Rozdział 3

Protokół komunikacyjny

Rozdział opisuje dokładny sposób łączenia się klientów z serwerem oraz klientów ze sobą za pośrednictwem serwera. Omawia on również kwestie przypisywania nazw urządzeniom oraz ich kanałom danych i kanałom informacyjnym w bazie danych. W skład opisu wchodzi sposób użycia poszczególnych technologii, jak również wiadomości używane do wzajemnej komunikacji między komponentami.

3.1. Wiadomości w protokole MQTT

Każda z wiadomości w protokole MQTT korzysta z jednej struktury wiadomości. Ogólna struktura wiadomości przedstawiona została w tabeli 3.1.

Tab. 3.1: Ogólna struktura wiadomości[8]

Fixed Header, present in all MQTT Control Packets
Variable Header, present in some MQTT Control Packets
Payload, present in some MQTT Control Packets

Każda wiadomość protokołu MQTT posiada swój stały nagłówek (Fix Header), który określa rodzaj wiadomości (4 bity), flagi kontrolne, specyficzne dla danego rodzaju wiadomości (4 bity) oraz długość pakietu (1-4 bajty).

Zależnie od rodzaju wiadomości obecny może być zmienny nagłówek (Variable Header) o zmiennej długości, zwierający dodatkowe informacje o połączeniu, takie jak identyfikator pakietu oraz długość właściwości i same właściwości.

Ostatnią częścią, również obecną tylko w niektórych pakietach, jest pole wiadomości (Payload), zawierające np. treść wiadomości.

3.2. Przypisywanie nazw kanałom

Przypisywanie nazw urządzeniom korzysta z faktu, że w pamięci każdego układu ESP32 zapisany jest jego adres MAC karty sieciowej, który jest unikalny na skalę światową. Adres ten zapisywany jest w bazie danych i stanowi klucz rekordu opisującego urządzenie.

Jak opisano w rozdziale 2.2, każde urządzenie dysponuje trzema kanałami logicznymi. Kanał danych urządzenia opisany jest jako `XX:XX:XX:XX:XX:XXd`, kanał informacyjny: `XX:XX:XX:XX:XX:XXi`, natomiast kanał służący do zmian ustawień do urządzenia: `XX:XX:XX:XX:XX:XXs` (gdzie adres MAC urządzenia to `XX:XX:XX:XX:XX:XX`).

3.3. Nawiązanie komunikacji klient-serwer

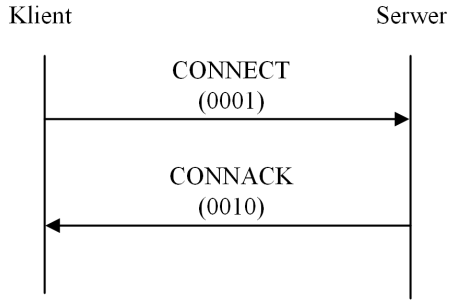
Połączenie z serwerem realizowane jest poprzez protokół MQTT v5.0, przy użyciu szyfrowanego (TLSv1.3) połączenia TCP.

3.3.1. Bezpieczeństwo połączenia

Aby zapewnić klientom bezpieczeństwo, posiadać muszą oni odpowiedni łańcuch certyfikatów (certificate chain) w celu identyfikacji serwera. Dzięki łańcuchowi certyfikatów, klienci weryfikują czy certyfikat serwera został wygenerowany przez zaufany urząd certyfikacji (CA). Dodatkowo po zweryfikowaniu autentyczności certyfikatu, klienci sprawdzają czy domena, dla której został wygenerowany odebrany certyfikat zgadza się z domeną, z którą połączyło się urządzenie.

3.3.2. Łączenie z brokerem MQTT

Po zestawieniu bezpiecznego połączenia, musi nastąpić faza łączenia z brokerem MQTT.



Rys. 3.1: Wymiana wiadomości podczas zestawiania połączenia z brokerem MQTT

Łączenie z brokerem odbywa się poprzez wysłanie przez klienta pakietu CONNECT (*Connection Request*) przez klienta. Pakiet CONNECT może składać się z trzech części wymienionych w punkcie 3.1. W przypadku aplikacji wykorzystywane są tylko nagłówek stały oraz zmienny z racji braku wiadomości wysyłanej w razie rozłączenia klienta (Will Message). W przypadku urządzenia wykorzystywane są trzy części wiadomości. Nagłówek stały określa typ wiadomości (0001). Nagłówek musi określić:

- wersję protokołu (v5.0),
- długość okresu „*Keep Alive*" (3.4),
- użycie nazwy użytkownika oraz samej nazwy użytkownika,
- użycie hasła oraz samego hasła,

Rysunek 3.2 przedstawia przykładową wiadomość CONNECT wysyłaną przez aplikację, natomiast 3.3 wysyłaną przez urządzenie.

```
Header Flags: 0x10, Message Type: Connect Command
    0001 .... = Message Type: Connect Command (1)
    .... 0000 = Reserved: 0
  Msg Len: 28
  Protocol Name Length: 4
  Protocol Name: MQTT
  Version: MQTT v5.0 (5)
Connect Flags: 0xc0, User Name Flag, Password Flag, QoS Level: At most once delivery (Fire and Forget)
    1... .... = User Name Flag: Set
    .1.. .... = Password Flag: Set
    ..0. .... = Will Retain: Not set
    ...0 0... = QoS Level: At most once delivery (Fire and Forget) (0)
    .... .0.. = Will Flag: Not set
    .... ..0. = Clean Session Flag: Not set
    .... ...0 = (Reserved): Not set
  Keep Alive: 60
Properties
    Total Length: 0
  Client ID Length: 1
  Client ID: 1
  User Name Length: 5
  User Name: admin
  Password Length: 5
  Password: admin
```

Rys. 3.2: Przykładowa wiadomość CONNECT wysyłana przez aplikację

```
Header Flags: 0x10, Message Type: Connect Command
  Msg Len: 57
  Protocol Name Length: 4
  Protocol Name: MQTT
  Version: MQTT v5.0 (5)
Connect Flags: 0xc6, User Name Flag, Password Flag, QoS Level: At most once delivery (Fire and Forget), Will Flag, Clean Session Flag
    1... .... = User Name Flag: Set
    .1.. .... = Password Flag: Set
    ..0. .... = Will Retain: Not set
    ...0 0... = QoS Level: At most once delivery (Fire and Forget) (0)
    .... .1.. = Will Flag: Set
    .... ..1. = Clean Session Flag: Set
    .... ...0 = (Reserved): Not set
  Keep Alive: 60
Properties
    Total Length: 3
    ID: Receive Maximum (0x21)
    Value: 20
  Client ID Length: 0
  Client ID:
Will Properties
    Total Length: 0
  Will Topic Length: 18
  Will Topic: XX:XX:XX:XX:XX:XXi
  Will Message Length: 4
  Will Message: 445f440a
  User Name Length: 5
  User Name: admin
  Password Length: 5
  Password: admin
```

Rys. 3.3: Przykładowa wiadomość CONNECT wysyłana przez urządzenie

Klient wysyłający wiadomość CONNECT otrzymuje od serwera odpowiedź w postaci wiadomości CONNACK (*Connect acknowledgement*), o kodzie wiadomości w stałym nagłówku: 0010. Nagłówek zmienny wiadomości zawiera kod błędu (ang. *Reason Code*) oraz (jeżeli kod błędu wynosi „0”) właściwości sesji narzucone przez ustawienia serwera.

W razie spełnienia wszystkich ww. wymagań, serwer odpowiada kodem błędu „0” oraz określa:

- maksymalną ilość aliasów dostępnych do przypisania przez klienta (w systemie określone jako 30; 3.5.2.1),
- maksymalną długość pakietu, która może zostać wysłana przez klienta (w systemie określone jako 1 460 bajtów),
- przypisany identyfikator klienta (taki sam jak podana nazwa użytkowika).

Wiadomość widoczna jest na rysunku 3.4.

```
Header Flags: 0x20, Message Type: Connect Ack
    0010 .... = Message Type: Connect Ack (2)
    .... 0000 = Reserved: 0
  Msg Len: 22
Acknowledge Flags: 0x00
    0000 000. = Reserved: Not set
    .... ...0 = Session Present: Not set
  Reason Code: Success (0)
Properties
    Total Length: 19
    ID: Topic Alias Maximum (0x22)
    Value: 30
    ID: Assigned Client Identifier (0x12)
    Length: 5
    Value: admin
    ID: Maximum Packet Size (0x27)
    Value: 1460
```

Rys. 3.4: Wiadomość CONNACK wysyłana w razie pomyślnego ustawienia parametrów sesji

W razie wyboru innej wersji protokołu serwer nie odrzuci połączenia, ale podczas pierwszej publikacji wiadomości klient nie będzie miał możliwości wyboru aliasu kanału. W razie braku dostarcza poświadczeń (tj. nazwy użytkownika lub hasła) lub dostarczenia nieprawidłowych poświadczeń, serwer odrzuci połączenie, informując klienta o przyczynie błędu.

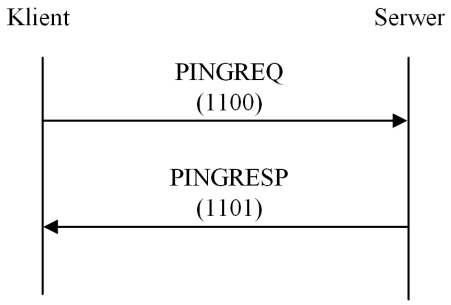
```
Header Flags: 0x20, Message Type: Connect Ack
    0010 .... = Message Type: Connect Ack (2)
    .... 0000 = Reserved: 0
  Msg Len: 3
> Acknowledge Flags: 0x00
  Reason Code: Not authorized (135)
Properties
    Total Length: 0
```

Rys. 3.5: Wiadomość CONNACK wysyłana w razie nieprawidłowych poświadczeń

W razie wysłania do serwera wiadomości niezgodnej ze standardem MQTT, serwer natychmiast zakończy połączenie.

3.4. Utrzymanie komunikacji klient-serwer

Utrzymywanie sesji odbywa się zgodnie ze standardem MQTT[8].



Rys. 3.6: Wydłużanie życia sesji MQTT

Każdy z połączonych z serwerem klientów zobowiązany jest wysyłać pakiet PINGREQ (ang. *PING request*) w razie braku wysyłania innych pakietów kontrolnych w określonym odstępie

czasu, aby utrzymać sesję. W przeciwnym razie połączenie zostaje zerwane. Pakiet składa się wyłączenie ze stałego nagłówkaokreślającego typ wiadomości (1100). Odstęp czasu nie jest z góry określony przez standard MQTT, dlatego czas ten określono w systemie jako 90 *s*.

W momencie otrzymania pakietu serwer odpowiada klientowi pakietem PINGRESP (ang. *PING response*), tym samym sesja zostaje przedłużona. Pakiet również składa się wyłączenie ze stałego nagłówkaokreślającego typ wiadomości (1101).

3.5. Komunikacja w protokole MQTT

Po prawidłowym nawiązaniu sesji, klient może połączyć się do danego kanału wybranego urządzenia. Standard MQTT zawiera dwie, niezależne od siebie operacje, możliwe do wykonania na kanałach:

1. SUBSCRIBE - subskrypcja danego kanału i jego nasłuchiwanie,
2. PUBLISH - publikacja wiadomości w danym kanale,

3.5.1. Subskrypcja kanałów

Subskrypcja kanałów jest wykonywana poprzez wysłanie pakietu SUBSCRIBE (*Subscribe request*), składającego się ze wszystkich części wymienionych w punkcie 3.1. Stały nagłówek określa typ wiadomości (0110) oraz jej całkowitą długość. Zmienny nagłówek zawiera niezerowy identyfikator pakietu (ang. *Packet Identyfier*), własności użytkownika (ang. *User Property*), które w systemie nie są używane, zatem pole jego długości jest zawsze równe zero.

Pole wiadomości obejmuje nazwę kanału, który klient chce nasłuchiwać oraz opcje subskrypcji. Klienci systemu muszą określić dwie opcje:

- *No Local*,
- *QoS*.

Domyślnie w protokole MQTT podczas wysyłania wiadomości (PUBLISH), serwer odsyła tę wiadomość z powrotem do „nadawcy”, jeśli subskrybuje on dany kanał. Ustawienie opcji *No Local* zapobiega takiemu zachowaniu. Pozytywnie wpływa to na pasmo wykorzystywane podczas komunikacji oraz ogranicza ilość wykorzystywanych kanałów (klient nie mógłby w łatwy określić czy jest nadawcą danej wiadomości).

Opcja *QoS* określa „jakość usługi” (ang. *Quality of Service*) i może ona przyjmować wartości [8]:

1. *0* - brak pewności odbioru wysłanej wartości przez pozostałych klientów,
2. *1* - pewność, że wiadomość zostanie odebrana co najmniej raz,
3. *2* - pewność, że wiadomość będzie dostarczona maksymalnie raz.

Klienci systemu używają QOS = 0. Pozostałe opcje pozwalają na ponowne wysłanie wiadomości przez serwer jeżeli dany klient nie odeśle potwierdzenia jej otrzymania, jednak dzieje się to kosztem potwierdzeń wysyłania potwierdzeń (w przypadku QOS = 1, wysyłane jest jedno potwierdzenie, przy QOS = 2, wysyłane są dwa potwierdzenia, zarówno przez nadawcę, jak i odbiorcę). Dodatkowo mechanizm ponownego wysyłania wiadomości wymaga rozłączenia z serwerem, a następnie ponownego łączenia, co nie jest odpowiednią metodą zapewnienia odbioru danych dla systemu, gdzie wymiana danych odbywa się w czasie rzeczywistym.

Urządzenie ma zawsze prawo do odczytu, jak i zapisu w każdym swoim kanale. W przypadku aplikacji (a zatem i użytkowników), prawo do danej operacji jest weryfikowane w bazie danych.

3.5.2. Publikacja w kanałach

3.5.2.1. Aliasy kanałów

Wybór kanału odbywa się na podstawie jego nazwy, których omówienie znajduje się w punkcie 3.2. Operacja odczytu (SUBSCRIBE) jest wywoływana jednorazowo i pozwala na odbiór wiadomości przez cały okres trwania połączenia, ale operacja zapisu (PUBLISH) w MQTT musi być wykonywana przy każdorazowo w razie wysłania nowej porcji danych. Stosowanie nazw kanałów stanowiłyby zdecydowanie zbyt dużą redundancję nawet w razie wysyłania pokaźnych ilości danych.

Protokół MQTT v5.0 przewiduje możliwość nadania aliasu, będącego liczbą całkowitą, danemu kanałowi[8]. Alias dla danego kanału dotyczy wyłącznie określonego klienta, zatem wielu klientów może używać tego samego aliasu dla różnych kanałów. Nadanie aliasu może odbyć się wyłączenie podczas publikacji wiadomości w danym kanale.

Klienci systemu powinni korzystać z mechanizmu aliasów. Podczas pierwszej publikacji w danym kanale, klient określa alias tego kanału. W systemie klienci wybierają alias będący najmniejszą liczbą całkowitą, która nie została jeszcze przydzielona jako alias innemu kanałowi lub która nie jest już aliasem kanału.

3.6. Nawiązanie komunikacji urządzenie-aplikacja

Zanim możliwa będzie wymiana danych pomiędzy urządzeniem a aplikacją, aplikacja określić musi dostępność urządzenia oraz wspólne parametry transmisyjne. Aby zwiększyć szansę na poprawny odbiór danych, aplikacja musi co dany okres czasu upewnić się, że urządzenie po drugiej stronie kanału jest wciąż aktywne. W tym celu wykorzystane są komendy, stworzone na potrzeby systemu. Następstwem wysłania komendy przez aplikację jest obowiązkowe odesłanie odpowiedzi przez urządzenie. Odpowiedź na komendę musi zostać odesłana w czasie 10 s.

3.6.1. Określenie odstępności urządzenia

Komunikacja aplikacji z urządzeniem rozpoczyna się od określenia czy urządzenie jest w danym momencie podłączone do systemu. Aż do momentu ustalenia wspólnych, zgodnych ze sobą parametrów transmisyjnych, aplikacja musi ignorować wszystkie informacje nadchodzące z kanału danych oraz z portu.

Aby to rozpocząć nową sesję, aplikacja wysyła w kanał ustawień „powitalną” komendę `M_HI` („Master Hi”). Jeżeli urządzenie jest w stanie odczytać informację, wówczas odpowiada aplikacji komendą `S_HI` („Slave Hi”) w kanale informacyjnym oraz resetuje licznik wiadomości kanału danych. Po odebraniu komendy `S_HI`, aplikacja również resetuje licznik wiadomości kanału danych.

Jeśli klient nie ma dostępu do kanału ustawień możliwe jest skorzystanie z `M_KA` **??**.

Jeśli komunikacja przebiegła poprawnie, aplikacja może przejść do wysyłania kolejnych komend, jeżeli nie, wówczas aplikacja musi poinformować klienta o braku możliwości komunikacji i co dany okres czasu (10 *s*) próbować „przywitać się z urządzeniem”.

W przypadku gdy urządzenie ukończyło wykonywanie subskrypcji wszystkich swoich kanałów, musi ono automatycznie wysłać komendę `S_HI`, aby powiadomić o swojej obecności drugiego klienta.

3.6.2. Ustawienia urządzenia

Odczyt i zmiana ustawień urządzenia może zostać dokonana w dowolnym momencie, jeżeli tylko nawiązanie komunikacji (3.6.1) przebiegło poprawnie. Na czas zmiany ustawień wymiana danych musi zostać zawieszona. Jeżeli odpowiedź urządzenia będzie nierozpoznana przez aplikację, wówczas aplikacja musi poinformować klienta o braku możliwości komunikacji i co dany okres czasu (10 *s*) próbować „przywitać się z urządzeniem".

Odpowiedzi nie muszą zostać wysłane w takiej samej kolejności, jak wysyłane były komendy. Aplikacja jest zobligowana do rozpoznania, która informacja jest odpowiedzią na daną komendę.

3.6.2.1. Odczytanie ustawień urządzenia

Odczytanie ustawień urządzenia jest dokonywane poprzez wysłanie do niego komendy `G_I` („Get Info"). W tabeli 3.2 zamieszczono odpowiedzi, które musi otrzymać aplikacja po wysłaniu komendy `G_I`. Odpowiedzi wysyłane są w kanał informacyjny (`XX:XX:XX:XX:XX:XXi`);

Tab. 3.2: Odpowiedzi na komendę `G_I`

Komenda	Rodzaj	Argument	Znaczenie argumentu	Znaczenie komendy
G_B	Liczba całkowita	9600 <= x <= 115200	n/d	Get Baud Rate
G_P	Łańcuch znaków	E	Even	Get Parity
		N	None	
		O	Odd	
G_C	Liczba całkowita	5 <= x <= 8	n/d	Get Char Size
G_S	Łańcuch znaków	O	One	Get Stop Bits
		OPF	One Point Five	
		T	Two	

Po pomyślnym pobraniu któregokolwiek z ustawień, aplikacja musi zaktualizować ustawienie swojego portu.

3.6.2.2. Zmiana ustawień urządzenia

W przeciwieństwie do pobierania ustawień, ich zmiana musi być dokonywana oddzielnie dla każdego ustawienia. Komendy zmieniające ustawienia urządzenia muszą być wysyłane w kanał do tego przeznaczony (`XX:XX:XX:XX:XX:XXs`). W tabeli 3.3 znajdują się komendy służące do zmiany ustawień urządzenia.

Następstwem wysłania komendy jest obowiązkowe otrzymanie na niej odpowiedzi. Poniżej widoczne są nazwy komend oraz przyporządkowane do nich nazwy odpowiedzi (definicje odpowiedzi są tożsame z tymi zamieszczonymi w tabeli 3.2).

Każda z wysyłanych komend (3.3) musi zostać wysłana z odpowiednim argumentem. Odpowiedzi na komendy (3.2) muszą zawierać argumenty korespondujące z tymi nadesłanymi w komendach. Jeżeli argumenty nie będą tożsame, wówczas urządzenie powinno zgłosić błąd na kanał informacyjny.

Tab. 3.3: Komendy zmiany ustawień

Komenda	Rodzaj	Argument	Znaczenie argumentu	Znaczenie komendy
S_B	Liczba całkowita	9600 <= x <= 115200	n/d	Set Baud Rate
S_P	Łańcuch znaków	E	Even	Set Parity
		N	None	
		O	Odd	
S_C	Liczba całkowita	5 <= x <= 8	n/d	Set Char Size
S_S	Łańcuch znaków	O	One	Set Stop Bits
		OPF	One Point Five	
		T	Two	

Tab. 3.4: Komendy i przyporządkowane do nich odpowiedzi

Komenda	Odpowiedź
S_B	G_B
S_P	G_P
S_C	G_C
S_S	G_S

3.7. Komunikacja urządzenie-aplikacja

Po spełnieniu wszystkich wymagań opisanych w punkcie 3.6, urządzenie i aplikacja mogą rozpocząć lub wznowić wymianę danych.

3.7.1. Licznik pakietów

Wiadomości wysyłane do kanału danych (PUBLISH) muszą zostać opatrzone numerem identyfikującym wiadomość. Numer nie jest wysyłany w treści wiadomości, ale w nagłówku zmiennym w polu właściwości (ang. *Properties*), jako właściwości użytkownika (ang. *User properties*) [8].

Właściwości użytkownika są zdefiniowane jako pole składające się z klucza oraz wartości. Oba typy danych to łańcuchy znaków UTF-8 [8].

Klucz numeru wiadomości to pojedynczy znak „*P*”. Wartość jest również pojedynczym znakiem, aby zaoszczędzić ilość wysyłanych danych. Może ona przybierać wartości od „ ” (spacja, numer znaku UTF-8: 0x20) do „∼” (tylda, numer znaku UTF-8: 0x7f). Przedział wartości jest spowodowany specyfikacją MQTT v5.0, gdzie inne wartości będą odrzucane, a wysyłana wiadomość usuwana przez serwer [8].

Dzięki numeracji pakietów klienci mogą określić który pakiet pojawi się jako następny. Zatem nawet jeżeli doszło do błędu (np. przepełnienie bufora i usunięcie wiadomości), wówczas gdy nadejdzie pakiet o nieoczekiwanym numerze, klient może zgłosić błąd i poprosić o prawidłowy pakiet, używając komendy: `I_N`, wraz z oczekiwanym numerem. Nieprawidłowy pakiet jest usuwany.

Wymusza to na klientach zapisywanie kilku ostatnich wiadomości. Ilość ta zdefiniowana jest jako 16.

Po odebraniu 8 wiadomości i przed odebraniem 8 kolejnych, klient powinien powiadomić drugą stronę o akceptacji ostatnich 8 pakietów komendą `A`, wraz z numerem ósmej wiadomości. Wówczas mogą być one bezpiecznie nadpisane przez nowe wiadomości. Jeżeli w terminie nie zostanie wysłane potwierdzenie, potrzebne jest poinformowanie o tym drugiej strony komendą `P_0`.

Nadpisanie niezaakceptowanych pakietów nie ma konsekwencji, dopóki druga strona nie zgłosi chęci ponownego wysłania wiadomości, która już została nadpisana. Wówczas dane w kanale zostały naruszone i komunikacja powinna rozpocząć się od nowa (punkt 3.6.1).

Rysunek 3.7 przedstawia przykładową wiadomość PUBLISH, używającą numeru pakietu.

```
▾ Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
    0011 .... = Message Type: Publish Message (3)
    .... 0... = DUP Flag: Not set
    .... .00. = QoS Level: At most once delivery (Fire and Forget) (0)
    .... ...0 = Retain: Not set
  Msg Len: 32
  Topic Length: 18
  Topic: B0:B2:1C:0B:15:CCi
▾ Properties
    Total Length: 7
    ID: User Property (0x26)
    Key Length: 1
    Key: P
    Value Length: 1
    Value:
  Message: 74657374
```

Rys. 3.7: Wiadomość PUBLISH używająca numeru pakietu

3.7.2. RTS/CTS

Standard RS232 przewiduje kontrolę przepływu [2]. Może być ona realizowana programowo, wysyłając określone znaki do drugiego urządzenia lub sprzętowo, zmieniając napięcie na wyprowadzeniach RTS oraz CTS interfejsu RS232.

System nie wspiera emulacji kontroli przepływu, ale pozwala na poinformowanie drugiej strony o zmianie stanu na pinie RTS/CTS.

Urządzenie, po wykryciu zmiany stanu na wejściu RTS z niskiego na wysoki (czyli z napięcia wysokiego na niskie), wysyła komendę `S_C` (*Set RTS*), jeżeli zmiana była odwrotna, wówczas `R_R` (*Reset RTS*). Aplikacja po odebraniu komendy musi zaktualizować stan swojego wyjścia RTS.

Aplikacja, po wykryciu zmiany stanu na wejściu CTS z niskiego na wysoki (czyli z napięcia wysokiego na niskie), wysyła komendę `S_C` (*Set CTS*), jeżeli zmiana była odwrotna, wówczas `R_C` (*Reset CTS*). Urządzenie po odebraniu komendy musi zaktualizować stan swojego wyjścia CTS.

3.7.3. Przerwanie komunikacji

Aby powiadomić o niedostępności urządzenia innych klientów, używany jest mechanizm „testamentu”, będący częścią standardu MQTT [8].

Jak opisano w punkcie 3.3.2, urządzenie określa swój testament w wiadomości CONNECT. Aby użyć testamentu, potrzebne jest określenie tematu, na który wiadomość zostanie wysłana w razie zerwania połączenia klienta z serwerem oraz samej wiadomości.

Urządzenie określa temat jako swój kanał informacyjny. Wiadomością jest komenda `D_D` (*Disconnect Detected*).

Przykładowa wiadomość jest widoczna na rysunku 3.3.