```
/*************_Uitleg van de schepper van dit programma_***********************
 *
 * 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.
 *
```

```
**********_Hoe alle hardware installeren_**********
 *
 *  DHT11 temperatuur en vochtigheidsmeter
 *  3pin Pinout :
 *               Signaal(D3) - 3.3V - GND
 *
 *  LDR Pinout :
 *              5V - Signaal(A0)
 *                 - Resistor 1000 Ohm naar GND
 *
 *  Relais Pinout :
 *              Signaal(D4) - 5V - GND
 *
 *  RGB Led Pinout :
 *              Blauw(D10)
 *              Groen(D11)
 *              Rood(D12)
 *              - => GND
 *
 *  Als het even is als "threshold = 28", dan groen licht aan.
 *  Als het warmer is als "threshold = 28", dan rood licht aan.
 *  Als het kouder is als "threshold = 28", dan blauw licht aan
 *          en relais aan voor ventilator.
 *              Ventilator heeft hoger Voltage nodig die Arduino niet kan voorzien.
 *              Ventilator aangesloten op 12V batterij als volgt:
 *                  venti + --> + batterij
 *                  venti - --> COM Relais
 *                NO Relais --> - batterij want willen pas AAN als signaal erdoor
 *                              NO = Normaal open, als signaal erdoor --> volle elektriciteitskring
 *                              NC = Normaal gesloten, als signaal erdoor --> geen contact, verbroken.
 *
 *  Geel LED licht voor BINNEN licht :
 *               Signaal(D5) - 1000 Ohm naar GND
 *
 *  Groen LED licht voor BUITEN licht :
 *               Signaal(D6) - 1000 Ohm naar GND
 *
 *  Created by Joachim Pham               Email: joachimpham@hotmail.com
 *  23/08/2020
 ***************************************************************************/
```

```
//********************_DHT needs_**********************
#include "DHT.h"
#define DHTTYPE DHT11          //type DHT sensor
#define DHTPIN 3            //input van DHT signal
DHT dht(DHTPIN, DHTTYPE);

const int thresholdTemp = 24;   //thermostaat waarde
float temp, vocht;
int tem, voc, lux, lichtBinnen, lichtBuiten;


//*******_Relay als schakelaar voor de Ventilator_*******

    #define RELAYPIN 4

//****************_LDR licht sensor_********************

    #define LDRPIN A0

    float licht;
    const int thresholdLicht = 600;       //lichtwaarde limiet

//***************_RGB LED  als thermostaat_**************

    #define ROODPIN  12        //Rood    -->   verwarming staat aan
    #define GROENPIN 11       //Groen   -->   alles staat af, tresholdTemperatuur
    #define BLAUWPIN 10        //Blauw   -->   ventilator staat aan

//***************_Binnen en buitenlicht_*****************

    #define BINNEN A1                     // <-- Aanpassing naar Analoge voor betere functie licht
    #define BUITEN A2

//******************_Licht schakelaar_*******************

    #define KNOPPIN 13

    int knopStatus;      // Status van de knop: niet ingedrukt = 0, ingedrukt = 1
    int lichtStatus = HIGH;      // huidige status lichtpin
    int laatsteknopstatus = LOW; // vorige lezing knop
```

```
    unsigned long lastDebounceTime = 0;     //laaste keer dat outpin werd gebruikt
    unsigned long debounceDelay = 20;     //de debounce tijd, als led flikkert, verhogen.


//***************_LoRa benodigdheden_********************

       // Deze info kun je krijgen als je een account aanmaakt op "thethingsnetwork.org".
       // Klik op "console", log in, en klik op "application"
       // Maak daar een nieuwe applicatie aan,
       //       "Application ID" is korte beschrijving in kleine letters bv: "auto-sauna"
       //       "Description" is voor jezelf duidelijk te maken waarvoor je deze app gaat gebruiken.
       //             bv: temperatuur en vochtigheid, auto lichten en updates.
       //       "App EUI" laat je door TTN beslissen
       //       "Handler" kies je de Europese;    ttn-handler-eu
       // Klik op "Add aplication"

       // Klik nadien op de applicatie, kijk bij "devices" en registreer één.
       //       "Device ID" is weer korte beschrijving in kleine letters waar en welk toestel je gebruikt
       //       "Device EUI" kun je zelf kiezen of laten kiezen voor TTN door shuffle icoon
       //       "App Key" laat je ook best kiezen door TTN
       //       "App EUI" zou al ingevuld moeten zijn
       // Klik op "Register"

       // Klik op jouw net-aangemaakte-toestel en dan op "Settings":
       //     Zorg dat "Activation Methode" op "ABP"
       // Klik op "Save"

       // Bij "Device Overview" krijg je alle info die je hieronder nodig hebt:
       // Al deze info staat in tekst, klik op "<>" om deze om te vormen naar HEX-taal voor computers:
       //       "Network Session Key"         = "0x.., 0x.., 0x.., 0x.., ..."
       //       "Application Session Key"     = "0x.., 0x.., 0x.., 0x.., ..."
       //       "Device Address"              = 26 .. .. ..       --> v0x26.........
       // Kopie en paste deze info hieronder in het programma en je bent klaar!
```

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


// LoRaWAN NwkSKey, network session key
static const PROGMEM u1_t NWKSKEY[16] = { 0x8C, 0xF5, 0xB4, 0x16, 0x69, 0x1E, 0x6E, 0xD3, 0xD6, 0xCB, 0xBF, 0xD5,
0x79, 0xC0, 0x32, 0x17 };

// LoRaWAN AppSKey, application session key
static const u1_t PROGMEM APPSKEY[16] = { 0x22, 0x28, 0xD8, 0x6F, 0x43, 0x7C, 0x0E, 0xCE, 0xB7, 0x51, 0xF0, 0xAA,
0x1A, 0x7B, 0x8A, 0x9B };

// LoRaWAN end-device address (DevAddr)
static const u4_t DEVADDR = 0x260111B7 ;            // <-- 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[17] = {0x01,0x67,0x00,0x00,0x02,0x68,0x00,0x03,0x65,0x00,0x00,0x04,0x0,0x00,0x05,0x1,0x00};
// bekijk de data type en structuur op      // meer info bij void alldata()
static osjob_t initjob, sendjob;                                                                    //
https://developers.mydevices.com/cayenne/docs/lora/#lora-cayenne-low-power-payload-payload-structure

// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations).
const unsigned TX_INTERVAL = 10;

// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 9,
    .dio = {2, 6, 7},
};
```

```
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){

      all4data_send();          // nieuwe void voor de data aan te passen

    // 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), 0);
        Serial.println(F("Packet queued"));
    }
    // Next TX is scheduled after TX_COMPLETE event.
}
//**************************************************
```

```
void setup() {                // bewaar de eigenschap van elke pin en zijn functie
    Serial.begin(115200);
    Serial.println(F("Starting LoRa"));

    // LDR licht sensor
    pinMode(LDRPIN, INPUT);

    // Relay
    pinMode(RELAYPIN, OUTPUT);

    // RGB led
    pinMode(ROODPIN, OUTPUT);
    pinMode(GROENPIN, OUTPUT);
    pinMode(BLAUWPIN, OUTPUT);

    // Binnen Buiten licht
    pinMode(BINNEN, OUTPUT);
    pinMode(BUITEN, OUTPUT);

    digitalWrite(BINNEN, lichtStatus);    //lichtstatus gaat veranderen naar HIGH/LOW
    digitalWrite(BUITEN, lichtStatus);    //waardoor deze direct de lichten aan of uit zet


    // Knop
    pinMode(KNOPPIN, INPUT);

    //DHT
      pinMode(DHTPIN, INPUT);
      dht.begin();

    Serial.print("DHT Temperatuur en Vochtigheid Start");
    Serial.println("");
    Serial.print("Temperatuur threshold is gezet op: ");
    Serial.print(thresholdTemp);
    Serial.print("°C");
    Serial.println("");
    Serial.print("Licht threshold is gezet op: ");
    Serial.print(thresholdLicht);
    Serial.print(" lux");
    Serial.println("");
```

```
//***************_Als er geen data is van DHT meter_*************

if (isnan(temp) ||  isnan(vocht)  ||  isnan(licht))  {
  Serial.println(F("Kijk even na of alle meet-instrumenten juist aangesloten zijn!"));
  return;
}

//*********************_Setup voor LoRa communicatie_*****************************

    #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

    // 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);
}


//*******_Nieuwe commando voor data te lezen en configureren voor verzending_*********

void all4data_send() {

          //************_Lees metingen_**********************

  float licht = analogRead(LDRPIN);         // lees lichtsterkte
  float temp = dht.readTemperature();       // lees temperatuur
  float vocht = dht.readHumidity();         // lees vochtigheid


      //****************_Bericht opmaken voor CayenneLPP_***************

        // in uint8_t mydata[7] bovenaan hebben we een structuur opgebouwd:
//  {   0x01,         0x67,             0x00,             0x00,     0x02,         0x68,             0x00,     0x03,
...}
//  in vak 1, data type 67, 1e deel temp 000, 2e deel temp 000, in vak 2, data type 68, vochtigheid 000, in vak 3,
...

// data type HEX = type,        data size (= grootte in bytes)
// data type 67 = temperatuur, krijgt 2 bytes (= 16 bits) <-- 0,1°C signed    <== als waarde 1 is = 0,1°C , als 100
is = 10°C
// data type 68 = vochtigheid, krijgt 1 byte (= 8 bits)   <-- 0,5% unsigned   <== als waarde 1 is = 0,5%  , als 100
is = 50%
// data type 65 = lichtigheid, krijgt 2 bytes (= 16 bits) <-- 1 lux unsigned  <== als waarde 1 is = 1 lux , als 100
is = 100 lux

// "data resolution per bit" = wilt zeggen hoeveel het is voor LPP als de waarde 1 is bij elke type.
// meer info op https://developers.mydevices.com/cayenne/docs/lora/#lora-cayenne-low-power-payload-data-types

// arrays beginnen vanaf 0 te tellen!       mydata[17](=array[16]+1)
```

```
// mydata[17] =  {0x01 -     0x67 -     0x00 -     0x00 -     0x02 -     0x68 -     0x00 -     0x03 -     0x65 -
0x00 -      0x00 -      0x04 -       0x0 -      0x00 -      0x05 -       0x0 -   0x00}
//            array[0] - array[1] - array[2] - array[3] - array[4] - array[5] - array[6] - array[7] - array[8] -
array[9] - array[10] - array[11] - array[12] - array[13] - array[14] - array[15] - array[16]

// we willen de 0x00 vervangen door data die we hierboven lezen met functie temp, vocht
// temperatuur gaat bv 22°C zijn --> 0,1°C per bit --> 220
// vochtigheid gaat bv 50   zijn --> 0,5%  per bit --> 100

    tem = dht.readTemperature()*1.0;           //22 --> 22.0
    voc = dht.readHumidity()*1.0;          //50 --> 50.0
    lux = analogRead(LDRPIN);
    lichtBinnen = digitalRead(BINNEN);
    lichtBuiten = digitalRead(BUITEN);

  int16_t tem_LPP;
  tem_LPP = temp*10;                    //22.0 --> 220 gaat in 16 bits staan: 01010101 00000000

  int16_t lux_LPP;
  lux_LPP = lux;


       //**********************__***********************

  mydata[2]  = tem_LPP>>8;              // tem_LPP>>8 = 00000000 01010101 = begin pas na bit 8
  mydata[3]  = tem_LPP;                 // tem_LPP =    01010101 00000000 = begin vanaf begin
  mydata[6]  = voc * 2;
  mydata[9]  = lux_LPP>>8;              // lux_LPP>>8 = 00000000 01010101 = begin pas na bit 8
  mydata[10] = lux_LPP;                 // lux_LPP =    01010101 00000000 = begin vanaf begin
  mydata[13] = lichtBinnen;             // digital output    1 of 0
  mydata[16] = lichtBuiten;             // digital output    1 of 0


     //*************_Print de waardes van temp en vocht_***************

Serial.print("");
Serial.println("------------------------");
Serial.print("De temperatuur is: ");
Serial.print(temp);
Serial.print(" °C");
```

```
Serial.println(" ");
Serial.print("De vochtigheid is: ");
Serial.print(vocht);
Serial.println(" %\t");

      //*****************_Print waarde licht_*************

Serial.print("Het licht is: ");
Serial.print(licht);
Serial.println(" lux");

delay(100);
}


void loop() {                 // In de Void loop() : worden alle onderstaande functies telkens opnieuw uitgevoerd.

        //***************_Verzending van data_**************

          os_runloop_once();

      //***********_Manuele knop besturing licht_******************

 int leesin = digitalRead(KNOPPIN);

if (leesin != laatsteknopstatus) {
  lastDebounceTime = millis();
}

if ((millis() - lastDebounceTime) > debounceDelay) {

  if (leesin != knopStatus) {
    knopStatus = leesin;

    if (knopStatus == HIGH) {
      lichtStatus = !lichtStatus;
    }
  }
}
```

```
digitalWrite(BINNEN, lichtStatus);
digitalWrite(BUITEN, lichtStatus);

laatsteknopstatus = leesin;

          //************_Lees metingen_**********************

  float licht = analogRead(LDRPIN);         // lees lichtsterkte
  float temp = dht.readTemperature();       // lees temperatuur
  float vocht = dht.readHumidity();         // lees vochtigheid

      //*************_Print de waardes van temp en vocht_***************

Serial.print("");
Serial.println("------------------------");
Serial.print("De temperatuur is: ");
Serial.print(temp);
Serial.print(" °C");

Serial.println(" ");
Serial.print("De vochtigheid is: ");
Serial.print(vocht);
Serial.println(" %\t");

      //*****************_Print waarde licht_*************

Serial.print("Het licht is: ");
Serial.print(licht);
Serial.println(" lux");

              //******************_Auto-Temperatuur_****************

if (temp < thresholdTemp)  {             //Als de gemeten temperatuur LAGER/KLEINER is dan vaste waarde, verwarming
gaat aan, ROOD.
  digitalWrite(ROODPIN, HIGH);
  digitalWrite(GROENPIN, LOW);
  digitalWrite(BLAUWPIN, LOW);
  digitalWrite(RELAYPIN, LOW);
  Serial.println("Verwarming AAN,");
  Serial.println("Ventilatie UIT.");
}
```

```
if (temp > thresholdTemp)  {             //Als de gemeten temperatuur HOGER/GROTER is dan vaste waarde, ventilatie
gaat aan, BLAUW.
  digitalWrite(ROODPIN, LOW);
  digitalWrite(GROENPIN, LOW);
  digitalWrite(BLAUWPIN, HIGH);
  digitalWrite(RELAYPIN, HIGH);
  Serial.println("Verwarming UIT,");
  Serial.println("Ventilatie AAN.");
}

if (temp == thresholdTemp)  {          //Als de gemeten temperatuur GELIJK is als vaste waarde, niets gaat aan, GROEN.
  digitalWrite(ROODPIN, LOW);
  digitalWrite(GROENPIN, HIGH);
  digitalWrite(BLAUWPIN, LOW);
  digitalWrite(RELAYPIN, LOW);
  Serial.println("Alles UIT.");
}

      //********************_Auto-licht_******************

if (licht < thresholdLicht) {              // Als het donkerder is dan thresholdLight, dan gaat licht aan.
  analogWrite(BINNEN, HIGH);
  analogWrite(BUITEN, HIGH);
  Serial.println("Lichten AAN");
}

if (licht > thresholdLicht) {              // Anders lichten uit
  analogWrite(BINNEN, LOW);
  analogWrite(BUITEN, LOW);
  Serial.println("Lichten UIT");
}
```

```
/*

if (knopstatus == HIGH) {
  digitalWrite(BINNEN, HIGH);
  digitalWrite(BUITEN, HIGH);
  Serial.println("Lichten AAN met knop");
}
else
{
  digitalWrite(BINNEN, LOW);
  digitalWrite(BUITEN, LOW);
  Serial.println("Lichten UIT met knop");
}


delay(1000);

*/
}
```