```
// A4 Clogging Monitor for a 3D Printer with 2 Extruders
// PAN, May 16th, 2023, COM 5, Normal Bootloader
// Here Settings for short test!

//1. Introduction

// A common issue is the clogging of the extruder nozzle in 3D-printing.
// This CM device detects clogging alerts by optical and acoustic alarm and pause the printing.
// If the alarm gets confirmed, the 3D- printing can be continued after Nozzle cleaning
// If the alarm gets NOT confirmed, the Power of the 3D-printer is switched off after 1/2 h
// The velleman display outputs are shown at the end of this PDF document

// 2. Settings of the parameters

// Standard operation:
// DelayTime  = 1000;                  // 1 sec for Display change
// alertTime = 20000,                  // Clogging detection after 20 sec
// BuzzerOffDelayTime = 60000;         // Buzzer Alarm during 1 Minute
// PowerOffDelayTime = 180000;         //Power Off after 30 Minutes
// int Factor = 60;                    //Timing Minutes

// For Fast test:
// DelayTime  = 1000;                  // 1 sec for Display change
// alertTime = 2000                    // Clogging detection after 2 sec
// BuzzerOffDelayTime = 30000;         // Buzzer Alarm during 1/2 Minute
// PowerOffDelayTime =  60000;         //Power Off after 1 Minute
// int Factor = 1;                     //Timing Seconds

// For Photo with 10 sec retarded shots:
// DelayTime  = 20000;                 // 20 sec for Display change
// alertTime = 10000                   // Clogging detection 10 sec
// BuzzerOffDelayTime = 30000;         // Buzzer Alarm during 1/2 Minute
// PowerOffDelayTime =  60000;         // Power Off after 1 Minute
// int Factor = 1;                     //Timing Seconds

// Time recording after Start Command, shows both Sensors if active
// YELLOW field below Time Values for purge former elapsed or alert values

// Shows after Start a green increasing green beam, end after 1 Minute
// Beam Start at 50, End at 180, Duration 130, Step 1, Clock 60sec/130 = 460 ms

// Supply 7.5 V at Arduino Vin, with wireless remote control
// If alarm and confirmed: back to first screen
// If 3D-Printer again ready, set "Continue" at 3D-Printer, Start monitoring again
// If alarm and not confirmed: Power off after PowerOFFDelayTime

// Note comments at the end of this program

int Factor = 1;             // 60 for Alarm in Minutes, or 1 for fast test with Seconds
int Pos = 120;               // Position first letter Elapsed Time and Alarm
int DelayTime = 1000;    // 1000 or 20000  for Foto
```

```
unsigned long alertTime = 2000;                  // set to 20000, for test set to 2000
unsigned long BuzzerOffDelayTime = 30000;  // set to 60000, for test set to 30000
unsigned long PowerOffDelayTime = 60000; // set to 180000, for test set to 60000
unsigned long currentTime;                        // for Start time
unsigned long signal1Arrival;                     // by clogging Sensor1, Opto Barrier 1 to 4
V
unsigned long signal2Arrival;                     // by clogging Sensor2, Opto Barrier 1 to 4
V
unsigned long alarmStart = 0;                     // for alarmStart related to currentTime;
unsigned long previousMillis = 0;

const int sensorPin1 ; // No Pin Nr, Signal from analogue Signal1 processing A6
const int sensorPin2 ; // No Pin Nr, Signal from analogue Signal 2 processing A7
const int sensorAnalogPin1 = A6; // Signal 1 from Opto Barrier, 1 to 4 V
const int sensorAnalogPin2 = A7; // Signal 2 from Opto Barrier, 1 to 4 V
const int sensorLed1 = 10;       // By sensor1Value HIGH
const int sensorLed2 = 11;       // By sensor2Value HIGH
const int relayPausePin = 12;    // If Alert, activate Pause
const int relayPowerOffPin = 13; // Remote 7.5V Supply OFF by wireless AC 230V switch
const int buzzerPin = A5;        // Buzzer pulsating, starting 20 sec after Clogging detection
const long interval = 460;       // Progress for green display during „printer ok“ @ 60 cec

int Reset = 4; // Reset by Power-ON
int alarm = 0;
int alarmValue = 0;
int sensor1Value = 0; // also needed for SensorLed1, On every ca. 4 mm
int sensor2Value = 0; // also needed for SensorLed1, On every ca. 4 mm
int sensor1AnalogValue = 0;
int sensor2AnalogValue = 0;
int extruder1Active = 0;
int extruder2Active = 0;
int extruder1Alarm = 0;
int extruder2Alarm = 0;
int sensor1PrevValue = 0;
int sensor2PrevValue = 0;
int relayPausePinValue = 0;
int buzzerValue = 0;
int relayPowerOFFPinValue = 0; // if set to HIGH in the void loop: remote control battery is
used very often!
int myState = 0;                    // for Velleman State machine
int step = 50;                      // for beginning green field if printer ok
int prestep = 50 ;                  // for beginning green field if printer ok
int StartMonitoring = 0;
int Time = 0;
int AlertMinutes = 0;
int ElapsedTime = 0;

// Here Velleman Definitions
#include <Adafruit_TFTLCD.h>
#include <Adafruit_GFX.h>
#include <TouchScreen.h>
```

```
#define LCD_CS A3
#define LCD_CD A2
#define LCD_WR A1
#define LCD_RD A0
#define LCD_RESET A4

#define TS_MINX 122
#define TS_MINY 111
#define TS_MAXX 942
#define TS_MAXY 890

#define YP A3
#define XM A2
#define YM 9
#define XP 8

#define BLACK      0x0000
#define BLUE       0x001F
#define RED        0xF800
#define GREEN      0x07E0
#define CYAN       0x07FF
#define MAGNETA 0xF81F
#define YELLOW   0xFFE0
#define WHITE      0xFFFF

// D0 to 07 to Villeman Display,pay attention order D0 and D1
Adafruit_TFTLCD tft (LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);
TouchScreen ts = TouchScreen (XP, YP, XM, YM, 364);
boolean buttonEnabled = true;  // Set the TouchScreen for Input
void (*resetFunc) (void) = 0; // for Auto reset after Touch Alarm Stop, needed

void setup() {
  pinMode(sensorLed1, OUTPUT); // Flash LED1 on Sensor1
  pinMode(sensorLed2, OUTPUT); // Flash LED2 on Sensor2
  pinMode(relayPausePin, OUTPUT);
  pinMode(relayPowerOffPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);

  Serial.begin(57600);

  // Velleman program in void setup for the first display

  tft.reset();
  uint16_t identifier = tft.readID();
  tft.begin(identifier);
  tft.setRotation(1); // Display: with 1 horizontal and 0 vertical
  tft.fillScreen (WHITE);
  tft.drawRect(0, 0, 319, 240, YELLOW);

  tft.setCursor(30, 30);
  tft.setTextColor(BLACK);
```

```
  tft.setTextSize(3);
  tft.print("Welcome Monitor");

  tft.setCursor(80, 80);
  tft.setTextColor(BLACK);
  tft.setTextSize(2);
  tft.print("May 16th, 2023 ");

  tft.setCursor(55, 110);
  tft.setTextColor(BLACK);
  tft.setTextSize(2);
  tft.print("www.ufo-doctor.ch ");

  tft.fillRect(25, 180, 270, 40, GREEN);
  tft.drawRect(25, 180, 270, 40, BLACK);
  tft.setCursor(43, 190);
  tft.setTextColor(BLACK);
  tft.print("Push here for Start!");              // Middle Position on Touchscreen is ok

}  // One Clamp for end of this setup

void loop() {
  digitalWrite(Reset, HIGH); // Power On Reset at start
  delay (200);
  pinMode(Reset, OUTPUT);
  delay (200);
  digitalWrite(Reset, LOW); // Reset stopped


  // Analog Sensor Signals Processing sensor
  sensor1Value = digitalRead(sensorPin1);
  sensor2Value = digitalRead(sensorPin2);

  sensor1AnalogValue = analogRead(sensorAnalogPin1);    // read sensor 1 at A6
  sensor2AnalogValue = analogRead(sensorAnalogPin2);    // read sensor 2 at A7

  if ((sensor1AnalogValue) < (int)( 1024 / 2 )) {       // Threshold 2.5V
    sensor1Value = HIGH;
  } else {
    sensor1Value = LOW;
  }

  if ((sensor2AnalogValue) < (int)( 1024 / 2 )) {       // Threshold 2.5V
    sensor2Value = HIGH;
  } else {
    sensor2Value = LOW;
  }

  // Here program sensorLed1 and/or sensorLed2 flashing at filament transport detected:

  if (sensor1Value == HIGH) { // If Encoder shows a HIGH
    digitalWrite(sensorLed1, HIGH); //Activation sensorLed1
```

```
    pinMode(sensorLed1, OUTPUT);
    delay(100); // short flash with 10 Ohm to D10, not impressing, but with 0 Ohm better, but
only
    // permitted supplied by Arduino Output D10, with direct 5V to much current!
  } else {
    pinMode(sensorLed1, INPUT);
  }// High impedance, prevents a LOW Output

  if (sensor2Value == HIGH) { // If Encoder shows a HIGH
    digitalWrite(sensorLed2, HIGH); //Activation sensorLed2
    pinMode(sensorLed2, OUTPUT);
    delay(100); // short flash with 10 Ohm to D11, (see note above for sensorLed1)
  } else {
    pinMode(sensorLed2, INPUT);
  }// High impedance, prevents a LOW Output

  if (sensor1Value != sensor1PrevValue && sensor1Value == LOW) {
    signal1Arrival = currentTime;
    extruder1Active = 1;
    extruder1Alarm = 0;
  }

  if (sensor2Value != sensor2PrevValue && sensor2Value == LOW) {
    signal2Arrival = currentTime;
    extruder2Active = 1;
    extruder2Alarm = 0;
  }

  sensor1PrevValue = sensor1Value;
  sensor2PrevValue = sensor2Value;

  if (!alarm) {
    // For Dual Extruder printing: First lower layer with Ext1, continued with Ext2 for
additional layers,

    if (!extruder2Active) {    // check if Ext2 is already active
      if (extruder1Active && currentTime - signal1Arrival > alertTime) {
        alarm = 1;
        alarmStart = currentTime;
        extruder1Alarm = 1;
        extruder1Active = 0;
      }
    }                                  // two clamps

    if (extruder2Active && currentTime - signal2Arrival > alertTime) {
      alarm = 1;
      alarmStart = currentTime;
      extruder2Alarm = 1;
      extruder2Active = 0;
    }
    if (alarm) {                       // if we set an alarm, then pause the 3D printing
      digitalWrite(relayPausePin, HIGH);
```

```
    relayPausePinValue = HIGH;
    delay(1000); digitalWrite(relayPausePin, LOW);
    buzzerValue = 1;
   }
  }                                  // Two clamps

  if (extruder1Alarm == 0 && extruder2Alarm == 0) {
    alarm = 0; // check if we managed to resume printing
  }                                  // One Clamps

  // Alarm Conditions
  currentTime = millis();               // for alert time, buzzer stop and powerOFF
  digitalWrite(relayPowerOffPin, LOW);

  if (alarm) {
    if (millis() % 2000 > 1000) {
      digitalWrite(buzzerPin, HIGH);      // buzzer pulsing all 2 sec ON for 1 sec
    }
    else digitalWrite(buzzerPin, LOW);
  }

  if (alarm && currentTime - alarmStart > BuzzerOffDelayTime) {   // buzzer Off
    digitalWrite (buzzerPin, LOW);            // Stop Buzzer
  }

  if (alarm && currentTime - alarmStart > PowerOffDelayTime) {
    digitalWrite (relayPowerOffPin, HIGH);    // check if Power is switched OFF
    delay (100);                            // if USB Cable connected: No Power Off, Battery of
remote Control used many times!
  }

  switch (myState) {                      // Variable(mystate), will be replaced by the next case
number
    case 0:                                 // Note  ":" after case!

      unsigned long elapsedTime = millis() - StartMonitoring;                      // Duration
since Start

      TSPoint p = ts.getPoint();               // p-Definition, not valid in later cases!
      if (p.z > ts.pressureThreshhold) {
        p.x = map(p.x, TS_MAXX, TS_MINX, 0, 320);        // 0, 320 for x-axis
        p.y = map(p.y, TS_MAXY, TS_MINY, 0, 480);        // 0, 480 for y-axis
        if (p.x > 10 && p.y > 200 && p.y < 250 && buttonEnabled);
        buttonEnabled = false;                    //Switch Display to false for Output, else no
Display!
        pinMode(XM, OUTPUT);
        pinMode(YP, OUTPUT);

        tft.fillScreen(YELLOW);
        tft.setCursor(30, 30);
        tft.setTextColor(BLACK);
        tft.setTextSize(3);
```

```
      tft.print("Monitor enabled");      // Text for Monitor ready

      tft.setCursor(40, 90);
      tft.setTextColor(BLUE);
      tft.setTextSize(2);
      tft.print("Elapsed Time minutes");

      tft.setCursor(20, 60);
      tft.setTextColor(RED);
      tft.setTextSize(2);
      tft.print(" Prepare Pause Printing");     // Prepare 3D-printer for "Pause"

      myState = 1;         // switch to next State 1
      break;               // break needed
    } else(extruder1Alarm = 0); (extruder2Alarm = 0); // Alarm deactivation

  }                        // ONE Clamps needed for termination of case 0

  switch (myState) {                          // ready for next case
    case 1:                                    // only ok with before the command “switch
(myState)”

     if ((extruder1Alarm) == HIGH) {

       for (int i = 0; i < 10; i++) {                // 5 times red /white flashing before  buzzer
alarm
         tft.fillScreen(RED);
         delay (200);
         tft.fillScreen(WHITE);
       }
       tft.fillRect(30, 25, 260, 90, RED);      // Top Red, Text “Alarm”
       tft.setCursor(90, 50);
       tft.setTextColor(BLACK);
       tft.setTextSize(5);
       tft.print("ALARM");

       tft.fillRect(40, 180, 240, 40, RED);
       tft.drawRect(40, 180, 240, 40, BLACK);
       tft.setCursor(45, 190);
       tft.setTextColor(BLACK);
       tft.setTextSize(2);
       tft.print("   Extr.1 clogged!");
       delay(DelayTime);                          // 1000 or 20000 for Foto

     } else {
       tft.drawRect(50, 180, 230, 40, BLACK);
       tft.setCursor(45, 190);
       tft.setTextColor(BLACK);
       tft.setTextSize(2);
       tft.print("    3D Printer ok    ");
       delay(1);                                  // 0.001 sec for fast repeating
     }
```

```
if ((extruder2Alarm) == HIGH) {

  for (int i = 0; i < 4; i++) {                  //5 times red flash during buzzer
    tft.fillScreen(RED);
    delay (500);
    tft.fillScreen(WHITE);
  }
  tft.fillRect(30, 25, 260, 90, RED);      // Top Red, Text “Alarm”
  tft.setCursor(90, 50);
  tft.setTextColor(BLACK);
  tft.setTextSize(5);
  tft.print("ALARM");

  tft.fillRect(40, 180, 250, 40, RED);
  tft.drawRect(40, 180, 250, 40, BLACK);
  tft.setCursor(45, 190);
  tft.setTextColor(BLACK);
  tft.setTextSize(2);
  tft.print("  Extr.2 clogged!");
  delay(DelayTime);                          // 1000 or 20000 for Foto
} else {
  tft.drawRect(50, 180, 230, 40, BLACK);
  tft.setCursor(45, 190);
  tft.setTextColor(BLACK);
  tft.setTextSize(2);
  tft.print("    3D Printer ok ");
  delay(1);                                  // 0.001 sec for fast repeating

  //  Fast increasing green field on display text „3D-Printer ok“ if still ok

  unsigned long currentMillis = millis();
  AlertMinutes = ElapsedTime / Factor;                    // 60 Minutes , 1 for Seconds

  // Adjust Position of AlertMinutes , not perfect, not applied here
  /*
      if (AlertMinutes > 9)  Pos = 60;
      if (AlertMinutes > 99) Pos = 40;
      if (AlertMinutes > 999) Pos = 20;
      else Pos = 150;
  */
  tft.fillRect(80, 110, 180, 60, YELLOW);             // purge former value
  tft.setCursor(Pos, 120);
  tft.setTextColor(BLUE);
  tft.setTextSize(5);
  tft.print(AlertMinutes);

  if (currentMillis - previousMillis >= interval) {         // store the momentary time
    previousMillis = currentMillis;
    ElapsedTime = currentMillis / 1000; // sec
    step = prestep + 1; // for full @ 60 sec
    prestep = step;
```

```
      tft.fillRect(step, 180, 20, 40, GREEN);
      // left edge 50, at hit 180
      // lent green field 20, hight 40
     }
    } //two clamp

    if ((extruder1Alarm) == HIGH || (extruder2Alarm) == HIGH) {     // OR-Function with ||
important!

     if ((extruder1Alarm) == HIGH) {
      tft.fillScreen(YELLOW);
      tft.setCursor(40, 20);
      tft.setTextColor(RED);
      tft.setTextSize(4);
      tft.print("Extruder 1!" );

      tft.setCursor(60, 60);
      tft.setTextColor(BLACK);
      tft.setTextSize(2);
      tft.print(" Alarm at Minutes:");

      tft.fillRect(80, 80, 180, 50, YELLOW);
      tft.setCursor(Pos, 90);
      tft.setTextColor(BLUE);
      tft.setTextSize(5);
      tft.print(AlertMinutes);

      tft.setCursor(33, 140);
      tft.setTextColor(BLUE);
      tft.setTextSize(3);
      tft.print("Pause Printing" );
     }

     if ((extruder2Alarm) == HIGH) {
      tft.fillScreen(YELLOW);
      tft.setCursor(40, 20);
      tft.setTextColor(RED);
      tft.setTextSize(4);
      tft.print("Extruder 2!" ) ;
      tft.setCursor(60, 60);
      tft.setTextColor(BLACK);
      tft.setTextSize(2);
      tft.print(" Alarm at Minutes:");

      tft.fillRect(80, 80, 180, 50, YELLOW);
      tft.setCursor(Pos, 90);
      tft.setTextColor(BLUE);
      tft.setTextSize(5);
      tft.print(AlertMinutes);

      tft.setCursor(33, 135);
      tft.setTextColor(BLUE);
```

```
        tft.setTextSize(3);
        tft.print("Pause Printing" );      // Big letters
       }
       tft.setCursor(30, 185);
       tft.setTextColor(BLACK);
       tft.setTextSize(3);                  // Big letters
       tft.print("Stop Alarm here");        // for Stop Alarm now!

       delay (DelayTime);                   //1000 or 20000 for Foto

       // Switch to Last case 2
       myState = 2;                         // for the last Case 2
     case 2:                              // call Case 2
       // Touch for Stop Alarms, takes some time
       TSPoint p = ts.getPoint();        // again defined in this new case state 2
       if (p.z > ts.pressureThreshhold) {
         p.x = map(p.x, TS_MAXX, TS_MINX, 0, 320);    // 320 for x top
         p.y = map(p.y, TS_MAXY, TS_MINY, 0, 480);     // 480 for y below
         if (p.x > 10 && p.y > 200 && p.y < 250 && buttonEnabled);
         buttonEnabled = false;             // Switch Display to false for Output, else no
Display!

         Serial.print(" Alarm Stop");       // only for test on Serial Monitor

         pinMode(XM, OUTPUT);
         pinMode(YP, OUTPUT);
         digitalWrite(buzzerPin, LOW);
         tft.fillScreen (GREEN);
         tft.drawRect(0, 0, 319, 240, GREEN);
         tft.setCursor(50, 50);
         tft.setTextSize(3);
         tft.setTextColor(BLACK);
         tft.print("Alarm stopped");

         tft.setCursor(42, 120);
         tft.setTextSize(2);
         tft.setTextColor(BLUE);
         tft.print(" Clean extruder now!");

         tft.setCursor(45, 180);
         tft.setTextSize(2);
         tft.setTextColor(BLACK);
         tft.print(" Ready to continue? ");

         delay(DelayTime);          // 1000 or 20000 for Foto
         resetFunc();               // call reset now
         //break;                   // NO break! else old time keeping
       }
     }
   }
} // four clamps
```

/* Comments:

The older displays on the velleman display remains present for all the time
Erasing is not possible, but filling Velleman display with the background color is ok before new data display.

Best Operation with an external 7.5 Supply, consumption 415 mA at Arduino Nano
The Velleman Display needs a stable 5V supply by the Arduino 5V regulator.

For debugging the system is supplied by an USB Cable: 4.94 V, 300 mA.
With USB connection the power OFF function is disabled!
The relayPowerOffPin becomes HIGH at the end of the PowerOffDelayTime without stopping the supply.
This means that the internal 3V Battery is many times used for nothing! Check it!

The Sensor Leds becomes HIGH during every filament transitions by the Pins 10 and 11.
The pinMode(sensorLed1&2, OUTPUT) must be returned to INPUT for high impedance after
this flashing instructions, else the Opto Barrier cannot be active for Analog Signal reading!

Current into sensor LEDs and opto-barrier WITHOUT Input Pin 11 or 12 active:
6.13 mA Sensor LED WEAK, Analog Signal 968/111, ok
Current into the serial sensor LED and opto-barrier WITH Pin 11 or 12 active
37 mA Sensor Led BRIGHT, Analog Signal 900/80, ok
*/