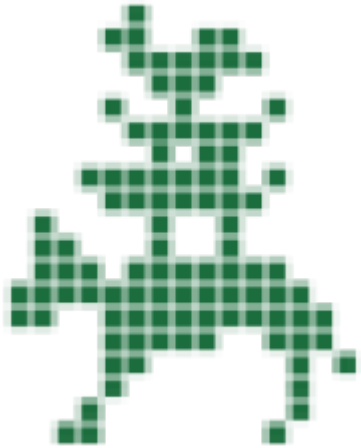
LoRa Gateway

Inventarnummer	g????
Benutzbar für	Mitglieder/Jeder
Einweisung benötigt	Nein
Ort	Hackerspace
Eigentümer	Verein



Ansprechpartner

- keiner
- (Mittelantrag: https://wiki.hackerspace-bremen.de/mittelantraege/antraege/finanziert/2019/lora_gateway?s[]=lora)

Dokumentation zur Inbetriebnahme eines Nodes

Bibliotheken:

- https://github.com/matthijskooijman/arduino-lmic

Weitere Informationen für Nodes entnehmt ihr bitte aus dem Quellcode der oben genannten Bibliothek oder aus der The Things Network Community. Man wird sofort fündig. Die Anleitungen im Netz sind Gateway unabhängig! Das heißt wenn bei euch zu Hause ein LoRa Gateway steht ist der Code für hier und bei euch gleich! Beispiel Sketch:

ttn-abp.ino

```
/*******************************************************************************
 * Copyright (c) 2015 Thomas Telkamp and Matthijs Kooijman
 *
 * Permission is hereby granted, free of charge, to anyone
```

```
 * obtaining a copy of this document and accompanying files,
 * to do whatever they want with them without any restriction,
 * including, but not limited to, copying, modification and
redistribution.
 * NO WARRANTY OF ANY KIND IS PROVIDED.
 *
 * This example sends a valid LoRaWAN packet with payload "Hello,
 * world!", using frequency and encryption settings matching those of
 * the The Things Network.
 *
 * This uses ABP (Activation-by-personalisation), where a DevAddr and
 * Session keys are preconfigured (unlike OTAA, where a DevEUI and
 * application key is configured, while the DevAddr and session keys
are
 * assigned/generated in the over-the-air-activation procedure).
 *
 * Note: LoRaWAN per sub-band duty-cycle limitation is enforced (1% in
 * g1, 0.1% in g2), but not the TTN fair usage policy (which is
probably
 * violated by this sketch when left running for longer)!
 *
 * To use this sketch, first register your application and device with
 * the things network, to set or generate a DevAddr, NwkSKey and
 * AppSKey. Each device should have their own unique values for these
 * fields.
 *
 * Do not forget to define the radio type correctly in config.h.
 *
*******************************************************************
********/

#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>

// LoRaWAN NwkSKey, network session key
// This is the default Semtech key, which is used by the early
prototype TTN
// network.
static const PROGMEM u1_t NWKSKEY[16] = { 0x2B, 0x7E, 0x15, 0x16, 0x28,
0xAE, 0xD2, 0xA6, 0xAB, 0xF7, 0x15, 0x88, 0x09, 0xCF, 0x4F, 0x3C };

// LoRaWAN AppSKey, application session key
// This is the default Semtech key, which is used by the early
prototype TTN
// network.
static const u1_t PROGMEM APPSKEY[16] = { 0x2B, 0x7E, 0x15, 0x16, 0x28,
0xAE, 0xD2, 0xA6, 0xAB, 0xF7, 0x15, 0x88, 0x09, 0xCF, 0x4F, 0x3C };

// LoRaWAN end-device address (DevAddr)
```

```
static const u4_t DEVADDR = 0x03FF0001 ; // <-- Change this address for
every node!

// These callbacks are only used in over-the-air activation, so they
are
// left empty here (we cannot leave them out completely unless
// DISABLE_JOIN is set in config.h, otherwise the linker will
complain).
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }

static uint8_t mydata[] = "Hello, world!";
static osjob_t sendjob;

// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations).
const unsigned TX_INTERVAL = 60;

// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 6,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 5,
    .dio = {2, 3, 4},
};

void onEvent (ev_t ev) {
    Serial.print(os_getTime());
    Serial.print(": ");
    switch(ev) {
        case EV_SCAN_TIMEOUT:
            Serial.println(F("EV_SCAN_TIMEOUT"));
            break;
        case EV_BEACON_FOUND:
            Serial.println(F("EV_BEACON_FOUND"));
            break;
        case EV_BEACON_MISSED:
            Serial.println(F("EV_BEACON_MISSED"));
            break;
        case EV_BEACON_TRACKED:
            Serial.println(F("EV_BEACON_TRACKED"));
            break;
        case EV_JOINING:
            Serial.println(F("EV_JOINING"));
            break;
        case EV_JOINED:
            Serial.println(F("EV_JOINED"));
            break;
        case EV_RFU1:
            Serial.println(F("EV_RFU1"));
```

```
            break;
        case EV_JOIN_FAILED:
            Serial.println(F("EV_JOIN_FAILED"));
            break;
        case EV_REJOIN_FAILED:
            Serial.println(F("EV_REJOIN_FAILED"));
            break;
        case EV_TXCOMPLETE:
            Serial.println(F("EV_TXCOMPLETE (includes waiting for RX
windows)"));
            if (LMIC.txrxFlags & TXRX_ACK)
              Serial.println(F("Received ack"));
            if (LMIC.dataLen) {
              Serial.println(F("Received "));
              Serial.println(LMIC.dataLen);
              Serial.println(F(" bytes of payload"));
            }
            // Schedule next transmission
            os_setTimedCallback(&sendjob,
os_getTime()+sec2osticks(TX_INTERVAL), do_send);
            break;
        case EV_LOST_TSYNC:
            Serial.println(F("EV_LOST_TSYNC"));
            break;
        case EV_RESET:
            Serial.println(F("EV_RESET"));
            break;
        case EV_RXCOMPLETE:
            // data received in ping slot
            Serial.println(F("EV_RXCOMPLETE"));
            break;
        case EV_LINK_DEAD:
            Serial.println(F("EV_LINK_DEAD"));
            break;
        case EV_LINK_ALIVE:
            Serial.println(F("EV_LINK_ALIVE"));
            break;
         default:
            Serial.println(F("Unknown event"));
            break;
    }
}

void do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        // Prepare upstream data transmission at the next possible
time.
```

```
        LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
        Serial.println(F("Packet queued"));
    }
    // Next TX is scheduled after TX_COMPLETE event.
}

void setup() {
    Serial.begin(115200);
    Serial.println(F("Starting"));

    #ifdef VCC_ENABLE
    // For Pinoccio Scout boards
    pinMode(VCC_ENABLE, OUTPUT);
    digitalWrite(VCC_ENABLE, HIGH);
    delay(1000);
    #endif

    // LMIC init
    os_init();
    // Reset the MAC state. Session and pending data transfers will be
discarded.
    LMIC_reset();

    // Set static session parameters. Instead of dynamically
establishing a session
    // by joining the network, precomputed session parameters are be
provided.
    #ifdef PROGMEM
    // On AVR, these values are stored in flash and only copied to RAM
    // once. Copy them to a temporary buffer here, LMIC_setSession will
    // copy them into a buffer of its own again.
    uint8_t appskey[sizeof(APPSKEY)];
    uint8_t nwkskey[sizeof(NWKSKEY)];
    memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
    memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
    LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);
    #else
    // If not running an AVR with PROGMEM, just use the arrays directly
    LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);
    #endif

    #if defined(CFG_eu868)
    // Set up the channels used by the Things Network, which
corresponds
    // to the defaults of most gateways. Without this, only three base
    // channels from the LoRaWAN specification are used, which
certainly
    // works, so it is good for debugging, but can overload those
    // frequencies, so be sure to configure the full frequency range of
    // your network here (unless your network autoconfigures them).
    // Setting up channels should happen after LMIC_setSession, as that
```

```
    // configures the minimal channel set.
    // NA-US channels 0-71 are configured automatically
    LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
    LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B),
BAND_CENTI);      // g-band
    LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
    LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
    LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
    LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
    LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
    LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
    LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK),
BAND_MILLI);      // g2-band
    // TTN defines an additional channel at 869.525Mhz using SF9 for
class B
    // devices' ping slots. LMIC does not have an easy way to define
set this
    // frequency and support for class B is spotty and untested, so
this
    // frequency is not configured here.
    #elif defined(CFG_us915)
    // NA-US channels 0-71 are configured automatically
    // but only one group of 8 should (a subband) should be active
    // TTN recommends the second sub band, 1 in a zero based count.
    //
https://github.com/TheThingsNetwork/gateway-conf/blob/master/US-global_
conf.json
    LMIC_selectSubBand(1);
    #endif

    // Disable link check validation
    LMIC_setLinkCheckMode(0);

    // TTN uses SF9 for its RX2 window.
    LMIC.dn2Dr = DR_SF9;

    // Set data rate and transmit power for uplink (note: txpow seems
to be ignored by the library)
    LMIC_setDrTxpow(DR_SF7,14);

    // Start job
    do_send(&sendjob);
}
```

```
void loop() {
    os_runloop_once();
}
```

Backup der Einstellungen des Gateways

backup-dragino-1cfaf8-2019-11-03.tar.gz

multiSF channel 4 enable ☑

multiSF channel 4 radio radio1

multiSF channel 4 IF -400000

multiSF channel 5 enable ☑

multiSF channel 5 radio radio1

multiSF channel 5 IF -200000

multiSF channel 6 enable ☑

multiSF channel 6 radio radio1

multiSF channel 6 IF 0

multiSF channel 7 enable ☑

multiSF channel 7 radio radio1

multiSF channel 7 IF +200000

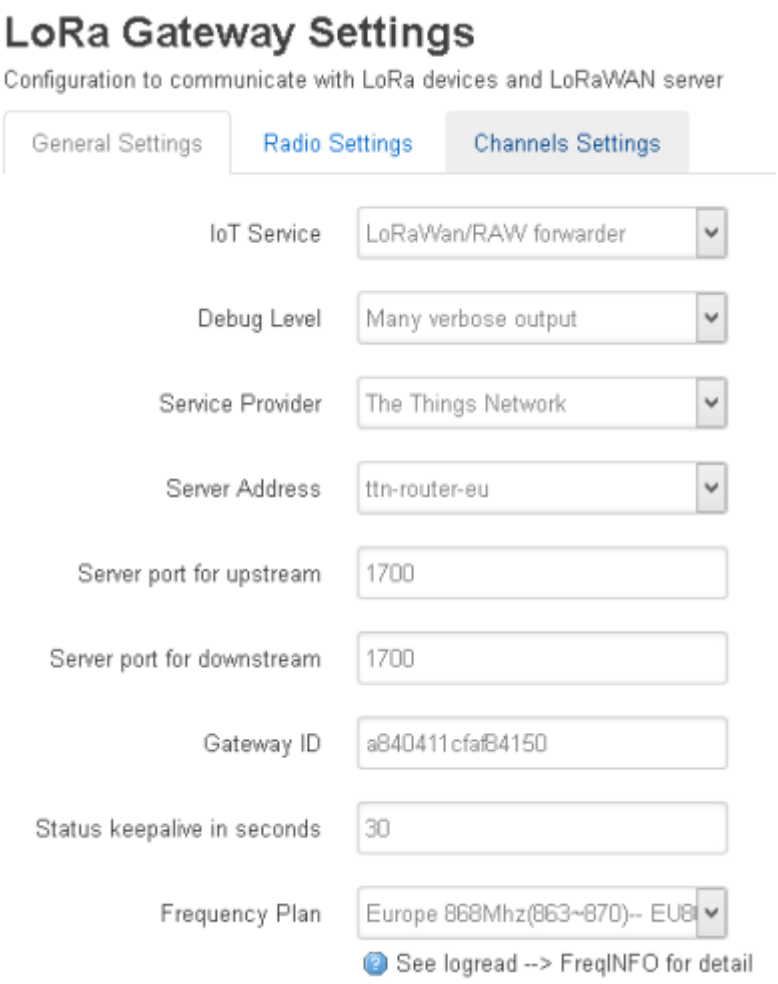
lorastd channel enable ☑

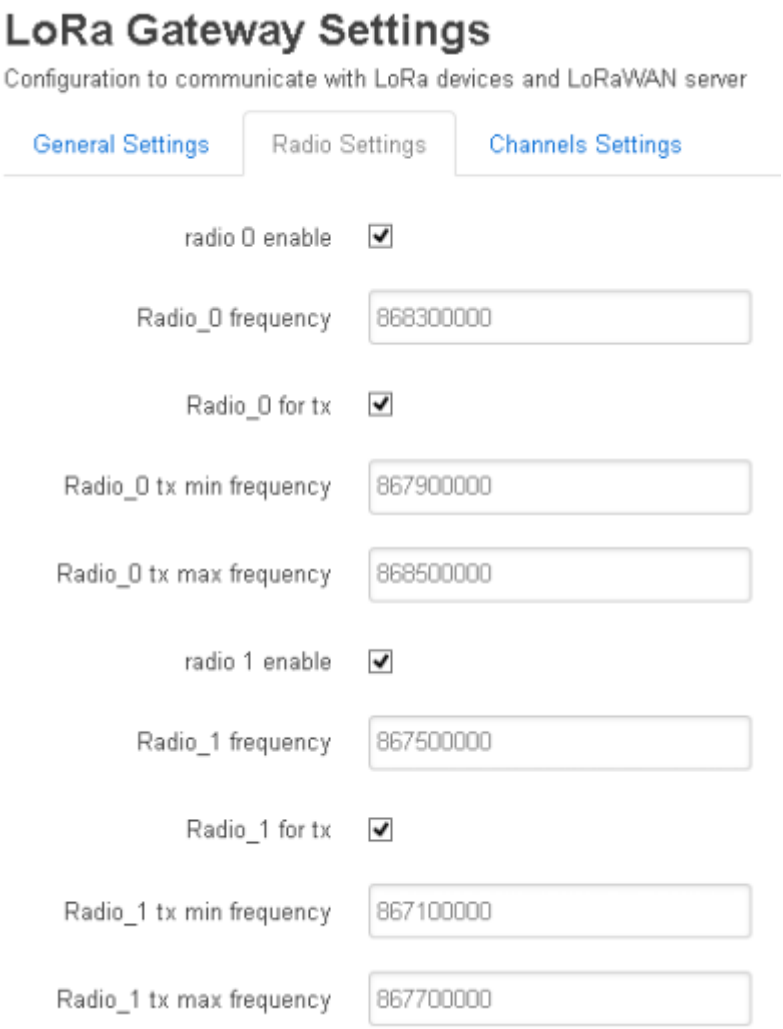
LoRa channel radio radio0

LoRa channel IF 868300000

LoRa channel SF 7

LoRa channel BW 125k


LoRa Gateway Settings
Configuration to communicate with LoRa devices and LoRaWAN server
General Settings
Radio Settings
Channels Settings
IoT Service
LoRaWan/RAW forwarder
Debug Level
Many verbose output
Service Provider
The Things Network
Server Address
ttn-router-eu
Server port for upstream
1700
Server port for downstream
1700
Gateway ID
a840411cfaf84150
Status keepalive in seconds
30
Frequency Plan
Europe 868Mhz(863~870)-- EU8
See logread --> FreqINFO for detail



From:
https://wiki.hackerspace-bremen.de/ - **Hackerspace Bremen e.V.**

Permanent link:
https://wiki.hackerspace-bremen.de/geraetschaften/infrastruktur/lora_gateway

Last update: **2023-02-27 01:15**