# Memento das Funções e Comandos AT empregados com a biblioteca "RoboCore_SMW_SW1276M0.h"

O módulo LoRaWAN Bee V2 - Chip Antenna SMD é baseado no transceptor **SMW-SX1276M0 da SMART Modular**, dispondo da biblioteca "RoboCore_SMW_SX1276M0.h" (v1.0.1 no Github). Class: A, B ou C. Possibilita utilização no modo P2P – modulação LoRa- e redes protocolo LoRaWAN (**AU915**) públicas e privadas (ativação ABP ou OTAA). **MCU** MS500. Interface de comunicação **UART por hardware** (115200 bps). **Fonte de energia**: 3,0 a 3,6 V. A montagem do módulo não disponibiliza (?) GPIO para propósito geral.
Transceiver RF Semtech: Datasheet SMW-SX1276M0. **ANATEL**: 00118603 (**eWBM eLR100-UL-00**).

## LoRa Comandos AT Manual

**Leitura ou execução: AT+___ ///// Configuração: AT+___ x ou AT+___=x**

**ACTIVATION COMMAND (Capítulo 6)**
- **AT+DEVEUI** End-device identifier (OTAA)
- **AT+APPEUI** AppEUI is a global application ID (OTAA)
- **AT+APPKEY** Application key (OTAA)
- **AT+DADDR** Device Address (4byte) (ABP)
- **AT+APPSKEY** Application Session Key (ABP)
- **AT+NWKSKEY** Network Session Key (ABP)
- **AT+PNM** Public Network Mode Status (=0 ou **1**)
- **AT+NJM** Network Join Mode (=0, **1** ou 2) ABP, OTAA ou P2P
- **AT+CLASS** LoRa Mac Class (=**A**, B ou C)
- **AT+JOIN** Execute JOIN request for LoRa Network Server
- **AT+NJS** Join (Network Server Connection) Status
- **AT+NWKID** Returns Network ID (4 bytes)
- **AT+AJOIN** Execute auto JOIN after the booting (=0 ou 1)
- **AT+AINF** Returns Activation setting value

**P2P COMMAND (Cap. 11)**
- **AT+P2PCH** Readout List of Channels and Config a Ch
- **AT+P2PDA** Set P2PDA with Peer Device Address
- **AT+P2PSW** Set Radio Sync Word; default (=18)

**LORAMAC CONFIGURE COMMAND (Cap. 8)**
- **AT+REGION** LoRaMAC Region Configuration; AU915(=1)
- **AT+ADR** ADR Command; ADR off(=0) e on(=1)
- **AT+DR** Data rate Command; DR SF10-BW125 (=2)
- **AT+RX2FQ** Rx Window 2 freq Command (=923300000)
- **AT+RX2DR** Rx Window 2 data rate (DR_X de 8 a 13) (=8)
- **AT+RX1DL** The Tx and the Rx Window 1 Delay (=5000)
- **AT+RX2DL** The Tx and the Rx Window 2 Delay (=6000)
- **AT+JN1DL** The Tx and the Rx Window 1 Join Accept Delay
- **AT+JN2DL** The Tx and the Rx Window 2 Join Accept Delay
- **AT+MUFR** Unconfirmed uplink Resend (1-15); default (=1)
- **AT+MCFR** Confirmed uplink resend (???)
- **AT+TXP** Tx Power index (EIRP-2x[0~10]); Max (=0)
- **AT+FCU** Uplink Counter
- **AT+FCD** Downlink Counter
- **AT+BAT** Battery Level(0:USB; 1~254; 255:Error)
- **AT+LCHK** Mac Command LinkCheckReq
- **AT+CRYPTO** Encryption Configuration; Standard AES (=0)
- **AT+CH** Channel Configuration

**UP/DOWN LINK COMMAND (Cap. 7)**
- **AT+SEND** LoRa Data Uplink (text-ASCII) (=port:payload)
- **AT+SENDB** LoRa Data Uplink (hexadecimal)
- **AT+RECV** Confirm Received Downlink Data (text-ASCII)
- **AT+RECVB** Confirm Received Downlink Data (hexadecimal)
- **AT+RSSI** Returns RSSI value of the last received data
- **AT+SNR** Returns SNR value from the last received data

**DEBUG COMMAND (Cap. 10)**
- **AT+DBG** Event and Debug Message Configuration (=0)
- **AT+TXCW** FSK Tx Continuous Wave mode (Tx Signal Strength Test)
- **AT+RXTT** LoRa Rx Signal Strength Test (desfaz: AT+TSTP)
- **AT+TXTT** LoRa Tx Signal Strength Test (desfaz: AT+TSTP)
- **AT+TSTP** Stop RF Test
- **AT+GPIO** MS500 GPIO Pin Information (???)

**SYSTEM COMMAND (Cap. 9)**
- **AT+RESET** System Reboot (+fraco)
- **AT+SINF** System Information
- **AT+VER** Firmware Version (0.16)
- **AT+SAG** Antenna Gain (-4~6 DBm)
- **AT+CFM** Uplink Packet Type; Unconfirmed (=**0** ou =1)
- **AT+SLEEP** Enters Low Power Mode (reboot para voltar)
- **AT+ALARM** RTC Wakeup time; default (=0)
- **AT+TIME** RTC time (=20:12:00)
- **AT+DATE** RTC Date (=2021:11:17)
- **AT+ECHO** AT Command Responding Message ECHO
- **AT+FRESET** Command to reset the configuration (+forte)

FUNÇÕES PÚBLICAS

```
void (*event_listener)(Event);
SMW_SX1276M0(Stream (&));
SMW_SX1276M0(Stream (&), int16_t);
void flush(void);
CommandResponse get_ADR(uint8_t (&));
CommandResponse get_AJoin(uint8_t (&));
```

CommandResponse **get_Alarm**(uint32_t (&));
CommandResponse **get_AppEUI**(char (&)[SMW_SX1276M0_SIZE_APPEUI]);
CommandResponse **get_AppKey**(char (&)[SMW_SX1276M0_SIZE_APPKEY]);
CommandResponse **get_AppSKey**(char (&)[SMW_SX1276M0_SIZE_APPSKEY]);
CommandResponse **get_DevAddr**(char (&)[SMW_SX1276M0_SIZE_DEVADDR]);
CommandResponse **get_DevEUI**(char (&)[SMW_SX1276M0_SIZE_DEVEUI]);
CommandResponse **get_DR**(uint8_t (&));
CommandResponse **get_Echo**(uint8_t (&));
void **get_buffer**(Buffer (&));
CommandResponse **get_JoinMode**(uint8_t (&));
CommandResponse **get_JoinStatus**(uint8_t (&));
CommandResponse **get_NwkSKey**(char (&)[SMW_SX1276M0_SIZE_NWKSKEY]);
CommandResponse **get_RSSI**(double (&));
CommandResponse **get_SNR**(double (&));
CommandResponse **get_Version**(char (&)[SMW_SX1276M0_SIZE_VERSION]);
bool **isConnected**(void);
bool **isSleeping**(void);
void **join**(void);
CommandResponse **listen**(bool = true);
CommandResponse **ping**(void);
CommandResponse **readT**(void);
CommandResponse **readT**(Buffer (&));
CommandResponse **readT**(uint8_t (&), Buffer (&));
CommandResponse **readX**(void);
CommandResponse **readX**(Buffer (&));
CommandResponse **readX**(uint8_t (&), Buffer (&));
CommandResponse **reset**(void);
CommandResponse **sendT**(uint8_t, const char *);
CommandResponse **sendT**(uint8_t, const String);
CommandResponse **sendX**(uint8_t, const char *);
CommandResponse **sendX**(uint8_t, const String);
CommandResponse **set_ADR**(uint8_t);
CommandResponse **set_AJoin**(uint8_t);
CommandResponse **set_Alarm**(uint32_t);
CommandResponse **set_AppEUI**(const char *);
CommandResponse **set_AppKey**(const char *);
CommandResponse **set_AppSKey**(const char *);
CommandResponse **set_DevAddr**(const char *);
CommandResponse **set_DevEUI**(const char *);
CommandResponse **set_DR**(uint8_t);
CommandResponse **set_Echo**(uint8_t);
CommandResponse **set_JoinMode**(uint8_t);
CommandResponse **set_NwkSKey**(const char *);
void **setPinReset**(int16_t);
CommandResponse **sleep**(uint32_t = 0);

**enum class CommandResponse : uint8_t { ERROR , OK , FAILED , FAILED_STRING , NOT_FOUND , DATA };**
**enum class Event : uint8_t { JOINED , RECEIVED , RECEIVED_X , SLEEP , WAKEUP , RESET };**