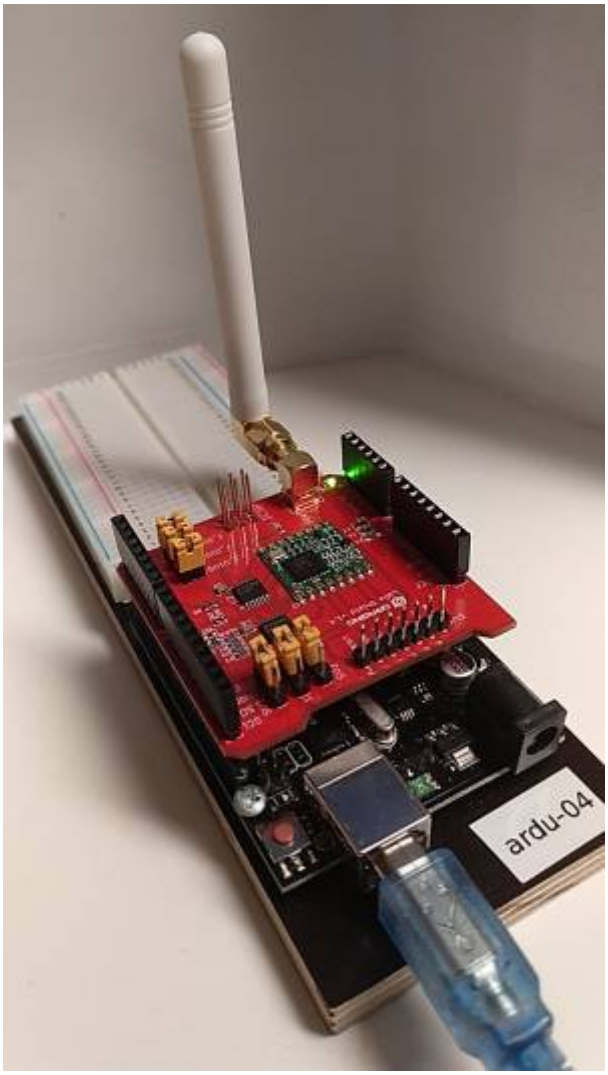
Ein "Standardsensor" mit dem LoRaShield

LoraShield aufstecken



Bibliothek installieren

Um das Shield ansprechen zu können, muss die LMIC Bibliothek installiert werden.

- Zip Datei herunterladen (Source: https://github.com/matthijskooijman/arduino-lmic)
- Installieren mit `Sketch→ Bibliothek einbinden → ZIP-Bibliothek hinzufügen`

Sketch

Klicken, um den Beispielsketch zu sehen

hello_lora.ino

```
// MIT License
//
https://github.com/gonzalocasas/arduino-uno-dragino-lorawan/blob/master
/LICENSE
// Based on examples from
https://github.com/matthijskooijman/arduino-lmic
// Copyright (c) 2015 Thomas Telkamp and Matthijs Kooijman

// Adaptions: Andreas Spiess

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

static const u1_t NWKSKEY[16] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00};
static const u1_t APPSKEY[16] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00};
static const u4_t DEVADDR = 0x00000000;


// 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 osjob_t sendjob;

// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations).
const unsigned TX_INTERVAL = 20;

// Pin mapping Dragino Shield
const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 9,
    .dio = {2, 6, 7},
};
void onEvent (ev_t ev) {
    if (ev == EV_TXCOMPLETE) {
        Serial.println(F("EV_TXCOMPLETE (includes waiting for RX
windows)"));
```

```
        // Schedule next transmission
        os_setTimedCallback(&sendjob,
os_getTime()+sec2osticks(TX_INTERVAL), do_send);
    }
}

void do_send(osjob_t* j){
    // Payload to send (uplink)
    static uint8_t message[] = "hi";

    // 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, message, sizeof(message)-1, 0);
        Serial.println(F("Sending uplink packet..."));
    }
    // Next TX is scheduled after TX_COMPLETE event.
}

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

    // LMIC init
    os_init();

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

    // Set static session parameters.
    LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);

    // 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_SF12,14);

    // Start job
    do_send(&sendjob);
}

void loop() {
```

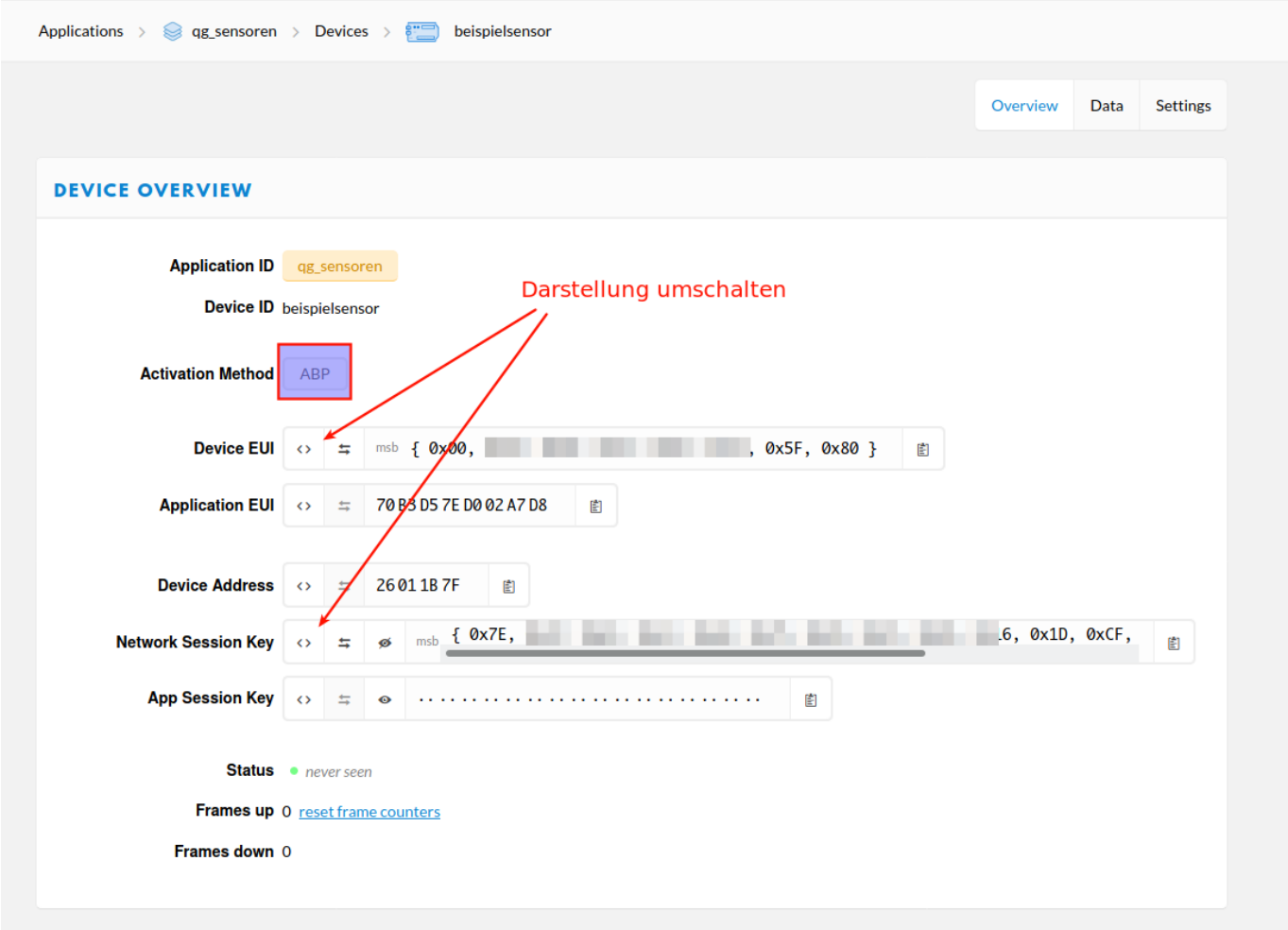
```
    os_runloop_once();
}
```

Neues Device bei TTN erstellen

Nun benötigt man in der TTN Console eine Application und muss dort ein neues Device anlegen. Man vergibt einen Namen und lässt die übrigen Werte berechnen. Das Ergebnis sieht so aus, wie im Screenshot zu sehen.

Wichtig:

- Das Device muss als Activation Method ABP haben
- Um die passenden IDs im Sketch einzutragen, müssen diese das passende Format haben - die Darstellung kann man durch anklicken von <> umschalten.



IDs in den Sketch eintragen

- NWKSKEY: Network Session Key
- APPSKEY: Application Session Key

- DEVADDR: Device Address

```
static const u1_t NWKSKEY[16] = { 0x7E, [redacted] 0xD3, 0x16, 0x1D, 0xCF, 0xA2, 0x54, 0xD6, 0x7E };
static const u1_t APPSKEY[16] = { 0x3A, 0x5A, 0x30, 0xDC, 0x43, [redacted] 0xFB, 0x1A, 0x70, 0x5A };
static const u4_t DEVADDR = 0x2[redacted]37F;
```

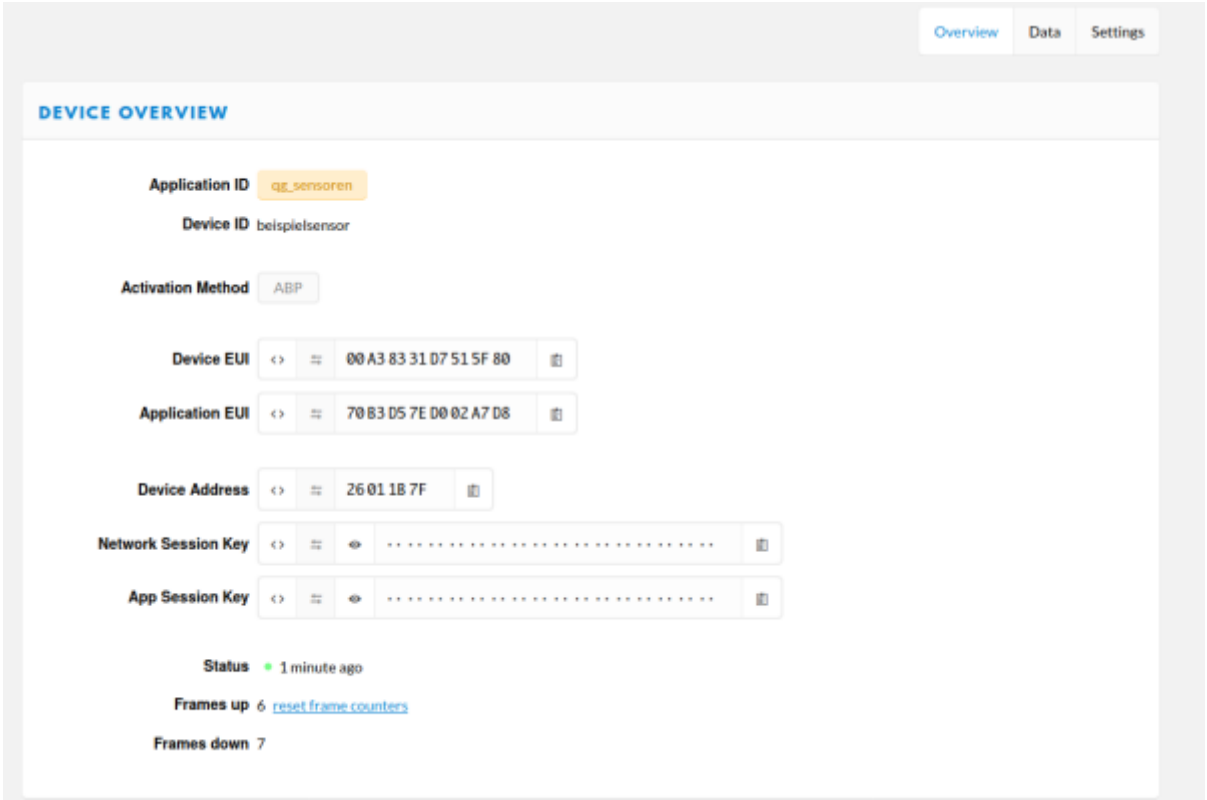
Kompilieren und Testen

Beim Kompilieren erscheinen Warnungen, die kann man ignorieren.

```
// 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 osjob_t sendjob;

// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations).
const unsigned TX_INTERVAL = 20;

// Pin mapping Dragino Shield
```

```
Hochladen abgeschlossen.
/tmp/mozilla_frank0/hello_lora/hello_lora.ino:76:52: warning: invalid conversion from 'const u1_t* {aka const unsigned char*}' to 'xref2u1_t {aka unsigned char*}' [-fpermis
     LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);
                                             ^
In file included from /home/frank/Arduino/libraries/arduino-lmic-master/src/lmic.h:5:0,
                 from /tmp/mozilla_frank0/hello_lora/hello_lora.ino:8:
/home/frank/Arduino/libraries/arduino-lmic-master/src/lmic/lmic.h:305:6: note:   initializing argument 3 of 'void LMIC_setSession(u4_t, devaddr_t, xref2u1_t, xref2u1_t)'
 void LMIC_setSession (u4_t netid, devaddr_t devaddr, xref2u1_t nwkKey, xref2u1_t artKey);
      ^~~~~~~~~~~~~~
/tmp/mozilla_frank0/hello_lora/hello_lora.ino:76:52: warning: invalid conversion from 'const u1_t* {aka const unsigned char*}' to 'xref2u1_t {aka unsigned char*}' [-fpermis
     LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);
                                             ^
In file included from /home/frank/Arduino/libraries/arduino-lmic-master/src/lmic.h:5:0,
                 from /tmp/mozilla_frank0/hello_lora/hello_lora.ino:8:
/home/frank/Arduino/libraries/arduino-lmic-master/src/lmic/lmic.h:305:6: note:   initializing argument 4 of 'void LMIC_setSession(u4_t, devaddr_t, xref2u1_t, xref2u1_t)'
 void LMIC_setSession (u4_t netid, devaddr_t devaddr, xref2u1_t nwkKey, xref2u1_t artKey);
      ^~~~~~~~~~~~~~
Der Sketch verwendet 19142 Bytes (59%) des Programmspeicherplatzes. Das Maximum sind 32256 Bytes.
Globale Variablen verwenden 867 Bytes (42%) des dynamischen Speichers, 1181 Bytes für lokale Variablen verbleiben. Das Maximum sind 2048 Bytes.
```

Wenn man den Sketch auf den Arduino lädt, sollte man bei TTN sehen, dass das Device aktiv ist:



Im Data-Tab kann man jetzt auch sehen, dass da Daten übermittelt werden: