy niż zadany, zmienna target
        if (led_brightness < target){
            // Jeśli tak to zwiększamy poziom jasności do zadanego
            while (led_brightness != target){
                led_brightness += 2;

                // Może się tak zdarzyć, że przeskoczymy poziom jasności o jeden na przykład,
                // gdy aktualny poziom to 21, a my zadany będzie 22
                // wtedy po wykonaniu się powyższej pętli otrzymamy wartość 23
                // dlatego poniżej porównywana jest wartość aktualna z zadaną,
                // jeśli się okaże, że doszło do przeskoku
                // wtedy ustawiana jest jasność dokładnie taka jak zadana
                // oraz kończy się wykonywanie metody
                if (led_brightness > target){
                    led_brightness = target;
                    pwm_set_duty (UINT16_MAX * (led_brightness/100.00));
                    break;
                }

                pwm_set_duty(UINT16_MAX * (led_brightness/100.00));
                vTaskDelay(30 / portTICK_PERIOD_MS);
            }
        }
        // taśma ma być włączona oraz aktualny poziom jasności jest większy
        // niż zadany, więc należy ściemnić taśmę.
        else{
            while (led_brightness != target){
                led_brightness -= 2;
                // Tak jak to miało miejsce podczas rozjaśniania, może się zdarzyć, że
                // Przeskoczymy wartość o 1, dlatego zawsze jest sprawdzany warunek
                // czy aktualna wartość jest mniejsza niż zadana,
                // jeśli tak, to ustawiamy aktualną wartość na zadaną
                // i kończymy wykonywanie pętli.
                if (led_brightness < target){
                    led_brightness = target;
                    pwm_set_duty (UINT16_MAX * (led_brightness/100.00));
                    break;
                }

                pwm_set_duty(UINT16_MAX * (led_brightness/100.00));
                vTaskDelay(30 / portTICK_PERIOD_MS);

            }
        }
    // taśma ma być wyłączona, więc dopóki aktualna wartość nie jest równa
    //zero, to taśma jest wygaszana.
    }else {
        while (led_brightness != 0){
            pwm_set_duty(UINT16_MAX * (led_brightness/100.00));
            led_brightness -=  2;
            // tak jak wyżej może się zdarzyć, że będzie przeskok o jeden
            if (led_brightness < 0){
                led_brightness = 0;
                pwm_set_duty(0);
                break;
            }
            pwm_set_duty(UINT16_MAX * (led_brightness/100.00));
            vTaskDelay(30 / portTICK_PERIOD_MS);
        }
    }
}
```

Listing 9. Reprezentacja akcesorium

```
homekit_accessory_t *accessories[] = {
    HOMEKIT_ACCESSORY(.id=1, .category=homekit_accessory_category_lightbulb, .services=(homekit_service_t*[]){
        HOMEKIT_SERVICE(ACCESSORY_INFORMATION, .characteristics=(homekit_characteristic_t*[]){
            HOMEKIT_CHARACTERISTIC(NAME, "Light strip"),
            HOMEKIT_CHARACTERISTIC(MANUFACTURER, "Michael's Software"),
            HOMEKIT_CHARACTERISTIC(SERIAL_NUMBER, "0x10"),
            HOMEKIT_CHARACTERISTIC(MODEL, "Make It Light"),
            HOMEKIT_CHARACTERISTIC(FIRMWARE_REVISION, "1.0"),
            HOMEKIT_CHARACTERISTIC(IDENTIFY, led_identify),
            NULL
        }),
       HOMEKIT_SERVICE(LIGHTBULB, .primary = true, .characteristics = (homekit_characteristic_t*[]) {
            HOMEKIT_CHARACTERISTIC(NAME, "Light strip"),
            HOMEKIT_CHARACTERISTIC(
                ON, true,
                .getter = led_on_get,
                .setter = led_on_set
            ),
            HOMEKIT_CHARACTERISTIC(
                BRIGHTNESS, 100,
                .getter = led_brightness_get,
                .setter = led_brightness_set
            ),
            NULL
        }),
        NULL
    }),
    NULL
};

void user_init(void) {
    uart_set_baud(0, 115200);
    wifi_config_init2("Apple Home 0x10", NULL, on_wifi_event);
    homekit_server_init(&config);
}
```

Listing 10. Modyfikacje pliku Makefile dla wersji RGB

```
RED_PIN ?= 14
GREEN_PIN ?= 5
BLUE_PIN ?= 13
FLASH_SIZE ?= 32

EXTRA_CFLAGS += -I../.. -DHOMEKIT_SHORT_APPLE_UUIDS -DRED_PIN=$(RED_PIN) -DGREEN_PIN=$(GREEN_PIN) -DBLUE_PIN=$(BLUE_PIN)
```

samemu dobrać wartość skoku oraz interwał czasowy. Z moich testów wynika, że to najlepsze wartości, gdyż czas pełnego zapalenia nie jest zbyt długi oraz efekt jest płynny. W analogiczny sposób przebiega gaszenie taśmy LED. Metodę można podzielić na kilka fragmentów:

- taśma LED ma być włączona,
  - zadany poziom jasności jest większy niż aktualny,
  - zadany poziom jasności jest mniejszy niż aktualny,
- taśma LED ma być wyłączana.

Listing 11. Sprawdzenie deklaracji pinów oraz dodanie niezbędnych zmiennych

```
#ifndef RED_PIN
#error RED_PIN is not specified
#endif
#ifndef GREEN_PIN
#error GREEN_PIN is not specified
#endif
#ifndef BLUE_PIN
#error BLUE_PIN is not specified
#endif

typedef union {
    struct {
        uint16_t blue;
        uint16_t green;
        uint16_t red;
    };
} rgb_color_t;

#define LED_RGB_SCALE 255

#define LFS 4

rgb_color_t current_color = {{ 0, 0, 0
}} ;
rgb_color_t target_color = {{ 0, 0, 0 }} ;

float led_hue = 0;
float led_saturation = 59;
float led_brightness = 100;
bool led_on = false;
```

Reprezentacja akcesorium (**listing 9**). Akcesorium reprezentuje tablica, gdyż jeden sterownik może pełnić funkcję dwóch urządzeń. Przykładem jest włącznik światła z podziałem na kilka punktów świetlnych. Podstawowe informacje o urządzeniu to:

- nazwa,
- producent,
- numer seryjny,
- model,
- wersja oprogramowania,
- metoda identyfikacji urządzenia.

Musimy również wskazać, jakiego typu jest to urządzenie, w tym przypadku jest to *lightbulb*.

Definiowane jest także to, czy źródło światła ma być włączone, czy nie. Ja ustawiłem, że tak i ma świecić z maksymalną jasnością. Dodatkowo wskazujemy nazwy funkcji odpowiedzialnych za sterowanie taśmą. Na koniec została metoda, która jest uruchamiana podczas podłączenia sterownika do zasilania *user_init*.

Sterowanie taśmą RGB

W oprogramowaniu do sterowania taśmą RGB dodatkowo pojawi się metoda odpowiedzialna za konwersję barwy oświetlenia z hsi na RGB. Najmniejsza zmiana jest w pliku *Makefile*, polega na dodaniu dodatkowych pinów oraz sprawdzeniu, czy piny rzeczywiście zostały dodane (**listing 10**).

Pierwsze miany w pliku programu to kod sprawdzający, czy wymagane piny rzeczywiście zostały dodane oraz deklaracja kilku zmiennych, których potem będziemy używać, m.in. skala RGB i przesunięcie bitowe (**listing 11**). Musimy też przerobić metodę inicjalizującą urządzenie. W tej wersji taśma zapali się na czerwono na sekundę, po czym zgaśnie (**listing 12**).

Czas na najważniejszy fragment oprogramowania. Z aplikacji dostajemy kod koloru w formacie HSI i należy go przeformatować do RGB. HSI jest to przestrzeń barw, w której każdy kolor opisywany jest przez trzy wartości:

H (*Hue*) – barwa/odcień,
S (*Saturation*) – nasycenie,
I (*Intensity*) – intensywność.

Barwa przyjmuje wartości od 0 do 360°, gdzie 0° to kolor czerwony, 120° zielony, 240° niebieski. Nasycenie reprezentuje stopień zanieczyszczenia barwy przez kolor biały, przyjmuje ono wartość między 0 a 1. Intensywność mieści się w zakresie 0 a 1, gdzie 0 to czerń, a 1 to biel. Szczegółowy sposób

Listing 12. Zmieniona metoda inicjalizująca

```
void led_identify_task(void *_args) {
    printf("LED identify\n");

    rgb_color_t color = target_color;
    rgb_color_t black_color = {0, 0, 0};
    rgb_color_t red_color = (255, 0, 0)};

    target_color = red_color;
    vTaskDelay(1000 / portTICK_PERIOD_MS);
    target_color = black_color;
    vTaskDelay(1000 / portTICK_PERIOD_MS);
    target_color = color;
    vTaskDelete(NULL);
}
```

Listing 13. Metoda odpowiedzialna za zamianę formatu koloru

```
static void hsi2rgb(float h, float s, float i, rgb_color_t* rgb) {
    int r, g, b;

    while (h < 0) { h += 360.0F; }; // cycle h around to 0-360 degrees
    while (h >= 360) { h -= 360.0F; };
    h = 3.14159F*h / 180.0F;    // convert to radians.
    s /= 100.0F;    // from percentage to ratio
    i /= 100.0F;    // from percentage to ratio
    s = s > 0 ? (s < 1 ? s : 1) : 0;    // clamp s and i to interval [0,1]
    i = i > 0 ? (i < 1 ? i : 1) : 0;    // clamp s and i to interval [0,1]
    //i = i * sqrt(i);  // shape intensity to have finer granularity near 0

    if (h < 2.09439) {
        r = LED_RGB_SCALE * i / 3 * (1 + s * cos(h) / cos(1.047196667 - h));
        g = LED_RGB_SCALE * i / 3 * (1 + s * (1 - cos(h) / cos(1.047196667 - h)));
        b = LED_RGB_SCALE * i / 3 * (1 - s);
    }
    else if (h < 4.188787) {
        h = h - 2.09439;
        g = LED_RGB_SCALE * i / 3 * (1 + s * cos(h) / cos(1.047196667 - h));
        b = LED_RGB_SCALE * i / 3 * (1 + s * (1 - cos(h) / cos(1.047196667 - h)));
        r = LED_RGB_SCALE * i / 3 * (1 - s);
    }
    else {
        h = h - 4.188787;
        b = LED_RGB_SCALE * i / 3 * (1 + s * cos(h) / cos(1.047196667 - h));
        r = LED_RGB_SCALE * i / 3 * (1 + s * (1 - cos(h) / cos(1.047196667 - h)));
        g = LED_RGB_SCALE * i / 3 * (1 - s);
    }

    rgb->red = (uint8_t) r;
    rgb->green = (uint8_t) g;
    rgb->blue = (uint8_t) b;
}
```

czystej białej barwy, gdyż będzie widoczne rozbicie na trzy poszczególne kolory. Względem wersji oprogramowania dla taśmy RGB zmianie ulega:

- liczba zadeklarowanych pinów,
- struktura opisująca barwę,
- zmienne *current color* oraz *target color*,
- konwersja z HSI na RGB,
- główna metoda sterująca.

Warto zwrócić uwagę na fakt, że jeśli wartości barw czerwonej, zielonej i niebieskiej są takie same, to wtedy uzyskujemy barwę białą, której jasność jest proporcjonalna do wartości barw RGB. W takim przypadku dioda biała powinna świecić, diody RGB powinny być wygaszone.

Pełne źródła programów znajdują się w materiałach dodatkowych, dlatego nie będziemy szczegółowo opisywać tej wersji. Cały kod znajduje się także na moim githubie (***https://bit.ly/3ibbdLw***). Wersja kodu dla sterownika RGB oraz RGB+W opiera

Listing 14. Modyfikacje w opisie akcesorium

```
homekit_accessory_t *accessories[] = {
    HOMEKIT_ACCESSORY(.id = 1, .category = homekit_accessory_category_lightbulb, .services = (homekit_service_t*[]) {
        HOMEKIT_SERVICE(ACCESSORY_INFORMATION, .characteristics=(homekit_characteristic_t*[]){
            HOMEKIT_CHARACTERISTIC(NAME, "Light strip"),
            HOMEKIT_CHARACTERISTIC(MANUFACTURER, "Michael's Software"),
            HOMEKIT_CHARACTERISTIC(SERIAL_NUMBER, "0x10"),
            HOMEKIT_CHARACTERISTIC(MODEL, "Make It Light"),
            HOMEKIT_CHARACTERISTIC(FIRMWARE_REVISION, "1.0"),
            HOMEKIT_CHARACTERISTIC(IDENTIFY, led_identify),
            NULL
        }),
        HOMEKIT_SERVICE(LIGHTBULB, .primary = true, .characteristics = (homekit_characteristic_t*[]) {
            HOMEKIT_CHARACTERISTIC(NAME, "LED Strip"),
            HOMEKIT_CHARACTERISTIC(
                ON, true,
                .getter = led_on_get,
                .setter = led_on_set
            ),
            HOMEKIT_CHARACTERISTIC(
                BRIGHTNESS, 100,
                .getter = led_brightness_get,
                .setter = led_brightness_set
            ),
            HOMEKIT_CHARACTERISTIC(
                HUE, 0,
                .getter = led_hue_get,
                .setter = led_hue_set
            ),
            HOMEKIT_CHARACTERISTIC(
                SATURATION, 0,
                .getter = led_saturation_get,
                .setter = led_saturation_set
            ),
            NULL
        }),
        NULL
    }),
    NULL
};
```

zamiany został opisany na blogu saikoLED (***https://bit.ly/2Vrp7iO***). Metoda odpowiedzialna za zamianę znajduje się na **listingu 13**.

Następną niedużą zmianą jest opis akcesorium, należy tutaj uwzględnić, że taśma ma trzy kolory, a nie jeden, więc sterujemy: odcieniem jasnością i nasyceniem (**listing 14**). W opisie akcesorium warto zwrócić uwagę na gettery i setery, wyglądają teraz tak jak na **listingu 15**.

Metoda sterująca taśmą, z racji tego, że sterowanie ma odbywać się każdym kanałem osobno, wymaga użycia biblioteki multipwm. Z tej metody utworzymy *task*, który będzie wykonywany cały czas. Dodatkowo, metoda ta musi zostać umieszczona w pamięci RAM naszego kontrolera (**listing 16**).

Na koniec została główna metoda, która uruchamiana jest w momencie podłączenia sterownika do zasilania user_init. Wymaga dodania jednej linii:

```
xTaskCreate(multipwm_task,
"multipwm", 256, NULL, 2,
NULL);
```

Sterowanie taśmami RGB+W

Taśmy RGB+W mają dodatkową białą diodę pomiędzy każdą parą diod RGB. Z diody RGB jest możliwe uzyskanie

Listing 15. Modyfikacja getterów i setterów

```
homekit_value_t led_on_get() {
    return HOMEKIT_BOOL(led_on);
}

void led_on_set(homekit_value_t value) {
    led_on = value.bool_value;
}

homekit_value_t led_brightness_get() {
    return HOMEKIT_INT(led_brightness);
}

void led_brightness_set(homekit_value_t value) {
    led_brightness = value.int_value;
}

homekit_value_t led_hue_get() {
    return HOMEKIT_FLOAT(led_hue);
}

void led_hue_set(homekit_value_t value) {
    led_hue = value.float_value;
}

homekit_value_t led_saturation_get() {
    return HOMEKIT_FLOAT(led_saturation);
}

void led_saturation_set(homekit_value_t value) {
    led_saturation = value.float_value;
}
```

Rysunek 5. Kod QR

```
Listing 16. Metoda sterująca taśmą RGB

IRAM void multipwm_task(void *pvParameters) {
    uint8_t pins[] = {RED_PIN, GREEN_PIN, BLUE_PIN}; // tablica pinów

    pwm_info_t pwm_info;
    pwm_info.channels = 3; // liczba kanałów

    multipwm_init(&pwm_info);
    multipwm_set_freq(&pwm_info, 1000); // ustawienie czestotliwości
    for (uint8_t i=0; i<pwm_info.channels; i++) {
        multipwm_set_pin(&pwm_info, i, pins[i]); // ustawienie pinów
    }

    while(1) { // ta pętla będzie wykonywać się w kółko
       if (led_on) {
           // jeśli taśma ma być włączona, to konwertujemy hsi
           // który przyszedł w zapytaniu na kod rgb
           hsi2rgb(led_hue, led_saturation, led_brightness, &target_color);
           // jeśli taśma ma być wyłączona, to wszystkie barwy są wygaszone
       } else {
           target_color.red = 0;
           target_color.green = 0;
           target_color.blue = 0;
       }

       // sterowanie rozjaśnianiem odbywa się tutaj w inny sposób, niż przy taśmach
       // jednobarwnych do aktualnego koloru jest dodawany tylko kawałek różnicy między
       // zadanym a aktualnym poprzez przesunięcie bitowe
       current_color.red += ((target_color.red * 256) - current_color.red) >> LFS ;
       current_color.green += ((target_color.green * 256) - current_color.green) >> LFS ;
       current_color.blue += ((target_color.blue * 256) - current_color.blue) >> LFS ;

       // gdy mamy już wartości, to ustawiamy je na konkretnych kanałach
       multipwm_stop(&pwm_info);
       multipwm_set_duty(&pwm_info, 0, current_color.red);
       multipwm_set_duty(&pwm_info, 1, current_color.green);
       multipwm_set_duty(&pwm_info, 2, current_color.blue);
       multipwm_start(&pwm_info);
       vTaskDelay(1);
    }
}
```

się na projekcie magic home dostępnym na githubie użytkownika maximkulkin (*https://bit.ly/3g6OkH9*).

Generowanie kodu QR

Z poziomu konsoli musimy uruchomić program gen_qrcode, jako parametry podajemy id kategorii urządzenia, kod potrzebny do sparowania sterownika z aplikacją, setupId oraz ścieżkę i nazwę obrazka z kodem qr. Dla opisanego sterownika komenda będzie wyglądała następująco:

```
tools/gen_qrcode 5 248-12-524 1A20 qrcode.png
```

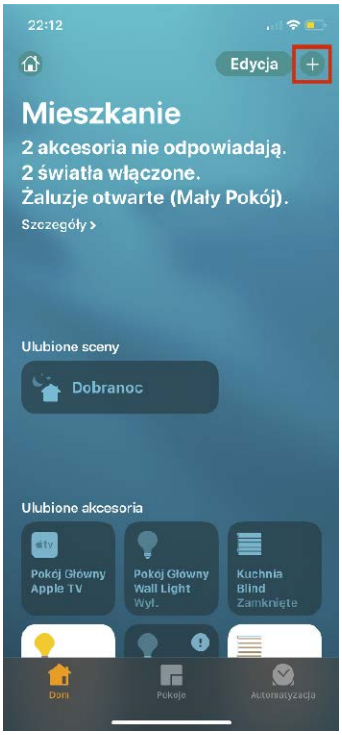
Kod QR dla urządzenia z tego artykułu pokazuje **rysunek 5**.

Uruchomienie

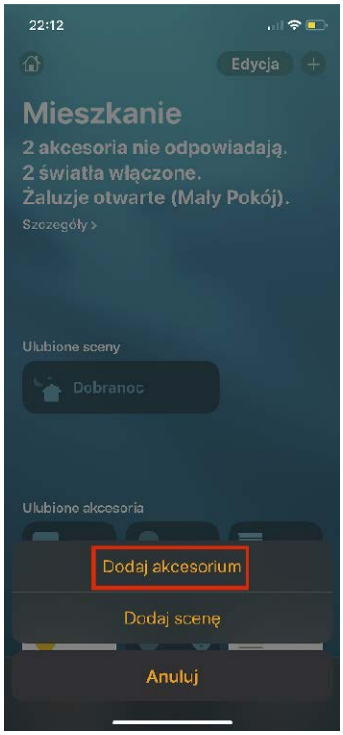
Gdy włączymy pierwszy raz sterownik, nie ma on zapisanej żadnej sieci. Należy otworzyć ustawienia sieci Wi-Fi na urządzeniu mobilnym lub komputerze i przyłączyć się do sieci Apple Home 0x10. Gdy już to zrobimy i sterownik pomyślnie połączy się z naszym routerem, to następnym krokiem jest uruchomienie aplikacji Home i kliknięcie „+” (**rysunek 6**). Następnie wybieramy „Dodaj akcesorium” (**rysunek 7**) i skanujemy kod QR. Sterownik zostanie sparowany z aplikacją i będzie można go użytkować.

Interfejs do sterowania jedną barwą jest bardzo prosty, pierwsze kliknięcie to włączenie, drugie to zgaszenie taśmy, a po przytrzymaniu możemy sterować jasnością za pomocą suwaka. W przypadku RGB/RGB+W interface jest bardziej rozbudowany. Możemy sterować kolorem, temperaturą oraz jasnością taśmy (**rysunek 8**, **rysunek 9**).

inż. Michał Urban
dev.michael.urban@gmail.com



Rysunek 6. Uruchomienie aplikacji Home



Rysunek 7. Dodawanie akcesorium

Rysunek 8. Interfejs wyboru barwy

Rysunek 9. Interfejs ustawiania jasności