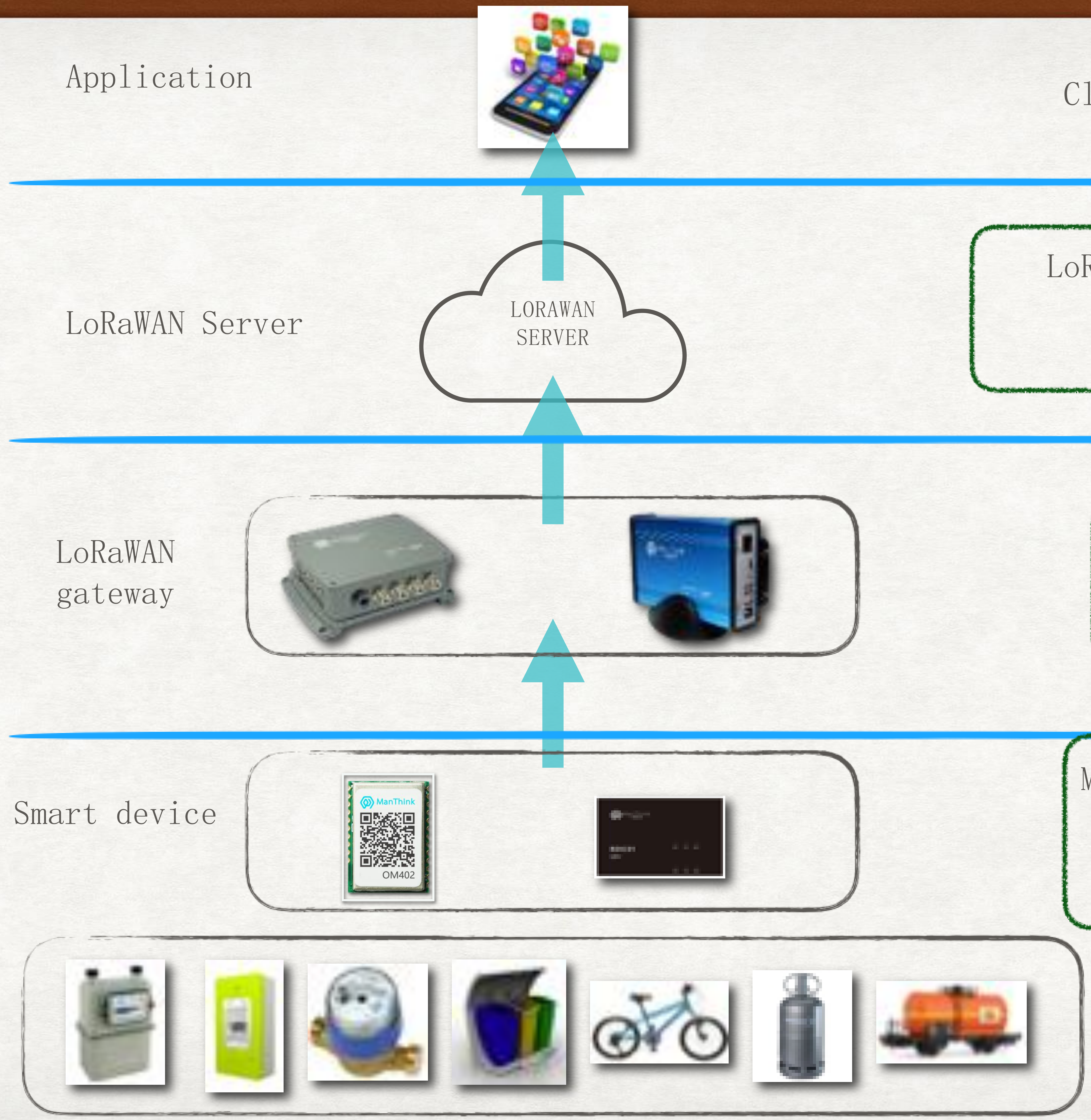
QUICKSTART

R&D BASED ON OMX02
LWS+MPOS
LORA/LORAWAN



Cloud for the specific application

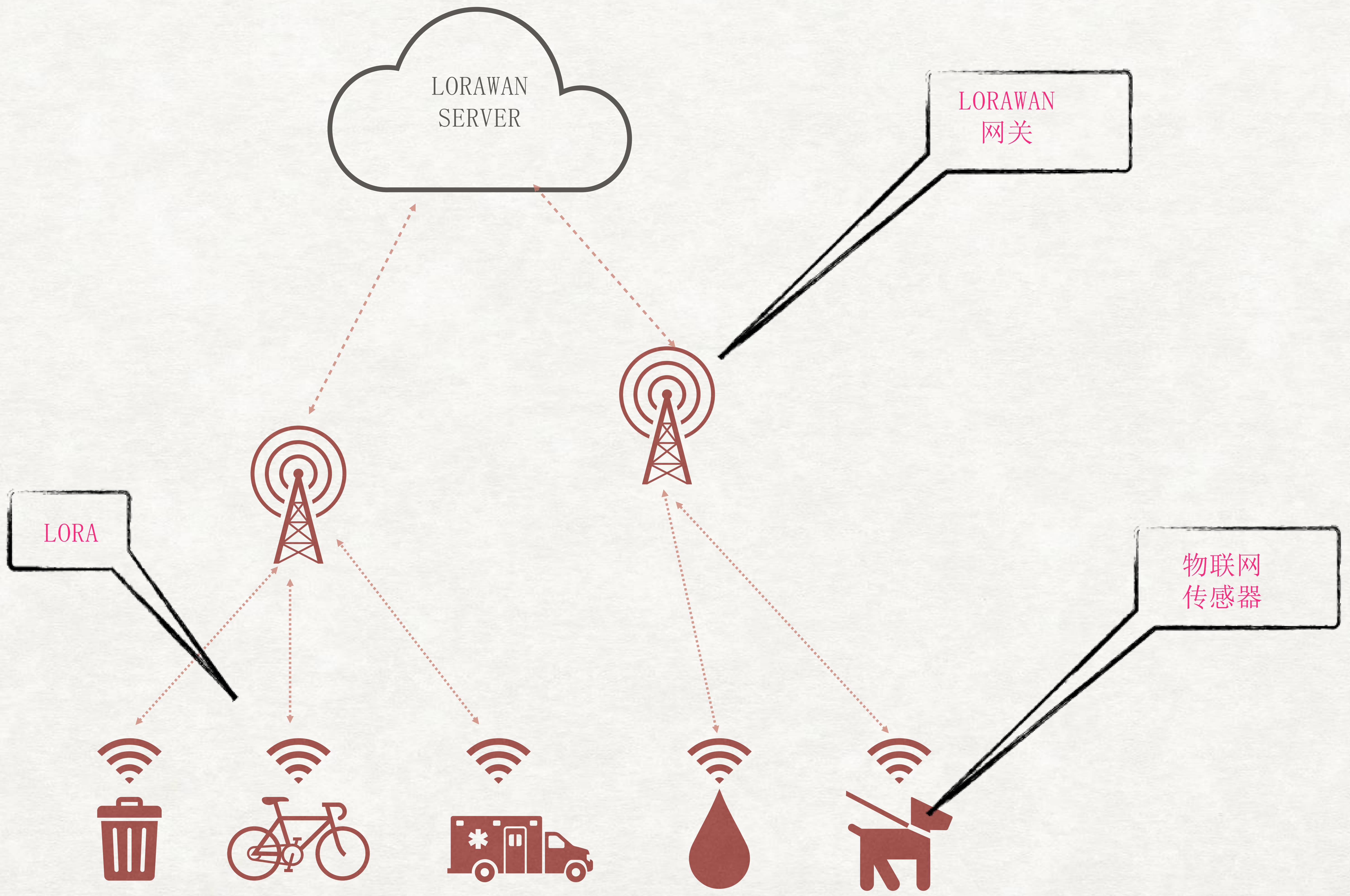
LoRaWAN Server get data from gateway

ManThink can supply different gateway
Include : indoor , outdoor, Full-duplex, SIG
And so on.

ManThink Supply module which open hardware
Resourse and MPOS+LWS SDK for developer
to complete their IOT application quickly

OMx02 module inside

DONEC QUIS NUNC



ManThink 门思科技

HARDWARE FEATRUE OF OM402/OM411

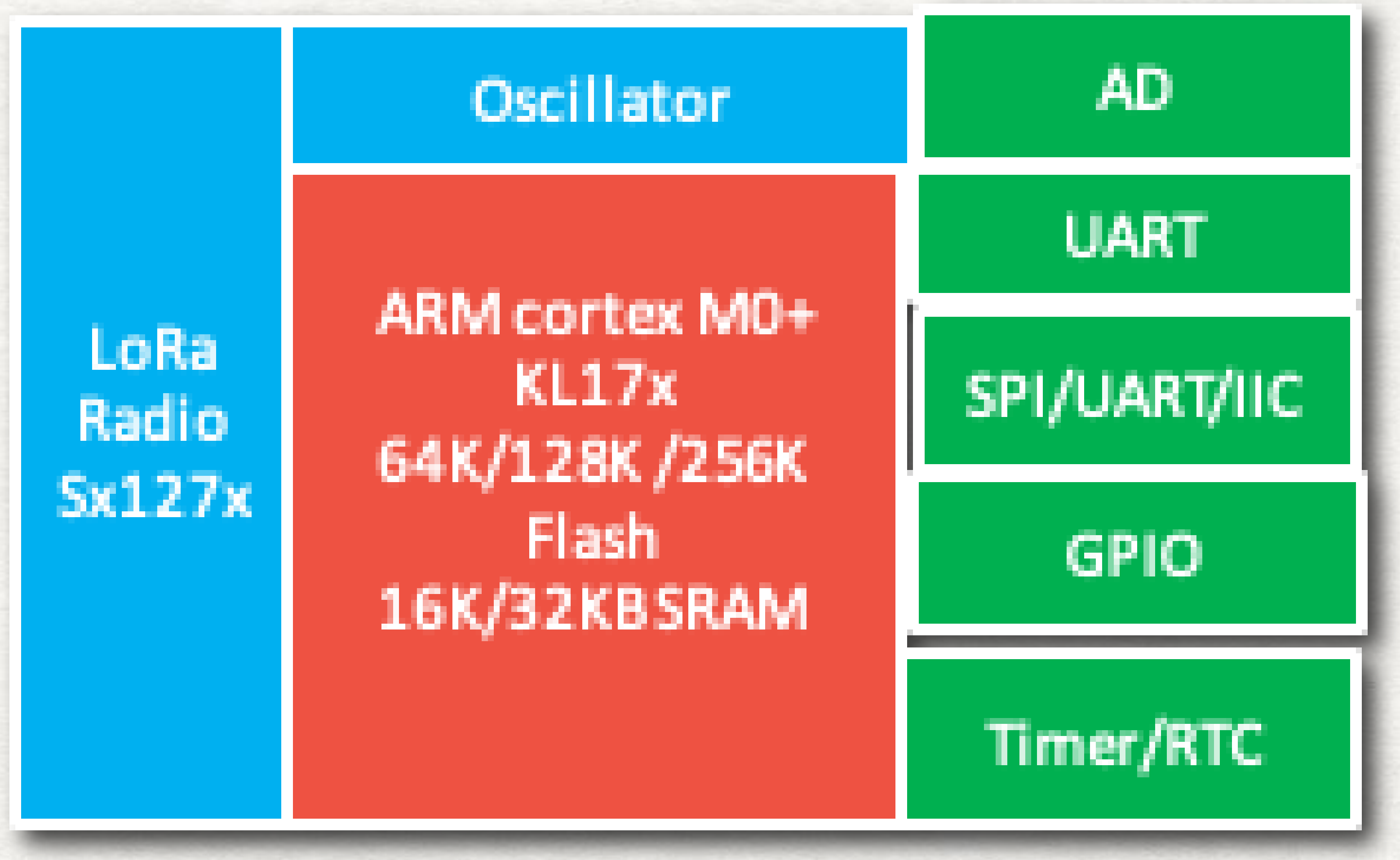
Low power
- <3uA sleeping current
- Auto sleep quickly(<1ms)

High performance
- -138dBm receive sensitivity
- Based on Cortex-M0+ MCU

Super small size
- 13mmx17.8mm
- LCC FootPrint

Parameters	
Radio Frequency	410~510MHz(OM402),860~1020MHz(OM802)
Transmission Power	5~20dBm
Receiving Sensitivity	-138dBm@292bps
Harmonic Suppression	≤1GHz：<-36dBm, ＞1GHz：＜-30dBm
Sleep Current(typical)	<3uA
Size	17.8mm x 13.0mm x 2.0mm
MCU	KL17x(32bit cortex-M0+)
Peripherals	SPI/UART/I2C/GPIO/AD
System Memory	64KFlash ,16K SRAM

参数



硬件框图

OM402/OM411 软件特性

Open-System

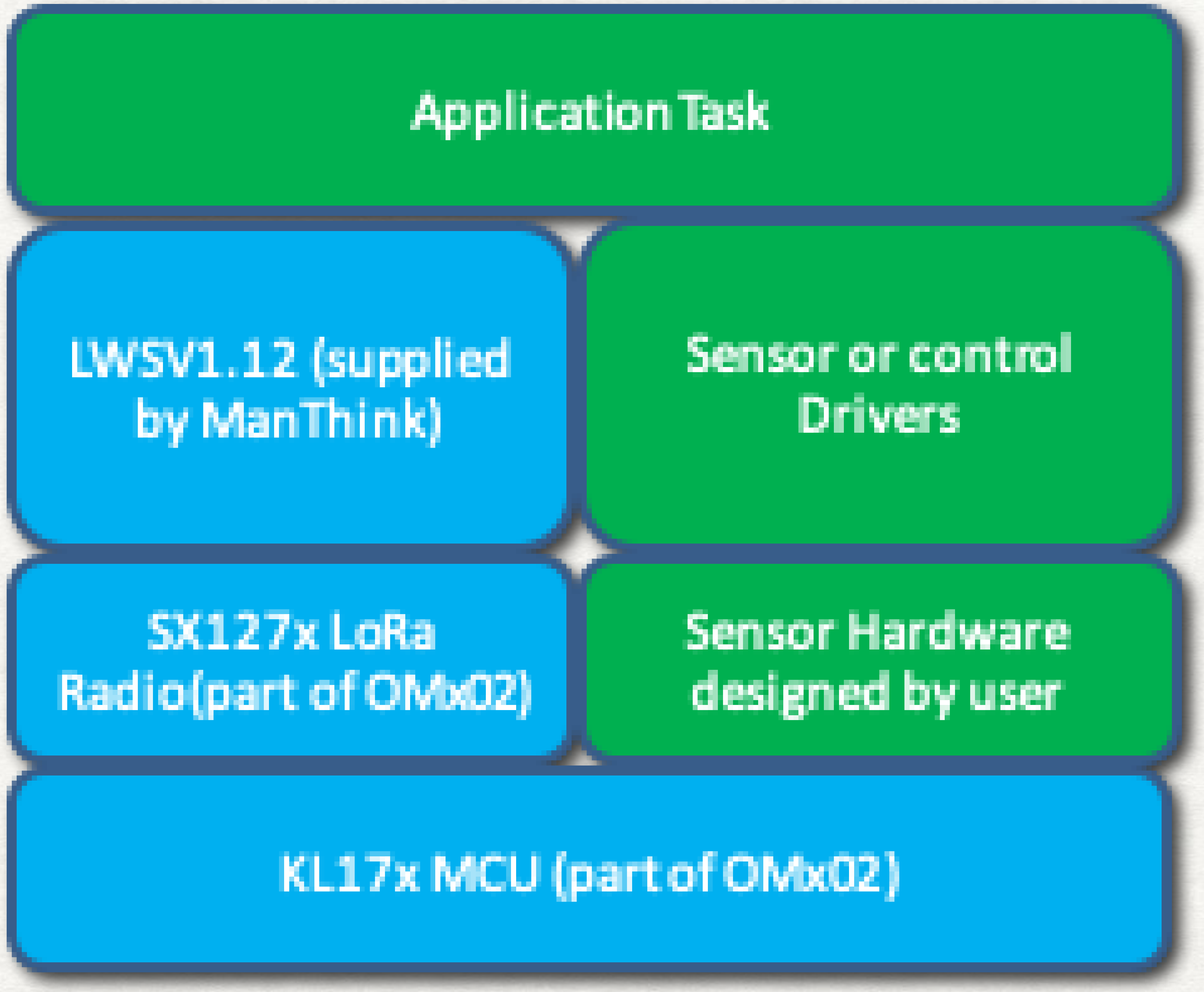
- High precise timing OS(Man-Pragnante)
- Auto sleep
- Open hardware resource ofKL17x
- Supply all drivers of KL17x
- Supply SDK of LoRaWAN

LoRaWAN protocol

- Class-A, Class-B and Class-C
- R&D by API
- Suppport FUOTA
- Support multi-bin
- Support SW mode

Low cost

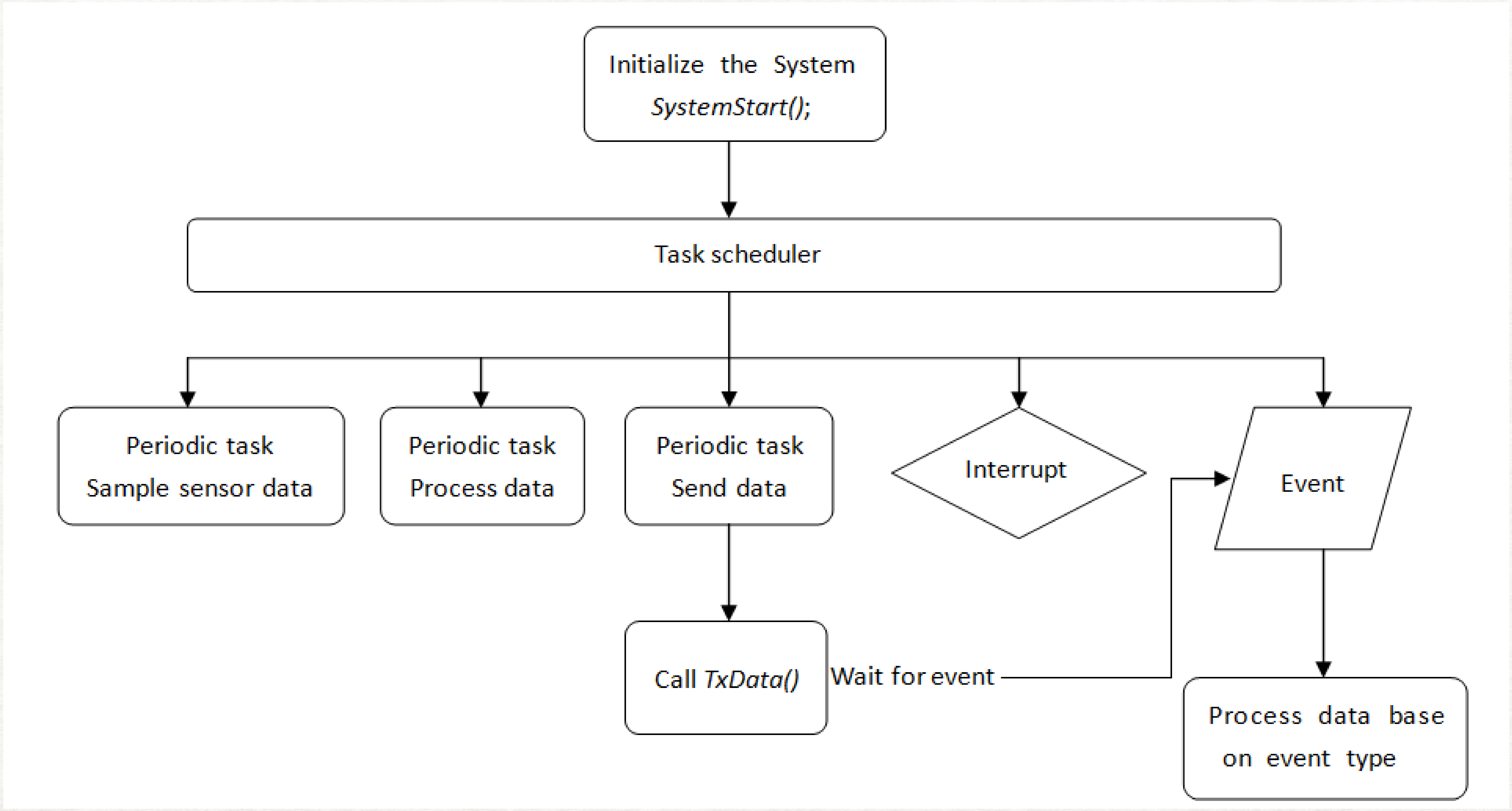
- Don’t need another MCU any more
- quickly



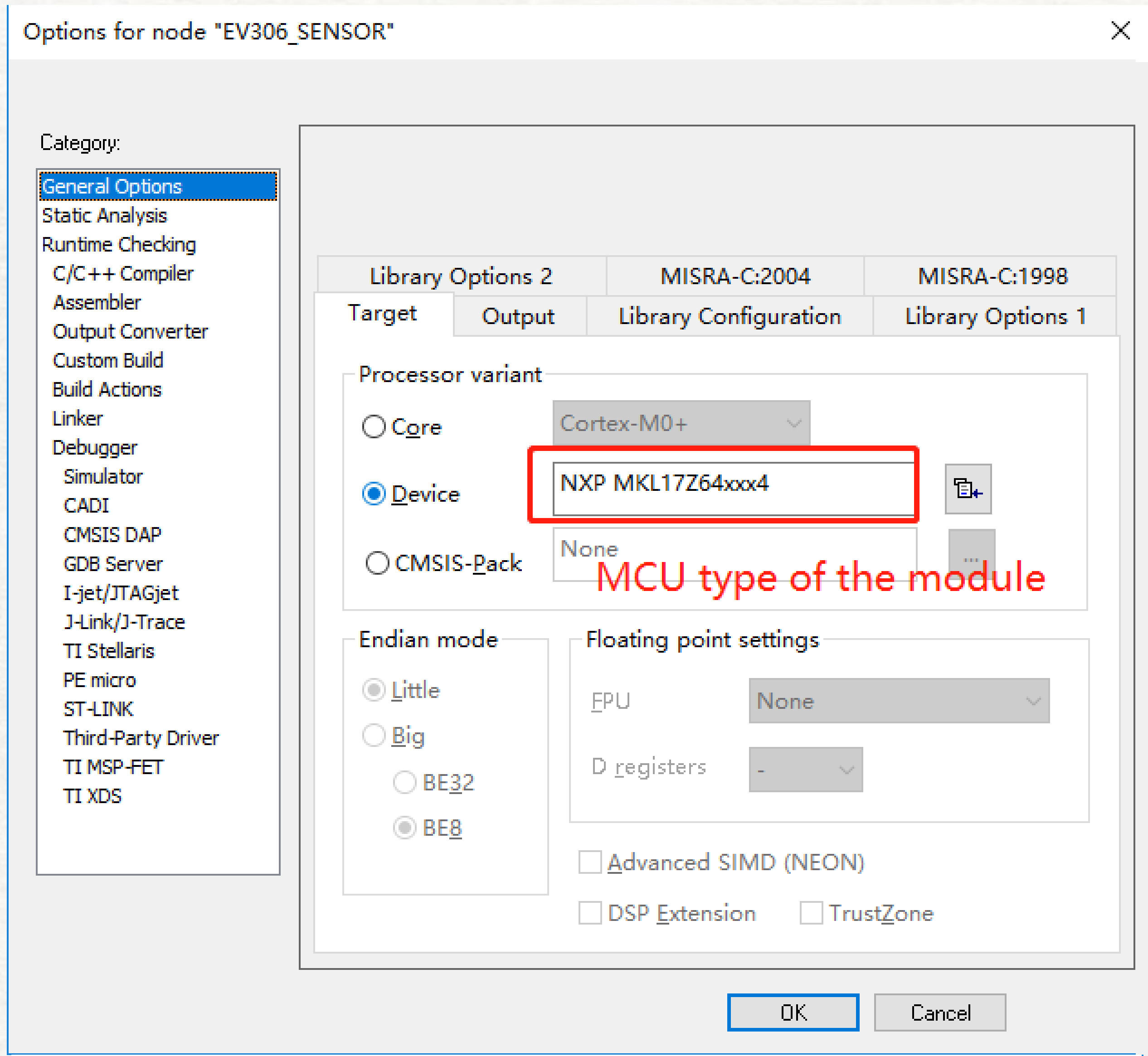
Application Task
LWSV1.12 (supplied by ManThink)
Sensor or control Drivers
SX127x LoRa Radio(part of OMx02)
Sensor Hardware designed by user
KL17x MCU (part of OMx02)

软件架构

CLASSIC PROCESS



IAR的工程配置


Options for node "EV306_SENSOR"

Category: General Options

Target

Processor variant

( ) Core: Cortex-M0+

(●) Device: NXP MKL17Z64xxx4

( ) CMSIS-Pack: None

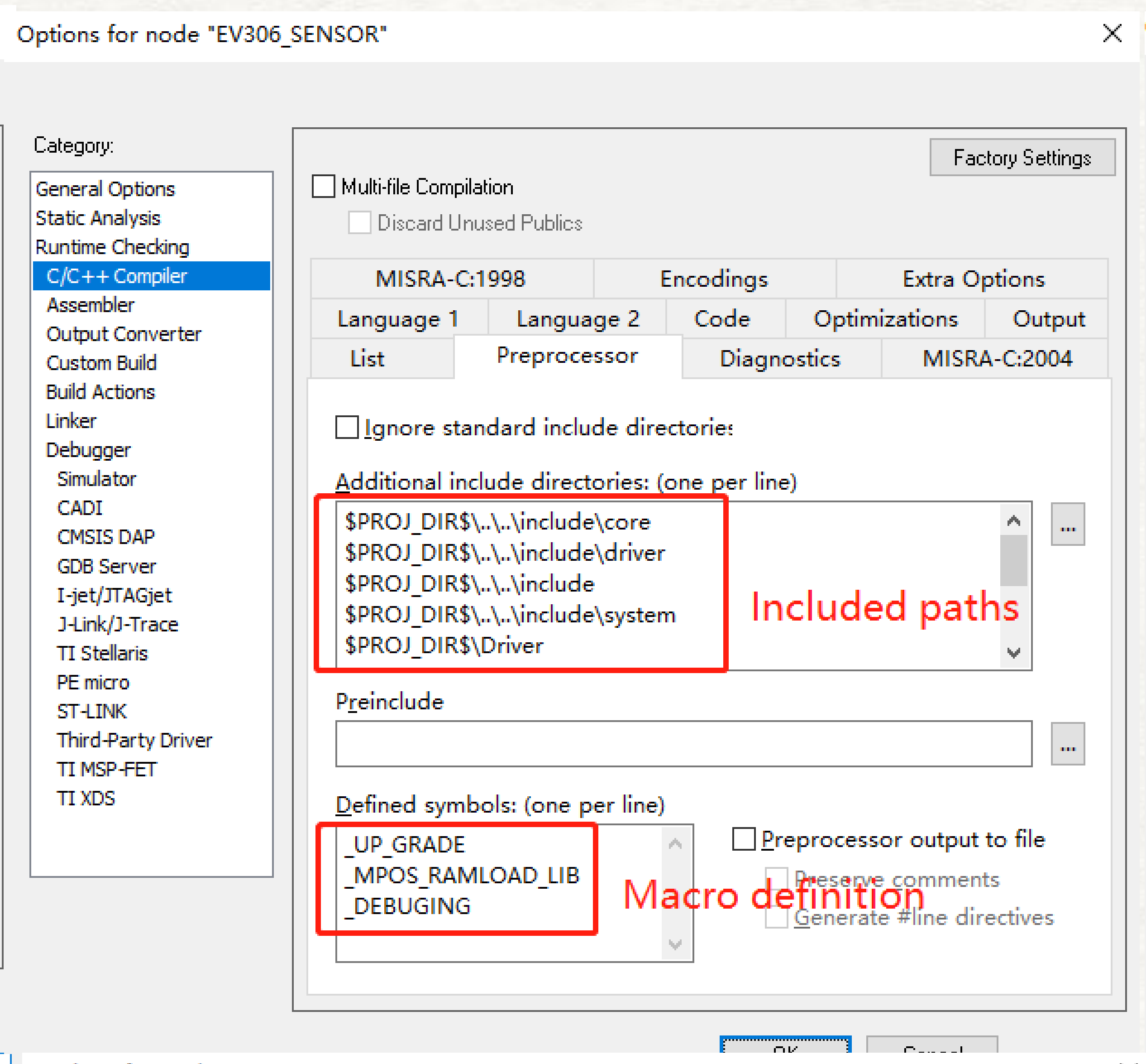
MCU type of the module

Endian mode: (●) Little ( ) Big

Floating point settings: FPU: None; D registers: -

[ ] Advanced SIMD (NEON)

[ ] DSP Extension [ ] TrustZone


Options for node "EV306_SENSOR"

Category: C/C++ Compiler

Preprocessor

[ ] Multi-file Compilation

[ ] Discard Unused Publics

[ ] Ignore standard include directories

Additional include directories: (one per line)

```
$PROJ_DIR$\..\..\include\core
$PROJ_DIR$\..\..\include\driver
$PROJ_DIR$\..\..\include\system
$PROJ_DIR$\..\..\include
$PROJ_DIR$\Driver
```

Included paths

Preinclude:

Defined symbols: (one per line)

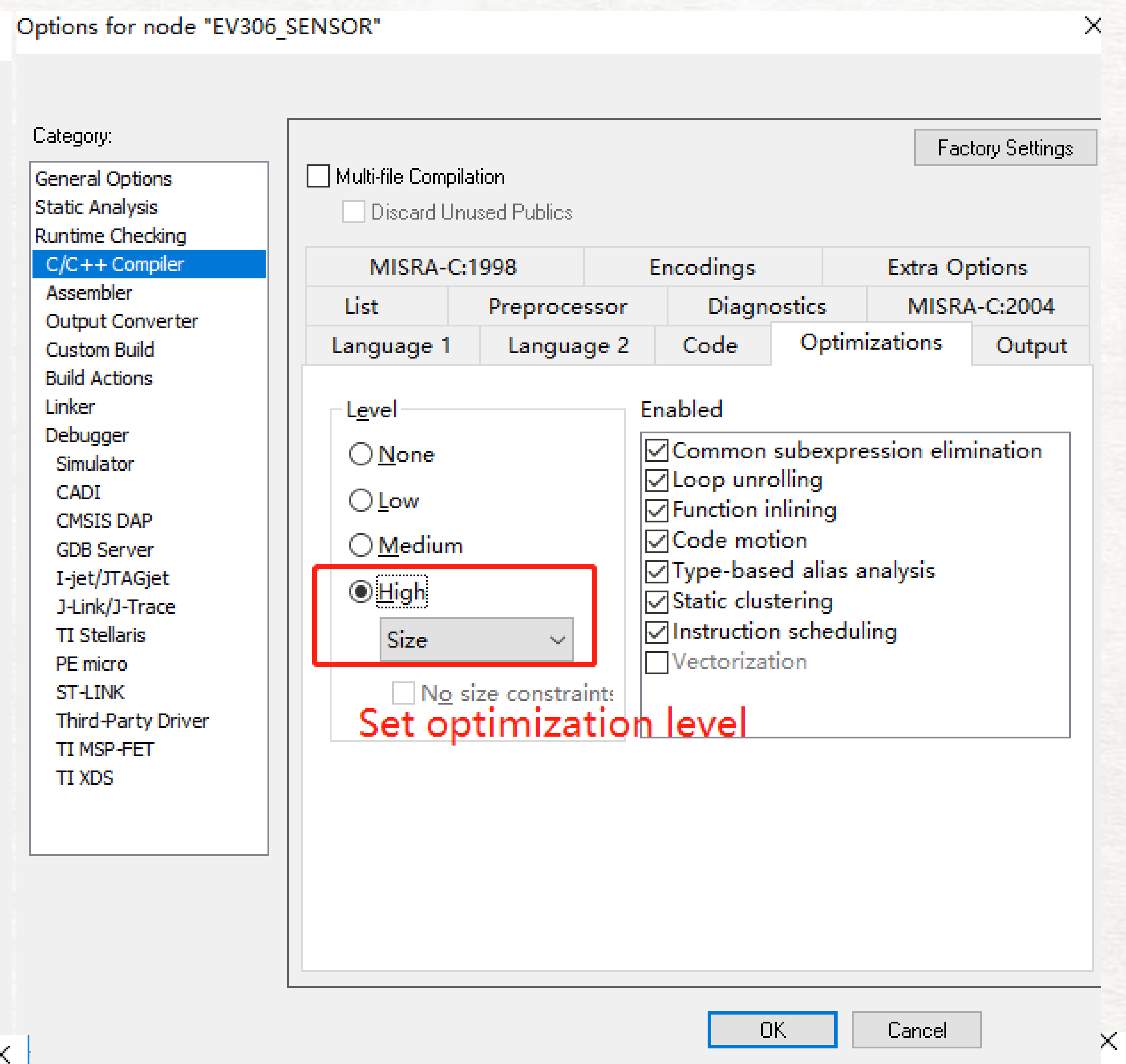
```
_UP_GRADE
_MPOS_RAMLOAD_LIB
_DEBUGING
```

Macro definition

[ ] Preprocessor output to file

[ ] Preserve comments

[ ] Generate #line directives


Options for node "EV306_SENSOR"

Category: C/C++ Compiler

Optimizations

[ ] Multi-file Compilation

[ ] Discard Unused Publics

Level: ( ) None ( ) Low ( ) Medium (●) High — Size

[ ] No size constraints

Enabled:

[✓] Common subexpression elimination

[✓] Loop unrolling

[✓] Function inlining

[✓] Code motion

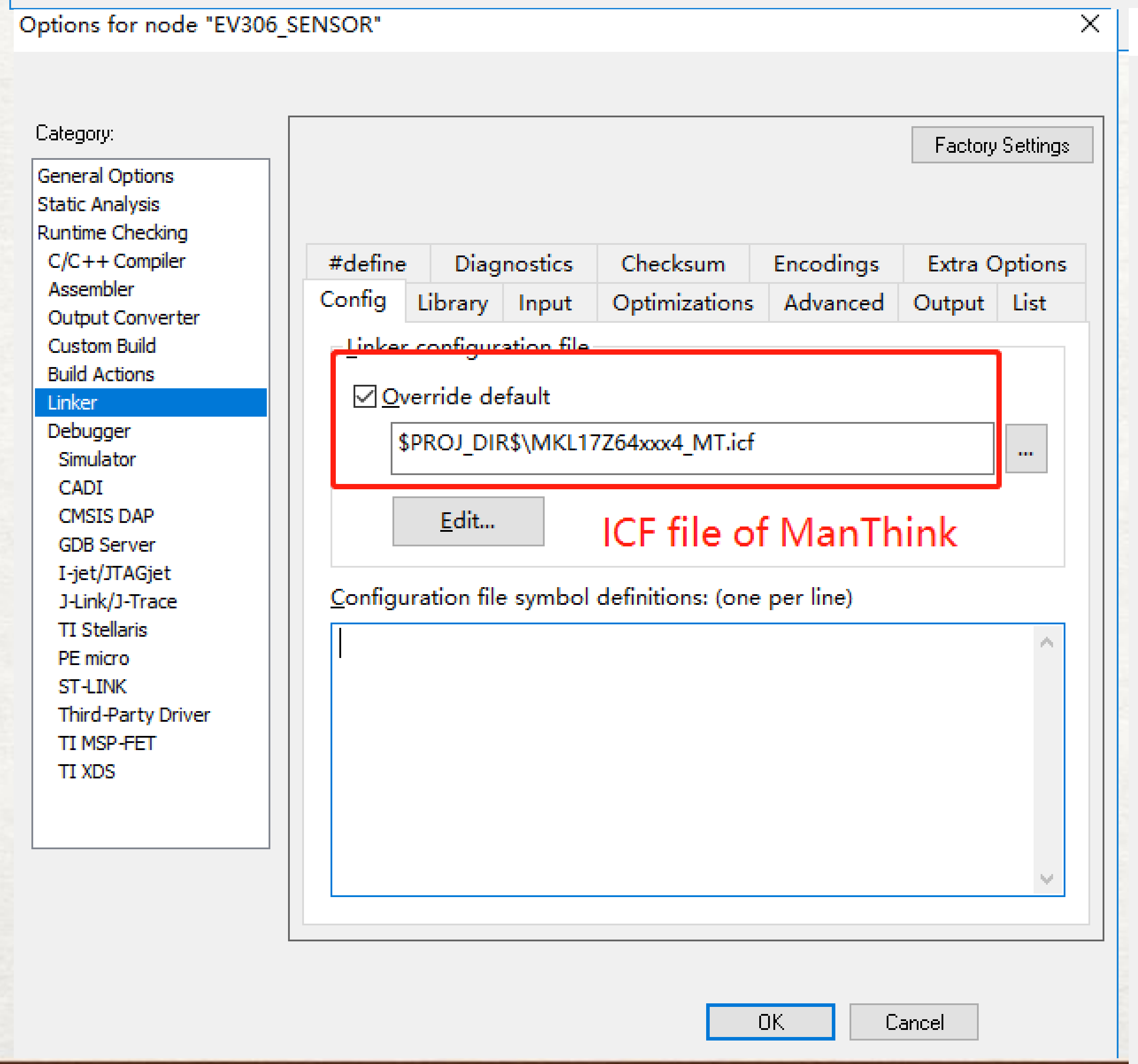
[✓] Type-based alias analysis

[✓] Static clustering

[✓] Instruction scheduling

[ ] Vectorization

Set optimization level


Options for node "EV306_SENSOR"

Category: Linker

Config

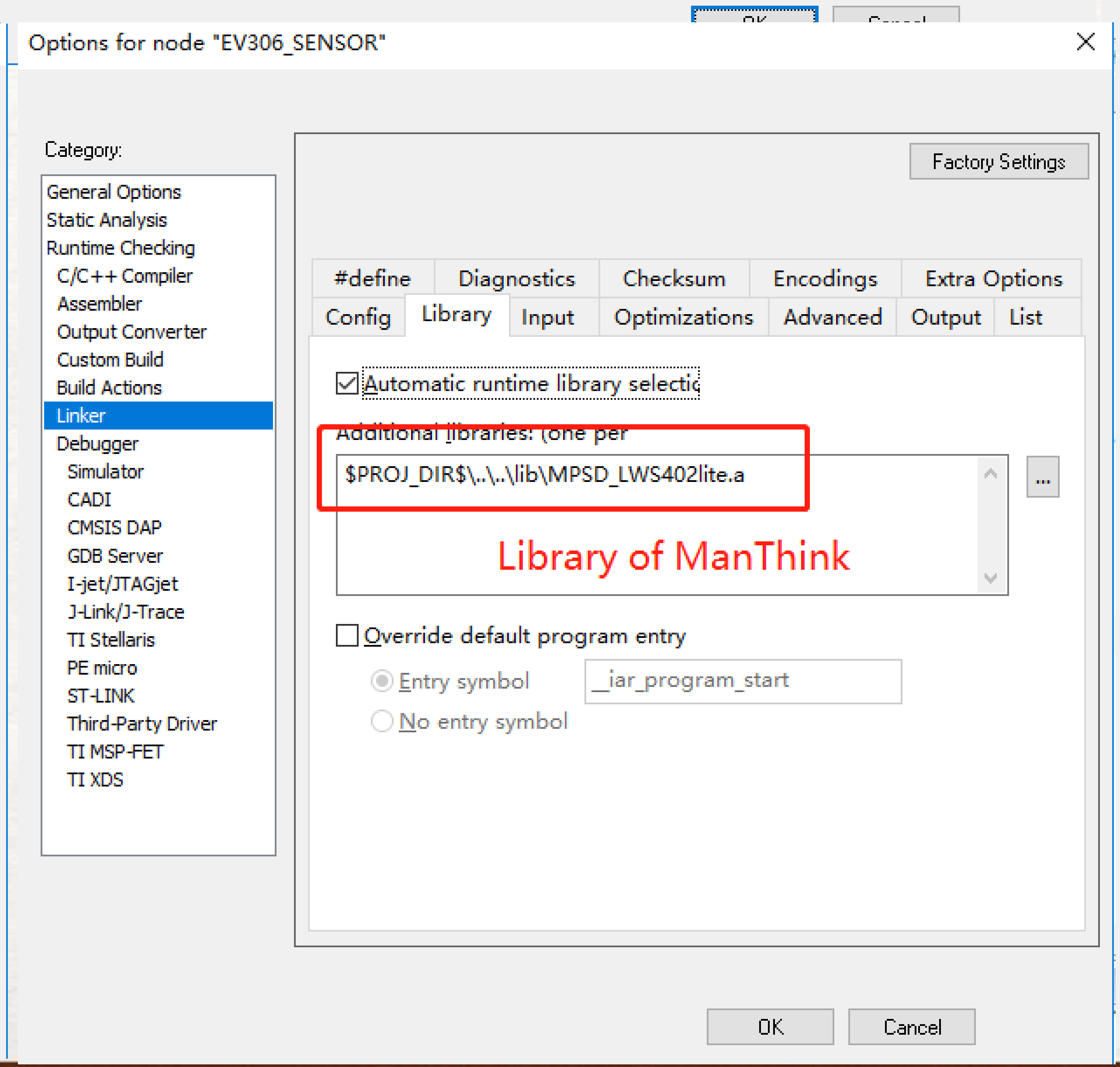
Linker configuration file

[✓] Override default

$PROJ_DIR$\MKL17Z64xxx4_MT.icf

ICF file of ManThink

Configuration file symbol definitions: (one per line)


Options for node "EV306_SENSOR"

Category: Linker

Library

[✓] Automatic runtime library selection

Additional libraries: (one per line)

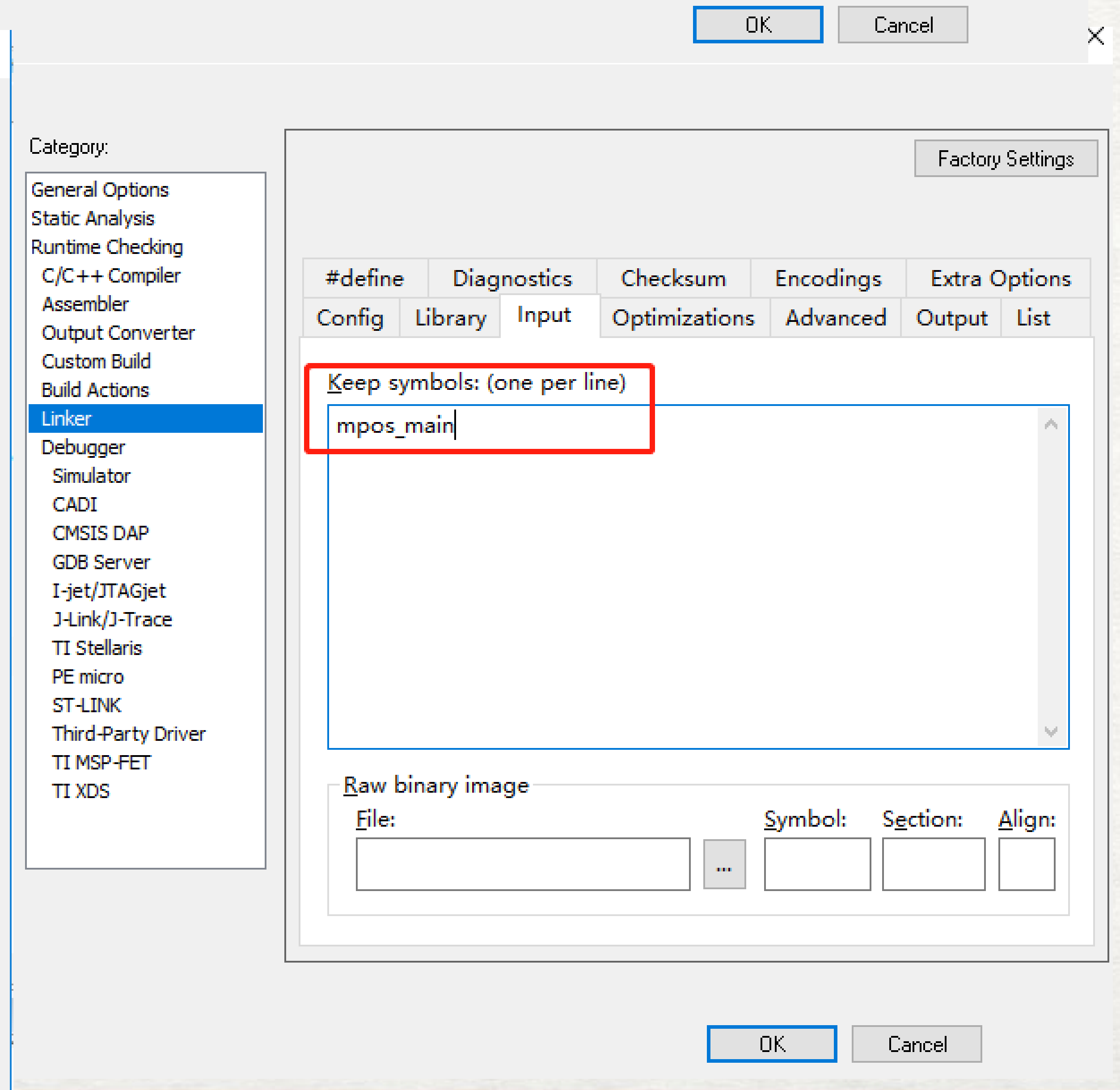
$PROJ_DIR$\..\..\lib\MPSD_LWS402lite.a

Library of ManThink

[ ] Override default program entry

(●) Entry symbol: __iar_program_start

( ) No entry symbol


Options for node "EV306_SENSOR"

Category: Linker

Input

Keep symbols: (one per line)

mpos_main

Raw binary image

File: Symbol: Section: Align:

系统初始化

```
  StackFunction.watchdog=true;    // Function bit for watch dog, must be false when debug
  StackFunction.nosleep=false;    // MPOS supply a virtual way for debug, must be true for release version
  StackFunction.uart1=true;       //if true, users will use the uart1 service from ManThink internally
  StackFunction.FlashBackup=true;// reserved
  StackFunction.MTprotocol=false; // if true, system will use the ManThink’ s uart protocol and the head of packet must be 0xFF, 0xAA
  mpos_driver.Clock_Init();        // initialze for system
  mp_userInit=MT_MyInit;           //Install user’ s initial functiion

void MT_MyInit()
{
   mpos_osfun.SysParaInit(); //Initialize system, and load the parameters from flash
   MT_MyHookInit();           //Install hook function
   MT_MyParaInit();           //Initialize the user’ s parameters
   MT_MyBoardInit();          //Initialize the hardware
   MT_LoRaWANParaInitial();   //Initialize the parameters of LoRaWAN
   mpos_driver.kickdog();     //Kick watch dog
   mpos_lws.LWS_init();       //Install the parameters for LWS stack
   mpos_driver.kickdog();     //Kick watch dog
   mpos_lws.LoRaWANInit();    //Initialize the parameters of LoRaWAN’ s system
   MT_SetupSensorTxTask();    //install the periodic task
   MT_SetupOnceEventTask();   //Install One-off task
}
```

ManThink 门思科技

LORAWAN参数初始化

```
RunStatus.Varible.State.Bits.Mode=1;// Set the mode as ClassA
RunStatus.Varible.State.Bits.ADR=1; //enable ADR function
RunStatus.Varible.State.Bits.OTA=0; //Set oRaWAN as ABP mode
RunStatus.Varible.State.Bits.FDD=1; //working at FDD mode
RunStatus.Varible.DR=0; //Initialize the data rate of LoRaWAN
mpos_lws.paraFWGet (&paraFwReg);  //Get the parameters of FW
mpos_lws.paraRDGet (&paraRdReg);  //Get the frequency parameters
paraFwReg.SFwRegister.RxWinDelay1=1;  //Set the first window delay value as 1 seconds
paraFwReg.SFwRegister.RxWinDelay2=2;  //Set the second window delay value as 2 seconds
paraFwReg.SFwRegister.JoinDelay1=5;   //Set the join delay value as 5 seconds
paraFwReg.SFwRegister.JoinDelay2=6;   //Set the second join delay value as 6 seconds
for(int i=0;i<4;i++)                  //duty cycle for 4 bands
{
   paraFwReg.SFwRegister.DutyBand[i][0]=0;
   paraFwReg.SFwRegister.DutyBand[i][1]=0;

mpos_osfun.os_wlsbf8 (paraFwReg.SFwRegister.AppEui, 0x8100000002000001); //Set APPEUI
mpos_osfun.memcpy1 (paraFwReg.SFwRegister.DevKey,AppKey);  //Set DevKey(APPKey)
mpos_lws.paraFWSave (&paraFwReg);                          //save the modified parameters

paraRdReg.SRdRegister.dn2Dr=0;    //Set the DR for second window
paraRdReg.SRdRegister.Power=22;  //Set RF power, actual power= value-2.when 22, power= 20dBm。 OM402 support 20dBM maximuly, OM411 can support22dBm
mpos_osfun.os_wlsbf2 (paraRdReg.SRdRegister.channelMap,0x00FF); //set the freq channelMap
for(int i=0;i<16;i++)                            //set the 16 freq
{
  mpos_osfun.os_wlsbf4 (paraRdReg.SRdRegister.Freq[i].SFreq,(470300000+200000*i),(i/2));
  paraRdReg.SRdRegister.Freq[i].DRRange.Bits.HiDr=5;
  paraRdReg.SRdRegister.Freq[i].DRRange.Bits.LoDr=0;
}
mpos_lws.paraRDSave (&paraRdReg);//Save parameters to Flash
```

ManThink
门思科技

周期任务

Developer can build periodic task
Can Set the cycles of task to be excuted, After cycles, Task will stoped
Developer can add sleeping deal function and wake up deal function
For low-power application

```
S_periodTask  MT_TaskSensorTx;         //Deifine a periodic task
MT_TaskSensorTx.Task=MT_SensorTx;      // Hook the function to the task
MT_TaskSensorTx.interval=600000;       //Set the period, unit :mS
MT_TaskSensorTx.cycles=0xFFFFFFFF;     //Set the cycles, If set as 0xFFFFFFFF, the task is infinite task
mpos_osfun.Task_Setup(&MT_TaskSensorTx);    // Install the task
mpos_osfun.Task_Remove(&MT_TaskSensorTx);   // Remove the task
```

LORAWAN API

```
void mpos_lws.LWS_SetClassMode (uint8_t classMode);//Set the working mode, 1=ClassA, 2=ClassB, 0=ClassC 。Result of switch will notify users by Event
LWERRO_t mpos_lws.JoinReset (LWOP_t mode); //Rejoin the network according to the set value :OTAA or ABP

Mode:LWOP_REJOIN
LWERRO_t mpos_lws.TxData(uint8_t * txBuffer,u1_t lenth,u1_t port,LWOP_t mode);
txBuffer: address of data to be sent
lenth: size of the data
port: port of LoRaWAN
mode: LWOP_LTC=confirm packet. LWOP_LTU=unconfirmed packet
```

LORAWAN EVENT

```
void HookUserEvent( mt_ev_t ev ,u1_t port,u1_t * Buffer, u2_t len)
{
  switch( ev )
  {
  case MT_EV_TXDONE:          break; // RF tx data done
  case MT_EV_JOINED:          break; // join success
  case MT_EV_JOIN_FAILED:     break; // join failed
  case MT_EV_REJOIN_FAILED:   break; //rejoin failed

  case MT_EV_TXOVER_NOPORT:   break; // packet transmit success without down data
  case MT_EV_TXOVER_NACK:     break; // failed to transmit a confirmed packet
  case MT_EV_TXOVER_DNW1:     break;
 // packet transmitted success and  get down data at first window. The port of down data in port and get data from buffer and size of data saved in len
  case MT_EV_TXOVER_DNW2:     break;
//// packet transmitted success and  get down data at the second window. The port of down data in port and get data from buffer and size of data saved in len
  case MT_EV_TXOVER_PING:     break;
 // get data under classB mode,The port of down data in port and get data from buffer and size of data saved in len
  default:                    break;
  }
}
```