Sygnał odebrany jest przez antenę i trafia do selektywnego wzmacniacza w.cz. Następnie poddawany jest obróbce wstępnej i próbkowany przez przetwornik analogowo-cyfrowy (A/C). Spróbkowany sygnał trafia do układu cyfrowego przetwarzania danych DSP (Digital Signal Processing), gdzie odbywają się na nim przekształcenia takie jak filtracja, mieszanie i demodulacja poprzez zastosowanie odpowiednich algorytmów matematycznych na sygnale dyskretnym. Ostatecznie sygnał po cyfrowej obróbce trafia do przetwornika cyfrowo-analogowego CA, gdzie z sygnału dyskretnego uzyskiwany jest użyteczny sygnał ciągły niskiej częstotliwości. W ostatnim etapie trafia on do wzmacniacza m.cz., gdzie jest wzmacniany i jest odtworzony przy użyciu głośnika.

PODSTAWOWE ZAŁOŻENIA PROJEKTOWE

Celem projektu jest przedstawienie sposobu budowy prostego radioodbiornika cyfrowego współpracującego z mikrokontrolerem, a także programowa implementacja jego obsługi. Zbudowane urządzenia ma być zasilane z akumulatora Li-ion, a także posiadać układ kontroli jego rozładowania. Wymiary radioodbiornika mają być jak najmniejsze, tak aby uzyskać urządzenie kieszonkowe. W przypadku rozładowania akumulatora, stan ten powinien zostać zasygnalizowany i uniemożliwione powinno zostać jego uruchomienie. Zakładany jest odbiór stacji radiowych modulowanych częstotliwościowo w paśmie 87,5-108 MHz (pasmo CCIR).

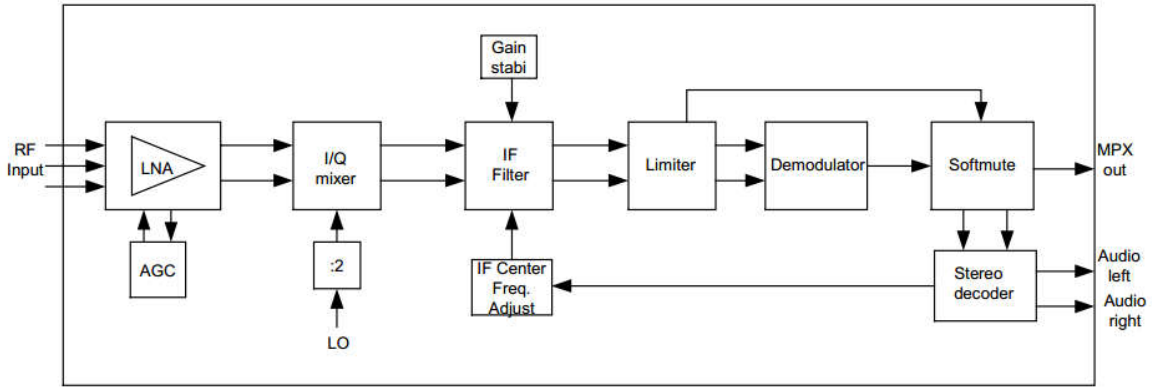
DOBÓR SDR

Na rynku dostępnych jest wiele układów radioodbiorników, które różnią się parametrami takimi jak napięcie zasilania, pasmo, obudowa, sposób modulacji sygnału, sposób sterowania, pobór prądu itp. Porównanie kilku układów zostało przedstawione w tab. 1.

Tabela 1. Porównanie dostępnych układów radioodbiorników [3, 4, 5, 6, 7, 8]

	TEA5767/68	RDA5807m	SI4703	SI4735	SI4689
Producent	Philips semi-conductors	RDA micro-electronics	Silicon Labs	Silicon Labs	Silicon Labs
Zasilanie	2,5-5V	2,7-3,3V	2,7-5,5V	2,7-5,5V	1,62-3,6V
Pobór prądu	<10,5mA	<20mA	<15,8mA	<21,3mA	-
Pasmo	FM	FM	FM	FM, AM, SW, LW	FM, AM, DAB, DAB+
THD	<1%	<0,2%	<0,5%	<0,1%	-
RDS	brak	jest	jest	jest	jest
Komunikacja	I^2C, 3Wire	I^2C	I^2C	I^2C, 3Wire	I^2C, SPI
Inne	AGC, HCC, SNR, SNL	AGC, SNR	AGC, AFL, SNR	AFC, AGL, SNR	MPEG, EPG,ASRC

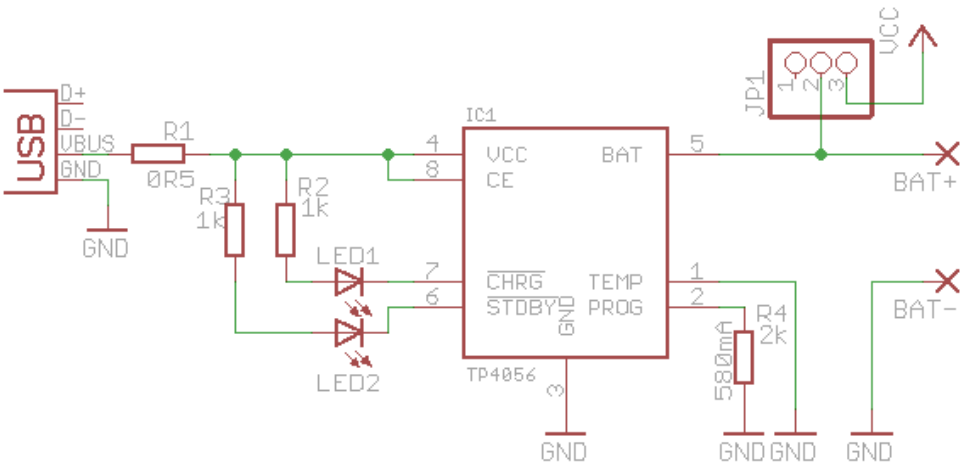
W urządzeniu zastosowany został układ TEA5767 z powodu łatwej implementacji programowej sterowania takim układem, nieskomplikowanego układu rejestrów wewnętrznych oraz ogólnej jego dostępności na rynku. Na rysunku 3 przedstawiony został jego schemat blokowy.



Rys. 3. Schemat blokowy układu tea5767 [3]

UKŁAD ZASILANIA

Zasilanie układu zrealizowane zostało przy użyciu ogniwa Li-ion o oznaczeniu 14650. Znamionowe napięcie ogniwa wynosi 3,7V jednakże przy naładowaniu wynosi ono 4,2V. Nie wolno dopuścić do rozładowania poniżej napięcia 3,2V, gdyż grozi to jego uszkodzeniem. Schemat układu zasilania przedstawia rys. 4.

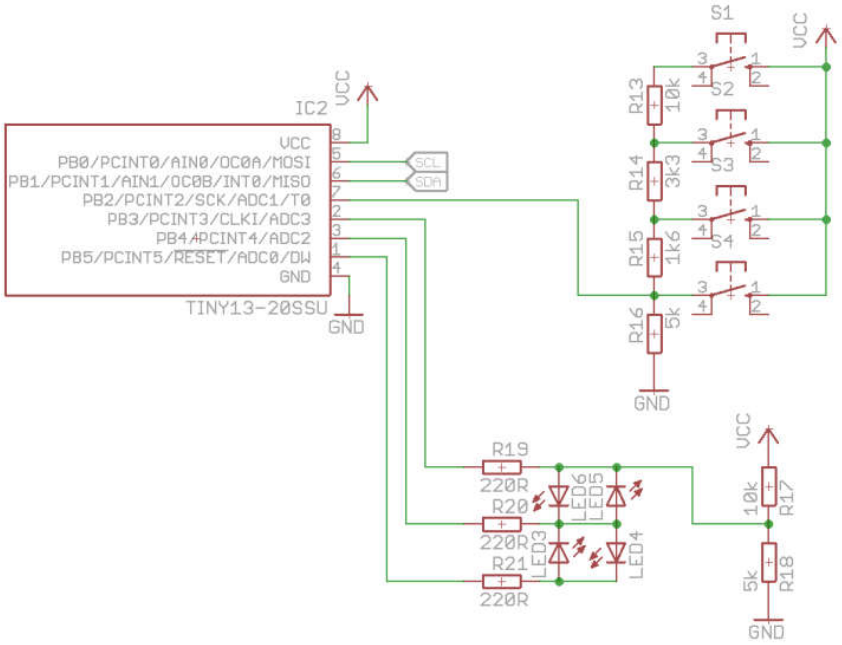


Rys. 4. Układ zasilania (źródło: opracowanie własne)

Do ładowania ogniwa użyty został układ TP4056 w standardowej aplikacji [11]. Diody LED1 oraz LED2 kolejno sygnalizują proces ładowania i podłączenie wtyczki USB do układu. Wartość rezystora R4 ustala prąd ładowania akumulatora na 580mA, co przy typowej pojemności akumulatora równej około 1000 mAh skutkuje czasem pełnego naładowania równym 1h 40min. Element JP1 jest to przełącznik jednotorowy dwupozycyjny służący do zadawania zasilania do dalszej części układu.

UKŁAD MIKROKONTROLERA

Jako element sterujący zastosowany został mikrokontroler ATtiny13V firmy AVR. Posiada on 1KB pamięci flash, 64 bajty pamięci RAM oraz EEPROM [9]. Napięcie zasilania zawiera się w przedziale 1,8 do 5,5V, a maksymalna częstotliwość taktowania to 20MHz. Rys. 5 przedstawia aplikację mikrokontrolera.

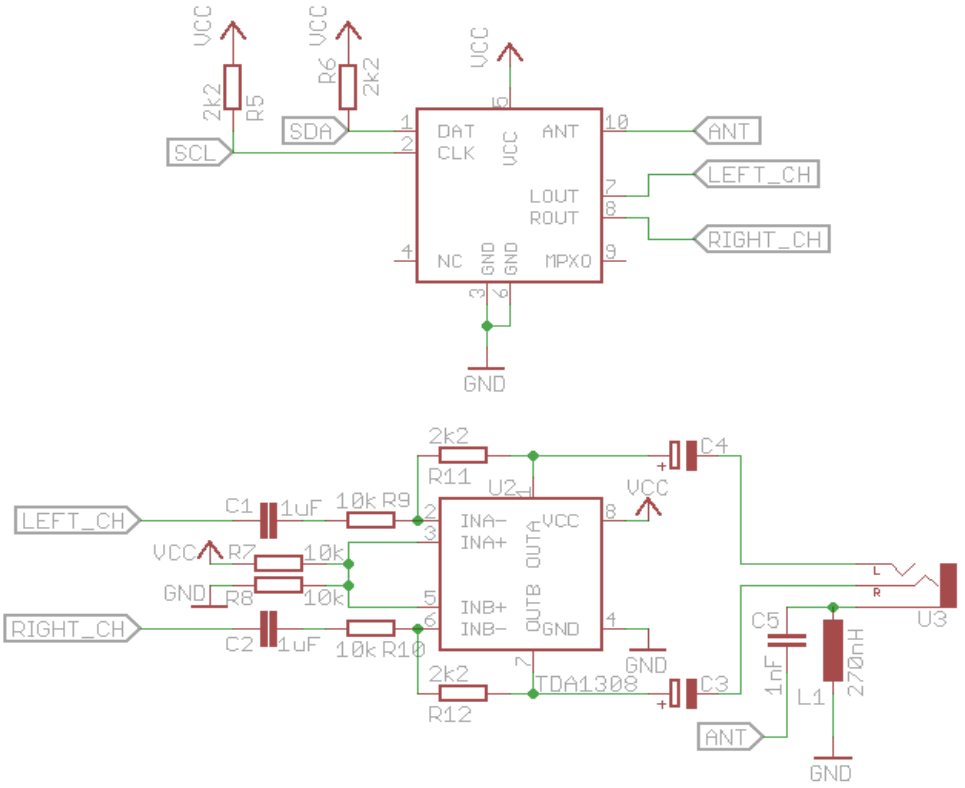


Rys. 5. Układ mikrokontrolera do sterowania układem TEA5767 (źródło: opracowanie własne)

Linie SCL i SDA są to kolejno linia zegarowa oraz linia danych interfejsu I^2C połączone z liniami sterującymi w układzie TEA5767. Przyciski S1–S4 połączone zostały z wielostopniowym dzielnikiem napięciowym i dołączone do wejścia przetwornika AC w mikrokontrolerze. Wciśnięcie przycisku spowoduję zmianę napięcia na wejściu przetwornika. Dzięki różnym wartościom rezystancji dołączonych do przycisków, wartość napięcia na wejściu będzie zależna od tego, który przycisk zostanie wciśnięty i pozwoli na jego identyfikację. Przyciski S1 oraz S2 służą do automatycznego wyszukiwania stacji radiowej, natomiast S3 i S4 do ręcznego dostrajania co 0,1 MHz. Diody LED3–6 sygnalizują poziom sygnału odbieranej stacji radiowej. Do ich sterowania wykorzystany został fakt, iż porty wyjściowe mikrokontrolera są trójstanowe. Możliwość ustawiania dowolnego wyjścia w stan wysokiej impedancji (R_{we}>1GΩ) pozwala na zaświecenie pojedynczej diody w układzie charlieplexingu. Rezystory R17 oraz R18 tworzą dzielnik napięciowy potrzebny do kontroli stanu rozładowania akumulatora.

UKŁAD ODBIORNIKA I WZMACNIACZA

Ostatnim elementem jest sam odbiornik radiowy oraz wzmacniacz wyjściowy. Sygnały sterujące podawane na wejścia DAT i CLK radioodbiornika konfigurują układ, a odebrany sygnał wystawiany jest na wyjścia LOUT oraz ROUT połączone następnie ze wzmacniaczem opartym o układ TDA1308. Specyfikacja magistrali I^2C określa, że stanem recesywnym na liniach jest stan wysoki i same linie są typu otwarty dren co wymusza zastosowanie rezystorów podciągających R5 oraz R6 do VCC. Napięcie zasilania wzmacniacza musi zawierać się w przedziale 3–5V, co gwarantuje nam użyte ogniwo Li-ion. Aplikacja wzmacniacza jest to typowy schemat połączeń zaczerpnięty z dokumentacji układu TDA1308 [12]. Dodatkowo do masy wejścia słuchawkowego dołączony zostaje filtr górnoprzepustowy zbudowany na elementach C5 oraz L1 i następnie podłączony jest on do wejścia ANT układu TEA5767. Powoduje on, że słuchawki będą pracować jako antena.



Rys. 6. Aplikacja mikrokontrolera do sterowania układem TEA5767(źródło: opracowanie własne)

ALGORYTM STEROWANIA

Sterownie układem odbywa się poprzez początkowe wpisanie do układu dolnej częstotliwości pasma FM wynoszącej 88MHz. Następnie uruchamiane jest skanowanie pasma w górę. W przypadku znalezienia stacji radiowej skanowanie zostaje zatrzymane a na diody sygnalizacyjne wystawiany jest poziom aktualnie odbieranego sygnału. W dalszej kolejności program czeka na przyciśnięcie któregoś z przycisków przez użytkownika. Wciśniecie przycisków S1 lub S2 powoduje skanowanie kolejno w górę oraz w dół pasma, aż do znalezienia kolejnej stacji radiowej. W przypadku naciśnięcia przycisku S3 lub S4 aktualna częstotliwość zmieniana jest o 0,1 MHz co pozwala dostroić pożądaną stację radiową ręcznie. Dodatkowo wraz z uruchomieniem urządzenia w pierwszej kolejności sprawdzane jest napięcie akumulatora. Jeżeli napięcie spadło poniżej wartości 3,2 V mikrokontroler ustawia układ radioodbiornika w tryb niskiego poboru energii, a następnie sam przechodzi w tryb uśpienia. Wybudzenie mikrokontrolera możliwe jest jedynie poprzez wyłączenie urządzenia i ponowne jego włączenie. Jeżeli po ponownym włączeniu, napięcie zasilania będzie znów zbyt niskie, sekwencja zostanie powtórzona.

KOD PROGRAMU

Program napisany został w programie Atmel Studio 7.0 z kompilatorem Atmel Toolchain 3.4.2. Ustawienia fusebitów mikrokontrolera wynoszą odpowiednio: LOW = 0x7A, HIGH = 0xFF, LOCK = 0x3F. Takie ustawienie pozwala na pracę mikrokontrolera z taktowaniem 9,6 MHz [2].

```
/*
 * tea+attiny.c
 * Created: 2016-06-05 15:36:13
 * Author : Pawel Lawnik, Tomasaz Luc
 */
#include <avr/io.h>
#include <avr/sleep.h>
#include <avr/power.h>
#include <util/delay.h>
#define TEA_WRITE_ADDRESS 0xC0
#define TEA_READ_ADDRESS 0xC1
#define SCLPORT   B
#define SDAPORT   B
#define SCL    PB0
#define SDA    PB1
#define PORT(x) XPORT(x)
#define XPORT(x) (PORT##x)
#define PIN(x) XPIN(x)
#define XPIN(x) (PIN##x)
#define DDR(x) XDDR(x)
#define XDDR(x) (DDR##x)
#define SDA_LO PORT(SDAPORT) &= ~(1<<SDA)
#define SDA_HI PORT(SDAPORT) |= (1<<SDA)
#define SCL_LO PORT(SCLPORT) &= ~(1<<SCL)
#define SCL_HI PORT(SCLPORT) |= (1<<SCL)
#define I2C_CLOCK_PULSE _delay_loop_1(2); SCL_HI;\
                  _delay_loop_1(2); SCL_LO;
#define BATTERY_ADC_MASK   0b11
#define KEYS_ADC_MASK      0b10
#define LOW_BATTERY_LVL    32*256/33
void i2c_start(void);
void i2c_stop(void);
void i2c_init(void);
void i2c_send_byte(uint8_t data);
uint8_t i2c_receive_byte(void);
void tea_write_registers(uint8_t *tab);
void tea_read_registers(uint8_t *tab);
void tea_seek_up(void);
void tea_seek_down(void);
uint8_t tea_read_sig_lvl();
uint8_t get_adc_val(uint8_t channel_mask);
void check_battery_state(void);
uint8_t check_buttons(void);
uint8_t tea_read_tab[5];
uint8_t tea_write_tab[5] =
{
   0x70,
   0x08,
   0xD0,
   0x16,
   0x00,
};
int main(void)
{
   PORTB = 0x00;
   DDRB=0x00;
   i2c_init();
   tea_write_registers(tea_write_tab);
    while (1)
    {
```

```
check_battery_state();
uint8_t buttons_status;
uint16_t pll_val;
uint8_t sig_lvl;
sig_lvl = tea_read_sig_lvl();
buttons_status = check_buttons();
if(sig_lvl==0){
   DDRB &= 0xC7;
   PORTB &= 0xC7;
   DDRB |= (1<<PB3)|(1<<PB4);
   DDRB |= (1<<PB3);
}
else if(sig_lvl<5){
   DDRB &= 0xC7;
   PORTB &= 0xC7;
   DDRB |= (1<<PB3)|(1<<PB4);
   DDRB |= (1<<PB4);
}
else if(sig_lvl<9){
}
else if(sig_lvl<13){
   DDRB &= 0xC7;
   PORTB &= 0xC7;
   DDRB |= (1<<PB4)|(1<<PB5);
   DDRB |= (1<<PB4);
}
else {
   DDRB &= 0xC7;
   PORTB &= 0xC7;
   DDRB |= (1<<PB4)|(1<<PB5);
   DDRB |= (1<<PB5);
}
switch(buttons_status){
   case 1:
      tea_read_registers(tea_read_tab);
      pll_val = tea_read_tab[1];
      pll_val |= (tea_read_tab[0]&0x3F)<<8;
      if(pll_val<13212UL){
         pll_val+=12;
      }
      tea_write_tab[1] = pll_val;
      pll_val >>=8;
      tea_write_tab[0] &= ~0x3F;
      tea_write_tab[0] |= pll_val;
      tea_write_registers(tea_write_tab);
   break;
   case 2:
      tea_read_registers(tea_read_tab);
      pll_val = tea_read_tab[1];
      pll_val |= (tea_read_tab[0]&0x3F)<<8;
      if(pll_val>10709UL){
         pll_val-=12;
      }
      tea_write_tab[1] = pll_val;
      pll_val >>=8;
      tea_write_tab[0] &= ~0x3F;
      tea_write_tab[0] |= pll_val;
      tea_write_registers(tea_write_tab);
   break;
   case 3:
```

```
            tea_seek_up();
         break;
         case 4:
            tea_seek_down();
         break;
      }
   }
}
void tea_write_registers(uint8_t *tab){
   i2c_start();
   i2c_send_byte(TEA_WRITE_ADDRESS);
   for(uint8_t a=0;a<5;a++){
      i2c_send_byte(tab[a]);
   }
   i2c_stop();
}
void tea_read_registers(uint8_t *tab){
   i2c_start();
   i2c_send_byte(TEA_READ_ADDRESS);

   for(uint8_t a=0;a<5;a++){
      tab[a]=i2c_receive_byte();
   }
   i2c_stop();
}
void tea_seek_up(void){
   tea_read_registers(tea_read_tab);
   tea_read_tab[0] &= 0x3F;
   tea_write_tab[0] |= tea_read_tab[0];
   tea_write_tab[1] = tea_read_tab[1];
   tea_write_tab[2] |= 0x80;
   tea_write_tab[0] |= 0x40;
   tea_write_registers(tea_write_tab);
}
void tea_seek_down(void){
   tea_read_registers(tea_read_tab);
   tea_read_tab[0] &= 0x3F;
   tea_write_tab[0] |= tea_read_tab[0];
   tea_write_tab[1] = tea_read_tab[1];
   tea_write_tab[2] &= ~0x80;
   tea_write_tab[0] |= 0x40;
   tea_write_registers(tea_write_tab);
}
void i2c_start(void){
   SDA_LO;
   _delay_loop_1(1);
   SCL_LO;
}
void i2c_stop(void){
   SDA_LO;
   _delay_loop_1(2);
   SCL_HI;
   _delay_loop_1(1);
   SDA_HI;
   _delay_loop_1(2);
}
void i2c_init(void){
   DDR(SDAPORT) |= (1<<SDA);
   DDR(SCLPORT) |= (1<<SCL);
   SDA_HI;
```

```
    SCL_HI;
}
void i2c_send_byte(uint8_t data){
    for(uint8_t mask=0b1000000;mask;mask>>=1){
        if(mask&data)SDA_HI;
        else SDA_LO;
        I2C_CLOCK_PULSE;
    }
    SDA_HI;
    DDR(SDAPORT) &= ~(1<<SDA);
    _delay_loop_1(2);
    SCL_HI;
    _delay_loop_1(2);
    SCL_LO;
    _delay_loop_1(2);
}
uint8_t i2c_receive_byte(void){
    uint8_t data = 0, mask = 0;
    SDA_HI;
    DDR(SDAPORT) &= ~(1<<SDA);
    for(uint8_t a=0;a<8;a++){
        _delay_loop_1(2);
        SCL_HI;
        mask = PIN(SDAPORT) & (1<<SDA);
        if(mask) data|=1;
        data<<=1;
        _delay_loop_1(2);
        SCL_LO;
    }
    DDR(SDAPORT) |= (1<<SDA);
    SDA_LO;
    I2C_CLOCK_PULSE;
    SDA_HI;
    return data;
}
uint8_t get_adc_val(uint8_t channel_mask){

    ADMUX =(1<<REFS0)|(1<<ADLAR)|channel_mask;
    ADCSRA =(1<<ADEN)|(1<<ADSC)|(1<<ADPS0)|
    (1<<ADPS1)|(1<<ADPS2);
    while(ADCSRA&(1<<ADSC));
    return ADCH;
}
void check battery state(void){
    uint8_t battery_lvl;
    PORTB &= ~(1<<PB3);
    DDRB &= ~(1<<PB3);
    battery_lvl = get_adc_val(BATTERY_ADC_MASK);
    if(battery_lvl<LOW_BATTERY_LVL){
        tea_write_tab[3] = 0x56;
        tea_write_registers(tea_write_tab);
        power_all_disable();
        set_sleep_mode(SLEEP_MODE_PWR_DOWN);
        sleep_enable();
        sleep bod disable();
        sleep_cpu();
    }
}
uint8_t check_buttons(void){
    uint8_t adc_button_lvl;
```

```
    adc_button_lvl = get_adc_val(KEYS_ADC_MASK);
    while(adc_button_lvl>3);
    if(adc_button_lvl>215) return 1;
    if(adc_button_lvl>150) return 2;
    if(adc_button_lvl>85) return 3;
    if(adc_button_lvl>28) return 4;
    return 0;
}
uint8_t tea_read_sig_lvl(void){
    tea_read_registers(tea_read_tab);
    return tea_read_tab[3]>>4;
}
```

PODSUMOWANIE

Zaproponowane rozwiązanie aplikacji bazującej na mikrokontrolerze ATtiny13V firmy AVR pozwala wykorzystać je i zaprogramować jako sterowalny odbiornik radiowy. Otwartość kodu, dostępność i niskie koszty elektronicznych części pozwalają wykonać własne rozwiązania prostych urządzeń a działający projekt jest tego dowodem.

LITERATURA

[1] Trusz J., Trusz W., *Odbiorniki radiowe, telewizyjne i magnetofony – opisy i dane techniczne*, Wydawnictwo komunikacji i łączności, Warszawa 1974

[2] Kardaś M., *Mikrokontrolery AVR język C – podstawy programowania*, Wydawnictwo Atnel, Szczecin 2011

[3] Philips semiconductors, AN10133 Low voltage Fm stereo radio with TEA5767/68, 18.06.2002

[4] NXP semiconductors, TEA5767HN Low-power FM stereo radio for handheld applications, REV.05 26.01.2007

[5] RDA Microelectronics, Single-Chip Broadcast FM Radio Tuner RDA5806M, Rev 1.1-07.2011

[6] Silicon Labs, Broadcast AM/FM/SW/LW Radio Receiver Si4730/31/34/35-D60, Rev.1.2 08.2013

[7] Silicon Labs, Broadcast FM Radio Tuner for Portable Applications Si4702/03-C19, Rev. 1.1 07.2009

[8] Silicon Labs, Single-Chip, Am/FM/HD/DAB/DAB+ Radio Receiver Si4689-A10, 08.11.2014

[9] Atmel, 8-bit AVR Microcontroller with 1K Bytes In-System Programmable Flash ATtiny13A, Rev. 8126F-AVR-05/12

[10] http://elektronikab2b.pl/biznes/1658-radiowy-odbiornik-cyfrowy-czyli-radio-programowalne (zasoby z dnia 12.06.2016)

[11] NanJing Top Power, Tp4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation In SOP-8

[12] NXP semiconductors, TDA1308 Class-AB stereo headphone driver, Rev. 5 14.03.2011