**0 BLINK**

// the setup function runs once when you press reset or power the board

void setup() {

// initialize digital pin LED\_BUILTIN as an output.

pinMode(LED\_BUILTIN, OUTPUT);

**Serial**.begin(115200);

while (!**Serial**);

**Serial**.println(LED\_BUILTIN);

}

// the loop function runs over and over again forever

void loop() {

digitalWrite(LED\_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second

digitalWrite(LED\_BUILTIN, LOW); // turn the LED off by making the voltage LOW

delay(1000); // wait for a second

}

**1 Get Parameters from Board**

/\*

STFW IoT

This sketch demonstrates the usage of MKR WAN 1300/1310 LoRa module.

\*/

#include <**MKRWAN**.h>

// LoRa Modem initialisieren:

**LoRaModem** modem;

void setup() {

// Serial Monitor initialisieren

**Serial**.begin(115200);

// Warten bis Serial Port bereit ist

while (!**Serial**);

**Serial**.println("Arduino MKRWAN 1310: Geräteparameter auslesen");

// EU868 => EU (Europa) + 868 MHz

if (!modem.begin(EU868)) {

**Serial**.println("[Fehler] LoRa-Modem konnte nicht gestartet werden.");

while (1) {}

};

**Serial**.print("[Info] LoRa-Modem (Modul) Version: ");

**Serial**.println(modem.version());

**Serial**.print("[Info] Device\_EUI: ");

**Serial**.println(modem.deviceEUI());

// initialize digital pin LED\_BUILTIN as an output.

pinMode(LED\_BUILTIN, OUTPUT);

}

void loop() {

// HIGH = 3.3V, LOW = 0V

digitalWrite(LED\_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second (1000ms)

digitalWrite(LED\_BUILTIN, LOW); // turn the LED off by making the voltage LOW

delay(1000); // wait for a second

}

**2.1 Minimal Working Example**

// Minimal working example

#include <**MKRWAN**.h> // Library

#include "arduino\_secrets.h" // Keys des Geräts

**LoRaModem** modem; // modem Objekt initialisieren

String appEui = SECRET\_APP\_EUI;

String appKey = SECRET\_APP\_KEY;

void setup() {

modem.begin(EU868); // Modem starten

modem.joinOTAA(appEui, appKey); // Am LoRaWAN-Netzwerk (z.B. TTN) anmelden (join)

modem.minPollInterval(300); // Nicht häufiger als alle 300s = 5min senden

}

void loop() {

modem.beginPacket(); // Übertragungspaket eröffnen

modem.write(77); // 77 ist der payload, hier beliebigen Wert reinschreiben

modem.endPacket(true); // Mit endPacket wird das Paket geschlossen und abgeschickt

delay(300\*1000);

}

**2 Send Message with LoRaWAN**

/\*

Lora Send And Receive

This sketch demonstrates how to send and receive data with the MKR WAN 1310 LoRa module.

\*/

#include <**MKRWAN**.h>

// LoRaWAN-Keys in diese Datei eintragen:

#include "arduino\_secrets.h"

**LoRaModem** modem;

// Please enter your sensitive data in the Secret tab (or arduino\_secrets.h)

String appEui = SECRET\_APP\_EUI;

String appKey = SECRET\_APP\_KEY;

void setup() {

// put your setup code here, to run once:

**Serial**.begin(115200);

while (!**Serial**);

// change this to your regional band (eg. US915, AS923, ...)

if (!modem.begin(EU868)) {

**Serial**.println("Failed to start module");

while (1) {}

};

**Serial**.print("Your device EUI is: ");

**Serial**.println(modem.deviceEUI());

// Now the magic begins ;-)

// Let's connect via LoRaWAN to the next gateway

// OTAA = Over The Air Activation

// Wir möchten uns jetzt mit dem Netzwerk verbinden:

int connected = modem.joinOTAA(appEui, appKey);

if (connected) {

**Serial**.println("Yay! Connected! Helloooo big IoT-world!");

} else {

**Serial**.println("Could not connect; are you indoor? Move near a window and retry");

while (1) {}

}

// Set poll interval to 300 secs.

modem.minPollInterval(300);

}

void loop() {

// was bedeutet '77'?

// Aufgabe: Wie kann eine von Euch definierte Zahl übertragen werden?

// - int

// - float

// Wie kann diese Zahl als Byte geschrieben werden?

//

// Bsp: Temperatur von 23.33 °C (float)

// Bsp: CO2: 666 ppm (int)

// ...

// Tip: Dokumentation lesen: Byte

// int (0..255) (2^8) => 1 byte

// int (0 .. 65'536) (2^16) => 2 bytes

// byte msg = 77;

// Beispiele für inputs:

// Datentyp: word = 2 bytes

int humidity = 45; // Ganzzahl positiv

int temperature = -12; // Ganzzahl negativ

word msg = 666; // Ganzzahl > 255 mit zwei Bytes

// Hier kommt unser Code hin

**Serial**.print("Sending: ");

**Serial**.print(msg);

**Serial**.print(" - converted to HEX: ");

**Serial**.print(msg, HEX);

**Serial**.print(" ");

**Serial**.println();

int err;

// Datenpaket / Payload beginnen

modem.beginPacket();

// Payload schreiben (Nutzdaten)

byte msg1 = 77;

modem.write(msg1);

byte msg2 = 255;

modem.write(msg2);

// Payload beenden

err = modem.endPacket(true);

if (err > 0) {

**Serial**.println("Message sent correctly!");

} else {

**Serial**.println("Error sending message :(");

**Serial**.println("(you may send a limited amount of messages per minute, depending on the signal strength");

**Serial**.println("it may vary from 1 message every couple of seconds to 1 message every minute)");

}

// 300 \* 1000ms

delay(300\*1000);

}

**3.2: Anderer Sensor**

#include <**MKRWAN**.h>

// Variablen um die eingelesenen Sensorwerte zu speichern

int digitalData;

int analogData;

void setup() {

// D0, digital pin 0 als Input definieren

pinMode(0, INPUT);

// A0, analog pin 0 als Input definieren

pinMode(A0, INPUT);

// Serielle Verbindung öffnen um auf der Console

// Debug-Nachrichten auszugeben

**Serial**.begin(115200);

while (!**Serial**);

// Starten mit Info auf Console

**Serial**.println("Sensortest:");

}

// Alles in dieser Funktion wird immer wieder ausgeführt...

void loop() {

// Digitaleingang einlesen

// Resultat (digitalData) ist HIGH (3.3V) oder LOW (0V)

digitalData = digitalRead(0);

**Serial**.println(digitalData);

// Resultat (analogData) ist zwischen 0..1023

// Der Analogeingang hat eine Auflösung von 10bits

// 0 entspricht 0V, 1023 entspricht 3.3V

analogData = analogRead(A0);

**Serial**.println(analogData);

// Verzögerung in Millisekunden bis die Funktion loop() erneut aufgerufen wird.

delay(200);

}

**3: ENV Shield**

#include <**Arduino\_MKRENV**.h>

void setup() {

**Serial**.begin(9600);

while (!**Serial**);

if (!**ENV**.begin()) {

**Serial**.println("MKR ENV Shield konnte nicht initialisiert werden");

while (1); // unendlich lange warten...

}

}

void loop() {

float temperature = **ENV**.readTemperature();

float humidity = **ENV**.readHumidity();

float pressure = **ENV**.readPressure();

float illuminance = **ENV**.readIlluminance();

**Serial**.print("Temperature = ");

**Serial**.print(temperature);

**Serial**.println(" °C");

**Serial**.print("Humidity = ");

**Serial**.print(humidity);

**Serial**.println(" %");

**Serial**.print("Pressure = ");

**Serial**.print(pressure);

**Serial**.println(" kPa");

**Serial**.print("Illuminance = ");

**Serial**.print(illuminance);

**Serial**.println(" lx");

delay(5000);

}

**6 Send and receive**

/\* Lora Send And Receive \*/

#include <MKRWAN.h>

LoRaModem modem;

#include "arduino\_secrets.h"

// Please enter your sensitive data in the Secret tab or arduino\_secrets.h

String appEui = SECRET\_APP\_EUI;

String appKey = SECRET\_APP\_KEY;

void setup() {

// put your setup code here, to run once:

**Serial**.begin(115200);

while (!**Serial**);

// change this to your regional band (eg. US915, AS923, ...)

if (!modem.begin(EU868)) {

**Serial**.println("Failed to start module");

while (1) {}

};

**Serial**.print("Your module version is: ");

**Serial**.println(modem.version());

**Serial**.print("Your device EUI is: ");

**Serial**.println(modem.deviceEUI());

int connected = modem.joinOTAA(appEui, appKey);

if (!connected) {

**Serial**.println("Something went wrong; are you indoor? Move near a window and retry");

while (1) {}

}

// Set poll interval to 60 secs.

modem.minPollInterval(60);

// NOTE: independently by this setting the modem will

// not allow to send more than one message every 2 minutes,

// this is enforced by firmware and can not be changed.

}

void loop() {

**Serial**.println("--------------------------------------------");

**Serial**.println("Enter a message to send to network");

**Serial**.println("(make sure that end-of-line 'NL' is enabled)");

while (!**Serial**.available());

String msg = **Serial**.readStringUntil('\n');

**Serial**.println();

**Serial**.print("Sending: " + msg + " - ");

for (unsigned int i = 0; i < msg.length(); i++) {

**Serial**.print(msg[i] >> 4, HEX);

**Serial**.print(msg[i] & 0xF, HEX);

**Serial**.print(" ");

}

**Serial**.println();

int err;

modem.beginPacket();

modem.print(msg);

err = modem.endPacket(true);

if (err > 0) {

**Serial**.println("Message sent correctly!");

} else {

**Serial**.println("Error sending message :(");

**Serial**.println("(you may send a limited amount of messages per minute, depending on the signal strength");

**Serial**.println("it may vary from 1 message every couple of seconds to 1 message every minute)");

}

delay(1000);

if (!modem.available()) {

**Serial**.println("No downlink message received at this time.");

return;

}

char rcv[64];

int i = 0;

while (modem.available()) {

rcv[i++] = (char)modem.read();

}

**Serial**.print("Received: ");

for (unsigned int j = 0; j < i; j++) {

**Serial**.print(rcv[j] >> 4, HEX);

**Serial**.print(rcv[j] & 0xF, HEX);

**Serial**.print(" ");

}

**Serial**.println();

}