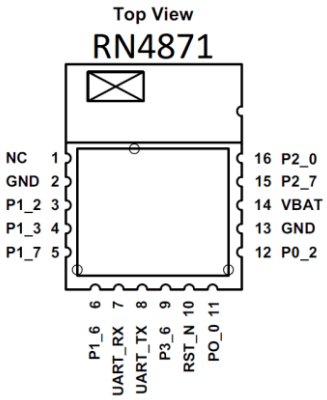
RN4871 – Informace / použití

Petr Novák / novakpe@fel.cvut.cz / 16.02.2021

Obsah

1 Zapojení pinů



(1) GND
(2) P1_2
(3) P1_3
(4) P1_7
(5) P1_6
(18) P2_0
(17) P2_7
(16) VBAT
(15) P0_2
(14) P0_0

(6) UART-RX (12) RST_N
(7) UART-TX (12) P3_6
(8) RESET (tlačítko) (11) USB-UART5
(9) USB-UART1 (10) MODE (tlačítko)

(MODE před 4k2 na VCC, tlačítko MODE spíná na GND, Pin18 = Pin10, přímo spojeno)
(RESET před 4k2 na VCC, tlačítko MODE spíná na GND)
(JUMPER zapojuje/odpojuje LEDku indikující VCC)

I2C: SCL = P1_2 / SDA = P1_3

(1) USB-UART1; (2) TxD; (3) RxD; (4) 3V3; (5) USB-UART1; (6) GND; (7)

2 Testovací modul / deska

- Slouží pro snadné testování BLE modulu RN4871.
- RESET – Tlačítko blíže ke konektoru seriál/USB.
- MODE – Tlačítko dále od konektoru seriál/USB.

3 Stručný postup použití

- Nejprve je vždy nutno modul pomocí znaků $$$ přepnout do tzv. „Command“ (dále je uvedeno pouze „Cmd“) režimu pro příjem / vysílání informací a povelů v textové podobně. Před a po zaslání těchto tří znaků nesmí být zasláno nic jiného po dobu minimálně 300ms (jinak nejsou znaky akceptovány). Tyto znaky tedy nejsou ukončeny entrem (nebyla by splněna podmínka klidu komunikace před a za nimi).
- Každý běžný povel / příkaz / data zaslaná do modulu musí být ukončeny odřádkováním (entrem) „\r“, což představuje ukončovací znak povelu pro modul.
- Po každém povelu zaslaném do modulu se musí čekat na odpověď modulu. Teprve poté lze zaslat další povel.
- O vykonání některých povelů modul informuje pomocí událostí (například na RESET) a o jiných pomocí stavu vykonání povelu (AOK / Err). Naprostá většina povelů po vykonání zobrazí prompt „CMD>“ pro možnost zaslání dalšího povelu.
    - Události jsou ve výstupním textu z modulu uzavřeny mezi „%...%“ (v tomto textu se dále doporučuje toto nastavení změnit na „[…]“).
    - Běžné odpovědi na povely obsahují „AOK“, „Err“ nebo data, následuje odřádkování a na novém řádku je „CMD>“.

4 Úvodní poznámky

- Všechny číselné / bytové hodnoty zapisované / čtené z modulu RN4871 jsou v binárním AsciiHex formátu a neobsahují úvodní „0x“.
- Při výchozím nastavení jsou textové události vystupující z modulu uzavřeny v „%...%“. Toto je velmi nevhodné, protože není možné dobře rozpoznat začátek a konec události (obsahuje

stejný znak). Proto se zde v tomto textu doporučuje změnit znaky pro uzavření události na „[…]“.
- Při přepnutí do „Cmd“ režimu není za znaky „$$$“ nikdy enter „\r“ (jinak se přepnutí nemusí nevykonat).
- V ukázkách jsou komentáře vždy před řádkem zasílaným do modulu RN4871 pro přehlednost (snad).

5 Obecné a často používané povely

- **$$$** - Přepnutí modulu do textového „Cmd“ režimu.
- **SF,2** – Uvedení modulu do továrního nastavení
- **R,1** – Reboot modulu (jako tlačítko RESET, nebo vypnutí / zapnutí)
- **V** – Zobrazení aktuálního FW v modulu
- **SN,ModulTest** – Nastavení názvu modulu, max. 20 znaků (???)
- **A** – Spuštění „Advertising“ (spustí vysílání ADVS paketů)
- **Y** – Zastavení „Advertising“ (zastaví vysílání ADVS paketů)
- **D** – Zobrazení informací o (aktuálním nastavení) modulu
- **PZ, PS, PC** – Nastavení „Serv(s)“ a „Char(s)“ (viz příklady)
- **LS** – Zobrazení seznamu „Serv(s)“ a „Char(s)“ na (připojeném) serveru
- **SHW, SHR** – Zápis / čtení „Char“ na straně serveru (viz příklady)
- **K,1** – Násilné odpojení vzdáleného / druhého zařízení.

6 Vytvoření „Serv(s)“ a „Char(s)“

- Informace vychází ze souboru „BlueToothLE-Info.docx/pdf“
- Nejprve se vygeneruje základní UUID pro „Serv(s)“ a „Chra(s)“
- Připraví se UUID čísla pro jednotlivé „Serv(s)“ a „Chra(s)“ změnou nějakého byte/bytes v základním UUID
- „Serv(s)“ se zadává pomocí povelu „PS,UUID“
- „Char(s)“ se zadává pomocí povelu „PC,UUID,params,bytes“
    - „params“ – parametry pro „Char(s)“, tedy bitové příznaky pro „Write (0x08)“, „Read (0x02)“, „Notify (0x10)“, „Indicate (0x20)“, …
    - „bytes“ – kolik bytes se bude danou „Char(s)“ přenášet (zapisovat/číst), v normálním provozu max. 20 bytes

7 Příklad nastavení a použití modulu jako „serveru“

- Jde o příklad nastavení modlu pro testovací použití.
- Modul opracuje jako „server“ a tedy poskytuje připojení a přenos dat.
- Před použitím tohoto příkladu je vhodné tlačítkem vykonat RESET modulu, aby byl správně použit povel pro nastavení modulu do „Cmd“ režimu. RESET v modulu v podstatě nic nenuluje.

/ nejprve je nutno nastavit textový „Cmd“ režim (znaky zaslat bez „\r“)
$$$

/ počkat na odpověď modulu „CMD>“
/ uvést modul do továrního nastavení, tedy vymazat všechno uživatelské nastavení
SF,2
/ počkat na odpověď „**Reboot after Factory Reset%REBOOT%**“ (informace a událost)
/ po tomto povelu se modul RESETuje a je potřeba jej opět přepnout do textového „Cmd“ režimu
$$$
/ počkat na odpověď modulu „CMD>“
/ pro snadné rozpoznání začátku/konce události se místo znaků „%“, použijí znaky „[“ a „]“
/ (povel „**S%,znak_na_začátku_události,znak_na_konci_události**“)
/ (stejný znak, tedy „%“, na začátku i na konci události je velmi nevhodný pro pársování textu)
S%,[,]
/ počkat na odpověď modulu „AOK“ a „CMD>“
/ vykonat měkký RESET modulu, aby se nastavení „**S%,[,]**“ aktivovalo
R,1
/ počkat na odpověď modulu „Rebooting[REBOOT]“
/ (událost je již uzavřena pomocí nových znaků)
/ po tomto povelu se modul RESETuje a je potřeba jej opět přepnout do textového „Cmd“ režimu
$$$
/ počkat na odpověď modulu „CMD>“

/ vymazání všech „Serv(s)“ a „Char(s)“
/ zde asi není potřeba, bylo tovární nastavení, vhodné před zadáváním nových „Serv(s)“ a „Char(s)“
PZ
/ počkat na odpověď modulu „AOK“ a „CMD>“

/ vytvoření „Serv(s)“ a „Char(s)“
/ vytvoření (první) „Serv“ …0101…, UUID = A1FC010178D340C29B6F3C5F7B2797DF
PS,A1FC010178D340C29B6F3C5F7B2797DF
/ počkat na odpověď modulu „AOK“ a poté „CMD>“
/ vytvoření „Char“ …0102…, UUID = A1FC010278D340C29B6F3C5F7B2797DF
/ „Char“ bude mít parametry „Read, Write“ a přenáší maximálně 20 bytes
PC,A1FC010278D340C29B6F3C5F7B2797DF,0A,14
/ počkat na odpověď modulu „AOK“ a poté „CMD>“
/ vytvoření „Char“ …0102…, UUID = A1FC010378D340C29B6F3C5F7B2797DF
/ „Char“ bude mít parametry „Read, Write, Notify“ a přenáší maximálně 20 bytes
PC,A1FC010378D340C29B6F3C5F7B2797DF,1A,14
/ počkat na odpověď modulu „AOK“ a poté „CMD>“

/ vytvoření (druhé) „Serv“ …0111…, UUID = A1FC011178D340C29B6F3C5F7B2797DF
PS,A1FC011178D340C29B6F3C5F7B2797DF
/ počkat na odpověď modulu „AOK“ a poté „CMD>“
/ vytvoření „Char“ …0112…, UUID = A1FC011278D340C29B6F3C5F7B2797DF
/ „Char“ bude mít parametry „Read, Write, Notify“ a přenáší maximálně 5 bytes
PC,A1FC011278D340C29B6F3C5F7B2797DF,1A,14
/ počkat na odpověď modulu „AOK“ a poté „CMD>“

/ po zadání „Serv(s)“ a „Char(s)“ je potřeba vykonat „REBOOT“ modulu
R,1

/ počkat na odpověď modulu „Rebooting[REBOOT]“
/ po tomto povelu se modul RESETuje a je potřeba je opět přepnout do textového „Cmd“ režimu
$$$
/ počkat na odpověď modulu „CMD>“
/ generování náhodné MAC adresy
&R
/ počkat na odpověď „647EA52F6384“ (například) a „CMD>“

/ nastavení ADVS paketu
/ vymazání původního ADVS paketu
IA,Z
/ počkat na odpověď modulu „AOK“ a poté „CMD>“
/ nastavit v ADVS sekci „Connect Flags“ (typ 0x01)
IA,01,05
/ počkat na odpověď modulu „AOK“ a poté „CMD>“
/ nastavit v ADVS sekci „Manufacturer Specific Data“ (typ 0xFF)
IA,FF,FFFF504E0102112233445566778899001122015566 88
/ počkat na odpověď modulu „AOK“ a poté „CMD>“

/ nastavení jména modulu
S-,RN4871-TEST
/ po zadání nazvu BLE je potřeba vykonat „REBOOT“ modulu
R,1
/ počkat na odpověď modulu „Rebooting[REBOOT]“
/ po tomto povelu se modul RESETuje a je potřeba je opět přepnout do textového „Cmd“ režimu
$$$
/ počkat na odpověď modulu „CMD>“

/ spuštění „Advertising“ (většinou se spustí automaticky, ale ne vždy)
A
/ počkat na odpověď modulu „AOK“ a poté „CMD>“

- Zde uvedený příklad nastavení / inicializace modulu je rovněž v souboru „RN4871InitServerTest.txt“ pro aplikaci „BTRN4871.exe“.
- Zadaná nastavení lze po vykonání zkontrolovat:
  - Použití znaků „[“ a „]“ místo „%“ je zřejmé z událostí vysílaných z modulu.
  - **LS** – Vypíše seznam definovaných „Serv(s)“ a „Char(s)“ včetně přiřazených „Handle“ k „Char(s)“.
  - **D** – Vypíše některé základní informace o nastavení modulu, mezi nimi i MAC adresu (BTA=X...X).
- Velmi důležitá jsou čísla tzv. „Handle“ vypsaná povelem LS pomocí nichž se využívají „Char(s)“ a tím se zapisují / čtou jejich hodnoty. Výstup povelu LS je následující (čísla „Handle“ se mohou lišit podle verze FW aktuálně použitého v modulu a samozřejmě při zadání vlastních „Serv(s)“ a „Char(s)“):

A1FC**0101**78D340C29B6F3C5F7B2797DF
A1FC**0102**78D340C29B6F3C5F7B2797DF,**0072**,0A
A1FC**0103**78D340C29B6F3C5F7B2797DF,**0074**,0A

A1FC**0103**78D340C29B6F3C5F7B2797DF,**0075**,10,0
A1FC**0111**78D340C29B6F3C5F7B2797DF
A1FC**0112**78D340C29B6F3C5F7B2797DF,**0092**,0A
A1FC**0112**78D340C29B6F3C5F7B2797DF,**0093**,10,0
END / výpis je vždy ukončen pomocí „END“

- Červená čísla udávají rozlišení UUID pro „Serv(s)“ a „Char(s)“.
- Modrá čísla udávají „Handle“ pro zápis / čtení dat „Char(s)“.
- Zelená čísla udávají „Handle“ události pro aktivaci / deaktivaci „Notify“ ze strany clienta. První „Char“ nemá příznak / možnost „Notify“ a tudíž nemá ani „Handle“ pro jeho aktivaci / deaktivaci.
- Využití čísel „Handle“ pokud modul pracuje v režimu serveru:

/ zápis hodnoty do „Char(s)“ s UUID = A1FC**0102**78D340C29B6F3C5F7B2797DF
/ SHW (server-handle-write) – Zápis ze strany serveru
/ 0x0072 – Handle pro zápis / čtení „Char“
/ 1122334455 (0x11,0x22,0x33,0x44,0x55) – Zapisovaná hodnota, 5 bytes
SHW,0072,1122334455
/ odpověď je „OK“ a poté „CMD>“

/ čtení hodnoty z „Char(s)“ s UUID = A1FC**0102**78D340C29B6F3C5F7B2797DF
/ SHR (server-handle-read) – Čtení ze strany serveru
/ 0x0072 – Handle pro zápis / čtení „Char“
SHR,0072
/ odpověď je „1122334455“ (původně zapsaná hodnota) a poté „CMD>“

- Informace o připojení vzdáleného clienta:

/ pokud se někdo na modul (server) připojí, vznikne událost
/ 0 = public address, 5CF3707167F9 – MAC adresa kdo se připojil
[CONNECT,0,5CF3707167F9]
/ pokud client povolí (zaregistruje si) „Notify“, vznikne událost
/ „Handle“ pro zapnutí / vypnutí „Notify“ je vždy o jednu větší než „Handle“ pro zápis/čtení hodnot
/ jde pouze o informaci, že má někdo zájem o změnu hodnoty v „Char“ s „Handle“ 0x0074
[WC,0075,0100]
/ pokud client zapíše do „Char“ data, tak na serveru vznikne událost
/ nějaký client do „Char“ s „Handle“ 0x0074 zapsal hodnotu 0x11
[WV,0074,11]
/ pokud client vypne na „Char“ s „Handle“ 0x0074 „Notify“ vznikne událost
[WC,0075,0000]
/ pokud se klient odpojí, nebo spojení se nějak přeruší, vznikne na serveru událost
[DISCONNECT]

Příklad využití modulu jako „clienta“

- Modul opracuje jako „client“ a tedy se připojuje na server a získává data.

- Před použitím tohoto příkladu je vhodné tlačítkem vykonat RESET modulu, aby byl správně použit povel pro nastavení modulu do „Cmd“ režimu. RESET v modulu v podstatě nic nenuluje.
- Zde jsou popsány pouze základní úkony pro přenos dat z jednoho modulu RN4871 do druhého.
- Pro další práci musí být modul samozřejmě přepnut v textovém „Cmd“ režimu a je vykonán povel „S%,[,]“.

/ Spuštění skenování / hledání dalších BLE zařízení
F
/ postupně (na řádcích) jsou zobrazeny informace o nalezených BLE zařízeních
/ D88039F612BE – MAC adresa nalezeného zařízení, 0 – jedná se o veřejnou adresu
[D88039F612BE,0,,,AA]
/ zastavení skenování / hledání BLE zařízení
X
/ připojení na vzdálené BLE zařízení (server)
/ 0 – zadána je veřejná adresa, D88039F612BE – adresa cílového BLE zařízení
C,0,D88039F612BE
/ na clientu vznikne událost „[CONNECT,0,D88039F612BE]“
/ připojeno na BLE zařízení s veřejnou adresou „D88039F612BE“
/ (na serveru vznikne stejná událost obsahující adresu clienta)

/ zjištění všech „Serv(s)“ a „Chra(s)“, které poskytuje „server“
CI
/ vypsání seznamu zjištěných „Serv(s)“ a „Chra(s)“ poskytovaných „serverem“
LC
/ výpis je velmi podobný jako v případě „serveru“ při použití povelu „LS“
…
A1FC010178D340C29B6F3C5F7B2797DF
A1FC010278D340C29B6F3C5F7B2797DF,0072,0A
A1FC010378D340C29B6F3C5F7B2797DF,0074,0A
A1FC010378D340C29B6F3C5F7B2797DF,0075,10
A1FC011178D340C29B6F3C5F7B2797DF
A1FC011278D340C29B6F3C5F7B2797DF,0092,0A
A1FC011278D340C29B6F3C5F7B2797DF,0093,10
…
END / výpis končí označením „END“
/ pozor že výpis může obsahovat mnohem více „Serv(s)“ a „Chra(s)“ než bylo na serveru nastaveno
/ server může poskytovat některé výchozí „Serv(s)“ a „Chra(s)“, které nelze odstranit

/ zaslání / zápis hodnoty do „Char“ na serveru vzdáleném zařízení (clientem)
/ CHW (client-handle-write) – Zápis ze strany clienta (clientem)
CHW,0074,1122334455
/ na serveru vznikne událost „[WV,0074,1122334455]“

/ čtení hodnoty z „Char“ ze serveru
/ CHR (client-handle-read) – Čtení ze strany clienta (clientem)
CHR,0074

/ zobrazí se přečtená hodnota (na serveru žádná událost nevzniká)

- Pokud server zapíše hodnotu do „Char(s)“, tak na clientu nevznikne žádná událost informující o zápisu nové hodnoty serverem do „Char(s)“.
- Pokud však „Char(s)“ na serveru poskytuje „Notify“, tak jej může client využít / aktivovat.

/ zapnutí „Notify“ clientem od zápisu nové hodnoty serverem do „Char“
/ „Handle“ pro aktivaci / deaktivaci „Notify“ clientem
/ toto „Handle“ je (snad) vždy o jednu větší než „Handle“ pro čtení/zápis „Char“
CHW,0075,0100
/ na serveru vznikne událost „**[WC,0075,0100]**“
/ nyní když se na serveru vykoná „**SHW,0074,1122334455**“
/ na clientu vznikne událost (změna na „Char“ s Handle 0x0074 a nová hodnota je 0x11,0x22,…)
[NOTI,0074,1122334455]

/ odpojení od „serveru“
K,1
/ na clientu i na serveru vznikne událost „[DISCONNECT]“

8 Využití skriptu

BLE modul RN4871 umožňuje vložit tzv. „skript“ obsahující množství povelů a ten autonomně vykonávat. Omezení / parametry skriptu jsou zhruba následující:

- Celková velikost skriptu nesmí přesáhnout 1000 bytes
- Jedna řádka skriptu nesmí přesáhnout 100 bytes
- Celý skript nesmí přesáhnout 50 řádek
- Řádek s komentářem začíná znakem „#“ (ukusuje však z celkové velikosti skriptu)

Ve skriptu mohou být definovány tyto události (následované posloupností povelů). Čísla uvedená v závorkách jsou vysvětlena dále:

- **@PW_ON** (00) – Spuštěno po zapnutí napájení modulu
- **@TMR1** (01) / **@TMR2** (02) / **@TMR3** (03) – Událost od vypršení intervalu časovače
- **@CONN** (04) / **@DISCON** (05) – Někdo se připojil / odpojil
- **@PIO1H** (06) / **@PIO1L** (07) / **@PIO2H** (08) / **@PIO2L** (09) / **@PIO3H** (0A) / **@PIO3L** (0B) – Změna H/L na pinu

V skriptu je možno použít téměř všechny povely určené pro modul a rovněž tři časovače označené „SM,1,…“, „SM,2,…“, „SM,3,…“ například:

- **SM,X,YYYY** – Spuštění časovače „X“ s periodou „YYYY“, přesněji řečeno s počtem kroků:
  - Pro časovač „1“ je krok „640ms“ (perioda = YYYY x 640ms)
  - Pro časovač „2“ a „3“ je krok „10ms“ (perioda = YYYY x 10ms)
- **SM,X,0000** – Zastavení časovače „X“

Se skriptem jsou spojeny tyto povely:

- **LW** – Výpis obsahu akruálního skriptu
- **WC** – Vymazaní / odstranění současného skriptu z modulu
- **WW** – Zápis nového skriptu do modulu

- **WP** – Pozastavení činnosti současného skriptu v modulu
- **WR** – Spuštění aktuálního skriptu umístěného v modulu. Spuštění od některé definované události podle čísla dříve uvedeného za příslušnou událostí (například **WR,00** – spuštění od značky „**@PW_ON**“, tedy v podstatě od začátku).

8.1 A) Vysílající opakovaně nějakou simulovanou hodnotu.

Ve skriptu zřejmě nelze udělat inkrement / přičtení hodnoty. Tento nedostatek lze však pro testovací účely vhodně obejít. Skript je umístěn v souboru „RN4871ServerSendValues.txt“.

Činnost skriptu:

- Pin 1.2 signalizuje zda je někdo připojen (výstup log. = 1).
- Po připojení klienta se nastaví P1_2 = 1. Spustí se „Timer1“.
- Po uplynutí „Timer1“ se zapíše číslo 0x11 na handle 0x0072. Spustí se „Timer2“.
- Po uplynutí „Timer2“ se zapíše číslo 0x22 na handle 0x0072. Spustí se „Timer3“.
- Po uplynutí „Timer3“ se zapíše číslo 0x33 na handle 0x0072. Spustí se (znova) „Timer1“.
- Po odpojení klienta se nastaví P1_2=0 a zastaví se všechny „TimerX“.

Požadavky:

- Vytvoření charakteristiky s „handle“ 0x0072 jako READ (čtení klientem), NOTIFY (událost pro klienta).

```
@PW_ON      / slouží pro spuštění skriptu po zapnutí nebo povelem „WR,00“
SW,0A,00    / odpojení speciálních funkcí z pinu 1_2
@CONN       / událost při připojení klienta (někdo se připojil)
|O,08,08    / pin P1_2 = 1 (signalizace připojení klienta)
SM,1,0002   / spustí se časovač „1“ na čas „2x640ms = 1.3s“
@TMR1       / událost od uplynutí periody časovače „1“
SM,1,0000   / zastavení časovače „1“
SHW,0074,11 / vyslání hodnoty 0x11 na „Chas“ s „Handle“ 0x0072
SM,2,1770   / spustí se časovač „2“ na čas „6000x10ms = 1m“
@TMR2       / událost od uplynutí periody časovače „2“
SM,2,0000   / zastavení časovače „2“
SHW,0074,22 / vyslání hodnoty 0x22 na „Chas“ s „Handle“ 0x0072
SM,3,1770   / spustí se časovač „3“ na čas „6000x10ms = 1m“
@TMR3       / událost od uplynutí periody časovače „3“
SM,3,0000   / zastavení časovače „3“
SHW,0074,33 / vyslání hodnoty 0x33 na „Chas“ s „Handle“ 0x0072
SM,1,005D   / spustí se časovač „1“ na čas „6000x10ms = 1m“
@DISCONN    / událost při odpojení clienta (někdo se odpojil)
|O,08,00    / pin P1_2 = 0 (signalizace odpojení clienta)
SM,1,0000   / zastaveni časovače „1“, „2“ i „3“
SM,2,0000
SM,3,0000
```

8.2 B) Zápis / čtení pinu přes „Char-Write“ a „Char-Read“.

Označení „%0072“ znamená přiřazení „Handle“ 0x0072 nějakému parametru. V tomto případě povel „|O,08,XX“ zapisuje hodnotu XX na pin P1_2 (viz poznámky). Zápis „|O,08,%0072“ tedy udává, že hodnota zaslaná klientem na „Char“ s „Handle“ 0x0072 bude automaticky (jakmile je na „Char“ zapsána) použita jako parametr pro zápis na pin P1_2, tedy jako parametr / hodnota do povelu „|O,08,XX“.

Pro čtení hodnoty z pinu P1_3 by mělo být zapsáno „%0075=|I,10“. Tedy čtení klientem „Char“ s „Handle“ 0x0075 by mělo vrátit stav pinu P1_3 (tedy jeho stav bude v bit4 vrácené hodnoty). Toto jsem však nerozchodil. Lze to obejít pomocí časovače a čtení stavu portu přes proměnnou a poté zápis hodnoty do „Char“. Toto řešení obsahuje jednu nevýhodu, pokud je na „Char“ nastaveno „Notify“, tak každý zápis této (i stejné) hodnoty se přenáší na klienta a generuje událost o zápisu (a to i stejné) nové hodnoty.

Popis činnosti:

- Při připojení se nastaví výchozí stav P1_2 = 0.
- Zápisem hodnoty na „handle“ 0x0072 se nastaví hodnota pinu „P1_2“. Zapisují se hodnoty 0x00 (pro P1_2 = 0) nebo 0x04 (pro P1_2 = 1). Spustí se „Timer1“.
- Po vypršení „Timer1“ je čtena hodnota P1_2 a ta se zapíše na „handle“ 0x0074. Ukládají se hodnoty 0x00 (pokud P1_2 = 0) nebo 0x04 (pokud P1_2 = 1).
- Při odpojení se nastaví výchozí stav P1_2 = 0. Zastaví se „Timer1“.

Požadavky:

- Vytvoření charakteristiky s „handle“ 0x0072 jako WRITE (zápis klientem).
- Vytvoření charakteristiky s „handle“ 0x0074 jako READ (čtení klientem), NOTIFY (událost pro klienta).

```
@PW_ON        / slouží pro spuštění skriptu po zapnutí nebo povelem „WR,00“
SW,0A,00      / odpojení speciálních funkcí z pinu 1_2
SW,0B,00      / odpojení speciálních funkcí z pinu 1_3
|O,08,%0072   / zápisem na „Handle“ 0x0072 hodnoty 0x08/0x00 bude nastaveno P1_2 = 1/0
%0075=|I,10   / čtením „Handle“ 0x0075 bude čten stav pinu P1_3 (stav je v bit4)
@CONN         / událost při připojení klienta (někdo se připojil)
|O,08,00      / pin P1_2 = 0 (výchozí stav)
SM,1,0005     / spuštění časovače „1“ s periodou 5x640ms=3.2s
@DISCONN      / událost při odpojení klienta (někdo se odpojil)
SM,1,0000     / zastavení časovače „1“
|O,08,00      / pin P1_2 = 0 (výchozí stav)
@TMR1         / událost od uplynutí periody časovače „1“
$VAR1 = |I,10 / čtení stavu z pinu P1_3
SHW,0074,$VAR1 / zápis čteného stavu z pinu P1_3 do „Char“ s „Handle“ 0x0074
SM,1,0005     / (opětovné) spuštění časovače „1“ s periodou 5x640ms=3.2s
```

8.3 C) Zápis pinu přes „Char-Write“, zachycení změny na pinu a využití „Char-Notify“.

Aby nebylo potřeba neustále zasílat čtenou hodnotu z pinu P1_3 do klienta periodicky, lze pouze detekovat její změnu a tu zaslat.

Popis činnosti:

- Po zapnutí odpojení speciálních funkcí a P1_2 a inicializace P1_3 jako „Pin Trigger 1“, tedy detekce změny svého stavu.
- Po připojení nastavení P1_2 do výchozího stavu, tedy na „P1_2 = 0“.
- Zápisem hodnoty do „handle“ 0x0072 se nastaví hodnota pinu „P1_2“. Zapisují se hodnoty 0x00 (pro P1_2 = 0) nebo 0x04 (pro P1_2 = 1). Spustí se „Timer1“.
- Při detekci změny na pinu P1_3 („Pin Trigger 1“) z „0“ a „1“ se na handle „0x0074“ zapíše hodnota 0x01 (tedy hodnota 1).
- Při detekci změny na pinu P1_3 („Pin Trigger 1“) z „1“ a „0“ se na handle „0x0074“ zapíše hodnota 0x00 (tedy hodnota 0).
- Po odpojení nastavení P1_2 do výchozího stavu, tedy na „P1_2 = 0“.

Požadavky:

- Vytvoření charakteristiky s „handle“ 0x0072 jako WRITE (zápis klientem).
- Vytvoření charakteristiky s „handle“ 0x0074 jako READ (čtení klientem), NOTIFY (událost pro klienta).

```
@PW_ON        / slouží pro spuštění skriptu po zapnutí nebo povelem „WR,00“
SW,0A,00      / odpojení speciálních funkcí z pinu 1_2
SW,0B,09      / nastavení funkce pinu P1_3 na „Pin Trigger 1“
|O,08,%0072   / zápisem na „Handle“ 0x0072 hodnoty 0x08/0x00 bude nastaveno P1_2 = 1/0
@CONN         / událost při připojení klienta (někdo se připojil)
|O,08,00       / pin P1_2 = 0 (výchozí stav)
@DISCONN      / událost při odpojení klienta (někdo se odpojil)
|O,08,00       / pin P1_2 = 0 (výchozí stav)
@PIO1H        / událost změny 0->1 na „Pin Trigger 1“, tedy na P1_3 z log.0 na log.1
$VAR1 = |I,10   / čtení stavu z pinu P1_3 (?nějak nefunguje?)
SHW,0074,$VAR1 / zápis čteného stavu z pinu P1_3 do „Char“ s „Handle“ 0x0074
(SHW,0074,01 – toto půjde určitě)
@PIO1L        / událost změny 1->0 na „Pin Trigger 1“, tedy na P1_3 z log.1 na log.0
$VAR1 = |I,10   / čtení stavu z pinu P1_3 (?nějak nefunguje?)
SHW,0074,$VAR1 / zápis čteného stavu z pinu P1_3 do „Char“ s „Handle“ 0x0074
(SHW,0074,00 – toto půjde určitě)
```

8.4 D) Vysílání hodnot z akcelerometru MPU9255.

Příklad skriptu pro připojení pohybového senzoru MPU9255 na modul RN4871 pomocí I2C pro vysílání pohybových dat, například do PC. Činnost je následující:

- Po zapnutí (@PW_ON) se inicializuje I2C, nastaví se MPU9255 do výchozího stavu, přečte se jeho identifikace (ta se zahodí) a nastaví výchozí rychlost zasílání dat na periodu 1s. Zatím se data nevysílají.
- Po připojení (@CONN) klienta se začnou data vysílat s nastavenou periodou. Vysílá se 6bytes, tedy 2bytes osa X, 2bytes osa Y a 2bytes osa Z. Jsou přenášena data přímo vyčtená z MPU9255, tedy v 16bits dvojkovém doplňku. Po připojení jsou první data odeslána vždy za 2s.
- Nyní lze nastavit jinou rychlost vysílání dat zápisem do charakteristiky 0x0072. Hodnota musí být 16bits AsciiHex. Jednotka 1 znamená 10ms. Tedy například pro hodnoty 0x0064 (100d) je tedy 100x10ms=1s.

- Po odpojení (@DISCONN) klienta se data vysílat přestanou a opět se nastaví výchozí rychlost vysílání na 1s.

Podmínky pro činnost:

- Vytvoření (jedné) charakteristiky s „handle“ 0x0072 přijímající 2bytes s parametry WRITE (zápis klientem), READ (čtení klientem), NOTIFY (událost do klienta o změně hodnoty na serveru).
    - WRITE – Perioda vysílání dat. Číslo do časovače „2“.
    - READ – Hodnoty aktuálních pohybových dat.

```
@PW_ON        / slouží pro spuštění skriptu po zapnutí nebo povelem „WR,00“
SW,0A,00      / odpojení speciálních funkcí z pinu 1_2 (bude použit pro I2C)
SW,0B,00      / odpojení speciálních funkcí z pinu 1_3 (bude použit pro I2C)
]A,D0,3       / inicializace I2C, adresa I2C zařízení je „D0“, frekvenci sběrnice je 50kHz
]W,6B00       / probuzeni obvodu zápisem do registru „PWR_MGMT_1 (0x6B)“ hodnoty „0x00“
SHW,0072,0064 / výchozí rychlost pro vysílání hodnot je 100x10ms=1s (pro příští připojení)
/]X,75,01     / čtení registru „WHO_AM_I (0x75)“
@CONN         / událost při připojení klienta (někdo se připojil)
SM,2,00CB     / první spuštění časovače „2“ s periodou „200x10ms = 2000ms“
@DISCONN      / událost při odpojení klienta (někdo se odpojil)
SM,2,0000     / zastavení časovače „2“
SHW,0072,0064 / výchozí rychlost pro vysílání hodnot je 100x10ms=1s (pro příští připojení)
@TMR2         / událost od uplynutí periody časovače „2“
$VAR1 = ]X,3B,06 / čtení dat akcelerometru v ose X (2bytes), Y (2bytes) a Z (2bytes)
                 / nastavení / zápis adresy pro čtení „ACCEL_XOUT_H (0x3B)“ a čtení 6 bytes
SHW,0074,$VAR1 / zápis dat do „Char“ s „Handle“ 0x0074 (musí být min. 6bytes)
$VAR2=SHR,0072 / vyzvednutí hodnoty do časovače pro rychlost vysílání hodnot
SM,2,$VAR2    / (opětovné) spuštění časovače „2“ s periodou … (podle aktuálního nastavení)
```

8.5 E) Příklad využití funkcí.

Modul umožňuje vytvořit až tři interní funkce s názvy „?FUNC1“, „?FUNC2“ a „?FUNC3“. Funkcím mohou být předány vstupní parametry, které jsou označeny „$PM1“, „$PM2“, „$PM3“, „$PM4“ a „$PM5“. Pokud je předáno méně než pět parametrů, tak jsou hodnoty zbývajících parametrů prázdné. Parametry pro funkci jsou odděleny čárkou, ale nejde o běžné hodnoty zapisované do „Char(s)“. Hodnota předávaná funkci je sice binární AsciiHex, ale musí být zaslaná jako text obsahující kódy znaků představující tyto číslice. Například pokud je potřeba funkci předat dva parametry „0050“ a „0010“:

- Požadované parametry pro funkci: „0050,0010“ („$PM1“ bude „0050“, „$PM2“ bude „0010“).
- Do „Char“ je nutno clientem zapsat: „303035302C30303130“ (číslice „0“ = AsciiHex kód znaku je „0x30“, číslice „5“ = AsciiHex kód znaku je „0x32“, čárka „,“ = AsciiHex kód znaku je „0x2C“, číslice „1“ = AsciiHex kód znaku je „0x31“).

Důvodu je následující. Pokud jsou dva moduly vzájemně spojeny, tak na clientu je potřeba pro vyvolání funkce na serveru zaslat požadavek „SHW,handle,func_params“. Povel „SHW“ však povoluje zadat pouze dva parametry / hodnoty („SHW,handle,value“), pokud by se zapsalo „SWH,handle,func_param_1, func_param_2,…“, tak by povel „SHW“ obsahoval více jak dvě hodnoty / parametry a tím by byl vyhodnocen jako neplatný. Z tohoto důvodu je tedy potřeba ze všech

parametrů pro volanou funkci vytvořit pouze jeden jediný parametr pro povel „SHW“. Proto jsou parametry pro volanou funkci v podstatě přes „Char“ zaslány jako nic neříkající text a teprve až při skutečném volání funkce na serveru jsou opět převedeny na správné parametry / číselné hodnoty oddělené čárkou.

```
@PW_ON        / slouží pro spuštění scriptu po zapnutí nebo povelem „WR,00“
%0072=?FUNC1 / „?FUNC1“ je aktivována / spuštěna zápisem hodnoty do „Handle“ 0x0072
?FUNC1        / zde začíná definice obsahu funkce „?FUNC1“
              / první parametr se vyzvedne do proměnné „$PM1“ a druhý do „$PM2“
              / parametry jsou zadány jako text a odděleny čárkou
SHW,0074,$PM1 / první vyzvednutý parametr z „$PM1“ se zapíše do „Char“ s „Handle“ 0x0074
SHW,0092,$PM2 / druhý vyzvednutý parametr z „$PM0“ se zapíše do „Char“ s „Handle“ 0x0092
```

Poznámky ke scriptu

- Ve „scriptu“ nesmí být definice „Serv(s)“ a „Char(s)“, tedy povely PZ, PS a PC! Script tyto definice / povely nevykoná a potom nelze přes tyto „Serv(s)“ a „Char(s)“ s modulem komunikovat!
- Povely ve scriptu se vykonávají vždy od určitého návěstí / události (když je aktivováno) až do dosažení dalšího jiného návěstí / události. Script návěstí / událost ve svém vykonávání (snad nikdy) nepřeleze.
- Ve scriptu mohou být použity pouze dvě proměnné a to označené „$VAR1“ a „$VAR2“. Žádné jiné proměnné, tedy ani vlastní definované, nejsou dovoleny.
- Pokud je potřeba stručně ověřit zda nějaká vytvořená událost ve scriptu skutečně vykoná to co je potřeba, tak ji lze přímo spustit pomocí jejího návěstí. Například povel „WR,06“ spustí script od události „@PIO1H“, tedy v podstatě scriptu (programově) nasimuluje vznik / generování této události.

9 Další poznámky

- Experimentálně bylo zjištěno, že po znacích „$$$“ pro přepnutí do „Cmd“ režimu může následovat odřádkování („\r“).
- Pokud je „Char“ pro čtení (Read) a není do ní zapsána zatím žádná hodnota (serverem/clientem), tak povel pro čtení vrací textovou hodnotu „N/A“ (Not-Available). Tedy nikoli „Err“ nebo nějaké výchozí číslo (třeba 0x00).
- Při povelu „|O,XX,YY“ pro zápis / čtení úrovně pinu se vychází z této tabulky:

Bitmap	RN4870 Pins	RN4871 Pins
01	P2_2	—
02	P2_4	—
04	P3_5	—
08	P1_2	P1_2
10	P1_3	P1_3

XX určuje, jaké piny jsou „bitově“ vybrány a YY určuje jaká „bitová“ hodnota bude na tyto piny uložena (přes všechny vybrané piny). První číslo je v podstatě (bitová) maska určující jaké piny (stavy pinů) podle tabulky budou ovlivněny druhým číslem, tedy v podstatě jaké bity z druhého čísla budou použity. Lze si tedy představit, jako by piny byly na jednom portu (bit0=P2_2, bit1=P2_4, bit2=P3_5, bit3=P1_2, bit4=P1_3), první parametr určuje masku pro

použité bity na tomto portu a druhé číslo určuje, jaká hodnota přes tuto masku na bity tohoto portu bude zapsána (XX & YY => bity_portu). Zde jsou některé příklady:

- |O,08,08 – P1_2=1 (maska bit3=1 a bude uložena hodnota 08 kde je bit3=1)
- |O,08,00 – P1_2=0 (maska bit3=1 a bude uložena hodnota 00 kde je bit3=0)
- |O,18,08 – P1_2=1 a P1_3=0 (maska bit3=1 a bit4=1 a bude uložena hodnota 08 kde je bit3=1 a bit4=0)

Stejný princip platí i pro čtení pinů povelem „|I,XX“. Všechny piny jsou čteny přes masku XX a tedy pouze ty piny mající v bitové masce bit=1 se uloží do výsledku, ostatní bity / piny jsou vždy „0“ protože mají v masce hodnotu „0“ (bity_portu & XX => YY).

- Funkci některých pinů lze nastavit podle vlastní potřeby. Slouží k tomu povel „SW,pin_index,pin_func“. První parametr „pin_index“ lze získat z následující tabulky a udává jakému pinu je / bude funkce přiřazena. Pouze piny v tabulce jsou dostupné pro nastavení nějaké vlastní funkce.

Pin Index	RN4870 Pins	RN4871 Pins	Default Function
00	P07	—	Low Battery Indication
01	P10	—	Status 2
02	P11	—	Status 1
03	P22	—	None
04	P24	—	None
05	P31	—	RSSI Indication
06	P32	—	Link Drop
07	P33	—	UART Rx Indication
08	P34	—	Pairing
09	P35	—	None
0A	P12	P12	None
0B	P13	P13	None
0C	—	P16	UART Rx Indication
0D	—	P17	None

Druhý parametr „pin_fun“ určuje funkci, která bude pinu přiřazena, tedy jak se bude pin chovat. Pouze funkce uvedené v následující tabulce lze pinu / pinům přiřadit.

Function Index	Function Description
00	None
01	Low Battery Indication
02	RSSI Indication
03	Link Drop
04	UART RX Indication
05	Pairing
06	RF Active Indication
07	Status 1
08	Status 2
09	Pin Trigger 1
0A	Pin Trigger 2
0B	Pin Trigger 3
0C	UART Mode Switch: Rising edge for UART Transparent mode; falling edge for Command mode.

Například povelem „SW,0B,09“ se přiřadí pinu P1_3 (pin_index = 0x0B) funkce „Pin Trigger 1“ (pin_func = 0x09) a tím lze ve scriptu využít události „@PIO1H“ a „@PIO1L“.

- Před použitím pinů P1_2 a P1_3 například zejména pro I2C je potřeba je nejprve (de)inicializovat. Tedy použít povely SW,0A,00 (pro P1_2) a SW,0B,00 (pro P1_3).
- Nastavení periody časovače „SM,1,XXXX“, “SM,2,XXXX“ a „SM,3,XXXX“ znamená pouze na jednu periodu, tedy jedno „vypršení“. Po vypršení nastaveného času se časovač zastaví, perioda se tedy neopakuje. Pro další periodu je potřeba časovač opět nastavit / spustit.
- Handle pro „Char(s)“ jsou velmi důležité. Je však poněkud složité výstup povelu „LS“ párovat v (malém) procesoru. Jak bylo zjištěno, tak jsou tato čísla „Handle“ pořád stejná i když se

opakovaně vytváří ty samé „Serv(s)“ a „Char(s)“ po nulování modulu. Z tohoto důvodu postačuje tato čísla zjistit například pomocí terminálu (vytvořit „Serv(s)“ a „Char(s)“ a přečíst jejich nastavení pomocí povelu „LS“) a poté je v programu na pevno použít. Pozor však na skutečnost, že při aktualizaci FW v RN4871 nebo vytvoření zcela nových / jiných „Serv(s)“ a „Char(s)“ s jinými UUID se mohou čísla „Handle“ změnit.

- Modul RN4871 je vhodný pouze pro jedno spojení, nebo případně pro spojení dvou modulů navzájem. Například událost „[DISCONNECT]“ neinformuje, jaké zařízení se odpojilo (pokud jich bude připojeno více).
- Všechny dostupné (textové) povely a možnosti modulu RN4871 jsou uvedeny v dokumentu „RN4870/71 Bluetooth Low Energy Module User’s Guide (DS50002466C)“.

10 Technické poznámky:

- Může se stát, že modul nechce vykonat povel pro „tovární nastavení“ nebo i jiný pro jeho reset / nulování. Nejčastější příčina spočívá, že je na modul někdo napojen (modul je s někým komunikačně spojen). Samotné ukončení aplikace na telefonu, ještě neznamená ukončení spojení s modulem (systém telefonu může spojení stále držet na pozadí pro jeho opětovné možné použití, například brzké znovu spuštění aplikace). Rovněž manuální reset modulu RN4871 neznamená nutné ukončení / přerušení spojení. Po resetu modulu se mobilní zařízení po přerušení tohoto spojení může k modulu po jeho resetu okamžitě automaticky opět připojit, což lze velmi dobře poznat z průběhu zobrazené komunikace.
- Stručný princip činnosti I2C sběrnice. Komunikace přes I2C sběrnici se skládá ze zaslání nejprve „adresy(bits7-1)+W/R(bit0)“, což je adresa cílového zařízení, s nímž se bude na sběrnici I2C komunikovat a indikace zde se bude do něho zapisovat nebo číst. Dále následují data / bytes pro zápis nebo čtení do/z cílového I2C obvodu. Velmi často je první byte v datech „adresa“ od které se další data / bytes zapisují nebo čtou do/ze zařízení na I2C sběrnici. Příklad použití pro MPU9255:
    - Nastavení adresy cílového obvodu na I2C sběrnici. MPU9255 má adresu „0x68“, ta je 7 bitová a je potřeba ji převést do bits7-1 (bit0 je příznak W/R), tedy posunout vlevo, výsledkem je hodnota „D0“. Z tohoto vychází povel „]A,D0,3“.
    - Zápis do cílového obvodu na I2C sběrnici. Při zápisu se jako první byte uvádí adresa paměti / registru v cílovém I2C obvodu a další bytes se zapisují od této adresy s automatickým inkrementem adresy zápisu. Z tohoto je následující formát povelu „]W,6B00“. Zapisuje se do obvodu na adresu / registru „0x6B“ a to hodnota „0x00“.
    - Čtení z cílového obvodu na I2C sběrnici. Při čtení se jako první byte uvádí adresa paměti / registru v cílovém I2C obvodu a poté se čtou z obvodu bytes od této adresy s automatickým inkrementem adresy čtení. Nejprve je tedy potřeba do cílového obvodu jeden byte zapsat, tedy adresu pro budoucí čtení, a teprve poté lze vyčíst požadovaný počet bytes. Z tohoto je následující formát povelu „]X,3B,06“. Nejprve se do obvodu zapíše adresa registru pro čtení „0x3B“ a teprve potom lze od této adresy vyčíst postupně 6 bytes (v podstatě 6 bytových registrů).

11 Extra nastavení

- Některé vlastnosti / chování modulu je potřeba nastavit extra. K tomuto slouží povel „**SR,XXXX**“. Hodnota „XXXX“ je složena z bitových příznaků. Některé užitečné jsou zde uvedeny (pro skutečnou aktivaci po nastavení příznaků musí být vždy reboot):
    - **0x0080** (Reboot after Disconnection) – Po odpojení clienta se vykoná reboot modulu (v podstatě inicializace jako po zapnutí).
    - **0x0040** (Running Script after PowerOn) – Po zapnutí modulu se automaticky spustí vložený script (není jej tedy nutno spouště manuálně například pomocí WR,00).
    - **0x0010** (Dala Length Extension - DLE) – Pozor, při nastavení bitu je DLE vypnuto / blokováno (nikoli povoleno)!

12 Verze

2021.02.16
- Přidán povel LW do popisu skriptu.