Podstawowe parametry:

- maksymalna liczba użytkowników: 40 (+ administrator),
- maksymalna liczba zdarzeń: 1000,
- czas automatycznego wylogowania administratora: 60 s,
- czas automatycznego wygaszenia podświetlenia: 40 s bezczynności użytkownika,
- czas automatycznego wyjścia do menu głównego: 30 s bezczynności użytkownika,
- maksymalny prąd styków przekaźnika wykonawczego: 1 A/220 VAC (szczegóły w dokumentacji elementu),
- napięcie zasilania: 9 VDC,
- maksymalny pobór prądu (przekaźnik wyłączony/załączony): 90/110 mA.

Dodatkowe materiały do pobrania ze strony www.ulubionykiosk.pl/media

----	NFC Lock (EP 4/2022)
AVT5186	Bezstykowy zamek RFID (EP 5/2009)
AVT969	Bezstykowy zamek RFID (EP 2/2007)
AVT3129	Zamek elektroniczny/immobilizer (EdW 7/2015)
AVT886	System bezstykowej kontroli dostępu (EP 10/2000)

W ofercie AVT*

AVT5934

* 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 stronie sklepu osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt Via e-mail: kity@avt.pl.

Simple Access System 2 (1)

Zaprezentowany projekt jest wyrazem mojej dalszej fascynacji technologią RFID, z którą po raz pierwszy spotkałem się podczas implementacji urządzenia NFClock, którego opis ukazał się w naszym miesięczniku w wydaniu kwietniowym (EP 4/22). Wtedy zaprojektowałem prosty zamek, który był otwierany przy użyciu karty RFID lub też urządzenia z interfejsem NFC. Z uwagi na prostotę konstrukcyjną było to jednak urządzenie, w którym musiałem pójść na pewne kompromisy. Tym razem chciałem zaprojektować „prawdziwy" system kontroli dostępu z pełną rejestracją zdarzeń i użytkowników.

Zasadniczym kompromisem zaimplementowanym w urządzeniu NFC Lock, było zastosowanie wyłącznie numeru seryjnego karty lub urządzenia NFC (UID), jako klucza autoryzacyjnego. Ktoś powie, w czym problem? Otóż problem jest w tym, że na rynku dostępne są „czyste" karty RFID, do których możemy wgrać dowolny numer seryjny i w ten sposób sklonować to z założenia niepowtarzalne peryferium. Co ciekawe, przypomina mi to trochę sytuację z interfejsem 1-wire. Producent i pomysłodawca interfejsu, firma MAXIM INTEGRATED, zapewniał, iż wyposażony w unikalny 64-bitowy numer seryjny układ jest niepowtarzalny, gdyż numer ten wypalany jest w trakcie procesu produkcji. Cóż, dobrych kilka lat temu, bo w wydaniu EP 2/2009 zaprezentowałem projekt „cButton. Emulator pastylek Maxim-Dallas", za pomocą którego mogliśmy skopiować dowolną pastylkę DS1990 i potem emulować ją z użyciem mikrokontrolera. W ten sposób łatwo dało się oszukać wszelkie czytniki tego typu układów, co sprawdziłem wielokrotnie w praktyce.

Tym razem chciałem zaprojektować „prawdziwy" system kontroli dostępu z pełną rejestracją zdarzeń i użytkowników pozwalający na zastosowanie bardziej wyszukanych sposobów autoryzacji. Z pomocą przyszła mi specyfikacja samych kart Mifare, które są flagowym przykładem technologii RFID. Nie będę w tym miejscu przytaczał wszystkich szczegółów dotyczących samych kart, jak i sposobu ich obsługi, gdyż ten temat drobiazgowo opisałem we wspomnianym wcześniej artykule. Skupię się na elementach nowych, istotnych z punktu widzenia procesu autoryzacji. Mowa o pamięci EEPROM, której nie tak mała ilość, bo 8 kilobitów, znajduje się na każdej karcie tego typu, a której obsługa dostarcza nam zaawansowanych mechanizmów autoryzacji.

Karty Mifare

Jak wspomniałem wcześniej, znaczniki w tym standardzie (1 k) wyposażone są w pamięć EEPROM o rozmiarze 8 kb. Pamięć ta zorganizowana jest w 16 sektorach, zawierających po 4 bloki pamięci, każdy o rozmiarze 16 bajtów. Łatwo policzyć, iż teoretycznie możemy pomieścić 1 kB danych. W praktyce jest to jednak nieco mniej, gdyż każdy sektor posiada zarezerwowany jeden blok (16 bajtów) noszący nazwę Sector Trailer, który przechowuje informacje dotyczące kluczy dostępowych oraz ustawionych reguł dostępu (*Access Bits*) do bloków tego sektora. Co więcej, pierwszy sektor tejże pamięci EEPROM, oprócz bloku Sector Trailer ma blok producenta Manfacturer Block przechowujący unikatowy identyfikator karty oraz dane producenta. Łatwo więc policzyć, że faktyczna ilość danych jaką możemy zapisać wynosi: (16 sektorów×3 bloki)–1=47 bloków danych co daje 47 bloków danych×16 bajtów=752 bajty. Niby niewiele, ale do całej rzeszy ciekawych zastosowań w zupełności wystarczy, wszak wiele

bankowych rozwiązań korzysta z tej funkcjonalności.

Na **rysunku 1** pokazano logiczną budowę pamięci EEPROM karty Mifare. Zgodnie z wcześniejszymi ustaleniami, każdy sektor danych zawiera blok o nazwie Sector Trailer, który przechowuje klucze autoryzacyjne (dwa – A i B) umożliwiające dostęp do poszczególnych bloków sektora, jak i reguły zarządzające tym dostępem – oddzielne dla każdego z bloków, w tym bloku Sector Trailer.

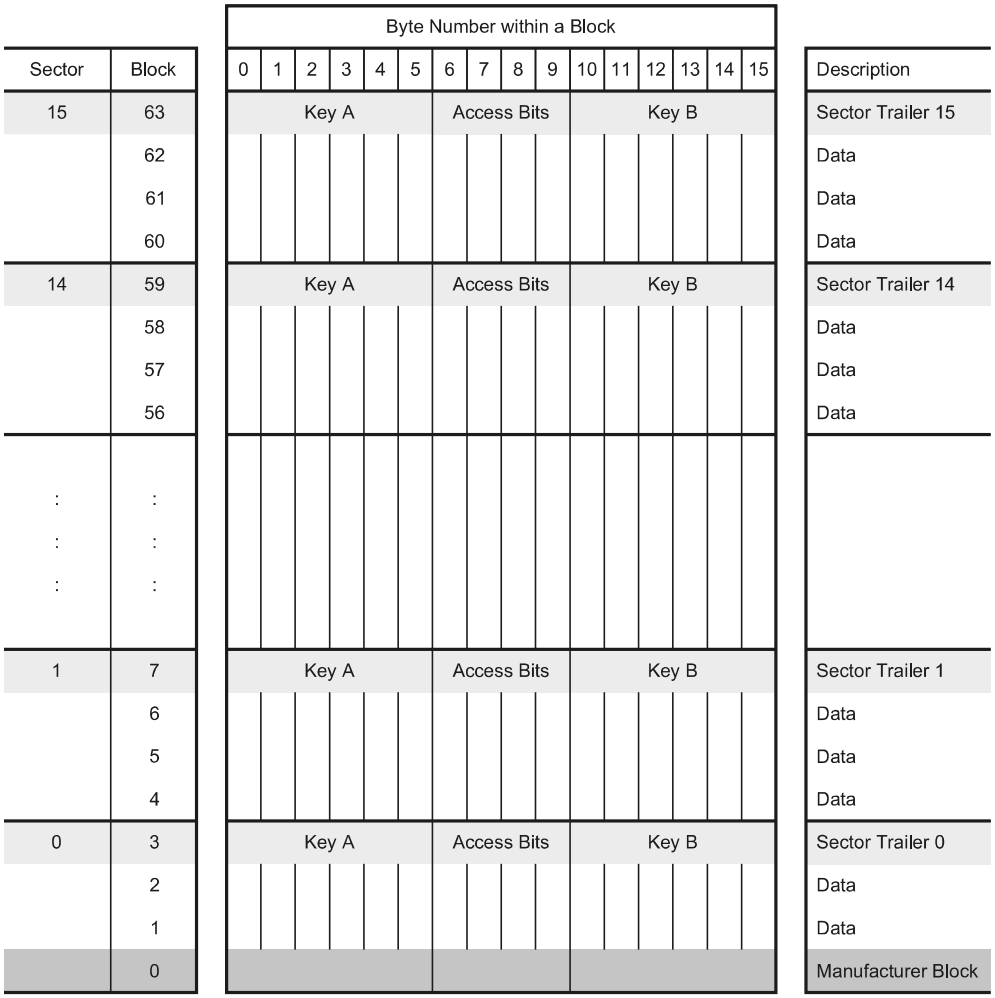
Na **rysunku 2** pokazano, jak wygląda struktura każdego bloku Sector Trailer:

- pierwsze sześć bajtów (0...5) przechowuje klucz dostępowy A;
- kolejne cztery bajty (6...9) przechowują informacje o ustawionych bitach dostępu do wszystkich czterech bloków pamięci w sektorze z osobna, przy czym bajt czwarty nie jest używany. Co ważne, każdy blok w sektorze może mieć inne reguły dostępu;
- ostatnie sześć bajtów (10...15) przechowuje opcjonalny, drugi klucz B.

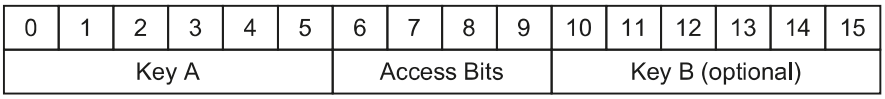
Za pomocą kluczy dostępu uzyskujemy prawa odczytu i/lub zapisu do poszczególnych bloków w sektorze, przy czym już teraz warto przyswoić informację, iż **wartości klucza A nie da się odczytać, można go tylko zapisać**. Skoro nie da się go odczytać (zwraca zawsze wartość 0x000000000000) to jak po raz pierwszy autoryzować odczyt sektora nie znając klucza? To proste, w świeżo zakupionych znacznikach RFID oraz NFC zarówno klucz A jak i klucz B mają wartość 0xFFFFFFFFFFFF, zaś bity dostępu wartość domyślną 0xFF0780, co oznacza następujące ustawienia autoryzacyjne:

- klucz A nie jest możliwy do odczytania (ale wiemy jaką ma fabrycznie ustawioną wartość),
- klucz B jest możliwy do odczytania i jest ustawiony jako 0xFFFFFFFFFFFF,
- klucz B nie jest używany do żadnych operacji na blokach pamięci,
- za pomocą klucza „A" możemy wykonywać dowolne operacje – zapisywać i odczytywać dane, ustawiać własne klucze oraz modyfikować bity dostępu.

Co ważne, klucze znajdują zastosowanie dla całego sektora (każdego bloku danych w ramach sektora), zaś bity dostępu dla każdego z bloków z osobna. Uprzedzam jednak, iż należy zachować szczególną ostrożność podczas ustawiania bitów dostępu, gdyż przez ustawienie niepoprawnych ich wartości możemy sobie zablokować jakikolwiek dostęp do całego sektora i to bezpowrotnie. Dla ułatwienia, w sieci znalazłem kilka kalkulatorów tychże bitów dostępu, ale najlepszy, z jakim się spotkałem znajduje się pod adresem: https://bit.ly/3rw8XET. Tyle po krótce o pamięci EEPROM kart Mifare, przejdźmy zatem do zagadnień implementacyjnych.



Rysunek 1. Logiczna budowa pamięci EEPROM karty Mifare



Rysunek 2. Struktura logiczna każdego bloku Sector Trailer karty Mifare

Listing 1. Funkcja autoryzująca dostęp danych do wybranego sektora karty

```
//Tries to authenticate a block of memory on a MIFARE 1k card
//UID – pointer to a byte array containing the card UID (4 or 7 bytes) and size byte (UID[7])
//Key – pointer to a byte array containing the 6 byte key value
//blockNumber – the block number to authenticate (0..63 for 1KB cards)

uint8_t pn532AuthenticateBlock(uint8_t *UID, uint8_t *Key, uint8_t blockNumber){
    uint8_t Result, UIDsize = UID[7];
    //Prepare the authentication command
    pn532PacketBuffer[0] =PN532_INDATAEXCHANGE;  //Data Exchange Command
    pn532PacketBuffer[1] = 1;  //Max card numbers
    pn532PacketBuffer[2] = PN532_AUTH_WITH_KEYA;  //Key type A
    pn532PacketBuffer[3] = blockNumber;  //Authenticated block number (0..63 for 1KB cards)
    memcpy(&pn532PacketBuffer[4], Key, 6);  //Key, bytes 4...9
    memcpy(&pn532PacketBuffer[10], UID, UIDsize);  //UID, bytes 10...13 or 10...16

    if((Result = pn532SendCommandCheckAck(pn532PacketBuffer, UIDsize == 4? 14: 17)) != NO_ERROR)
        return Result;

    //Read the response packet
    pn532ReadData(pn532PacketBuffer, 12);

    //Check if we are authenticated (Status byte == 0)
    if(pn532PacketBuffer[7] != 0x00) return AUTHENTICATION_ERROR;

    return NO_ERROR;
}
```

Listing 2. Funkcja pozwalająca na zapis autoryzowanego wcześniej bloku danych

```
//Tries to write an entire 16-byte data block at the specified block address.

uint8_t pn532WriteBlock(uint8_t *Data, uint8_t blockNumber){
    uint8_t Result;
    //Prepare write command
    pn532PacketBuffer[0] =PN532_INDATAEXCHANGE;  //Data Exchange Command
    pn532PacketBuffer[1] = 1;  //Max card numbers
    pn532PacketBuffer[2] = PN532_MIFARE_WRITE;  //Mifare write command
    pn532PacketBuffer[3] = blockNumber;  //Written block number (0..63 for 1KB cards)
    memcpy(&pn532PacketBuffer[4], Data, 16);  //Data

    if((Result = pn532SendCommandCheckAck(pn532PacketBuffer, 20)) != NO_ERROR) return
Result;
    _delay_ms(10);

    //Read the response packet
    pn532ReadData(pn532PacketBuffer, 26);

    return NO_ERROR;
}
```

Listing 3. Funkcja pozwalająca na odczyt autoryzowanego wcześniej bloku danych

```
//Tries to read an entire 16-byte data block at the specified block address.

uint8_t pn532ReadBlock(uint8_t *Data, uint8_t blockNumber){
    uint8_t Result;
    //Prepare read command
    pn532PacketBuffer[0] =PN532_INDATAEXCHANGE;  //Data Exchange Command
    pn532PacketBuffer[1] = 1;  //Max card numbers
    pn532PacketBuffer[2] = PN532_MIFARE_READ;  //Mifare read command
    pn532PacketBuffer[3] = blockNumber;  //Read block number (0..63 for 1KB cards)

    if((Result = pn532SendCommandCheckAck(pn532PacketBuffer, 4)) != NO_ERROR) return Result;

    //Read the response packet
    pn532ReadData(pn532PacketBuffer, 26);

    //Check if there were no errors (Status byte == 0)
    if(pn532PacketBuffer[7] != 0x00) return READ_BLOCK_ERROR;
    //Copy the 16 data bytes to the output buffer
    memcpy(Data, &pn532PacketBuffer[8], 16);

    return NO_ERROR;
}
```

Listing 4. Definicje błędów funkcji narzędziowych obsługi kart Mifare

```
#define NO_ERROR 0x00
#define ACKNOWLEDGE_FRAME_ERROR 0x01
#define STATUS_TIMEOUT 0x02
#define FIRMWARE_HEADER_ERROR 0x03
#define RFCONFIGURATION_ERROR 0x04
#define SAMCONFIGURATION_ERROR 0x05
#define NO_TAGS_FOUND 0x06
#define AUTHENTICATION_ERROR 0x07
#define READ_BLOCK_ERROR 0x08
```

Listing 5. Funkcja pozwalająca na odczyt numeru seryjnego karty

```
uint8_t cardReadUID(uint8_t *UID){
    uint8_t Result;
    Result =
pn532ReadPassiveTargetID(UID);
    return Result;
}
```

Listing 6. Funkcja pozwalająca na przeprowadzenie procesu uwierzytelniania karty

```
uint8_t cardAuthenticate(uint8_t *UID, uint8_t *Key){
    uint8_t Result;
    Result = pn532AuthenticateBlock(UID, Key, SECTOR1_TRAILER);
    return Result;
}
```

Listing 7. Funkcja pozwalająca na zapis klucza dostępu A na karcie

```
uint8_t defaultKey[6] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
uint8_t defaultAccessBits[4] = {0xFF, 0x07, 0x80, 0x69};
uint8_t cardWriteKey(uint8_t *Key){
    uint8_t Result;
    uint8_t trailerBlock[16];

    memcpy(trailerBlock, Key, 6);  //Key A: bytes 0...5
    memcpy(&trailerBlock[6], defaultAccessBits, 4);  //Default access bits: bytes 6...9
    memcpy(&trailerBlock[10], defaultKey, 6);  //Default Key B: bytes 10...15

    Result = pn532WriteBlock(trailerBlock, SECTOR1_TRAILER);
    return Result;
}
```

Ustawienia Fuse-bitów (ważniejszych):
CKSEL3...0: 0010
SUT1...0: 10
CKDIV8: 1
EESAVE: 0

Program sterujący

Wybaczcie, ale nie będę w tym miejscu powielał informacji z artykułu o NFClock (EP 4/22) tylko przedstawię nowe, niezbędne z punktu widzenia autoryzacji, funkcje narzędziowe. Pierwsza z nich to funkcja autoryzująca dostęp danych do wybranego sektora karty, której to ciało pokazano na **listingu 1**. Funkcja jako argument przyjmuje numer seryjny autoryzowanej karty (*UID*, 4 lub 7 bajtów), który zawiera również pole mówiące o jego rozmiarze (*UID[7]*), Klucz dostępu A (*Key*) oraz numer bloku danych, do którego dostęp chcemy uzyskać (*blockNumber*). Funkcja zwraca 0 (*NO_ERROR*) w przypadku udanej autoryzacji, lub kod błędu.

Kolejną istotną funkcją jest funkcja pozwalająca na zapis autoryzowanego wcześniej bloku danych (w tym bloku Sector Trailer przechowującego klucze itd). Jej ciało pokazano na **listingu 2**. Funkcja ta jako argument przyjmuje wskaźnik na dane (16 bajtów) przeznaczone do zapisu oraz numer bloku danych, do którego zapis będziemy realizować. Funkcja zwraca 0 (*NO_ERROR*) w przypadku powodzenia, lub kod błędu. Na koniec funkcja pozwalająca na odczyt autoryzowanego wcześniej bloku danych (w tym bloku Sector Trailer przechowującego klucze itd). Jej ciało pokazano na **listingu 3**.

Funkcja ta jako argument przyjmuje wskaźnik na tablicę (16 bajtów), do której zapisane zostaną odczytane dane oraz numer bloku danych, z którego odczyt będziemy realizować. Funkcja zwraca 0 (*NO_ERROR*) w przypadku powodzenia, lub kod błędu. Na koniec przypomnę specyfikację potencjalnych błędów, których definicje znalazły się w pliku nagłówkowym, a które pokazano na **listingu 4**.

Korzystając z powyższych funkcji narzędziowych napisałem 3 inne, bardzo proste funkcje umożliwiające przeprowadzenie podstawowych operacji na karcie RFID/NFC a wyłączną ideą ich powstania była chęć

WYKAZ ELEMENTÓW, które możesz zamówić w sklepie AVT na stronie sklep.avt.pl lub bezpośrednio (ul. Leszczynowa 11, 03-197 Warszawa, tel. 48222578451, e-mail: handlowy@avt.pl):

Rezystory: (obudowa miniaturowa 1/8 W, raster 0,2")
R1: 47 kΩ
R2, R5: 1 kΩ
R3, R4: 2,2 kΩ
R6: 22 Ω

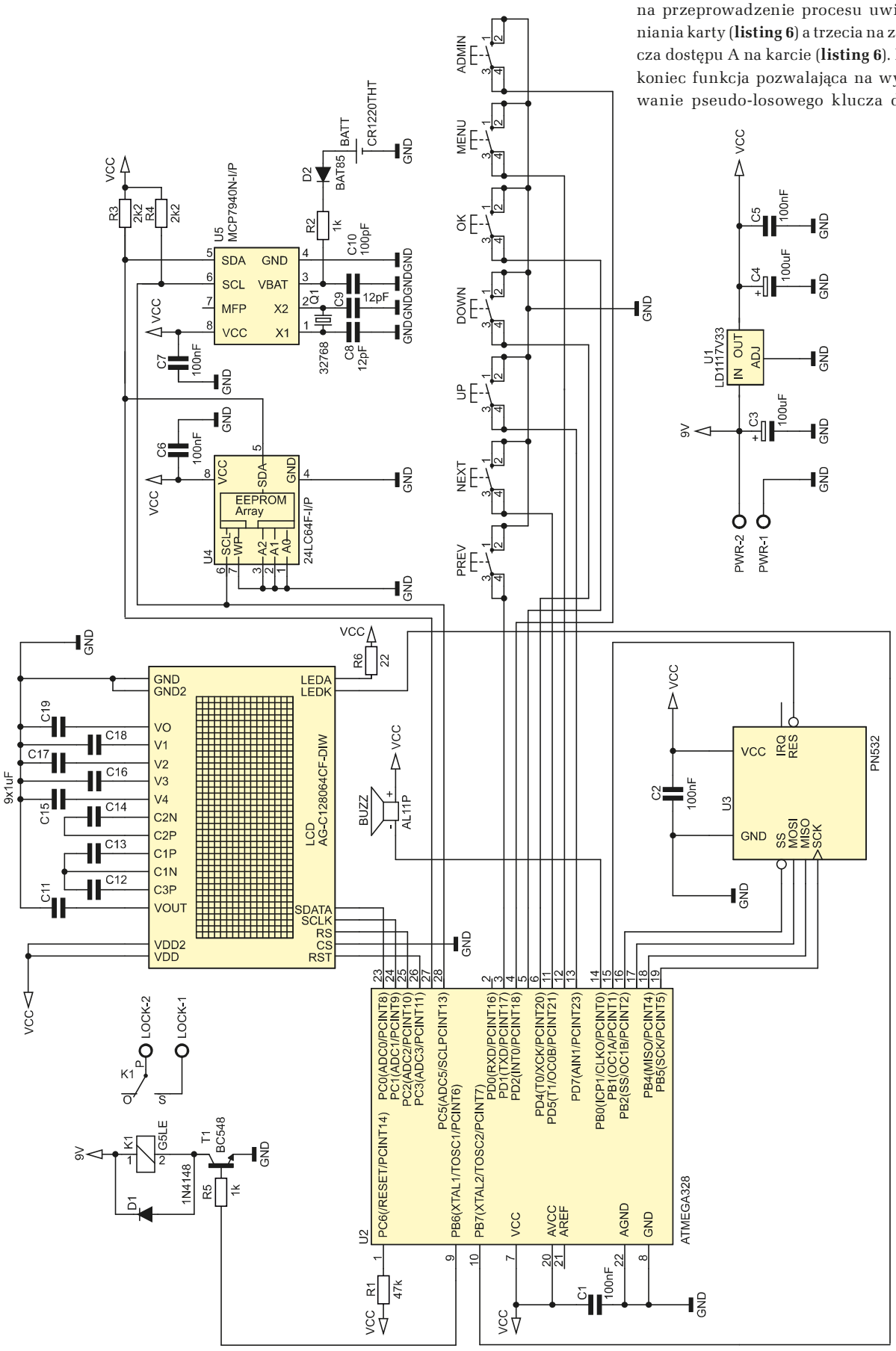
Kondensatory:
C1, C2, C5...C7: ceramiczny MLCC 100 nF (raster 0,1")
C3, C4: elektrolityczny 100 µF/16 V (raster 0,1")
C8, C9: ceramiczny MLCC 12 pF (raster 0,1")
C10: ceramiczny MLCC 100 pF (raster 0,1")
C11...C19: ceramiczny MLCC 1 µF (raster 0,2")

Półprzewodniki:
D1: 1N4148 (DO-34-7)
D2: BAT85 (DO-34-7)
T1: BC548 (TO-92)
U1: LD1117V33 (TO-220)
U2: ATMEGA328 (DIL28)
U3: moduł RFID/NFC typu PN532
U4: 24LC64F-I/P (DIL8)
U5: MCP7940N-I/P (DIL8)
LCD: wyświetlacz graficzny graficzny LCD-AG-C128064CF-DIW

Pozostałe:
K1: przekaźnik G5LE-14-9
Q1: rezonator kwarcowy zegarkowy 32768 Hz
BUZZ: buzzer piezoelektryczny z generatorem 5 V
PREV, NEXT, UP, DOWN, OK, MENU: microswitch TACT z ośką 16 mm do montażu przewlekanego
ADMIN: microswitch TACT z ośką 1 mm do montażu przewlekanego
PWR, LOCK: złącze śrubowe AK500/2 (raster 0,1")
BATT: gniazdo baterii CR1220 THT typu DS1092-15-B6P-C CONNFLY
VCC: bateria CR1220

```
Listing 8. Funkcja pozwalająca na wygenerowanie pseudo-losowego klucza dostępu

void cardGenerateKey(uint8_t *Key){
    for(uint8_t i=0; i<6; i++) Key[i] = rand() & 0xFF;
}
```

uproszczenia i zwiększenia czytelności kodu źródłowego. Pierwsza z nich pozwala na odczyt numeru seryjnego karty i pokazano ją na **listingu 5**, druga pozwala na przeprowadzenie procesu uwierzytelniania karty (**listing 6**) a trzecia na zapis klucza dostępu A na karcie (**listing 6**). I na sam koniec funkcja pozwalająca na wygenerowanie pseudo-losowego klucza dostępu,



Rysunek 3. Schemat ideowy urządzenia SAS2

której to ciało pokazano na **listingu 8**. Tyle na tą chwilę w kwestiach implementacyjnych, w związku z czym przejdźmy do schematu naszego urządzenia.

Budowa i działanie

Schemat ideowy urządzenia SAS2 pokazano na **rysunku 3**. Zbudowano dość prosty system mikroprocesorowy, którego „sercem" jest nowoczesny mikrokontroler ATmega328 taktowany wewnętrznym, wysokostabilnym generatorem RC o częstotliwości 8 MHz, który przy udziale sprzętowego interfejsu TWI (odpowiednik I²C) zapewnia obsługę pamięci EEPROM (U4) oraz obsługę układu MCP7940N-I/P (U5) realizującego funkcjonalność dokładnego zegara czasu rzeczywistego (RTC) wyposażonego w zintegrowany, automatyczny mechanizm podtrzymania zasilania (co zapewnia bateria CR1220).

Mikrokontroler jest odpowiedzialny również za obsługę interfejsu użytkownika zbudowanego z użyciem kilku przycisków typu microswitch, buzzera piezoelektrycznego i niedrogiego, graficznego wyświetlacza LCD COG o rozdzielczości 128×64 piksele, którego obsługa odbywa się z zastosowaniem programowej realizacji interfejsu SPI. Wybór tego, konkretnego typu mikrokontrolera nie był bynajmniej podyktowany niezbędną pojemnością pamięci Flash, gdyż program obsługi aplikacji napisany przy pomocy AVR-GCC zajmuje około 11 kB tej pamięci. Pokaźna część przypada na dwujęzykowe teksty interfejsu użytkownika zapisane w pamięci Flash oraz definicje czcionki ekranowej 5×7 pikseli. Wybór ten wynika wyłącznie z potrzeby dobrania mikrokontrolera o odpowiedniej liczbie wyprowadzeń niezbędnej z uwagi na rodzaj zastosowanego interfejsu użytkownika i liczbę obsługiwanych peryferiów. Nie mniej ważnym elementem przy wyborze zastosowanego typu mikrokontrolera był także wymóg wielkości wbudowanej pamięci EEPROM, która to w naszym wypadku została określona na poziomie 1024 bajtów. Można by, co prawda, zastosować dodatkową pamięć zewnętrzną lecz zastosowane rozwiązanie wydawało mi się optymalne, jeśli chodzi o możliwości urządzenia i cenę.

W implementacji programu obsługi aplikacji nie mogło również zabraknąć miejsca na realizację obsługi modułu RFID/NFC pod postacią peryferium PN532, która to odbywa się z zastosowaniem sprzętowego interfejsu SPI mikrokontrolera. Uważny czytelnik zauważy zapewne, iż zarówno wyświetlacz COG, jak i moduł RFID wymagają interfejsu SPI, zatem dlaczego nie zastosowano do ich obsługi jednego interfejsu sprzętowego mikrokontrolera? To dość proste. Po pierwsze oba peryferia mają inną prędkość, jak i kolejność bitów transmisji (LSB czy MSB), w związku z czym nie chciałem co chwilę zmieniać ustawień sprzętu a po drugie, i dość prozaiczne, łatwiej było mi zaprojektować płytkę drukowaną realizując ww. obsługę za pomocą różnych wyprowadzeń mikrokontrolera. Inną sprawą jest, że dobrze było rozdzielić magistralę krytyczną (PN532) od tej mniej wymagającej (LCD COG). Tyle w kwestiach budowy urządzenia SAS2.

Robert Wołgajew
robert.wolgajew@ep.com.pl

Podstawowe parametry:

- maksymalna liczba użytkowników: 40 (+ administrator),
- maksymalna liczba zdarzeń: 1000,
- czas automatycznego wylogowania Administratora: 60 s,
- czas automatycznego wygaszenia podświetlenia: 40 s bezczynności użytkownika,
- czas automatycznego wyjścia do menu głównego: 30 s bezczynności użytkownika,
- maksymalny prąd styków przekaźnika wykonawczego: 1 A/220 VAC (szczegóły w dokumentacji elementu),
- napięcie zasilania: 9 VDC,
- maksymalny pobór prądu (przekaźnik wyłączony/załączony): 90/110 mA.

Dodatkowe materiały do pobrania ze strony www.ulubionykiosk.pl/media

----	NFC Lock (EP 4/2022)
AVT5186	Bezstykowy zamek RFID (EP 5/2009)
AVT969	Bezstykowy zamek RFID (EP 2/2007)
AVT3129	Zamek elektroniczny/immobilizer (EdW 7/2015)
AVT886	System bezstykowej kontroli dostępu (EP 10/2000)

W ofercie AVT*

AVT5934

* 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 stronie sklepu osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt Via e-mail: kity@avt.pl.

SAS2

Simple Access System 2 (2)

Zaprezentowany projekt to rozbudowany system kontroli dostępu z pełną rejestracją zdarzeń i użytkowników współpracujący ze znacznikami RFID oraz NFC. W pierwszej części artykułu (EP5/22) zostały opisane zagadnienia związane z budową i działaniem kart Mifare oraz została opisana konstrukcja elektryczna projektu. Były to informacje niezbędne do zrozumienia działania opisanego urządzenia oraz niezwykle przydatne w przypadku własnej implementacji podobnego rozwiązania. W drugiej i ostatniej części zawarty jest dokładny opis funkcjonalności oraz obsługi systemu SAS2.

Funkcjonowanie systemu

Przejdźmy zatem do ciekawych zagadnień dotyczących funkcjonowania systemu, zwłaszcza w zakresie autoryzacji kart i użytkowników. Po pierwsze w zakresie interfejsu użytkownika zdefiniowano szereg zasad:

- zmiana ustawień zegara czasu rzeczywistego RTC, edycja nazwy oraz usuwanie zdefiniowanego wcześniej użytkownika możliwe jest wyłącznie po zalogowaniu karty Administratora;
- usuwanie zdefiniowanych wcześniej użytkowników możliwe jest wyłącznie wtedy, gdy na liście zdarzeń nie zanotowano żadnego zdarzenia z identyfikatorem usuwanego użytkownika;
- nie jest możliwe usuwanie zdarzeń z listy zdarzeń, zaś sama lista ma charakter kołowy, tzn. gdy wyczerpana zostanie maksymalna liczba zarejestrowanych zdarzeń (1000), nowo rejestrowane zdarzenia zastępują zdarzenia najstarsze;
- rejestrowane są również zdarzenia zalogowania/wylogowania Administratora jak i zdarzenia prób logowania niezarejestrowanych użytkowników (używających niezarejestrowanych kart);
- wbudowano mechanizm samowylogowywania się Administratora po czasie 60 sekund, co zapobiega nieautoryzowanym zmianom, w przypadku niewylogowania się Administratora;
- wbudowano mechanizm automatycznego powrotu do ekranu głównego urządzenia po czasie 30 sekund bezczynności ze strony użytkownika;
- dodawanie nowego użytkownika poprzedzone jest każdorazowym sprawdzeniem obecności numeru seryjnego nowej karty na liście użytkowników;
- w procesie dodawania nowego użytkownika przyznawany jest automatycznie najniższy, wolny identyfikator ID z listy użytkowników zachowanej w pamięci EEPROM mikrokontrolera;
- obsługa logowania kart dostępna jest jedynie z Menu głównego urządzenia. Każda inna pozycja Menu automatycznie uniemożliwia sprawdzanie zbliżanych kart RFID;
- wbudowano mechanizm wygaszania podświetlenia wyświetlacza graficznego po czasie 30 sekund bezczynności klawiatury w celu ograniczenia poboru mocy.

W związku z powyższym aplikacja programu obsługi wyświetla dodatkowe komunikaty dotyczące następujących zdarzeń:

- braku uprawnień do zmiany nastaw przeznaczonych wyłącznie dla Administratora;
- istnieniu użytkownika na liście użytkowników o numerze karty, która ma zostać dodana jako nowa;
- osiągnięciu maksymalnej, dopuszczalnej liczby (40) użytkowników na liście zdefiniowanych użytkowników;

- braku możliwości usunięcia użytkownika z listy użytkowników w przypadku istnienia identyfikatora tegoż użytkownika na liście zdarzeń;
- zalogowaniu i wylogowaniu (w tym automatycznym wylogowaniu) klucza administratora;
- wyczyszczeniu listy zdarzeń lub zapamiętaniu klucza administratora (wyłącznie podczas włączania urządzenia);
- braku zdarzeń na liście zdarzeń;
- braku użytkowników na liście użytkowników.

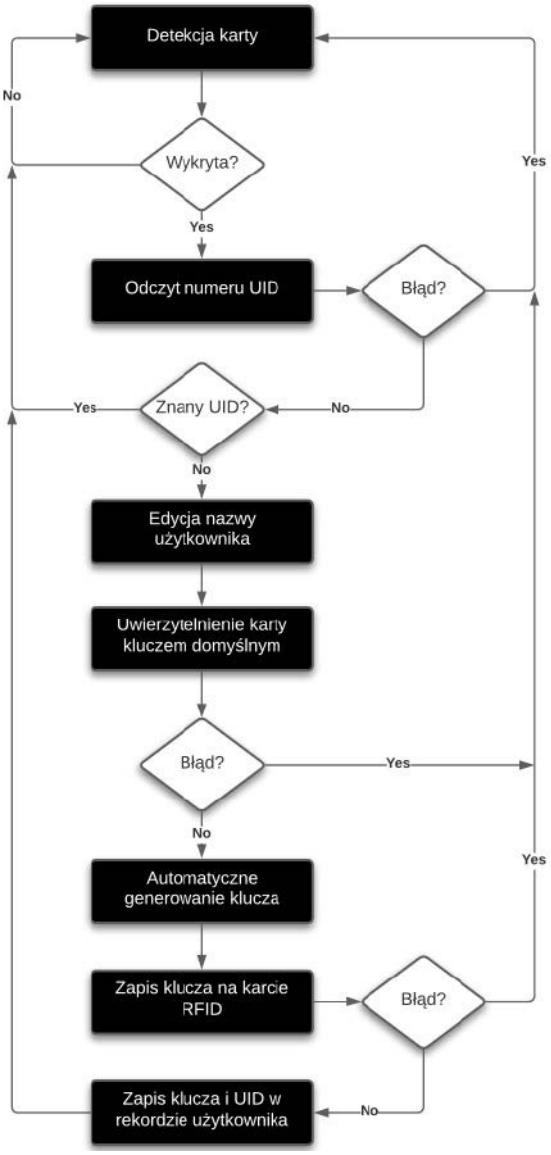
Listing 9. Fragment kodu aplikacji, który odpowiedzialny za proces dodawania użytkownika

```
//Czekamy na przyłożenie karty
while(cardReadUID(UID) != NO_ERROR){

}
//Uwierzytelniamy kartę kluczem defaultowym i odczytanym wcześniej numerem UID
//Na tym etapie znamy już User.userID, User.userName i User.UID
if(cardAuthenticate(User.UID, defaultKey) == NO_ERROR){
    //Generujemy klucz
    cardGenerateKey(User.Key);
    //Zapisujemy nowy klucz na karcie
    if(cardWriteKey(User.Key) == NO_ERROR){
        //Zapisujemy nowego użytkownika (wraz z danymi klucza)
        eeprom_write_block(&User, &UserEE[User.userID], sizeof(User));
        usersNumber++;  //Zwiększenie liczby użytkowników
    }
}
```

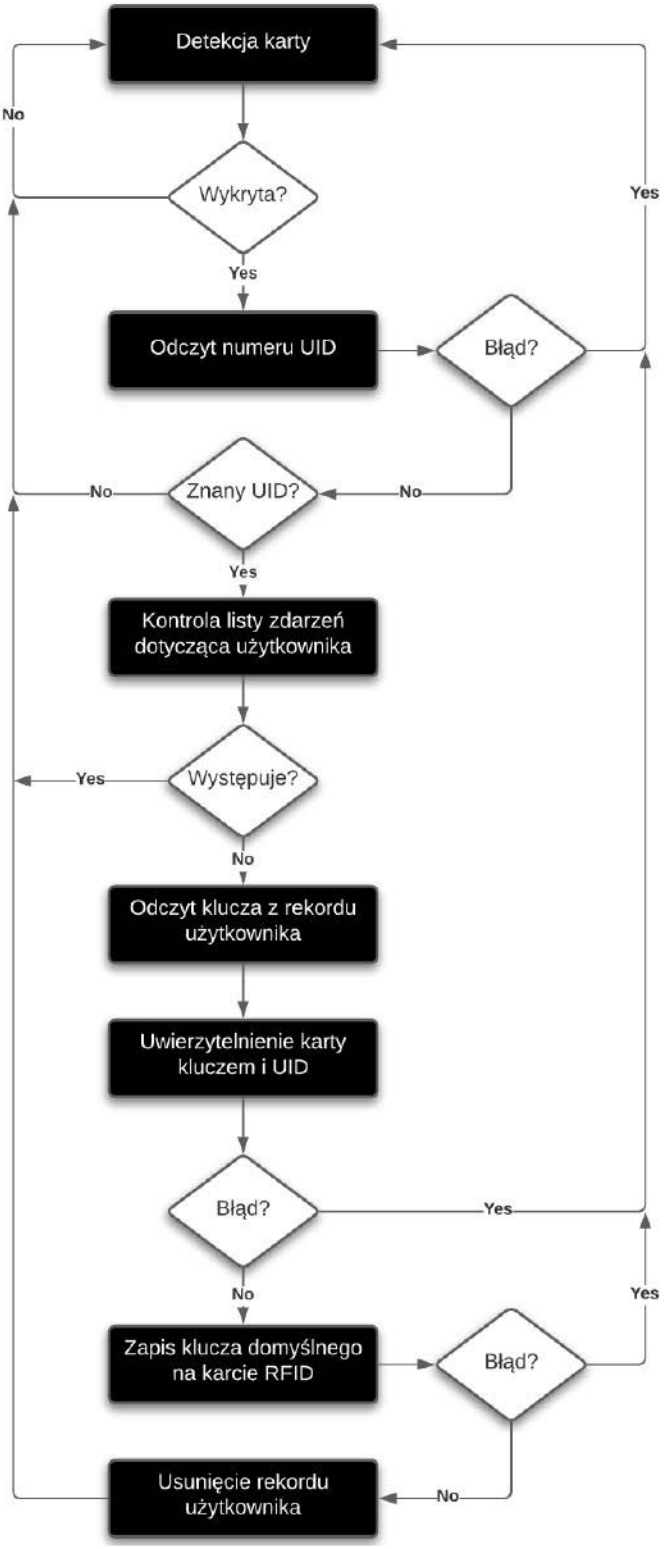
W tym miejscu przejdźmy zatem do szczegółów. Program obsługi aplikacji urządzenia SAS2 realizuje kilka głównych procesów, jeśli chodzi o obsługę kart RFID użytkowników. Są to następujące procesy:

- dodawanie nowej karty, czyli poniekąd dodawanie nowego użytkownika,
- usuwanie karty użytkownika,
- dodawanie karty Administratora,
- usuwanie karty Administratora,
- obsługa kart w procesie logowania.

Jako, że obsługa karty Administratora przebiega w bardzo zbliżony sposób, jak obsługa kart zwykłego użytkownika (poza brakiem konieczności nadawania nazwy użytkownika), tyle tylko że możliwa



Rysunek 4. Graf funkcjonalny procesu dodawania karty użytkownika



Rysunek 5. Graf funkcjonalny procesu usuwania karty użytkownika

Listing 10. Fragment kodu aplikacji, który odpowiedzialny jest za proces usuwania użytkownika

```
//Sprawdzamy czy użytkownik, którego chcemy właśnie usunąć nie znajduje się na liście Event,
//bo jeśli się znajduje (czyli się logował) to nie możemy go usunąć – takie przyjęliśmy założenia
commonPos = 0;  //Wskaźnik, że można usunąć użytkownika – zmienna tymczasowa

//Wczytujemy dane tego Usera kierując się numerem relatywnym i jednocześnie zapamiętujemy numer bezwzględny
//w tablicy Userów, którego dotyczy ten numer relatywny, by go później wykasować w odpowiednim miejscu
absoluteUserNumber = readRelativeUserNr(userListStartPos+userHighlightPos);

for(uint16_t idx=0; idx<eventsNumber; ++idx){
    FRAMreadBlock(idx, &Event);
    if(Event.userID == User.userID) {commonPos = 1; break;}  //Znaleziono go na liście
}

if(!commonPos){
    //Czekamy na przyłożenie karty lub przycisk DOWN, który usuwa użytkownika bez obecności karty
    while(cardReadUID(UID) != NO_ERROR){
        if(KeyPressed(&DOWN_PIN, DOWN_KEY, &keyDown, 1) == LONG) goto DELETE_USER;
    }
    //Sprawdzamy czy wczytana karta należy do użytkownika, którego chcemy usunąć
    if(memcmp(UID, User.UID, 8) == 0){
        //Uwierzytelniamy kartę kluczem i numerem UID
        if(cardAuthenticate(User.UID, User.Key) == NO_ERROR){
            //Zapisujemy na karcie defaultowy klucz uwalniając ją z obsługiwanych kart
            if(cardWriteKey(defaultKey) == NO_ERROR){
                DELETE_USER:
                //Użytkownika kasujemy pod jego bezwzględnym adresem w tabeli
                //User – zapamiętany powyżej
                User.userID = EMPTY_USER;
                eeprom_write_block(&User, &UserEE[absoluteUserNumber], sizeof(User));
                usersNumber--;  //Zmniejszamy liczbę użytkowników
            }
        }
    }
}
```

jest wyłącznie podczas włączania urządzenia (i przyciśnięcia przycisku „ADMIN"), nie będę jej opisywał oddzielnie, zaś skupię się na pozostałych procesach wskazując dla jasności odpowiednie grafy funkcjonalne.

Zacznijmy od procesu obejmującego dodawanie karty użytkownika, którego to graf pokazano na **rysunku 4**. Jak widać, możliwe jest wyłącznie dodanie nowej, jeszcze nieobsługiwanej karty RFID. Cały proces rozpoczyna się od przyłożenia karty RFID/urządzenia NFC do czytnika. Następnie urządzenie przechodzi do nadania nazwy użytkownika (za pomocą interfejsu GUI), po czym karta autoryzowana jest kluczem domyślnym. Następnie generowany jest przez system SAS2 unikalny klucz dostępowy karty RFID (klucz A), który zapisywany jest zarówno w pamięci EEPROM mikrokontrolera jak i pamięci EEPROM karty RFID. Dalej proces podlega finalizacji poprzez dodanie rekordu nowego użytkownika do listy obsługiwanych użytkowników. Warto podkreślić, i co nasuwa się z lektury grafu funkcjonalnego, iż obecność karty w przypadku tego procesu niezbędna jest dwukrotnie, a mianowicie podczas pierwszego odczytu numeru seryjnego karty (UID) oraz drugi raz po przeprowadzeniu edycji nazwy użytkownika i zapisu kluczy dostępu. Fragment kodu aplikacji, który odpowiedzialny za proces dodawania użytkownika pokazano na **listingu 9**.

Kolejny proces to proces usuwania użytkownika z listy użytkowników, którego graf pokazano na **rysunku 5**. Jak widać możliwe jest usunięcie wyłącznie tych kart użytkowników, dla których nie zarejestrowano zdarzeń w dzienniku zdarzeń, zaś do przeprowadzenia samego procesu, co oczywiste, niezbędna jest obecność karty, którą chcemy usunąć. Karta taka w pierwszej kolejności autoryzowana jest kluczem odczytanym z rekordu użytkownika o podanym numerze seryjnym (UID), następnie klucz dostępowy na karcie aktualizowany jest wartością domyślną (jak dla fabrycznie nowej karty) a rekord użytkownika usunięty z listy obsługiwanych użytkowników. Nasuwa się pytanie, czy nie wystarczyłoby usunięcie samego rekordu użytkownika bez operacji na karcie RFID? Dla samego systemu SAS2 taka operacja byłaby z pewnością wystarczająca, jednak w takim wypadku na karcie RFID pozostałby unikalny klucz, który zostałby usunięty z pamięci EEPROM mikrokontrolera. Jak mielibyśmy ponownie autoryzować (dodać) taką kartę bez znajomości klucza, którego z niej samej nie można odczytać? Stałaby się ona bezużyteczna dla naszego systemu kontroli dostępu (co nie znaczy, że dla innych urządzeń też), dlatego trzeba ją odpowiednio przygotować.

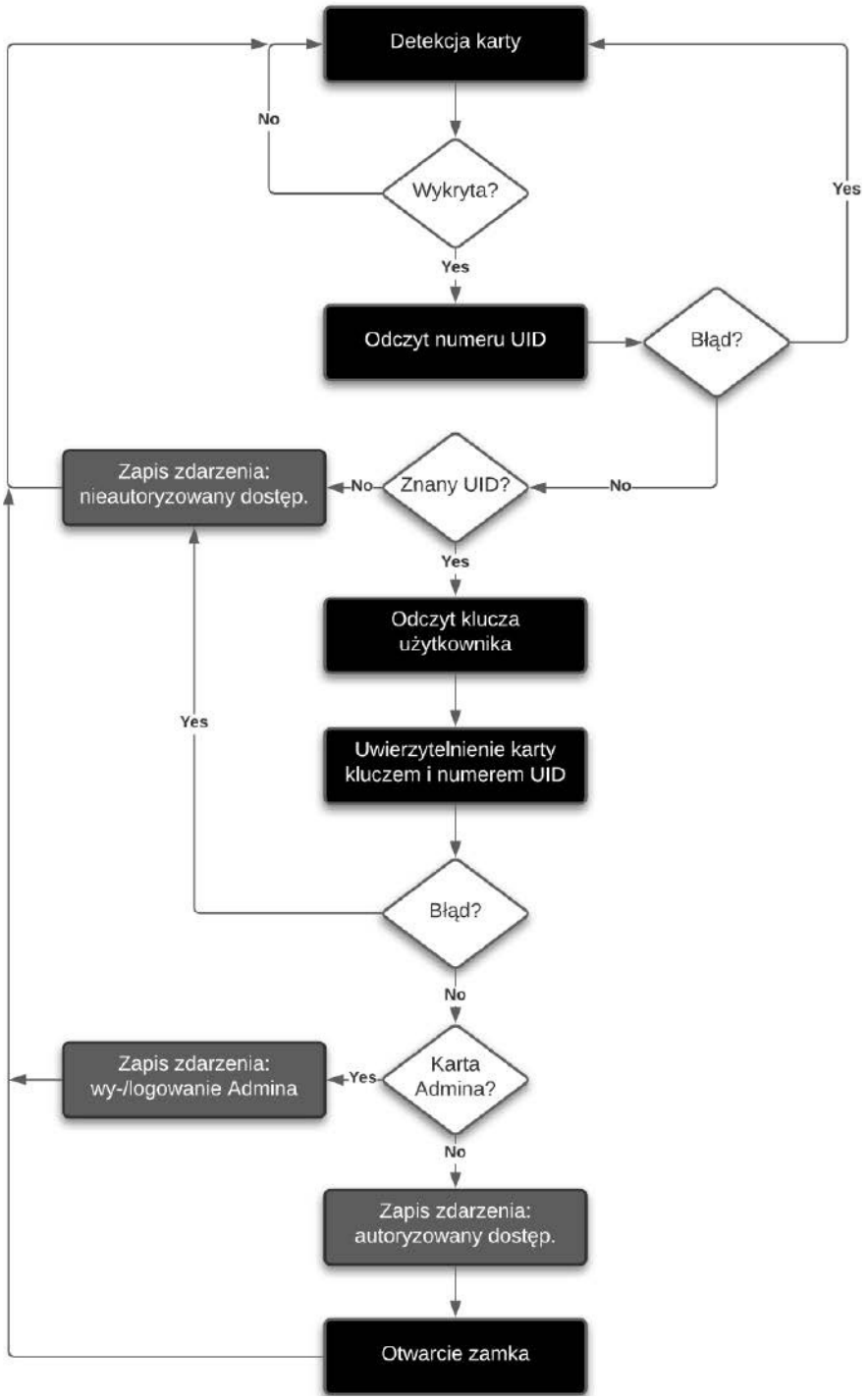
Uważny Czytelnik zada sobie z pewnością pytanie, co w przypadku zagubienia karty RFID znanego użytkownika? Czy już nigdy nie będziemy mogli usunąć go z listy użytkowników, skoro nie mamy karty? Przewidziałem taką możliwość i Menu systemu SAS2 udostępnia opcję usunięcia użytkownika (i danych jego karty) bez posiadania karty RFID, pod warunkiem, jak zawsze, iż usuwany użytkownik nie widnieje na liście zdarzeń systemu. Aby tego dokonać należy podczas oczekiwania na zbliżenie karty RFID nacisnąć i przytrzymać przycisk DOWN. Fragment kodu aplikacji, który odpowiedzialny jest za proces usuwania użytkownika pokazano na **listingu 10**.

Funkcja powyższa korzysta z dość ciekawej funkcji narzędziowej o nazwie *readRelativeUserNr()*, której zadaniem jest odczyt danych użytkownika o relatywnym numerze w tablicy użytkowników czyli funkcja ta pomija wszystkich nieaktywnych użytkowników pomiędzy istniejącymi i zwraca bezwzględny numer elementu tablicy, w którym znajduje się względny numer użytkownika. Ciało wspomnianej funkcji pokazano na **listingu 11**.

Ostatni proces, o którym należy wspomnieć to proces „normalnej" obsługi karty realizowany na ekranie głównym urządzenia, który ma na celu rejestrację zdarzeń z użyciem kart RFID oraz otwieranie rygla zamka podłączonego do złącza LOCK urządzenia. Graf tego procesu pokazano na **rysunku 6**. Proces obsługi karty użytkownika rozpoczyna się z chwilą zbliżenia karty do czytnika i powoduje odczytanie jej numeru seryjnego UID. Jeśli numer UID jest dla naszego systemu nieznany to na liście zdarzeń urządzenia SAS2 zapisywane jest zdarzenie (ze znacznikiem czasowym) nieautoryzowanego dostępu do systemu. W innym przypadku proces przechodzi do autoryzacji bieżącej karty kluczem dostępu odczytanym z rekordu użytkownika (ze znanym numerem UID). Nieudanej autoryzacji

Listing 11. Kod funkcji readRelativeUserNr()

```
uint8_t readRelativeUserNr(uint8_t relativeUserNr){
    uint8_t idx, userCounter = 0;
    for(idx=0; idx<MAX_USERS; ++idx){
        eeprom_read_block(&User, &UserEE[idx], sizeof(User));
        if(User.userID != EMPTY_USER){
            userCounter++;
            if((userCounter-1) == relativeUserNr) break; else User.userID = EMPTY_USER;
        }
    }
    return idx;
}
```



Rysunek 6. Graf funkcjonalny procesu obsługi kart użytkowników

towarzyszy, jak poprzednio, dodanie zdarzenia nieautoryzowanego dostępu do systemu, w przeciwnym razie proces przechodzi do następnego kroku, w którym sprawdzane jest czy autoryzowana karta nie jest przypadkiem kartą Administratora systemu. Jeśli jest kartą Administratora to stosowne zdarzenie (wylogowania/zalogowania Administratora) zapisywane jest w dzienniku zdarzeń, zaś system przechodzi w tryb uprawnień Administratora (możliwe jest wykonanie dodatkowych operacji dostępnych wyłącznie dla Administratora). Jeśli autoryzowany użytkownik jest z kolei „zwykłym" użytkownikiem to fakt jego logowania zapisywany jest, jak poprzednio, w dzienniku zdarzeń po czym rygiel zamka podłączonego do złącza LOCK uruchamiany jest na czas 1,5 sekundy.

Kilka słów uwagi na temat dodawania/usuwania karty Administratora. Tak, jak wspomniano na wstępie, proces dodawania czy usuwania karty Administratora możemy zainicjować wyłącznie podczas włączania urządzenia przy naciśniętym przycisku ADMIN. Jeśli w urządzeniu nie zapisano jeszcze karty Administratora i zbliżona karta będzie nową kartą dla systemu SAS2 to zostanie ona zapamiętana, jako karta Administratora. Aby ją usunąć należy ponownie zbliżyć ją do czytnika podczas włączania urządzenia z wciśniętym, jak poprzednio, przyciskiem ADMIN, co spowoduje odpowiednie jest „sformatowanie" i możliwość ponownego użycia w systemie SAS2 (jako karty Administratora lub użytkownika). Fragment kodu aplikacji, który odpowiedzialny jest za proces obsługi karty pokazano na **listingu 12**.

Kilka słów uwagi należy się potencjalnej możliwości obsługi urządzeń NFC w rodzaju telefonu komórkowego. Jak wiadomo niektóre z tych urządzeń wyposażono w interfejs NFC umożliwiający wymianę danych lub obsługę płatności zbliżeniowych. Czy nasz system będzie w stanie odczytać numer seryjny takiego urządzenia? Oczywiście, że tak, bo musi ono spełniać standardy protokołu NFC. Problem jednak w tym, że systemy operacyjne telefonów komórkowych generują losowe numery seryjne (tzw. RID – *Random ID*), w związku z czym ich zapamiętywanie nie ma większego sensu, gdyż podczas kolejnego połączenia z takim urządzeniem wygeneruje ono zupełnie inny numer seryjny. Zachowanie to można zmienić, ale przynajmniej w Androidzie nie jest to takie proste i wymaga modyfikacji firmware lub użycia specjalnych aplikacji. Ponoć zdarzają się telefony z statycznym numerem UID (takie informacje odnalazłem na specjalistycznych forach), lecz ja na takie nie natrafiłem, więc należy założyć, iż standardowo generują one losowe numery RID.

Obsługa urządzenia

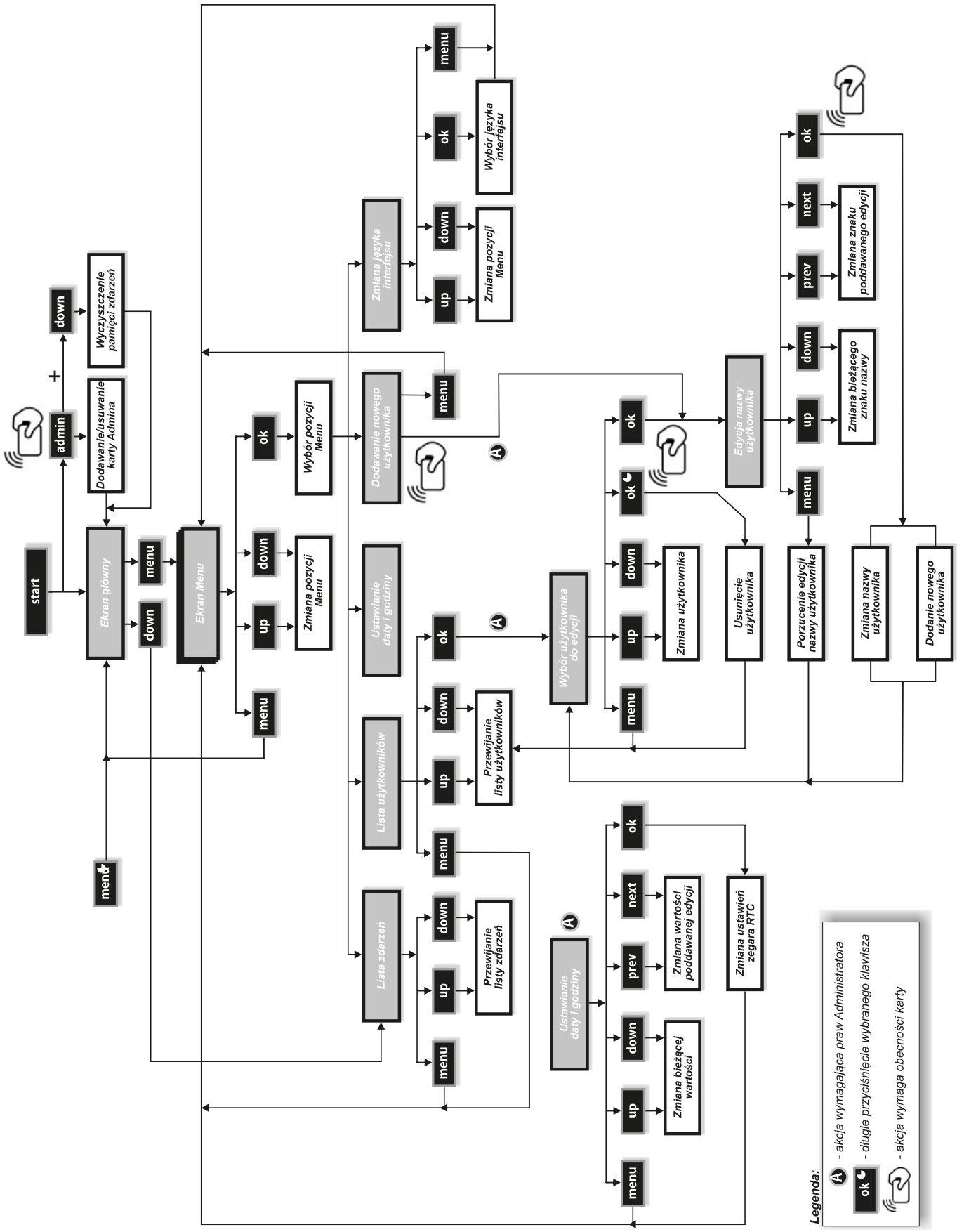
Tyle o procesach, przejdźmy zatem do Menu samego urządzenia i sposobu jego obsługi. Projektując Menu systemu SAS2 oraz sposób obsługi tego urządzenia przyjąłem, że ergonomia i prostota jego użytkowania jak i czytelność interfejsu użytkownika powinna być najważniejszym kryterium przy konstruowaniu stosownych procedur sterujących. Zgodnie z tymi podstawowymi założeniami, na płytce sterownika przewidziano aż 7 przycisków sterujących dających bezpośredni dostęp do podstawowej funkcjonalności. Jako że Menu obsługi urządzenia udostępnia wiele funkcji, stosowne przyciski mają różnoraką funkcjonalność zależną od miejsca w układzie Menu, przy czym ich podstawowe znaczenie przedstawia się następująco:

- przyciski oznaczone jako **NEXT**, **PREV** służą do zmiany pozycji edytowanego elementu;
- przyciski oznaczone jako **UP**, **DOWN** służą do zmiany wartości edytowanego elementu oraz poruszania się po kolejnych opcjach Menu urządzenia;
- przycisk oznaczony jako **OK** służy do zatwierdzania wyboru zarówno w zakresie opcji Menu jak i zmiany parametrów poddawanych edycji;
- przycisk oznaczony jako **MENU** służy do wejścia w system Menu jak i wyjścia do Menu nadrzędnego bez zmiany parametrów edytowanego elementu (w przypadku Menu umożliwiającego edycję).

Dodatkowo, długie przytrzymanie tego przycisku powoduje każdorazowo wyjście do ekranu głównego aplikacji;

- przycisk oznaczony jako **ADMIN** służy do realizacji funkcji administracyjnych: kasowania całej pamięci zdarzeń i wejścia w tryb zapamiętywania/usuwania klucza Administratora.

Wszystkie wymienione operacje możliwe są do wykonania wyłącznie podczas włączania urządzenia, zaś sam przycisk **ADMIN** z założenia nie powinien być dostępny na panelu obsługi urządzenia (stąd też zastosowano dla niego przełącznik microswitch z bardzo krótką ośką). Jako



Rysunek 7. Diagram prezentujący kompletny algorytm obsługi systemu Menu urządzenia SAS2

że lista dostępnych opcji systemu Menu urządzenia SAS2 jest dość obszerna, na **rysunku 7** pokazano diagram prezentujący kompletny algorytm obsługi, zaś na **rysunku 8** pokazano wygląd ekranu interfejsu użytkownika dla głównych trybów pracy systemu Menu. Warto zauważyć, iż w przypadku edycji nazwy użytkownika w ramach procesu dodawania nowego użytkownika długie przyciśnięcie przycisków UP/DOWN powoduje szybkie zmiany edytowanych liter, co ma ułatwić stosowną edycję.

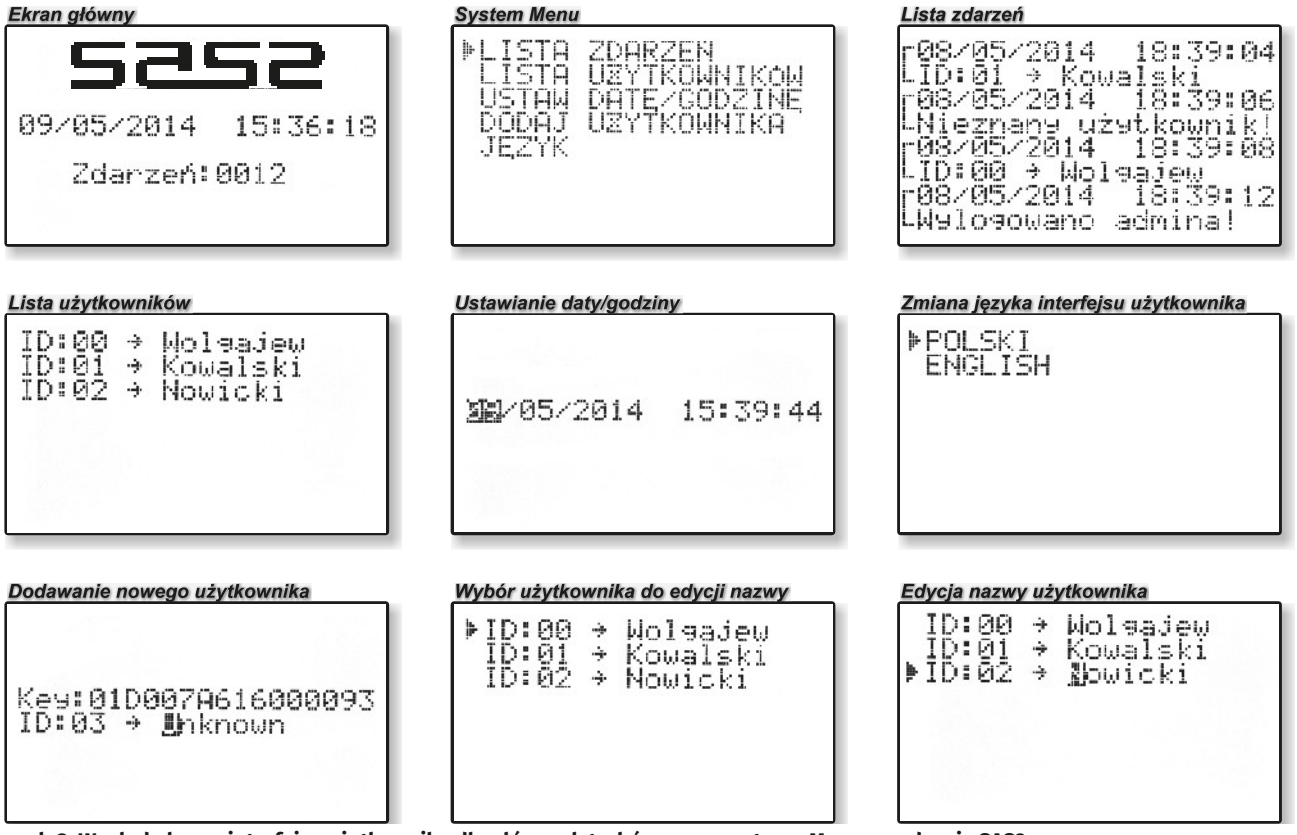
Montaż i uruchomienie

Schemat płytki PCB systemu SAS2 pokazano na **rysunku 9**. Jak widać, zaprojektowano bardzo zwartą konstrukcję obwodu drukowanego z zastosowaniem elementów THT, po to, by całe urządzenie wymiarami nie przekraczało niezbędnego, minimalnego obszaru dla wykonania interfejsu użytkownika. W tym celu w wybranych miejscach zastosowano montaż „piętrowy", w związku z czym kolejność implementacji ma tutaj szczególne znaczenie.

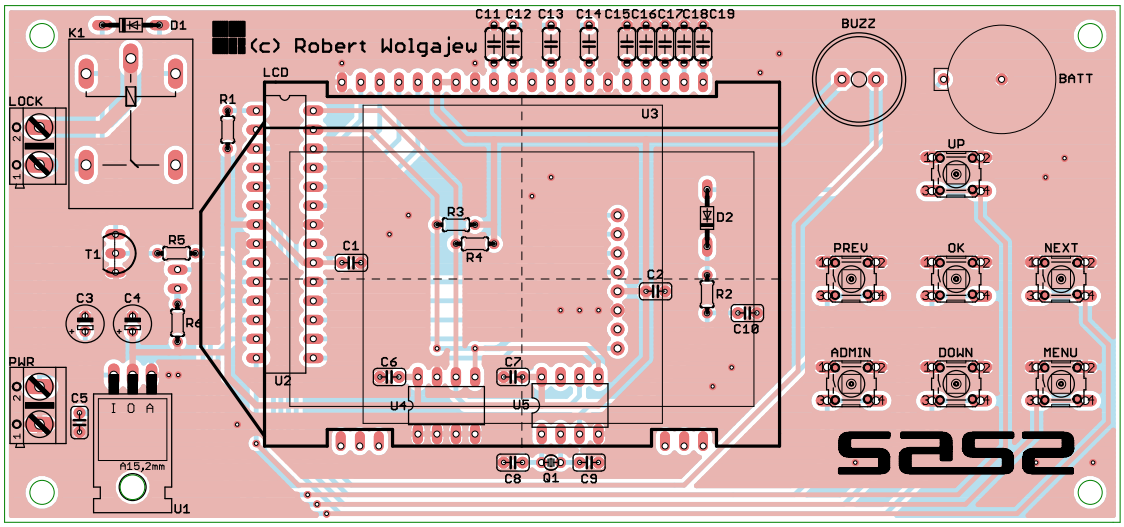
Warto również podkreślić, iż dla zminimalizowania zakłóceń, na płytce urządzenia poprowadzono obszerne pola masy po obu stronach obwodu drukowanego oraz zastosowano szereg przelotek pomiędzy nimi w celu zmniejszenia pojemności pasożytniczych. Montaż urządzenia rozpoczynamy od przylutowania wszystkich układów scalonych, następnie lutujemy pozostałe półprzewodniki, dalej elementy bierne a na samym końcu peryferia

Listing 12. Fragment kodu aplikacji, który odpowiedzialny jest za proces obsługi karty

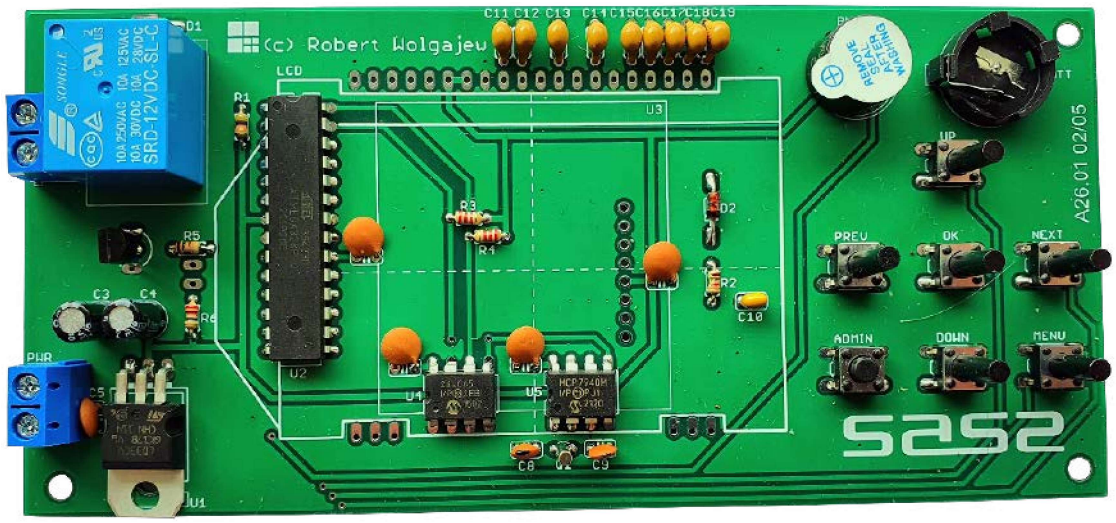
```
//Sprawdzamy obecność karty – tylko na ekranie głównym
if(workMode == MODE_NORMAL && cardReadUID(UID) == NO_ERROR){

    //Sprawdzamy rodzaj przyłożonej karty – w zmiennej globalnej User pojawią się
    //ewentualne dane użytkownika
    switch(findUserUID(UID)){
        case USER_NOT_FOUND:
            foundUserID = UNKNOWN_USER;
        break;

        case USER_FOUND:
            //Uwierzytelniamy kartę odczytanym kluczem i numerem UID
            if(cardAuthenticate(User.UID, User.Key) == NO_ERROR)
                foundUserID = User.userID;
            else foundUserID = UNKNOWN_USER;
        break;

        case USER_IS_ADMIN:
            //Uwierzytelniamy kartę odczytanym kluczem i numerem UID
            if(cardAuthenticate(User.UID, User.Key) == NO_ERROR){
                //Przyłożona karta to numer ADMINa,
                //więc zmieniamy wartość zmiennej logowania
                adminLoggedin ^= 0x01;
                if(adminLoggedin) foundUserID = ADMIN_LOGIN;
                else foundUserID = ADMIN_LOGOUT;
            }
        break;
    }

    //Przygotowyjemy dane struktury Event do zapisania zdarzenia w pamięci EEPROM
    Event.userID = foundUserID;  //ID znalezionego Usera, USER_UNKNOWN, ADMIN_LOGIN lub
    //ADMIN_LOGOUT
    RTCreadDateTime(&Event.dateTime);  //Bieżący znacznik czasu
    Event.Status = NEW_EVENT;  //Znacznik nowego zdarzenia, jeszcze nie przeglądanego

    //Zapis struktury pod bieżącym adresem zdarzenia
    EEPROMwriteBlock(newEventPointer, &Event);

    //Ustawienie nowego wskaźnika miejsca do kolejnego zapisu.
    //Zapętlenie rejestru zdarzeń, jeśli osiągnięto koniec rejestru
    if(++newEventPointer == MAX_EVENTS) newEventPointer = 0;
    //Zwiększenie liczby zdarzeń. Jeśli osiągnięto maksymalną liczbę zdarzeń to nie
    //przekraczamy jej,
    //bo lista zacznie się wypełniać od góry
    if(++eventsNumber > MAX_EVENTS) eventsNumber = MAX_EVENTS;

    //Zapisujemy wskaźnik do najbliższego wolnego miejsca i liczbę zdarzeń w EEPROM
    EEPROMwriteWord(EVENT_POINTER_ADDR, newEventPointer);
    EEPROMwriteWord(EVENT_NR_ADDR, eventsNumber);

    //W przypadku znanego użytkownika załączamy przekaźnik
    if(foundUserID != UNKNOWN_USER && foundUserID != ADMIN_LOGIN && foundUserID != ADMIN_
LOGOUT){
        Beep(BUZZER_SHORT); //Piknięcie buzzera
        RELAY_ON;
        _delay_ms(1500);
        RELAY_OFF;
    }
}
```



Rysunek 8. Wygląd ekranu interfejsu użytkownika dla głównych trybów pracy systemu Menu urządzenia SAS2



Rysunek 9. Schemat płytki PCB systemu SAS2



Fotografia 1. Wygląd obwodu drukowanego urządzenia SAS2 tuż przed przylutowaniem modułu RFID i wyświetlacza COG

mechaniczne. Kondensatory ceramiczne znajdujące się docelowo pod płytką modułu RFID (C1, C2, C6 i C7) montujemy poziomo by w ten sposób zmniejszyć wysokość tej „warstwy" elementów. Następnie przylutowujemy moduł czytnika RFID najbliżej jak to się da od obwodu drukowanego urządzenia. W tym momencie ustawiamy w odpowiednich pozycjach przełączniki SMD umieszczone na module RFID, dzięki którym wybieramy aktywny interfejs komunikacyjny (w naszym wypadku SPI). Aby przygotować moduł do pracy z interfejsem SPI przełączamy przełącznik oznaczony jako ON w pozycję On (blisko znacznika „1") zaś przełącznik oznaczony, jako KE w pozycję Off (blisko znacznika „KE").

W ostatnim kroku montujemy wyświetlacz graficzny COG łącznie z podświetleniem kierując się zasadą, iż powinien on znaleźć się jak najbliżej płaszczyzny modułu RFID, jak to tylko możliwe. Montując ten element możemy również posłużyć się gniazdem goldpin (żeńskim) co pozwoli na jego delikatne odsunięcie od płaszczyzny modułu RFID. Sam wyświetlacz przysłoni co prawda moduł RFID, lecz testy praktyczne wykazały, iż nie zawiera on elementów, które w sposób znaczący mogłyby zaburzyć funkcjonowanie anteny umieszczonej na tym peryferium.

Poprawnie zmontowany układ nie wymaga żadnych regulacji (poza koniecznością zamontowania baterii CR1220 podtrzymującej zegar RTC) i powinien działać tuż po włączeniu zasilania. Jedynym zabiegiem, jaki należy wykonać przed użytkowaniem urządzenia jest sformatowanie i wyczyszczenie pamięci zdarzeń EEPROM, do czego przewidziano odpowiednią opcję Menu (dostępną wyłącznie podczas włączania urządzenia).

Na **fotografii 1** pokazano wygląd obwodu drukowanego urządzenia SAS2 tuż przed przylutowaniem modułu RFID i wyświetlacza COG.

Robert Wołgajew
robert.wolgajew@ep.com.pl