**Supplementary Materials: Conceptualization and construction of a low-cost and self-made device for monitoring of Particulate Matter: a step-by-step guide**

Giacomo Fanti1,\*, Francesca Borghi1,3, Emanuele Cauda4, Cody Wolfe4, Justin Patts4, Carlo Dossi2, Andrea Cattaneo1, Andrea Spinazzè1, Domenico Maria Cavallo1

|  |
| --- |
|  |

1. Department of Science and High Technology, University of Insubria, Como, Italy;

[francesca.borghi@uninsubria.it](mailto:francesca.borghi@uninsubria.it) (F.B.); andrea.cattaneo@uninsubria.it (A.C.);

andrea.spinazze@uninsubria.it (A.S.); domenico.cavallo@uninsubria.it (D.M.C.)

1. Department of Theoretical and Applied Sciences, University of Insubria, Varese, Italy;

Carlo.Dossi@uninsubria.it (C.D.)

1. Department of Medical and Surgical Sciences, University of Bologna, Bologna, Italy;

[francesca.borghi12@unibo.it](mailto:francesca.borghi12@unibo.it) (F.B.)

1. Center for Direct Reading and Sensor Technologies, National Institute for Occupational Safety and Health, Centers for Disease Control and Prevention, Pittsburgh, Pennsylvania, PA, USA;

[cuu5@cdc.gov](mailto:cuu5@cdc.gov) (E.C.); [yvv5@cdc.gov](mailto:yvv5@cdc.gov) (C.W.); [jdq7@cdc.gov](mailto:jdq7@cdc.gov) (J.P.)

**\*** Correspondence: [giacomo.fanti@uninsubria.it](mailto:giacomo.fanti@uninsubria.it) (G.F.) Tel.: +39-0312386645

# *Script S1:* Arduino IDE sketch (text file)

*#include <SoftwareSerial.h>*

*SoftwareSerial pmsSerial(2, 3);*

*#include <SPI.h>*

*#include <SD.h>*

*#include <SimpleDHT.h>*

*#define \_DHTPIN 4*

*SimpleDHT22 dht22(\_DHTPIN);*

*#include <Wire.h>*

*#include <RTClib.h>*

*RTC\_DS3231 rtc;*

*int intestaColonne = 0;*

*void setup() {*

*// our debugging output*

*Serial.begin(115200);*

*// sensor baud rate is 9600*

*pmsSerial.begin(9600);*

*if (!SD.begin(10)){*

*Serial.println("errore SD");*

*while(1);*

*}*

*File dataFile = SD.open ("P.ALP.txt", FILE\_WRITE);*

*if (dataFile) {*

*dataFile.print ("Date"); dataFile.print (";"); dataFile.print ("Time"); dataFile.print (";"); dataFile.print("PM1"); dataFile.print(";"); dataFile.print("PM2.5"); dataFile.print(";"); dataFile.print("PM10"); dataFile.print(";");*

*dataFile.print("Npart\_0.3um"); dataFile.print(";"); dataFile.print("Npart\_0.5um"); dataFile.print(";"); dataFile.print("Npart\_1um"); dataFile.print(";");*

*dataFile.print("Npart\_2.5um"); dataFile.print(";"); dataFile.print("Npart\_5um"); dataFile.print(";"); dataFile.print("Npart\_10um"); dataFile.print(";");*

*dataFile.print("T"); dataFile.print(";"); dataFile.print("RH"); dataFile.println("\t");*

*dataFile.close();*

*}*

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

*Serial.println("verify connections");*

*while(true);*

*}*

*if (rtc.begin()) {*

*rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));*

*//rtc.adjust(DateTime(2021, 11, 22, 12, 00));*

*}*

*}*

*struct pms5003data {*

*uint16\_t framelen;*

*uint16\_t pm10\_standard, pm25\_standard, pm100\_standard;*

*uint16\_t pm10\_env, pm25\_env, pm100\_env;*

*uint16\_t particles\_03um, particles\_05um, particles\_10um, particles\_25um, particles\_50um, particles\_100um;*

*uint16\_t unused;*

*uint16\_t checksum;*

*};*

*struct pms5003data data;*

*void loop() {*

*float temp = 0;*

*float hum = 0;*

*dht22.read2(&temp, &hum, NULL);*

*if (!intestaColonne) {*

*Serial.print("\t"); Serial.print("Data e Ora"); Serial.print("\t"); Serial.print("\t"); Serial.print("PM1"); Serial.print("\t"); Serial.print("PM2.5"); Serial.print("\t"); Serial.print("PM10"); Serial.print("\t");*

*Serial.print("Npart\_0.3um"); Serial.print("\t"); Serial.print("Npart\_0.5um"); Serial.print("\t"); Serial.print("Npart\_1um"); Serial.print("\t");*

*Serial.print("Npart\_2.5um"); Serial.print("\t"); Serial.print("Npart\_5um"); Serial.print("\t"); Serial.print("Npart\_10um"); Serial.print("\t");*

*Serial.print("Temperature"); Serial.print("\t"); Serial.print("Humidity"); Serial.println("\t");*

*intestaColonne = 1;*

*}*

*DateTime now = rtc.now();*

*if (readPMSdata(&pmsSerial)) {*

*// reading data was successful!*

*Serial.print("\t"); Serial.print (now.year(), DEC); Serial.print ("/"); Serial.print (pad(now.month())); Serial.print ("/"); Serial.print (pad(now.day())); Serial.print (" "); Serial.print (pad(now.hour())); Serial.print (":"); Serial.print (pad(now.minute())); Serial.print (":"); Serial.print (pad(now.second())); Serial.print("\t"); Serial.print(data.pm10\_standard); Serial.print("\t"); Serial.print(data.pm25\_standard); Serial.print("\t"); Serial.print(data.pm100\_standard); Serial.print("\t");*

*Serial.print(data.particles\_03um); Serial.print("\t"); Serial.print("\t"); Serial.print(data.particles\_05um); Serial.print("\t"); Serial.print("\t"); Serial.print(data.particles\_10um); Serial.print("\t"); Serial.print("\t");*

*Serial.print(data.particles\_25um); Serial.print("\t"); Serial.print("\t"); Serial.print(data.particles\_50um); Serial.print("\t"); Serial.print("\t"); Serial.print(data.particles\_100um); Serial.print("\t"); Serial.print("\t");*

*Serial.print(temp); Serial.print("\t"); Serial.print("\t"); Serial.print(hum); Serial.println("\t");*

*}*

*File dataFile = SD.open ("P.ALP.txt", FILE\_WRITE);*

*if (dataFile) {*

*dataFile.print (pad(now.month())); dataFile.print ("/"); dataFile.print (pad(now.day())); dataFile.print ("/"); dataFile.print (now.year(), DEC); dataFile.print (";"); dataFile.print (pad(now.hour())); dataFile.print (":"); dataFile.print (pad(now.minute())); dataFile.print (":"); dataFile.print (pad(now.second())); dataFile.print (";"); dataFile.print(data.pm10\_standard); dataFile.print(";"); dataFile.print(data.pm25\_standard); dataFile.print(";"); dataFile.print(data.pm100\_standard); dataFile.print(";");*

*dataFile.print(data.particles\_03um); dataFile.print(";"); dataFile.print(data.particles\_05um); dataFile.print(";"); dataFile.print(data.particles\_10um); dataFile.print(";");*

*dataFile.print(data.particles\_25um); dataFile.print(";"); dataFile.print(data.particles\_50um); dataFile.print(";"); dataFile.print(data.particles\_100um); dataFile.print(";");*

*dataFile.print(temp); dataFile.print(";"); dataFile.print(hum); dataFile.println("\t");*

*dataFile.close();*

*}*

*else {*

*Serial.println("error opening P.ALP.txt");*

*}*

*}*

*char \*res = malloc (5);*

*String pad(int n) {*

*sprintf(res, "%02d", n);*

*return String(res);*

*}*

*boolean readPMSdata(Stream \*s) {*

*if (! s->available()) {*

*return false;*

*}*

*// Read a byte at a time until we get to the special '0x42' start-byte*

*if (s->peek() != 0x42) {*

*s->read();*

*return false;*

*}*

*if (s->available() < 32) {*

*return false;*

*}*

*uint8\_t buffer[32];*

*uint16\_t sum = 0;*

*s->readBytes(buffer, 32);*

*for (uint8\_t i=0; i<30; i++) {*

*sum += buffer[i];*

*}*

*/\* debugging*

*for (uint8\_t i=2; i<32; i++) {*

*Serial.print("0x"); Serial.print(buffer[i], HEX); Serial.print(", ");*

*}*

*Serial.println();*

*\*/*

*// The data comes in endian'd, this solves it so it works on all platforms*

*uint16\_t buffer\_u16[15];*

*for (uint8\_t i=0; i<15; i++) {*

*buffer\_u16[i] = buffer[2 + i\*2 + 1];*

*buffer\_u16[i] += (buffer[2 + i\*2] << 8);*

*}*

*// put it into a nice struct :)*

*memcpy((void \*)&data, (void \*)buffer\_u16, 30);*

*if (sum != data.checksum) {*

*Serial.println("Checksum failure");*

*return false;*

*}*

*// success!*

*return true;*

*}*

# *Script S2:* R-code (text file)

*### R script to handle .txt files from the P.ALP and convert the data into a usable .csv file ###*

*### Last version 6/23/2022 ###*

*### File From Folders*

*### setwd(dir = "C:/example")*

*### REMOVE THE 3 “###” HEADING THE PREVIOUS 2 LINES IF YOU WANT TO USE THE FOLDER AS DIRECTORY”*

*### File From SD card*

*### setwd(dir = "E:")*

*### REMOVE THE 3 “###” HEADING THE PREVIOUS 2 LINES IF YOU WANT TO USE THE SD-CARD AS DIRECTORY”*

*CW <- read.table("P.ALP.TXT", header = TRUE, sep = ";", dec = ".")*

*naming\_convention <- "P.ALP Session\_"*

*### In the following two lines you need to insert the correct date (MM/DD/YYYY) and time (HH:MM:SS)*

*start\_date <- "01/25/2023"*

*start\_time <- "15:37:45"*

*library("anytime")*

*library("lubridate")*

*#conversion from original date to current date and time*

*##CW$Date <- c(start\_date)*

*start\_dateTime <- as.data.frame(paste(start\_date, start\_time))*

*colnames(start\_dateTime)[1] <- "start\_dateTime"*

*start\_dateTime <- start\_dateTime$start\_dateTime[ 1]*

*badDate <- CW$Date[ 1]*

*badTime <- CW$Time[ 1]*

*times <- as.data.frame(paste(CW$Date, CW$Time))*

*colnames(times)[1] <- "updatedTime"*

*badDateTime <- times$updatedTime[ 1]*

*#calculate difference between times and add to the times variable and bind to CW*

*good <- anytime(start\_dateTime)*

*bad <- anytime(badDateTime)*

*difference <- paste(difftime(bad, good, units="secs"))*

*times2 <- anytime(times$updatedTime)*

*updatedTime <- times2 - as.numeric(difference)*

*times3 <- cbind(updatedTime, times)*

*#*

*CW <- cbind(times3$updatedTime, CW)*

*colnames(CW)[1] <- "updatedTime"*

*colnames(CW)[13] <- "Temp"*

*start\_Time <- CW$updatedTime[1]*

*end\_Time <- tail(CW$updatedTime, n=1)*

*unique\_seconds <- as.data.frame(unique(CW$Time))*

*colnames(unique\_seconds)[1] <- "unique\_seconds"*

*unique\_minutes <- as.data.frame(unique(substr(CW$Time,1,5)))*

*colnames(unique\_minutes)[1] <- "unique\_minutes"*

*unique\_hours <- as.data.frame(unique(substr(CW$Time,1,2)))*

*colnames(unique\_hours)[1] <- "unique\_hours"*

*total\_sampling\_minutes <- nrow(unique\_seconds)/60*

*# 1 second resolution*

*PM1 <- as.data.frame(tapply(CW$PM1, CW$updatedTime, mean))*

*colnames(PM1)[1] <- "PM1"*

*PM2.5 <- as.data.frame(tapply(CW$PM2.5, CW$updatedTime, mean))*

*colnames(PM2.5)[1] <- "PM2.5"*

*PM10 <- as.data.frame(tapply(CW$PM10, CW$updatedTime, mean))*

*colnames(PM10)[1] <- "PM10"*

*Npart\_0.3um <- as.data.frame(tapply(CW$Npart\_0.3um, CW$updatedTime, mean))*

*colnames(Npart\_0.3um)[1] <- "Npart\_0.3um"*

*Npart\_0.5um <- as.data.frame(tapply(CW$Npart\_0.5um, CW$updatedTime, mean))*

*colnames(Npart\_0.5um)[1] <- "Npart\_0.5um"*

*Npart\_1um <- as.data.frame(tapply(CW$Npart\_1um, CW$updatedTime, mean))*

*colnames(Npart\_1um)[1] <- "Npart\_1um"*

*Npart\_2.5um <- as.data.frame(tapply(CW$Npart\_2.5um, CW$updatedTime, mean))*

*colnames(Npart\_2.5um)[1] <- "Npart\_2.5um"*

*Npart\_5um <- as.data.frame(tapply(CW$Npart\_5um, CW$updatedTime, mean))*

*colnames(Npart\_5um)[1] <- "Npart\_5um"*

*Npart\_10um <- as.data.frame(tapply(CW$Npart\_10um, CW$updatedTime, mean))*

*colnames(Npart\_10um)[1] <- "Npart\_10um"*

*CW[, 13:14][CW[, 13:14] == 0] <- NA*

*Temp <- as.data.frame(tapply(CW$Temp, CW$updatedTime, mean, na.rm = TRUE))*

*colnames(Temp)[1] <- "Temp"*

*RH <- as.data.frame(tapply(CW$RH, CW$updatedTime, mean, na.rm = TRUE))*

*colnames(RH)[1] <- "RH"*

*second\_resolution <- cbind(PM1,PM2.5,PM10,Npart\_0.3um,Npart\_0.5um,Npart\_1um,Npart\_2.5um,Npart\_5um,Npart\_10um,Temp,RH)*

*second\_resolution\_round <- round(second\_resolution, digits = 1)*

*write.csv(second\_resolution\_round, paste0(naming\_convention,"AT\_1sec.csv"))*

*# 1 min resolution*

*times\_min <- as.data.frame(substr(CW$updatedTime,1,16))*

*colnames(times\_min)[1] <- "times\_min"*

*CW <- cbind(times\_min, CW)*

*PM1 <- as.data.frame(tapply(CW$PM1, CW$times\_min, mean))*

*colnames(PM1)[1] <- "PM1"*

*PM2.5 <- as.data.frame(tapply(CW$PM2.5, CW$times\_min, mean))*

*colnames(PM2.5)[1] <- "PM2.5"*

*PM10 <- as.data.frame(tapply(CW$PM10, CW$times\_min, mean))*

*colnames(PM10)[1] <- "PM10"*

*Npart\_0.3um <- as.data.frame(tapply(CW$Npart\_0.3um, CW$times\_min, mean))*

*colnames(Npart\_0.3um)[1] <- "Npart\_0.3um"*

*Npart\_0.5um <- as.data.frame(tapply(CW$Npart\_0.5um, CW$times\_min, mean))*

*colnames(Npart\_0.5um)[1] <- "Npart\_0.5um"*

*Npart\_1um <- as.data.frame(tapply(CW$Npart\_1um, CW$times\_min, mean))*

*colnames(Npart\_1um)[1] <- "Npart\_1um"*

*Npart\_2.5um <- as.data.frame(tapply(CW$Npart\_2.5um, CW$times\_min, mean))*

*colnames(Npart\_2.5um)[1] <- "Npart\_2.5um"*

*Npart\_5um <- as.data.frame(tapply(CW$Npart\_5um, CW$times\_min, mean))*

*colnames(Npart\_5um)[1] <- "Npart\_5um"*

*Npart\_10um <- as.data.frame(tapply(CW$Npart\_10um, CW$times\_min, mean))*

*colnames(Npart\_10um)[1] <- "Npart\_10um"*

*CW[, 14:15][CW[, 14:15] == 0] <- NA*

*Temp <- as.data.frame(tapply(CW$Temp, CW$times\_min, mean, na.rm = TRUE))*

*colnames(Temp)[1] <- "Temp"*

*RH <- as.data.frame(tapply(CW$RH, CW$times\_min, mean, na.rm = TRUE))*

*colnames(RH)[1] <- "RH"*

*min\_resolution <- cbind(PM1,PM2.5,PM10,Npart\_0.3um,Npart\_0.5um,Npart\_1um,Npart\_2.5um,Npart\_5um,Npart\_10um,Temp,RH)*

*min\_resolution\_round <- round(min\_resolution, digits = 1)*

*write.csv(min\_resolution\_round, paste0(naming\_convention,"AT\_1min.csv"))*

*total\_sampling\_minutes*

*### END ###*

# Additional Figures

Immagine che contiene Componente elettrico, Componente di circuito, elettronica, Componente di circuito passivo

Descrizione generata automaticamente

**Figure S1:** Arduino Uno-R3 microcontroller board

Immagine che contiene elettronica, Ingegneria elettronica, Componente elettrico, Componente di circuito

Descrizione generata automaticamente

**Figure S2:** Elegoo Uno-R3 microcontroller board

Immagine che contiene scatola, blu

Descrizione generata automaticamente

**Figure S3:** Plantower PMS5003 Sensor for Particulate Matter monitoring

Immagine che contiene cavo, Impianto elettrico, elettronica, Alimentazione elettrica

Descrizione generata automaticamente

**Figure S4:** DHT22 sensor

Immagine che contiene testo, Componente di circuito, Componente elettrico, Componente di circuito passivo

Descrizione generata automaticamente

**Figure S5:** DS3231 Real Time Clock module

Immagine che contiene elettronica, Componente elettrico, Componente di circuito, Componente di circuito passivo

Descrizione generata automaticamente

**Figure S6:** Micro-SD card adapter module

Immagine che contiene testo, elettronica, memoria flash

Descrizione generata automaticamente

**Figure S7:** Micro-SD card.

Immagine che contiene testo, schermata, interno

Descrizione generata automaticamente

**Figure S8:** Prototype breadboard

Immagine che contiene elettronica, Ingegneria elettronica, Componente di circuito, Componente elettrico

Descrizione generata automaticamente

**Figure S9:** Solderable board

Immagine che contiene testo, batteria

Descrizione generata automaticamente

**Figure S10:** USB type C (female) to USB type B (male) converter

Immagine che contiene connettore, cavo, aria aperta

Descrizione generata automaticamente

**Figure S11:** USB type C (male) to USB type A (male) connection cable

Immagine che contiene gadget, Attrezzi, smartphone, Dispositivo elettronico

Descrizione generata automaticamente

**Figure S12:** 10400mAh power bank battery

Immagine che contiene cavo, Impianto elettrico, Film termoretraibile

Descrizione generata automaticamente

**Figure S13:** Jumper wires Elegoo kit

Immagine che contiene interno, vassoio, pavimento, contenitore

Descrizione generata automaticamente

**Figure S14:** Airtight container

Immagine che contiene terra, interni

Descrizione generata automaticamente

**Figure S15:** Airtight box with the P.ALP’s environmental sensors