```
/*==================================
 *   MQTT-SN Client Application for Arduino
 *==================================*/
```

Xbee Baude rate,Serial device
MQTT-SN parameters

```
/*--------------------------------------------------
 *          Create Topic
 *--------------------------------------------------*/
```

Declare Topics Client useing

```
/*--------------------------------------------------
 *          Tasks invoked by Timer
 *--------------------------------------------------*/
```

Declare and register Tasks executed periodically

```
/*--------------------------------------------------
 * Tasks invoked by PUBLISH command Packet
 *--------------------------------------------------*/
```

Declare and register SUBSCRIBE

```
/*--------------------------------------------------
 *          Tasks invoked by INT0 interruption
 *--------------------------------------------------*/
```

INT0 procedure

```
/*--------------------------------------------------
 *          Arduino setup() function
 *--------------------------------------------------*/
```

Initialise Arduino

Xbee baud rate, Serial device MQTT-SN parameters

```
XBEE_APP_CONFIG = {
   {
      9600,          //Baudrate
      0,            //Serial PortNo (for Arduino App)
      0             //Device (for linux App)    i.e. /dev/ttyUSB0
   },
   {
      "ARD02",       //ClientId
      300,           //KeepAlive
      true,          //Clean session
      false,         //EndDevice  (sleep mode)
      "willTopic",    //WillTopic   or 0   DO NOT USE NULL STRING "" !
      "willMessage"   //WillMessage or 0   DO NOT USE NULL STRING
"" !
   }
 };
```

Declare Topics Client useing

```
MQString* topic1 = new MQString("ty4tw/tp1");
MQString* topic2 = new MQString("ty4tw/tp2");
MQString* tpMeasure = new MQString("ty4tw/soilReg");
```

MQString is UTF-8 format string.
Topic type is MQString.

Declare & registe Tasks executed periodically

```
int  task0(){
  return PUBLISH(tpMeasure,&pl,QOS1);
}

int task1(){
  Payload pl = Payload(36);
  pl.set_array(9);
  pl.set_int32(30);
  pl.set_int32(255);
return PUBLISH(topic1,&pl,QOS1);
}
/*--------------  List of task invoked by Timer ------------*/
TASK_LIST = {  //{ function , executing duration in second},
          {task0, 40},
          {task1,10},
END_OF_TASK_LIST};
```

Task0() runs every 40 secs.
Task1() runs every 10 secs.

Declare & register SUBSCRIBE

```
TOPICS_IN_CALLBACK = {
   topic1,
  END_OF_TOPICS
 };

int on_publish1(MqttsnPublish* msg){
    INDICATOR_OFF();
   return 0;
}

int on_publish2(MqttsnPu
   PUBLISH(topic1,  “resp”, 4,QOS0);
   return 0;
}
/*------------ Link Callback to Topic -------------*/
SUBSCRIBE_LIST = {
              {topic1, on_publish1, QOS1},
              {topic2, on_publish2, QOS1},
END_OF_SUBSCRIBE_LIST};
```

Declare Topics which used in callback routines

Callbacs which runs when the topic subscribed is received.

Declare Topics subscribed, callback and QOS