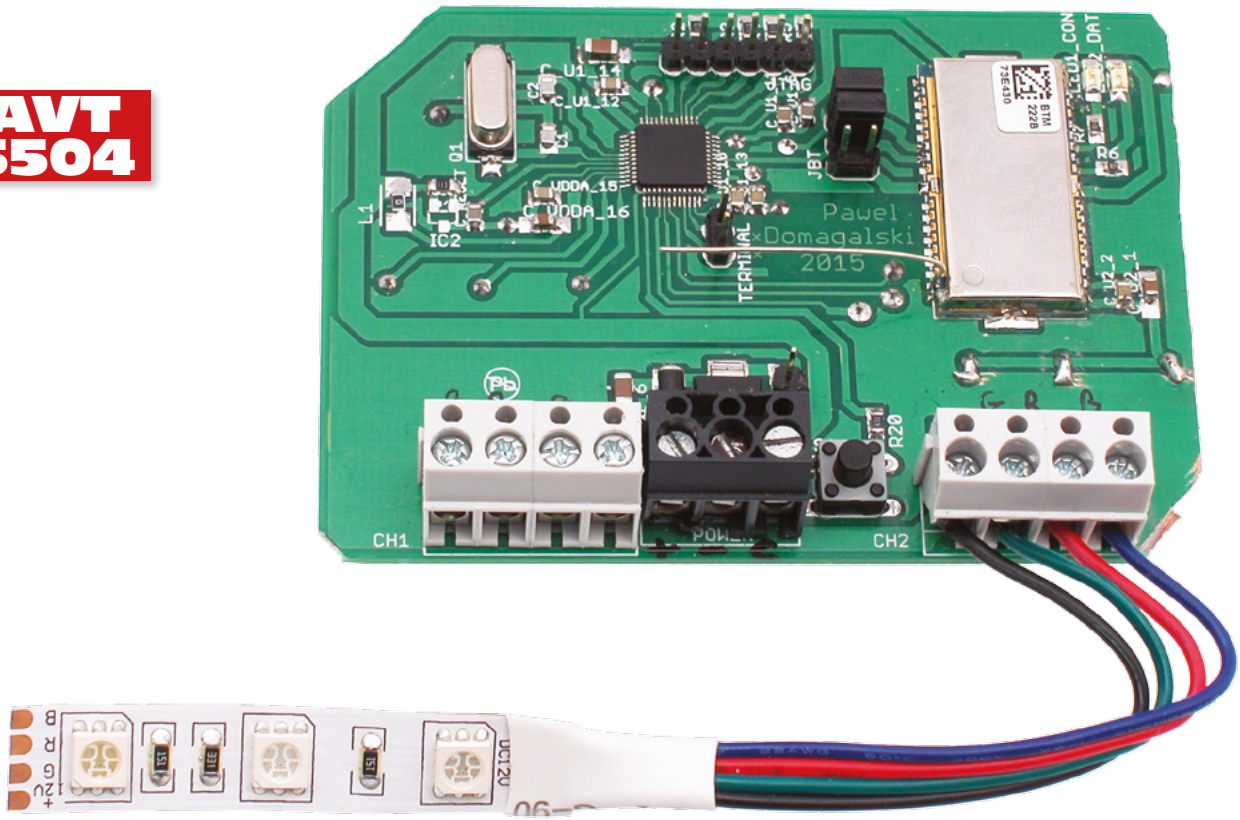
AVT 5504



Kontroler oświetlenia RGB z Bluetooth

Urządzenie, którego cały interfejs użytkownika zaimplementowano w komfortowej w użyciu aplikacji dla systemu Android. Komunikacja odbywa się przez Bluetooth, moduł sterujący oparto na mikrokontrolerze ARM Cortex M3. Do uruchomienia aplikacji można zastosować nieużywany smartfon wbudowując go w obudowę zamocowaną w łatwo dostępnym miejscu.

Rekomendacje: *kontroler przyda się do aranżowania oświetlenia wnętrza i tworzenia odpowiedniego nastroju. Dzięki łatwej kontroli koloru świecenia może przydać się w gabinecie medycyny alternatywnej.*

Oświetlenie LED zyskuje coraz większą popularność. Obniżenie cen taśm LED sprawiło rosnące zainteresowanie, są one szeroko stosowane nie tylko w drogich apartamentach czy inteligentnych domach. Rosnące konstrukcjami tego typu podyktowane jest przede wszystkim możliwością uzyskania atrakcyjnych wizualnie efektów rozproszenia światła i uzyskania efektownego podświetlenia elementów mebli lub ścian na całej długości. Oczywiście, duże znaczenie ma przy tym aspekt ekonomiczny, ponieważ – jak dobrze wiemy – diody LED wymagają do zasilania znacznie mniej energii niż tradycyjne żarówki.

Już jakiś czas temu myślałem o wykonaniu systemu oświetlenia opartego na taśmach z diodami RGB i dlatego postanowiłem zaprojektować sterownik trójkolorowych diod LED z interfejsem użytkownika w postaci aplikacji uruchomionej na smartphonie z systemem Android. Takie rozwiązanie bardzo upraszcza część sprzętową, która składa się wyłącznie z trzech bloków:

1. Modułu Bluetooth zapewniającego komunikację z aplikacją Android.
2. Modułu sterującego opartego na mikrokontrolerze z rdzeniem ARM Cortex-M3 firmy ST.

W ofercie AVT*	
AVT-5504 A	AVT-5504 B
AVT-5504 C	AVT-5504 UK

Podstawowe informacje:

- Kontrola koloru i natężenia światła za pomocą interfejsu Bluetooth i aplikacji dla systemu Android.
- Możliwość pracy automatycznej (płynna zmiana koloru) lub sterowania ręcznego.
- Paleta 16,7 miliona kolorów.
- Mikrokontroler STM32F103C8T6, moduł BTM-222.
- Dwa niezależne kanały sterujące pracą diod/taśm RGB.
- Maksymalny prąd obciążenia ok. 10 A (większy po zastosowaniu radiatora).
- Napięcie zasilające: 12 V DC (wyższe po zmianie stabilizatora).

Dodatkowe materiały na FTP:

ftp://ep.com.pl, user: 07641, pass: yus9jv2r

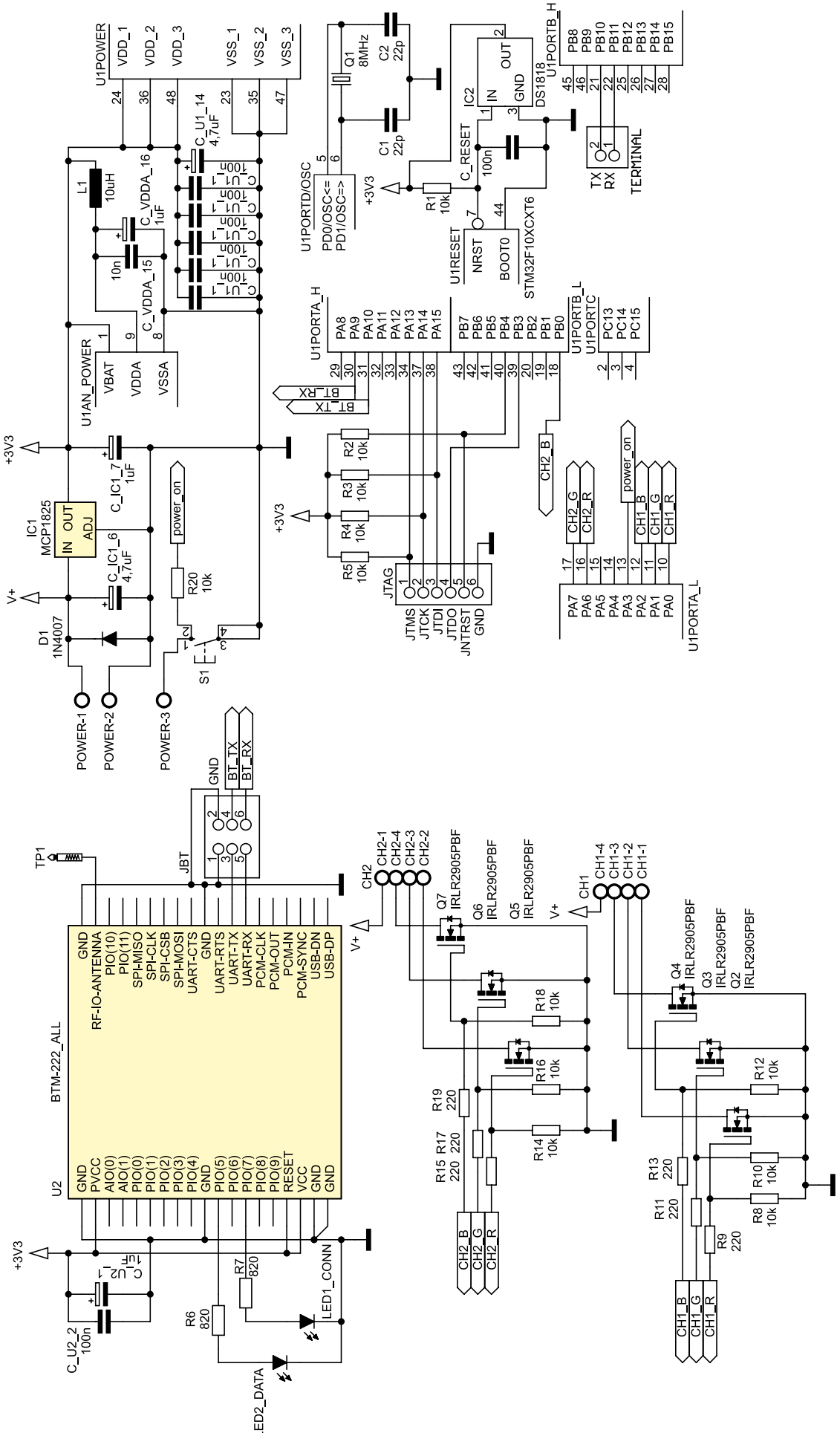
- wzory płytek PCB

Projekty pokrewne na FTP:

(wymienione artykuły są w całości dostępne na FTP)

AVT-1847	Miniaturowy sterownik taśmy LED (EP 2/2015)
AVT-5487	PWMLEDz: 10-kanałowy sterownik taśm LED z interfejsem Modbus lub SPPoB (EP 1/2015)
AVT-1800	LED Dimmer – regulator oświetlenia LED (EP 5/2014)

* Uwaga:
Zestawy AVT mogą występować w następujących wersjach:
AVT xxxx UK to zaprogramowany układ. Tylko i wyłącznie. Bez elementów dodatkowych.
AVT xxxx A płytka drukowana PCB (lub płytki drukowane, jeśli w opisie wyraźnie zaznaczono), bez elementów dodatkowych.
AVT xxxx A+ płytka drukowana i zaprogramowany układ (czyli połączenie wersji A i wersji UK) bez elementów dodatkowych.
AVT xxxx B płytka drukowana (lub płytki) oraz komplet elementów wymienionych w załączniku pdf
AVT xxxx C to nic innego jak zmontowany zestaw B, czyli elementy wlutowane w PCB. Należy mieć na uwadze, że o ile nie zaznaczono wyraźnie w opisie, zestaw ten nie ma obudowy ani elementów dodatkowych, które nie zostały wymienione w załączniku pdf
AVT xxxx CD oprogramowanie (nieczęsto spotykana wersja, lecz jeśli występuje, to niezbędne oprogramowanie można ściągnąć, klikając w link umieszczony w opisie kitu)
Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! (UK, A, A+, B lub C). *http://sklep.avt.pl*



Rysunek 1. Schemat ideowy kontrolera LED RGB z Bluetooth

3. Modułu wykonawczego z 6 tranzystorami MOSFET-N sterowanych sygnałami PWM generowanymi przez mikrokontroler.

Schemat ideowy urządzenia pokazano na **rysunku 1**.

Trójkolorowe diody RGB świecą w dowolnie wybranym kolorze. Kodowanie intensywności świecenia każdego koloru z rozdzielczością 8-bitową umożliwia uzyskanie ponad 16,7 miliona kombinacji. Daje to możliwość precyzyjnego ustawienia barwy. Jest to niewątpliwie ogromna przewaga nad typowymi diodami jednokolorowymi. Ponadto, w zaproponowanym przeze mnie rozwiązaniu zdecydowałem się na wykonanie dwóch niezależnych kanałów, co umożliwia zasilanie dwóch diod/taśm RGB.

Wykaz elementów

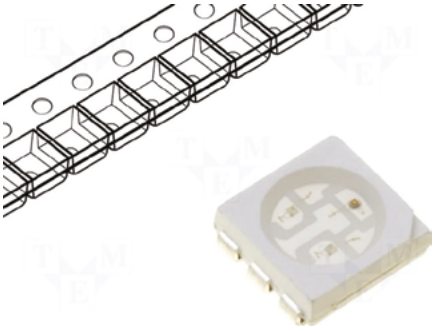
Rezystory: (SMD 0805)
R1…R5, R8, R10, R12, R14, R16, R18, R20: 10 kΩ.
R6, R7: 820 Ω
R9, R11, R13, R15, R17, R19: 220 Ω
Kondensatory:
C1: 22 pF (SMD 0805)
C_IC1_6, C_U1_14: 4,7 µF (SMD „A")
C_IC1_7, C_U2_1, C_VDDA_16: 1 µF (SMD „A")
C_RESET, C_U1_5, C_U1_10, C_U1_11, C_U1_12, C_U1_13, C_U2_2: 100 nF (SMD 0805)
C_VDDA_15: 10 nF (SMD 0805)
Półprzewodniki:
D1: 1N4007
LED1_CONN, LED2_DATA: dioda LED (SMD 1206)
IC1: MCP1755S-3302E/DB 3,3V
IC2: DS1818R (SOT23, opis w tekście)
Q2…Q7: IRLR2905PBF (D-PAK)
U1: STM32F103C8T6
Inne:
U2: BTM-222
Q1: 8 MHz
JBT: listwa goldpin 2×3
JTAG: listwa goldpin 1×6
TERMINAL: listwa goldpin 1×2
CH1, CH2: złącze śrubowe 1×4, raster 5 mm
POWER: złącze śrubowe 1×3, raster 5 mm
S1: microswitch
L1: dławik 10 µH (opis w tekście)

Przy wybieraniu taśm RGB należy zwrócić uwagę na obudowy diod LED oraz ich gęstość ich rozmieszczenia na taśmie. Zwykle taśmy są wykonane z diod w obudowie 3528 lub 5050 (**rysunek 2**). Podstawowa różnica pomiędzy nimi polega na tym, że dioda 5050 jest zbudowana z trzech odseparowanych od siebie struktur diod LED zintegrowanych wewnątrz jednej obudowy. Ponadto, ta obudowa jest prawie dwa razy większa, co ułatwia odprowadzanie ciepła. Dzięki wymienionym właściwościom, taśma zbudowana z tych diod może świecić jaśniej i jest przy tym trwalsza.

Typowo, w handlu są oferowane taśmy mające 30 lub 60 diod na metr. Zdecydowałem się na wersję z 60 diodami, mimo iż obawiałem się problemów z odprowadzaniem ciepła. Przykleiłem je do metalowej ramy łóżka, dzięki czemu przy prądzie 3 A diody zbytnio się nie nagrzewają. Myślę, że przy braku metalowych profili i maksymalnym prądzie zasilania wynoszącym 6 A diody mogłyby z łatwością przekroczyć temperaturę 60°C.

Android SDK

Jako środowisko programistyczne wybrałem Eclipse w wersji Kepler z wtyczką ADT (Android Development Tools). Możliwa jest również praca przy wykorzystaniu Android Studio, czyli z nowszym środowiskiem stworzonym przez Google do rozwijania aplikacji dla systemu Android.



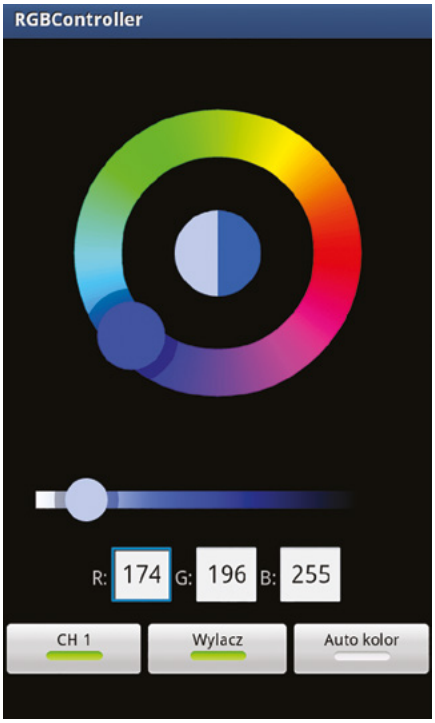
Rysunek 2. Dioda LED w obudowie 5050

Zdecydowałem się pozostać przy Eclipse z dwóch powodów: poprzedni projekt również realizowałem z wykorzystaniem Eclipse (przyzwyczajenie) oraz tego, że aplikację dla mikrokontrolera również tworzyłem w tym środowisku.

Aplikacja Android

Aplikację wykonano dla minimalnego „API level" równego 7, co odpowiada wersji 2.1. Podyktowane jest to tym, że dopiero od wersji 2.1 została oficjalnie wprowadzona do API obsługa Bluetooth. Z tego powodu dostęp oraz administrowanie urządzeniem Bluetooth to jedyne z dodatkowych praw dostępu, których wymaga aplikacja i które zostały zdefiniowane w pliku *manifest.xml*.

Wygląd aplikacji pokazano na **rysunku 3**. W górnej części umieszczono okrąg zawierający kolory z palety RGB. W celu wybrania koloru należy dotknąć wskaźnik i po prostu przesunąć go na pożądaną barwę. Dodatkowo możemy zmodyfikować jej odcień za pomocą leżącego poniżej paska regulującego nasycenie koloru oraz jego jasność (Saturation/Value Bar). Przesuwając go w lewo rozjaśniamy wybrany kolor aż do całkowitej bieli 0xFFFFFF,



Rysunek 3. Wygląd aplikacji dla systemu Android

natomiast przesuwając w prawo przyciemniamy aż do czerni (0x000000). Nastawy są natychmiast przesyłane do mikrokontrolera, dzięki czemu zmiana koloru świecenia diod RGB jest wykonywana w czasie rzeczywistym.

Zmiana koloru jest możliwa po „wciśnięciu" przycisku *Włącz/Wyłącz*. Przycisk w pozycji *Włącz* powoduje połączenie ze zdalnym modułem Bluetooth (nazwa modułu aktualnie połączonego wyświetlana jest na górnej belce), natomiast w pozycji *Wyłącz* wysyła rozkaz wyłączenia diod i rozłącza połączenie.

Mając ustawiony kolor dla kanału pierwszego, możemy przełączyć się na ustawianie koloru dla kanału drugiego za pomocą przycisku *CH1* znajdującego się na dole, po lewej stronie przycisku *Włącz/Wyłącz*.

Zmiana koloru jest również pokazana w środkowej części palety barw w formie koła. Lewa połowa wyświetla kolor ustawiony dla kanału pierwszego, natomiast prawa dla kanału drugiego. Będąc w trybie edycji kanału drugiego możemy skopiować kolor z kanału pierwszego poprzez kliknięcie na środek okręgu wyboru koloru. Kopiowanie w drugą stronę jest niemożliwe.

Przy zmianach koloru aktualizowane są również wartości składowych RGB wyświetlanych w trzech edytowalnych polach tekstowych. Oczywiście możliwa jest ich edycja skutkująca przeliczeniem koloru na palecie barw.

Pod ostatnim przyciskiem na dole dostępna jest funkcjonalność automatycznej zmiany koloru. Po jej uruchomieniu do mikrokontrolera zostanie przesłane polecenie, po którym zostanie uruchomiona nieskończona pętla płynnej zmiany koloru, w której następuje powolne i płynne wyświetlenie całej palety barw.

Wykorzystane projekty przykładowe

Udało mi się zaoszczędzić sporo czasu poświęconego na implementację palety barw dzięki wykorzystaniu części projektu **HoloColorPicker** (*https://github.com/LarsWerkman/HoloColorPicker*). Projekt jest udostępniony na warunkach licencji Apache, która pozwala na kopiowanie i modyfikowanie kodu źródłowego nawet w celach komercyjnych bez obowiązku upubliczniania. Wprowadzone przeze mnie modyfikacje dotyczyły między innymi:

- dodałem obsługę drugiego kanału,
- usunąłem zbędne paski nasycenia, wartości i przezroczystości.

Domyślny layout zmodyfikowałem do swoich potrzeb. Plik *dimens.xml* zawiera wszystkie parametry, jak na przykład długości promieni okręgów dla poszczególnych elementów przez co personalizacja aplikacji jest łatwa i wygodna.

Część odpowiadającą za komunikację z modułem Bluetooth skopiowałem ze swojego starego projektu. Wprowadziłem kilka niezbędnych poprawek i dodałem automatyzację procesu łączenia się z modułem (przy pierwszej próbie wysłania danych do modułu połączenie jest nawiązywane automatycznie). Połączenie jest sprawdzane i w razie potrzeby nawiązywane przy powrocie aplikacji z trybu uśpienia (funkcja *onStart*). Domyślnie szukaną nazwą zdalnego modułu Bluetooth jest „RGBController". Jeśli urządzenie o takiej nazwie znajduje się w zasięgu i dodatkowo jest na liście sparowanych urządzeń, to połączenie zostanie nawiązane automatycznie, bez wyświetlenia komunikatu.

Ramka danych przesyłana pomiędzy aplikacją a mikroprocesorem składa się łącznie z 9 bajtów (**rysunek 4**). Bajt startu i stopu służy jedynie do synchronizacji ramki. Nastawa koloru jest transmitowana w postaci trzech bajtów w kolejności czerwony, zielony i niebieski dla pierwszego kanału a następnie kolejne trzy dla drugiego kanału. Dodatkowo przesyłany jest jeszcze jeden bajt zawierający bity konfiguracyjne.

ARM toolchain

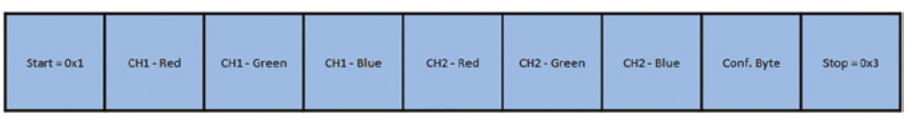
Do wykonania oprogramowania dla mikrokontrolera użyłem Eclipse z zainstalowaną wtyczką umożliwiającą debugowanie poprzez GDB. Obecnie jest dostępnych mnóstwo darmowych i komercyjnych środowisk programistycznych dla mikroprocesorów ARM, które bardzo ułatwiają pracę programiście dzięki możliwości przyśpieszenia projektu za pomocą różnego rodzaju kreatorów. Moim zdaniem jest to złudna oszczędność czasu, ponieważ gdy program przestanie być rozwijany, to będziemy zmuszeni do szukania nowego środowiska i nauki jego obsługi. To samo dotyczy przypadku darmowych wersji płatnych programów – mogą w pewnym momencie przestać być darmowe (jak np. CodeSourcery), a jeśli nawet dalej takie będą, to najprawdopodobniej licencja uniemożliwi nam wykorzystanie ich w celach komercyjnych. Wtedy pozostanie nam albo „przeportować" projekt i zmienić środowisko, albo zapłacić za licencję, która w wypadku profesjonalnych programów wcale nie jest tania. Dlatego mój projekt oparłem na pliku Makefile, przez co jest całkowicie niezależny od środowiska programistycznego i może być nawet rozwijany na komputerze pracującym pod Linuksem, bez środowiska graficznego (masochiści mogą nawet używać tylko edytora *vi*). Ponadto, szybciej i więcej uczymy się widząc, w jaki sposób są zależne od siebie poszczególne elementy projektu.

Do kompilowania używam darmowego pakietu **Linaro** wykorzystującego *gcc*. Za ten kompilator odpowiadają firmy: ARM, Freescale, IBM, Samsung, ST-Ericsson oraz Texas Instruments. W 2010 r. te firmy założyły organizację non-profit, której celem jest wspieranie rozwoju oprogramowania *open source* dla systemów wbudowanych. Już sama obecność firmy ARM w gronie założycieli przekonuje mnie, że jest to najlepsze dostępne rozwiązanie. Kilka lat wcześniej szeroko stosowany był pakiet CodeSourcery, który po przejęciu go przez firmę Mentor przestał być darmowy.

Skompilowany projekt w postaci pliku binarnego wgrywam do pamięci Flash mikrokontrolera za pomocą programu **openOCD**, który z jednej strony komunikuje się (poprzez sterowniki ftdi) z zamontowanym w interfejsie JTAG programatora chipem FTDI, a z drugiej nasłuchuje na porcie 3333 rozkazów z gdb. Używam JTAG'a **lock pick** autorstwa Freddiego Chopina.

Oprogramowanie mikrokontrolera

Oprogramowanie dla mikrokontrolera częściowo oparłem o biblioteki CMSIS od firmy ST. Konfigurację systemu zegara



Rysunek 4. Ramka protokołu transmisji

Listing 1. Konfigurowanie USART za pomocą biblioteki
```
USART_InitTypeDef USART_InitStructure;
USART_InitStructure.USART_BaudRate = BT_SPEED ;//460800;
USART_InitStructure.USART_WordLength = USART_WordLength_8b;
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Parity = USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_
None;
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
USART_Init( BT_USART, &USART_InitStructure);
USART_ITConfig(BT_USART, USART_IT_RXNE, ENABLE);
USART_Cmd(BT_USART, ENABLE);
```

Listing 2. Konfigurowanie USART – bezpośrednio (bez użycia bibliotek)
```
BT_USART->BRR = 0x9C; //230400 USART APB1 = 36Mhz// 0x753; == 19200 //
BT_USART->CR1 = USART_CR1_UE | USART_CR1_RXNEIE | USART_CR1_RE | USART_CR1_TE;
//Usart enable re te, interrupt rx, enable,
```

Listing 3. Funkcja sprawdzająca fazę automatycznego generowania koloru
```
bool checkStage( unsigned int color)
{
    unsigned int r = (color >> 16) & 0xFF;
    unsigned int g = (color >> 8) & 0xFF;
    unsigned int b = color & 0xFF;
    //two zero detecting
    if( r + g + b == 255)
    {
        return true;
    }
    else if ( r + g + b == 255*2) //two 255 detecting
    {
        return true;
    }
    return false;
}
```

Listing 4. Tablica LUT do automatycznego generowania koloru
```
unsigned int stepValues[6]= {
    +0x1,
    -0x10000,
    +0x100,
    -0x1,
    +0x10000,
    -0x100
};
```

oraz przerwań pozostawiłem domyślne, natomiast pozostałe wykorzystywane przeze mnie peryferia skonfigurowałem modyfikując odpowiednie rejestry.

Osobom programującym mikrokontrolery ST na pewno jest znany dylemat czy używać Standard Peripheral Library, czy wyrzucić je do kosza. Z pewnością zaletą jej użycia będzie przejrzystość konfiguracji oraz łatwość adaptacji na podstawie wielu przykładów dostępnych w Internecie. Wadą jest to, że osoby używające tych bibliotek najczęściej nie wykazują chęci przeczytania dokumentacji mikrokontrolera, przez co w razie problemów nie wiedzą od czego zacząć diagnostykę. Z drugiej strony. Osoby, które przeczytają *Reference Manual* są w stanie samodzielnie skonfigurować peryferia, więc nie muszą marnować czasu na czytanie dokumentacji bibliotek. Dodatkowo, podczas debugowania znają na pamięć wartości jakie powinny zawierać poszczególne rejestry (do tego celu bardzo przydaje się plugin EmbSysRegView umożliwiający podgląd zawartości wszystkich rejestrów). Porównanie konfiguracji modułu USART z użyciem standardowych bibliotek ST pokazano na **listingu 1**, natomiast bez nich na **listingu 2**. Niech czytelnik sam zdecyduje, co jest dla niego bardziej odpowiednie.

Funkcja *main*. W pierwszych liniach funkcji *main* zawarto wywołania funkcji konfigurujących zegary, porty GPIO, UART, moduł przerwań, oraz timery. Następnie umieszczono nieskończoną pętlę *while*, która odpowiada za generowanie automatycznych zmian koloru.

Zmianę koloru w najprostszym wypadku można by uzyskać dzięki zwykłej inkrementacji. Przy takiej metodzie dałoby się uzyskać wszystkie możliwe kolory, lecz nie zmieniałyby się one płynnie (np. jasny niebieski 0x0000ff a następnie skok do bardzo ciemnego zielonego 0x000100). Z tego powodu należy płynnie odejmować jasność z jednego koloru, a po dojściu do zera dodawać do innego, aż do osiągnięcia 0xFF. Aby rozróżnić poszczególne etapy napisałem funkcję checkStage (**listing 3**) wykrywającą sytuację, w której jeden lub dwa kolory są wygaszone. Jeśli funkcja zwróci prawdę, to inkrementujemy indeks *stepValues* (**listing 4**) do tablicy LUT, w której są zdefiniowane wartości, jakie mają być dodane lub odjęte od liczby definiującej kolor.

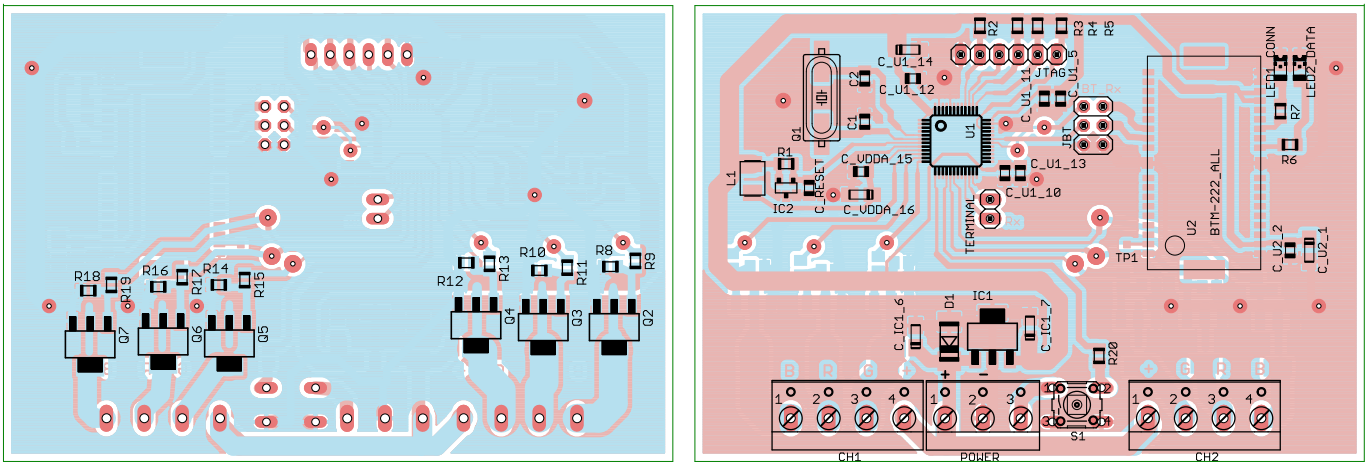
Sposób działania pętli *while* pokazano na **listingu 5**. Dla zagwarantowania prawidłowej zmiany koloru fragment kodu odpowiedzialny za przeliczanie koloru jest wykonywany przy wyłączonej obsłudze przerwań od UART. Pozostały czas procesora jest zużywany w funkcji *delay*, która sprawdza w pętli flagi błędów od UART i po wykryciu aktywnej flagi wysyła informację do terminala.

Przerwanie od USART3. USART3 odpowiada za komunikację wysyłanie logów na terminal do komputera oraz za przyjmowanie komend. Domyślnie ustawiony jest na prędkość 230400 kbs. Obsługiwane komendy pozwalają na ręczne modyfikowanie poszczególnych kolorów i składają się z następujących znaków: numer kanału, pierwsza litera koloru, spacja, wartość z zakresu 0 – 255. Przykładowe komendy: 1r 127 (ustawienie koloru czerwonego pierwszego kanału na 50 % mocy), 2b 255 (ustawienie koloru niebieskiego drugiego kanału na pełną moc).

Przerwanie od USART1. USART1 odpowiada za komunikację z modułem Bluetooth. W obsłudze przerwania sprawdzane są bajty początku ramki i po odebraniu ilości bajtów odpowiadającej długości ramki sprawdzany jest bajt końca ramki. Jeśli się zgadza następuje ustawienie rejestrów TIMx->CCRx (Channel Compare Register) dla poszczególnych kanałów.

Zdalne wyłączenie zasilania. Z uwagi na zastosowany w prototypie zasilacz ATX, który może być włączany i wyłączany poprzez linię sterującą, zdecydowałem się taką linię wyprowadzić z mikrokontrolera. Włączenie zasilacza następuje poprzez

Listing 5. Pętla *while* w funkcji *main*
```
while(1)
{
    BT_USART->CR1 &= ~USART_CR1_RXNEIE; //turn off it
    if(autoColor)
    {
        CH1_R_LED_CCR = (autoColor >> 16) & 0xFF;
        CH1_G_LED_CCR = (autoColor >> 8) & 0xFF;
        CH1_B_LED_CCR = (autoColor) & 0xFF;
        CH2_R_LED_CCR = (autoColor >> 16) & 0xFF;
        CH2_G_LED_CCR = (autoColor >> 8) & 0xFF;
        CH2_B_LED_CCR = (autoColor) & 0xFF;
        autoColor = autoColor + stepValues[autoColorStage];
        if( checkStage(autoColor)) autoColorStage++;
        if( autoColorStage >= 6 ) autoColorStage = 0;
    }
    BT_USART->CR1 |= USART_CR1_RXNEIE; //turn on it
    delay();
```



Rysunek 5. Schemat montażowy kontrolera LED RGB z Bluetooth

wciśnięcie microswitcha na płytce, natomiast wyłączenie z poziomu aplikacji. Linia ta jest dostępna na trzecim pinie środkowego złącza śrubowego.

Plik inc/hwConf.h. Plik *hwConf.h* zawiera stałe. Nie wszystkie mogą być łatwo modyfikowane bez zmian w innych plikach projektu ale te które mogą zainteresować użytkownika to: POWER_ON oraz POWER_OFF (definiują aktywny stan linii power on/off), LED_FRESH_RATE (definiuje częstotliwość pracy PWMa), AUTO_COLOR_LOOP_DELAY (definiuje wartość opóźnienia przy automatycznym generowaniu koloru).

Montaż

Schemat montażowy sterownika pokazano na **rysunku 5**. Montaż jest wykonywany typowo. Proponuję rozpocząć go od montażu elementów SMD na warstwie górnej, a następnie na spodzie płytki. Na końcu montujemy elementy przewlekane. Na płytce zostało przewidziane miejsce na układ generowania sygnału zerowania DS1818R, jednak w ostateczności można go zastąpić rezystorem. Cewkę L1 można zastąpić zworą, ponieważ urządzenie nie używa przetwornika A/C. Kwestię anteny do Bluetooth pozostawiam otwartą. Planowałem zastosować antenę fabryczną SMD na częstotliwość 2,4 GHz, jednak brak problemów związanych z zasięgiem przekonał mnie, że zwykły kawałek drutu o długości 3,1 cm również jest świetną anteną, wystarczającą dla tej aplikacji.

Uruchomienie i konfigurowanie

Mikrokontroler najlepiej zaprogramować za pomocą JTAG. Jego sygnały są dostępne na złączu 6-pinowym. Kolejność sygnałów jest następująca: GND, JNTRST, JTDO, JTDI, JTCK, JTMS.

Konfigurowanie Bluetooth. Moduł BTM-222 zastosowany w sterowniku nie wymaga przeprowadzania skomplikowanej konfiguracji. Domyślnie, moduł BTM-222 pracuje z prędkością 19200 kbs. Powinniśmy ją podnieść do 230400, ba na taką jest skonfigurowany UART mikrokontrolera. Dla wygody polecam jeszcze modyfikację rozgłaszanej nazwy urządzenia. Zmiana jej na „RGBController" pozwoli aplikacji Android na automatyczne łączenie zawsze, gdy urządzenie będzie w zasięgu. Dobrym pomysłem jest również zmiana domyślnego numeru PIN modułu Bluetooth. Jeśli tego nie zrobimy, może się zdarzyć, że inny czytelnik Elektroniki Praktycznej mieszkający obok będzie w stanie połączyć się z naszym kontrolerem.

Dla skonfigurowania modułu BTM-222 musimy przyłączyć jego linie RxD i TxD do portu szeregowego komputera (oczywiście poprzez konwerter napięć UART 3,3 V/RS232). Domyślna prędkość dla nowego modułu to 19200 kbs i taką też należy ustawić w terminalu. Pozostałe parametry połączenia można pozostawić domyślne. Po sprawdzeniu, że moduł odpowiada echem możemy przystąpić do wydawania poleceń konfiguracyjnych. W tym celu wpisujemy komendy AT zamieszczone na **listingu 6** kończąc każdą linijkę przyciśnięciem Enter (w razie problemów należy sprawdzić, jaki znak jest domyślnie wysyłany jako znak końca linii przez terminal). Pierwsza komenda zmieni nazwę urządzenia, druga zmieni numer PIN, natomiast trzecia ustawi prędkość transmisji na 230400 kbs. Do ponownej komunikacji z modułem niezbędna będzie oczywiście zmiana prędkości w konfiguracji terminala.

Po zakończeniu konfiguracji należy założyć zworki na goldpiny łączące linie RxD oraz TxD pomiędzy mikrokontrolerem a modułem Bluetooth.

Zasilanie. Do zasilania diod LED mocy zaleca się stosowanie dobrych zasilaczy impulsowych. Jest to podyktowane tym, aby jak najbardziej ograniczyć wahania napięcia przy zmianie obciążenia zasilacza. Częste przekraczanie dopuszczalnego napięcia zasilania diod LED prowadzi do ograniczenia ich żywotności.

Projektując rozprowadzenie linii zasilających do taśm RGB, należy wziąć pod uwagę znaczny spadek napięcia pomiędzy początkiem i końcem taśmy, który może

Listing 6. Konfigurowanie modułu Bluetooth za pomocą komend AT

```
ATN=RGBController
ATP=1234
ATL6
```

sięgać nawet kilku woltów prowadząc do wyraźnej różnicy w jasności świecenia diod na przeciwległych końcach. Zalecam doprowadzenie przewodów zasilających do obu końców taśmy, jeśli jest możliwość, to nawet dodatkowe dołączenie ich w środkowym odcinku taśmy. Dzięki temu znacznie ograniczymy spadek napięcia i zyskamy na równomierności świecenia. W swojej instalacji zasiliłem taśmę łącznie w trzech miejscach i przy napięciu na wyjściu zasilacza 11,5 V maksymalny prąd zasilający wynosi 3 A. Gdy zasilałem taśmę tylko z jednej strony, ten prąd był mniejszy o ponad 0,5 A.

Rozwój projektu. Niestety z braku czasu nie zrealizowałem wszystkich pomysłów. Jeśli ktoś chciałby poświęcić swój czas i zmodyfikować projekt, to służę pomocą. Zainteresowanym osobom udostępnię pliki źródłowe. Przydałoby się między innymi utworzyć dodatkowe pole edycyjne w aplikacji do przechowywania wartości opóźnienia przy automatycznym generowaniu koloru i przesyłać ją na bieżąco zamiast trzymać stałą wartość w procesorze. Innym niezrealizowanym pomysłem jest wykonanie transformaty FFT i zmiana koloru w zależności od częstotliwości muzyki, czyli stary, dobry kolorofon.

Tranzystory IRLR2905PBF mogą być wymienione na inne pod warunkiem, że będą miały całkowicie otwarty kanał przy napięciu Vgs=3,3 V. Tranzystor IRLR2905PBF ma progowe napięcia załączania mieszczące się granicach 1…2 V. Model IRLR2705PBF również może być stosowany. Stabilizator MCP1755S-3302E/DB może być zamieniony na inny, jeśli maksymalne napięcie wejściowe będzie wynosiło powyżej 12 V.

Paweł Domagalski
domagalski250@gmail.com