**Software inlezen van liturgische titel voor de livestream uit database (mysql) Mariakerk**

**pythonleestitel7.ino**

```
#include <Ethernet.h> //Load Ethernet Library
#include <EthernetUdp.h> //Load UDP Library
#include <SPI.h> //Load the SPI Library
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "RTClib.h"
// beperking: als een livestream geweest is, dan moet dit gereset, dus opnieuw gestart worden
// dit zal normaliter vanzelf gebeuren, of reset drukken
// indien beginuur bekend, dan voor dienst bij volgaan en micros uit automatisch naar de hoofddia met titel CTRL-Z bij volgmij
// indien geen beginuur bekend dan automatisch naar hoofddia zonder titel
// internetaansluiting is vereist
// de interne klok wordt gereset via python liturgietitelmet.exe die ook de correcte huidige tijd meestuurt
// er is geen pemanente usb verbinding nodig dus

#include <TM1637Display.h>
#define DIO A2  // voor het vorige display let gewisseld tgo vorige versies
#define CLK A3
#define UDP_TX_PACKET_MAX_SIZE 48 // standaard is 24, dan net 1 letter te kort!
TM1637Display display = TM1637Display(CLK, DIO);
RTC_DS3231 rtc; // dit is de klok
char daysOfTheWeek[7][12] = {"zo", "ma", "di", "wo", "do", "vr", "za"};
int pythoninfo, gezieninfo, gezieninfo2, begonnen;
int uur, minuut, seconde, jaar,toontijd;
String uurnu, minuutnu, secondenu, jaarnu, toontijdnu;
int gevonden, lengte, istijd;
const int autotitel     = 6; // geel, mag enkel aan indien begintijd bekend is en niet overschreden, moet naar volgmij
const int autostart     = 7; // indien hoog, dus bij titel en na begintijd levert autostart ... moet puls zijn?????
unsigned long misabsoltijd, nuabsoltijd;
String begintijd, begintijddef;
LiquidCrystal_I2C lcd(0x27,20,4);  // set the LCD address to 0x27 for a 16 chars and 2 line display
byte mac[] = {0x2C, 0xF7, 0xF1, 0x08, 0x33, 0xE6 }; //Assign a mac address
IPAddress ip(192, 168, 2, 167); //Assign my IP adress
unsigned int localPort = 5000; //Assign a Port to talk over
char packetBuffer2[UDP_TX_PACKET_MAX_SIZE];
int packetSize; //Size of Packet
EthernetUDP Udp2; //Define UDP Object

void setup()
{
  display.clear();
```

```
  display.setBrightness(5); // set the brightness to 7 (0:dimmest, 7:brightest)
  pinMode (CLK,OUTPUT);
  pinMode (DIO,OUTPUT);
  pinMode(autotitel, OUTPUT); // mag enkel hoog of aan indien begintijd bekend is en niet
overschreden
  pinMode(autostart, OUTPUT); // puls hoog bij aanvang dienst zodat standaardsettings mic,
muziek, volgmij ...
  digitalWrite(autotitel, LOW);
  digitalWrite(autostart, LOW);
  pythoninfo  = 2; // om niet straks over te slaan
  gezieninfo  = 2;
  gezieninfo2 = 2;
  begonnen    = 2;
  Serial.begin(9600); //Turn on Serial Port
  Ethernet.begin(mac, ip); //Initialize Ethernet
  Udp2.begin(localPort); //Initialize Udp
  delay(1500);
  lcd.init();      // initialize the lcd
  lcd.backlight();
  if (!rtc.begin())
  {
    Serial.println("Couldn't find RTC. Program stopped.");
    while (1);
  }
}

void loop()
{
    packetSize = Udp2.parsePacket(); //Read the packetSize
    if(packetSize > 0)
    { //Check to see if a request is present
      Udp2.read(packetBuffer2, UDP_TX_PACKET_MAX_SIZE); //Reading the data request
on the Udp
      String begintijdin(packetBuffer2); //Convert packetBuffer2 array to string begintijd
      Udp2.beginPacket(Udp2.remoteIP(), Udp2.remotePort());  //Initialize Packet send
      Udp2.print("Klaar"); // gegevens zijn binnen, kan tijd zijn of iets anders
      Udp2.endPacket(); //Packet has been sent
      /// hier gaan wij klok resetten
      int scheiding = begintijdin.indexOf('#');
      Serial.println("plaats scheidingsteken ");
      Serial.print(String(scheiding));
      Serial.println(begintijdin);
      String begintijd = begintijdin.substring(0,scheiding);
      String resettijd = begintijdin.substring(scheiding+1,begintijdin.length()+1);
      Serial.print("resettijd: ");
      Serial.println(resettijd);
      int reset_len = resettijd.length() + 1;
      char char_array[reset_len];
```

```
    resettijd.toCharArray(char_array, reset_len); // hieronder moeilijk doen om tijd van python in te lezen
    char *strings[6]; // an array of pointers to the pieces of the above array after strtok()
    char *ptr = NULL;
    int rjaar;
    int rmaand;
    int rdag;
    int ruur;
    int rminuut;
    int rseconde;
    byte index = 0;
     ptr = strtok(char_array, ",");  // scheidingsteken
     while (ptr != NULL)
     {
       strings[index] = ptr;
       index++;
       ptr = strtok(NULL, ",");
     }
     Serial.println("De delen na strtok()");
     for (int n = 0; n < index; n++)
     {
       Serial.print(n);
       Serial.print("  ");
       Serial.println(strings[n]);
     }
    rjaar = atoi(strings[0]);
    rmaand = atoi(strings[1]);
    rdag = atoi(strings[2]);
    ruur = atoi(strings[3]);
    rminuut = atoi(strings[4]);
    rseconde = atoi(strings[5]);
    rtc.adjust(DateTime(rjaar, rmaand, rdag, ruur, rminuut, rseconde));
    Serial.println("begintijd test ");
    Serial.println(begintijd);
    Serial.println("resettijd test ");
    Serial.println(resettijd);
    if (begintijd.indexOf(':') != -1)
    {
       istijd = 1;
       lengte = begintijd.length();
       if(lengte == 5) begintijd += ":00";
    }
    else
    {
      istijd = 0;
    }
    begintijddef  = begintijd; // let op door de += is begintijd hierna de kluts kwijt vandaar begintijddef
    memset(packetBuffer2, 0, UDP_TX_PACKET_MAX_SIZE);
```

```
}
DateTime now = rtc.now(); // Fetch the current time from the RTC
clearLCDLine(3);
lcd.setCursor(0,3);
lcd.print(daysOfTheWeek[now.dayOfTheWeek()]);
lcd.print(" ");
if(now.day()<10) lcd.print('0');
lcd.print(now.day(), DEC);
lcd.print("/");
if(now.month()<10) lcd.print('0');
lcd.print(now.month(), DEC);
lcd.print("/");
jaarnu = now.year();
jaar  = jaarnu.toInt();
lcd.print(jaar - 2000);
lcd.print(" ");
if(now.hour()<10) lcd.print('0');
lcd.print(now.hour(), DEC);
lcd.print(':');
if(now.minute()<10) lcd.print('0');
lcd.print(now.minute(), DEC);
lcd.print(':');
if(now.second()<10) lcd.print('0');
lcd.print(now.second(), DEC);
uurnu       = now.hour();
uur         = uurnu.toInt();
minuutnu    = now.minute();
minuut      = minuutnu.toInt();
secondenu   = now.second();
seconde     = secondenu.toInt();
nuabsoltijd = uur * 3600UL + minuut * 60 + seconde;
if(begintijddef != "")
{
 if(istijd == 1)
 {
   if(pythoninfo != 1) // tegen flikkeren
   {
     uurnu       = begintijddef.substring(0,2);
     uur         = uurnu.toInt();
     minuutnu    = begintijddef.substring(3,5);
     minuut      = minuutnu.toInt();
     secondenu   = begintijddef.substring(6,8);
     seconde     = secondenu.toInt();
     misabsoltijd = uur * 3600UL + minuut * 60 + seconde;
     Serial.println();
     Serial.print("misabsoltijd: ");
     Serial.print(misabsoltijd);
     Serial.println();
     Serial.print("Begintijd = ");
```

```
          Serial.print(begintijddef);
          Serial.println();
          toontijdnu = uurnu+minuutnu;
          toontijd  = toontijdnu.toInt();
          clearLCDLine(0);
          lcd.setCursor(0,0);
          lcd.print("Dienst begint ");
          lcd.print(uurnu+":"+minuutnu+"u");
          lcd.println();
          Serial.print(toontijd);
          Serial.println();
          display.showNumberDecEx(toontijd, 0b11100000, false, 4, 0); // lcd 7-segment
        }
        if(((misabsoltijd-40) < nuabsoltijd) && (misabsoltijd != 0)) // 40 seconden voor begin
gaat het om
        {
          digitalWrite(autotitel, LOW);
          if(begonnen != 1)
          {
            digitalWrite(autostart, HIGH);
            clearLCDLine(1);
            lcd.setCursor(2,1);
            lcd.print("viering begonnen");
            clearLCDLine(2);
            lcd.setCursor(2,2);
            lcd.print("titel verborgen");
            begonnen = 1;
            delay(200); // autostart is puls
            digitalWrite(autostart, LOW);
          }
        }
        else
        {
          if(gezieninfo != 1) // tegen flikkeren
          {
            digitalWrite(autotitel, HIGH);
            digitalWrite(autostart, LOW);
            clearLCDLine(1);
            lcd.setCursor(1,1);
            lcd.print("toon liturg. titel");
            clearLCDLine(2);
            lcd.setCursor(3,2);
            lcd.print("voor de dienst");
            gezieninfo = 1;
          }
        }
        pythoninfo = 1;
    }
    else
```

```
    {
      display.clear();
      digitalWrite(autotitel, LOW);
      digitalWrite(autostart, LOW);
      if(gezieninfo2 != 1) // tegen flikkeren
      {
       clearLCDLine(0);
       lcd.setCursor(1,0);
       lcd.print("Ontbrekende tijd:");
       clearLCDLine(1);
       lcd.setCursor(0,1);
       lcd.print("geen volgmij titel!");
       clearLCDLine(2);
       lcd.setCursor(0,2);
       lcd.print("enkel handmat. start");
       gezieninfo2 = 1;
      }
    }
  }
  else
  {
    misabsoltijd = 0;
    if(pythoninfo != 0)  // tegen flikkeren
    {
     clearLCDLine(0);
     lcd.setCursor(0,0);
     lcd.print("Begintijd ontbreekt!");
     clearLCDLine(1);
     lcd.setCursor(1,1);
     lcd.print("run liturgietitel");
    }
    pythoninfo = 0;
    gezieninfo   = 0;
    gezieninfo2  = 0;
  }
  delay(1000);
}

 void clearLCDLine(int line)
 {
   for(int n = 0; n < 20; n++)
   { // 20 indicates symbols in line. For 2x16 LCD write - 16
     lcd.setCursor(n,line);
     lcd.print(" ");
   }
}
```