LWM – MQTT Gateway

1. Opis

Projekt integruje LWM stack od firmy Atmel a Ethernet NIC Wiznet 5500 pripojený na kit s MCU Atmel ATMEGA256FR2. Výsledné zariadenie slúži brána medzi sietami LWM a Ethernet, resp. MQTT.

2. Popis fungovania

Po zapnutí, zariadenie ako prvé inicializuje LWM stack a periférie MCU – SPI, Timer, Watchdog. Následne prebehne inicializácia sieťovej karty podla zadaných nastavení (vid. Kapitola 3) a zariadenie vypíše stav karty a jej nastavenia.

Inicializáciu sieťovej karty nasleduje pokus o spojenie so vzdialeným MQTT serverom, podľa nastavení v zdrojovom kóde (vid. Kapitola 3). V prípade neúspešného pripojenia, zariadenie vypíše chybovú hlášku a reštartuje sa.

V prípade úspešného pripojenia, pokračuje zariadenie prihlásením k daným MQTT topicom pomocou MQTTSubscribe. Ak je prihlásenie úspešné, zariadenie do konzole vypíše „Subscribed “názov topicu””.

Po prihlásení do MQTT topicu, zariadenie prejde do cyklu, v ktorom pravidelne kontroluje, či mu na LWM neprišli nové správy. V intervale mqtt_timer zariadenie spúšta funkciu MQTTYield, pomocou ktorej dostane zariadenie priestor na spracovanie príchodzích MQTT správ.

V prípade prijatia správy cez LWM, je nastavený príznak data_ready. Pri ďalšom prechode while cyklom sú data spracované do JSON formátu a odoslané na príslušný MQTT topic podľa typu dát pomocou funkcie mqtt_pub.

Obdobne sú spracované aj príchodzie MQTT konfiguračné správy, ktoré sú prijaté vo formáte JSON a sú spracované externou knižnicou.

3. Nastavenie

Všetky nastavenia sa nachádzajú na začiatku súboru „main.c“.

- **#define CLIENT** – nastavuje názov zariadenia pre MQTT server
- **#define USER** – nastavuje meno používateľa pre MQTT server
- **#define PASSWORD** – nastavuje heslo používateľa pre MQTT server
- **#define PUBLISH_CONFIG_0** – nastavuje topic pre konfiguračné správy
- **#define SUBSCRIBE_TOPIC** – nastavuje topic na ktorom bude zariadenie odoberať správy
- **#define PUBLISH_TOPIC** – nastavuje základný topic, ku ktorému sa pridajú jednotlivé témy podľa zvolených senzorov
- **mqtt_target –** IP adresa MQTT serveru
- **netInfo –** nastavenia sietovej karty – IP adresa, Maska podsiete, MAC adresa, Gateway adresa.

Pod „// define sensors“ sa nachádzajú definície zariadení a ich senzorov vo forme poľa s názvami veličín. Pre definíciu ďalších zariadení je potrebné použiť nasledujúcu šablónu:

```
char *zariadenie[50] = {"Vel1", "Vel2", "Vel3","Vel4","Vel5","Vel6","Vel7","Vel8"};
```

, kde „Vel***n***“ je názov veličiny. Zároveň je potrebné upraviť funkciu „create_json“ pridaním ďalšieho switch case pre zariadenie s novou maskou.

4. Funkcie

- **bool appDataInd(NWK_DataInd_t *ind)**
  - spracovanie prijatých dát z LWM
  - *ind - pointer na prijaté dáta.
- **static bool appAddrInd(NWK_DataInd_t *ind)**
  - získa cluster ID od zariadenia so senzormi – použité na odlíšenie jednotlivých zariadení a ich senzorov.
  - *ind - pointer na prijaté dáta.
- **void mqtt_pub(Client* mqtt_client, char * mqtt_topic, char * mqtt_msg, int mqtt_msg_len)**
  - odosiela správy na MQTT server
  - **char * mqtt_topic –** mqtt topic
  - **char * mqtt_msg** - mqtt správa, ktorú chceme odoslať
  - **int mqtt_msg_len** – dĺžka mqtt správy
- **int32_t MQTTSubscribe(Client* c, const char* topicFilter, enum QoS qos, messageHandler messageHandler)**
  - prihlasuje sa k odberu správ s daným topicom
  - **const char* topicFilter –** mqtt topic
  - **enum QoS qos -** nastavenie QoS pre MQTT
  - **messageHandler** – callback na spracovanie správ
- **static void create_json()**
  - vytvorí struct z prijatých dát vo forme MQTT správy v JSON formáte a MQTT topicu podľa druhu prijatých dát.
- **void executeCommand(char *command)**
  - spracuje konfiguračnú správu vo formáte JSON a nastaví podľa nej dané parametre
  - **char *command –** prijatá správa