RFID TZBOT

Moduł RFID współpracujący z tagami 13.56MHz ISO 15693

Rysunek 1: TZBOT RFID

Data ostatniej kompilacji:
31 lipca 2020

1 Wymagania odnośnie tagów RFID

Urządzenie współpracuje tylko i wyłączenie z tagami zgodnymi z normą ISO15693. Przykładem tagiego taga jest `NXP - ICODE SLIX2`

2 Specyfikacja urządzenia

`Parametr`	`Opis`
Częstotliwość pracy	13.56 MHz
Moc wyjściowa	200mW
Odległość odczytu/zapisu	max 15cm(zależy od rozmiarów tagów)
Pomiar siły sygnału	8 stopni (0–7)
Rodzaj anteny	Zintegrowana
Czas odczytu (4 bajty)	13ms
Czas odczytu (8 bajtów)	15 ms
Czas zapisu (4 bajty)	16ms
Czas zapisu (8 bajtów)	32ms
Prędkość transmisji	26kbit/s
Napięcie pracy	12V DC – 36V DC
Pobór mocy	typ. 1.5W
Stopień ochrony IP	IP54
Waga	0.41 kg
Temperatura pracy	$-10°C \sim +50°C$
Wilgotność otoczenia	poniżej 80%

Tabela 1: Specyfikacja urządzenia

3 Opis wyprowadzeń

`Kolor przewodu`	`Opis`
Czerwony	+24 DC
Niebieski	GND
Czarny	RX (RS232)
Szary	TX (RS232)
Fioletowy	GND (RS232)
Biały	A(RS485)
Brązowy	B(RS485)
Zielony	GND(RS485)
Pomarańczowy	C-H(CAN)
Żółty	C-L(CAN)

Tabela 2: Definicja wyjść

4 RS232

4.1 Schemat połączeń

Czujnik należy podłączyć do komputera za pomocą konwertera USB-RS232 zgodnie z poniższymi zaleceniami.

4.1.1 Opis wyprowadzeń

Kolor przewodu	Opis
Czerwony	+24 DC
Niebieski	GND
Czarny	RX (RS232)
Szary	TX (RS232)
Fioletowy	GND (RS232)

Tabela 3: Schemat połączeń RS232

4.2 Specyfikacja połączenia

W celu nawiązania połączenia z urządzeniem należy użyć następujących parametrów połączenia

Parametr	Wartość
Prędkość transmisji [bps]	115200
Parzystość	Brak
Bit stopu	1 bit
Ilość bitów na bajt	8

Tabela 4: Specyfikacja połączenia

Przykładowa konfiguracja portu przy użyciu biblioteki `pyserial` w języku `Python`:

```
> ser = serial.Serial(
  port=portT,
  baudrate=115200,
  parity=serial.PARITY_NONE,
  stopbits=serial.STOPBITS_ONE,
  bytesize=serial.EIGHTBITS,
  timeout=1)
```

4.3 Protokół komunikacji

4.3.1 CRC16-MODBUS

Każda ramka wysyłana lub odbierana z czujnika RFID zakończona jest dwoma bajtami sumy kontrolnej CRC16-MODBUS. Fragment kodu w języku Python wykorzystujący bibliotekę `libscrc` obliczający wartość sumy kontrolnej, dzieląca ją na dwa bajty (HIGH oraz LOW) oraz dołączający ją do wysyłanej do czujnika wiadomości:

```
> import libscrc
> code = bytearray([0x52, 0x43, 0x6f, 0x64, 0x65, 0x00])
> def calcCrc(code):
>   crc16 = libscrc.modbus(code)
>   crc16 = hex(crc16).split('x')[-1]
>   crc16H = crc16[0:2]
>   crc16L = crc16[2:4]
>   return int(crc16H, 16), int(crc16L, 16)
> crc16 = calcCrc(code)
> code.append(crc16[1])
> code.append(crc16[0])
> ser.write(code)
```

4.3.2 Odczyt danych z karty

W celu odczytu danych z tagu RFID zgodnego z normą `ISO 15693` , należy przesłać do urządzenia zapytanie o ustalonej postaci: `0x52, 0x43, 0x6f, 0x64, 0x65, 0x00, 0x3F, 0xFD`, gdzie zapytanie jest tablicą liczb heksadecymalnych

`Numer bajtu`	`Wartość (hex)`	Znaczenie
1	0x52	ID
2	0x43	ID
3	0x6f	ID
4	0x64	ID
5	0x65	ID
6	0x00	Communication ID
7	0x3F	CRC16-MODBUS Low byte
8	0xFD	CRC16-MODBUS High byte

Tabela 5: Zapytanie do czytnika RFID

Konstrukcja zapytania do czujnika w języku `Python`:

```
> code = bytearray([0x52, 0x43, 0x6f, 0x64, 0x65, 0x00, 0x3f, 0xfd])
> ser.write(code)
```

W przeciągu maksymalnie sekundy od otrzymania zapytania urządzenie odpowiada wiadomością następującej postaci:

`Numer bajtu`	`Wartość (hex)`	Znaczenie
1	0x52	ID
2	0x43	ID
3	0x6f	ID
4	0x64	ID
5	0x65	ID
6	0x00	Communication ID(0x00)
7	0x??	Status odczytu (0–1)
8	0x??	Siła sygnału (0–7)
9	0x??	Dane z karty
10	0x??	Dane z karty
11	0x??	Dane z karty
12	0x??	Dane z karty
13	0x??	Dane z karty
14	0x??	Dane z karty
15	0x??	Dane z karty
16	0x??	Dane z karty
17	0x??	CRC16-MODBUS Low byte
18	0x??	CRC16-MODBUS High byte

Tabela 6: Odpowiedź czujnika RFID z odczytanymi danymi

Konstrukcja odczytu z czujnika w języku `Python`:

```
> read_val = ser.read(18)
```

Bajt 7. przyjmuje wartości:

- 0x00 w przypadku braku odczytu z karty oraz,
- 0x01 w przypadku gdy udało się dokonać odczytu
- Wartości zwracane w bajtach od 9 do 16 dotyczą ostatniego udanego odczytu z karty. Oznacza to że w przypadku gdy czytnik tagów RFID nie odczytał nowych danych z karty, w bajtach od 9 do 16 odpowiedzi, zakodowane będą dane z ostatniego udanego odczytu taga, lub w przypadku gdy od uruchomienia czytnika RFID, nie udało się odczytać dotąd żadnego tagu, wszystkie te bajty przyjmą wartość $0x00$
- Przerwa pomiędzy otrzymaniem odpowiedzi z czujnika, a wysłaniem kolejnego zapytania powinna trwać od minimum 5ms. Zalecany jest jednak czas przerwy wynoszący 30ms.

4.3.3 Zapis danych na karcie

W celu dokonania zapisu na karcie RFID należy przesłać do urządzenia ramkę postaci:

Numer bajtu	Wartość (hex)	Znaczenie
1	0x57	ID
2	0x43	ID
3	0x6f	ID
4	0x64	ID
5	0x65	ID
6	0x00	Communication ID(0x00)
7	0x??	Dane do zapisu
8	0x??	Dane do zapisu
9	0x??	Dane do zapisu
10	0x??	Dane do zapisu
11	0x??	Dane do zapisu
12	0x??	Dane do zapisu
13	0x??	Dane do zapisu
14	0x??	Dane do zapisu
15	0x??	CRC16-MODBUS Low byte
16	0x??	CRC16-MODBUS High byte

Tabela 7: Ramka zapisu na karcie

W odpowiedzi na wysłane przez nas pytanie otrzymujemy:

Numer bajtu	Wartość (hex)	Znaczenie
1	0x57	ID
2	0x43	ID
3	0x6f	ID
4	0x64	ID
5	0x65	ID
6	0x00	Communication ID
7	0x??	Status odczytu
8	0x??	CRC16-MODBUS Low byte
9	0x??	CRC16-MODBUS High byte

Tabela 8: Odpowiedź czujnika RFID po zapisie

Wartość zapisana w bajcie 7. sygnalizuje status dokonanego zapisu:

- 0x00 – zapis zakończony powodzeniem
- 0x01 – błąd zapisu lub karta jest zablokowana do zapisu
- 0x02 – nie wykryto karty do zapisu

5 Opis paczki `rfid` do ROS Melodic

5.1 Komunikacja z czujnikiem

W celu komunikacji z czujnikiem RFID należy:

- w oknie terminala uruchomić polecenie:

```
> rosrun
```

- w nowym oknie terminala uruchomić skrypt `listener.py`

```
> rosrun rfid listener.py
```

- w tym momencie momencie możemy komunikować się z czujnikiem RFID za pomocą publikowania wiadomości w topicu `rfid_listener`
  - r* – jednorazowy odczyt danych z czujnika, gdzie * to dowolny ciąg znaków
  - w* – jednorazowy zapis danych na tagu, gdzie * to maksymalnie 8 znaków do zapisu na tagu RFID
  - l* – odczyt danych w pętli, gdzie * to czas odświeżania w `ms`
- Do publikowania wiadomości w topicu rfid_listener można użyć programu `commands.py` uruchamianego przy wywołaniu polecenia:

```
> rosrun rfid commands.py
```

5.2 Dostęp do odpowiedzi z czujnika

Odpowiedzi otrzymywane z czujnika publikowane są w topicach:

- `rfid_result_read` – dane odczytywane z taga RFID
- `rfid_result_write` – dane o statusie zapisu na tagu RFID

Aby nasłuchiwać danych z topiców można użyć polecenia:

```
> rostopic echo (nazwa topicu)
```

5.2.1 Struktura odpowiedzi

Typ	Nazwa	Opis
uint16	ComIdByte	Bajt komunikacji. Typ 0x00
uint16	CardReadingByte	0x00 -> odczyt danych nieudany 0x01 -> odczyt udany
uint16	SignalStrength	Siła sygnału w skali 0 do 7
uint16	CardData8	8. bajt z odczytu
uint16	CardData7	7. bajt z odczytu
uint16	CardData6	6. bajt z odczytu
uint16	CardData5	5. bajt z odczytu
uint16	CardData4	4. bajt z odczytu
uint16	CardData3	3. bajt z odczytu
uint16	CardData2	2. bajt z odczytu
uint16	CardData1	1. bajt z odczytu

Tabela 9: Struktura wiadomości `rfid_result_read`

Typ	Nazwa	Opis
bool	Data	True -> zapis na tagu się powiódł, False -> zapis się nie udał

Tabela 10: Struktura wiadomości `rfid_result_write`