Deployment Architecture:

A typical deployment architecture would consist of

1. MQTT broker
2. MQTT client (app)
3. Modbus Driver (app)
4. Modbus Device

Specifications/Usage

The Application has MQTT broker server IP hardcoded in the MQTT client, if the broker location is to be changed then the code must be modified accordingly.

The MQTT client is subscribed to the following topics:

1. connect
2. discon
3. read
4. write

connect: Message instructing the app to connect to a particular device

- Format: {"h":"<IP/Machine name>","p":<port>,"id":"<id>optional"}
- Example: {"h":"192.168.0.136","p":502,”id”:”01”}
  App would try to connect to IP 192.168.0.136:502; on success the result is published on MQTT topic ‘status’, on error/timeout the app reverts to its default state and waits for another connect message.
  Id is the Modbus device id (default id is 00), App cannot detect if sent id is unavailable.

NOTE: only one device may be connected at one time sending multiple connect message disconnects the previous.

discon: Message instructing the app to disconnect from connected device

- Format/Example: {"disconnected":true}

read : Message instructing the app to read data from specified register addresses on connected device and publish to MQTT topic read_result

- Format:
  [
  {"fc":"<FC_1>","id":["<ID_1>","<ID_2>","<ID_3>"…"<ID_n>"],"t":"<Type>"},
  {"fc":"<FC_2>","id":["<ID_1>","<ID_2>","<ID_3>"…"<ID_n>"],"t":"<Type>"},
  .
  .
  .
  {"fc":"<FC_m>","id":["<ID_1>","<ID_2>","<ID_3>"…"<ID_n>"],"t":"<Type>"}
  ]
  Where;
  FC: function code to access the type of registers on Modbus device.
    - 01: Coil Status
    - 02: Input Coil
    - 03: Holding Register
    - 04: Input Register

  ID: address/location of register to be read (in decimal)

  Range: 0-65535 i.e. 0000 to FFFF

  Excepts multiple ids (Limit unknown)

  Type: How to format returned data; does not apply to FC01, FC02.
    - **s**: short. Reads register data at <ID> as an unsigned 16 bit int
    - **ss**: signed short. Reads register data at <ID> as a signed 16 bit int
    - **l**: long big endian. Reads register data at <ID>,<ID>+1 as an unsigned 32 bit int
    - **lu**: long little endian (not implemented). Reads register data at <ID>+1,<ID> as an unsigned 32 bit int
    - **sl**: signed long big endian. Reads register data at <ID>,<ID>+1 as an unsigned 32 bit int
    - **slu**: signed long little endian (not implemented). Reads register data at <ID>+1,<ID> as an unsigned 32 bit int
    - **f**: float big endian. Reads register data at <ID>,<ID>+1 as a 32 bit float
    - **fu**: float little endian (not implemented). Reads register data at <ID>+1>,<ID> as a 32 bit float
    - **d**: double big endian. Reads register data at <ID>,<ID>+1,<ID>+2,<ID>+3 as a 64 bit double
    - **du**: double little endian (not implemented).

- Example:
  [{"fc":"03","id":["100","105","108","109"],"t":"s"}]

Returns 4 values corresponding to holding registers 100,105,108,109 as unsigned short int.

[
{"fc":"03","id":["100","105","108","109"],"t":"ss"},
{"fc":"04","id":["106","105","103","111"],"t":"l"},
{"fc":"03","id":["100","105","108","109"],"t":"d"}
{"fc":"01","id": ["100","105","108","109"]}
]
Returns 16 values; holding registers 100,105,108,109 as signed short int
Input registers (100,101), (105,106), (108,109), (109,110) as unsigned long int
Holding registers (100,101,102,105), (105,106,107,108), (108,109,110,111), (109,110,111,112) as a double.
Coil status of 100,105,108,109 as Boolean numbers.

write: Message instructing the app to write data to specified register addresses on connected device and publish result (success/error) to MQTT topic "read_result".

- Format:
  [
  {"fc":"<FC>","id":["<ID>"],"t":"<Type>", "w":"<value>"},
  {"fc":"<FC>","id":["<ID>"],"t":"<Type>", "w":" <value>"},
  .
  .
  .
  {"fc":"<FC>","id":["<ID>"],"t":"<Type>", "w":" <value>"}
  ]
  Where;
  FC: function code to access the type of registers on Modbus device.
  - 05: write coil
  - 06: Write register (redundant)
  - 10: Write register

  ID: address/location of register to be written (in decimal)

  Range: 0-65535 i.e. 0000 to FFFF

  Does not excepts multiple ids

  Type: How to format returned data; does not apply to FC05.
  - **s**: short. Writes <value> at <ID> as an unsigned 16 bit int or Signed 16 bit int if <value> is negative
    Range: -32768 to 65535

- **l**: long big endian. Writes <value> at <ID>,<ID>+1 as an unsigned 32bit int or Signed 32 bit int if <value> is negative
  Range: -2147483648 to 4294967296
  lu: long little endian (not implemented). Writes <value> at <ID>+1,<ID> as an unsigned 32bit int or Signed 32 bit int if <value> is negative
  Range: -2147483648 to 4294967296
- **f**: float big endian. Writes <value> at <ID>,<ID>+1 as a 32bit float
  Range: -3.4E38 to 3.4E38
- **fu**: float little endian (not implemented). Writes <value> at <ID>+1,<ID> as a 32bit float
  Range: -3.4E38 to 3.4E38
- **d**: double big endian. Writes <value> at <ID>,<ID>+1,<ID>+2,<ID>+3 as a 64bit double
  Range: -1.7E308 to 1.7E308
- **du**: double little endian (not implemented).
  Range: -1.7E308 to 1.7E308

- Example:
  [
  {"fc":"06","id": ["107"],"t": "s", “w”:”100”},
  {"fc":"10","id": ["103"],"t": "l", “w”:”-10”},
  {"fc":"10","id": ["105"],"t": "d", “w”:”-13.5”}
  {"fc":"05","id": ["109"], “w”: ”1”}
  ]
  Writes; 100 (or 0x0064) to holding register 107
  65535, 65526 (or 0xFFFF, 0xFFF6) to holding registers 103,104 respectively
  49496, 0 (or 0xc158, 0x0000) to holding registers 105,106 respectively
  1 to status coil 109