Supplementary information

Low cost referenced luminescent imaging of oxygen and pH with a 2-CCD colour-near infrared camera

Josef Ehgartner, Helmar Wiltsche, Sergey M. Borisov and Torsten Mayr

Supplementary Figures

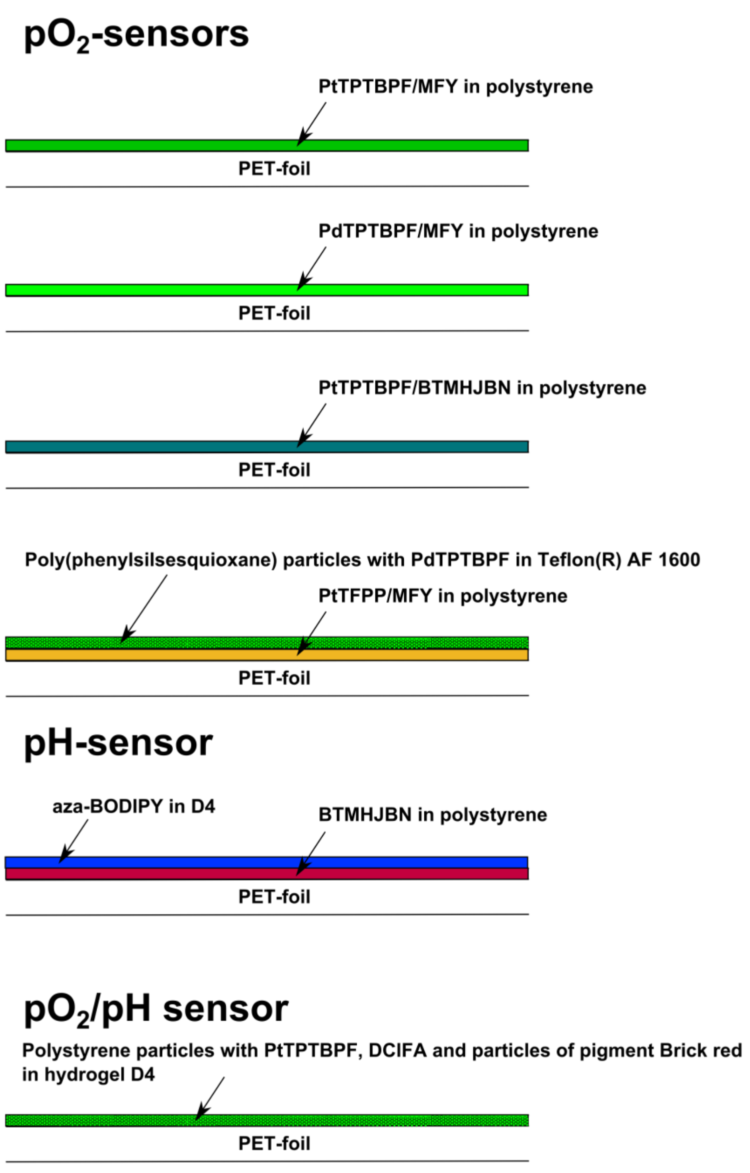
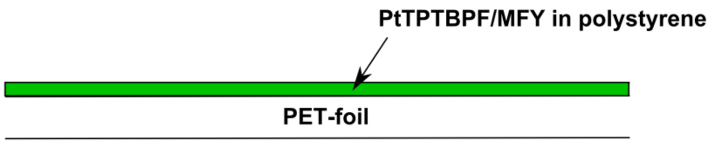
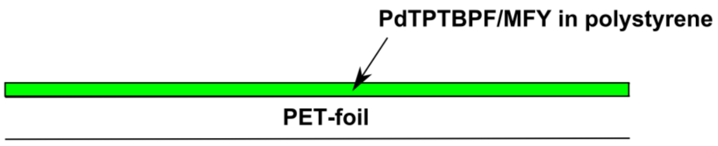
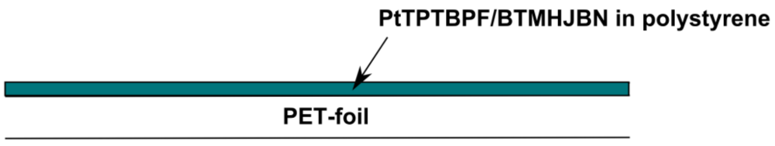

pO_2-sensors
PtTPTBPF/MFY in polystyrene
PET-foil
PdTPTBPF/MFY in polystyrene
PET-foil
PtTPTBPF/BTMHJBN in polystyrene
PET-foil
Poly(phenylsilsesquioxane) particles with PdTPTBPF in Teflon(R) AF 1600
PtTFPP/MFY in polystyrene
PET-foil
pH-sensor
aza-BODIPY in D4
BTMHJBN in polystyrene
PET-foil
pO_2/pH sensor
Polystyrene particles with PtTPTBPF, DCIFA and particles of pigment Brick red in hydrogel D4
PET-foil

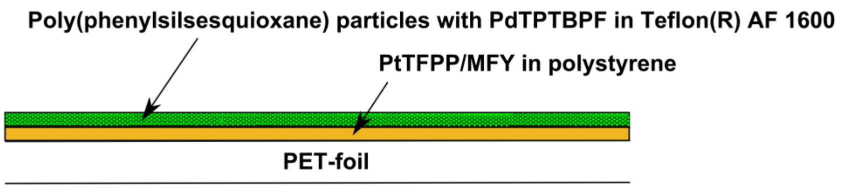
Figure 1 Schematic drawings of the different sensors are shown below.

pO_2-sensors

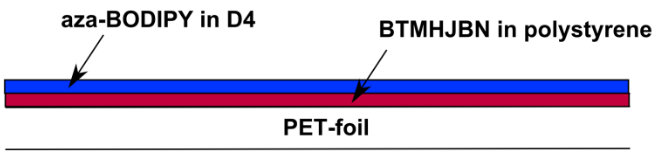




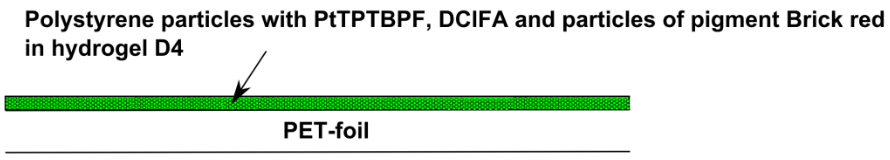



Poly(phenylsilsesquioxane) particles with PdTPTBPF in Teflon(R) AF 1600
PtTFPP/MFY in polystyrene
PET-foil

pH-sensor



pO_2/pH sensor


Polystyrene particles with PtTPTBPF, DCIFA and particles of pigment Brick red in hydrogel D4
PET-foil

The source code of the trigger box is provided below. The source code of the common vision blox (CVB 2011) program will be provided on request by contacting the author.

```
/*
****************************************************************************
***
*****************
Shutter Control
0.13 HgW 290913

****************************************************************************
***
*****************

RS232 Syntax

<Mode (SET, GET, RUN, TST)>_<Command (TME, )>_<DEVICE (C1, C2)>_<Parameter>VbCr
Example:
SET TME C1 02000 Sets the shutter open time of camera 1 to 2000 ms
RUN Starts the triggering process
SET DAC D1 03000 Sets the D1 DAC to 3000
TST Test the communication with the µcontroller


* LCD RS pin to digital pin 12
* LCD Enable pin to digital pin 11
* LCD R/W pin to Ground
* LCD D4 pin to digital pin 4
* LCD D5 pin to digital pin 5
* LCD D6 pin to digital pin 6
* LCD D7 pin to digital pin 7


*/
// include the library code:
#include <LiquidCrystal.h>
#include <Wire.h>


// Constants
// set pin numbers:
const int TTL1Pin = 13; // Camera TTL 1 pin
const int TTL2Pin = 8 ; // Camera TTL 2 pin
const int DAC_Select_A = 2; // DAC channel select pin A
const int DAC_Select_B = 3; // DAC channel select pin B


// variables will change:
int DelayCam1 = 1000;
int DelayCam2 = 1000;

char SerialBuffer[255];
int SBufferLength = 255;
char vbCr = 13;
String SerialBufferSTR = "";
String CommandSTR = "";
```

```

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 4, 5, 6, 7);




//************************************************************************
****
*****************
void setup() {
// set up the LCD's number of columns and rows:
lcd.begin(16, 2);
// Print a message to the LCD.
DoUpdateLCD();


// initialize the TTL pin as an output:
pinMode(TTL1Pin, OUTPUT);
pinMode(TTL2Pin, OUTPUT);

// initialize DAC selection pins as an output:
pinMode(DAC_Select_A, OUTPUT);
pinMode(DAC_Select_B, OUTPUT);


// initialize the RS232:
Serial.begin(9600);


// initialize the I2C communication
Wire.begin(); //Pin-Definition is set by the library to A4 (SDA) and A5 (SCL)
DAC_setVoltage(0, 1, true); //set the DAC to 0 and put this value in the EEPROM
DAC_setVoltage(0, 2, true); //set the DAC to 0 and put this value in the EEPROM
DAC_setVoltage(0, 3, true); //set the DAC to 0 and put this value in the EEPROM
DAC_setVoltage(0, 4, true); //set the DAC to 0 and put this value in the EEPROM
}




//************************************************************************
****
*****************
void loop(){

digitalWrite(TTL1Pin, LOW);
digitalWrite(TTL2Pin, LOW);
digitalWrite(DAC_Select_A, LOW);
digitalWrite(DAC_Select_B, LOW);


// Step 1: Wait until the RS232 sends a new command
while (Serial.available() > 0) {
//Read the command
Serial.readBytesUntil(vbCr, SerialBuffer, SBufferLength);
```

```
if (SBufferLength > 0) {
SerialBufferSTR = SerialBuffer;
if (SerialBufferSTR.length() > 2) {
// Command found - decode it
if (SerialBufferSTR.substring(0, 3) == "SET") {
// SET Command found - decode it
DoDecodeCommand_SET(SerialBufferSTR.substring(4));
}
else if (SerialBufferSTR.substring(0, 3) == "RUN") {
//RUN Command
DoDecodeCommand_RUN();
}
else if (SerialBufferSTR.substring(0, 3) == "TST") {
//TST Command
Serial.print("OK");
}
else {
//Send via RS232 an ERROR signal
Serial.print("ERR");
}
}
}
//Update the LCD
DoUpdateLCD();
}

}



//**************************************************************************
****
******************
void DoUpdateLCD(){
lcd.clear();
lcd.print("Camera 1: ");
lcd.print(DelayCam1);
lcd.setCursor(0, 2);
lcd.print("Camera 2: ");
lcd.print(DelayCam2);
/*Serial.print("*");
Serial.print(DelayCam1);
Serial.print("*");
Serial.print(DelayCam2);
Serial.print("*");*/
}



//**************************************************************************
****
******************
void DoDecodeCommand_SET(String CommandSTR){
//Decodiert den übergebenen SET-String

if (CommandSTR.substring(0, 3) == "TME"){
DoDecodeDeviceCommand_TME(CommandSTR.substring(4));
//Send via RS232 an OK signal
Serial.print("OK");
}
else if (CommandSTR.substring(0, 3) == "DAC"){
```

```
DoDecodeDeviceCommand_DAC(CommandSTR.substring(4));
//Send via RS232 an OK signal
Serial.print("OK");
}
else {
Serial.print("ERR: DoDecodeCommand_SET");
}
}



//************************************************************************
****
******************
void DoDecodeDeviceCommand_TME(String CommandSTR){
//Decodiert den übergebenen TME-String

char Buffer[6];
int BufferLen = 6;
String TempSTR = "";

if (CommandSTR.substring(0, 2) == "C1"){
// Camera 1 is the command target: Write to global variable
TempSTR = CommandSTR.substring(3);
TempSTR.toCharArray(Buffer, BufferLen);
DelayCam1 = atoi(Buffer); //convert STRING to INTEGER
if (DelayCam1 > 2000) DelayCam1 = 2000;
if (DelayCam1 < 50) DelayCam1 = 50;
}
else if (CommandSTR.substring(0, 2) == "C2"){
// Camera 2 is the command target: Write to global variable
TempSTR = CommandSTR.substring(3);
TempSTR.toCharArray(Buffer, BufferLen);
DelayCam2 = atoi(Buffer); //convert STRING to INTEGER
if (DelayCam2 > 2000) DelayCam2 = 2000;
if (DelayCam2 < 50) DelayCam2 = 50;
}

}



//************************************************************************
****
******************
void DoDecodeCommand_RUN(){
// RUN Command is processed
int TempDelay = 0;

if (DelayCam1 == DelayCam2) {
//Both Cameras have identical delay
lcd.clear();
lcd.print("Trigger running ...");
digitalWrite(TTL1Pin, HIGH); // turn the TTL Pin on (HIGH is the
voltage level)
digitalWrite(TTL2Pin, HIGH);
delay(DelayCam1); // wait
digitalWrite(TTL1Pin, LOW); // turn the TTL Pin off by making the
voltage LOW
digitalWrite(TTL2Pin, LOW);
DoUpdateLCD;
```

```
}
else if (DelayCam1 > DelayCam2) {
//Camera 1 has a larger delay than Campera2
TempDelay = DelayCam1 - DelayCam2;
lcd.clear();
lcd.print("Trigger running ...");
digitalWrite(TTL1Pin, HIGH); // turn the TTL Pin on (HIGH is the voltage level)
digitalWrite(TTL2Pin, HIGH);
delay(DelayCam2); // wait
digitalWrite(TTL2Pin, LOW);
delay(TempDelay);
digitalWrite(TTL1Pin, LOW);
}
else if (DelayCam2 > DelayCam1) {
//Camera 1 has a larger delay than Campera2
TempDelay = DelayCam2 - DelayCam1;
lcd.clear();
lcd.print("Trigger running ...");
digitalWrite(TTL1Pin, HIGH); // turn the TTL Pin on (HIGH is the voltage level)
digitalWrite(TTL2Pin, HIGH);
delay(DelayCam1); // wait
digitalWrite(TTL1Pin, LOW);
delay(TempDelay);
digitalWrite(TTL2Pin, LOW);
}

Serial.print("OK");
}



//*****************************************************************************
*****************
void DAC_setVoltage( uint16_t po_voltageDAC, int Channel, bool write_in_EEPROM
){
//Select the DAC
DoSelecDAC(Channel);



// Now communicate with the DAC
uint8_t twbrback = TWBR;
uint8_t _i2caddr;

_i2caddr = 0x60; //Hard-coded - selection is done by the Channel
TWBR = 12; // 400 khz

// First Byte
Wire.beginTransmission(_i2caddr);

// Second byte
if (write_in_EEPROM){
Wire.write(0x40);
}
else {
Wire.write(0x60);
}
```

```

Wire.write(po_voltageDAC / 16); // Upper data bits (D11.D10.D9.D8.D7.D6.D5.D4)
Wire.write((po_voltageDAC % 16) << 4); // Lower data bits (D3.D2.D1.D0.x.x.x.x)
Wire.endTransmission();
TWBR = twbrback;

}




//*************************************************************************
****
*****************
void DoDecodeDeviceCommand_DAC(String CommandSTR){
//Decodiert den übergebenen DAC-String

char Buffer[6];
int BufferLen = 6;
String TempSTR = "";
int DAC_value = 0;

lcd.clear();
lcd.print("Setting DAC...");

//DAC Werte berechnen
TempSTR = CommandSTR.substring(3);
TempSTR.toCharArray(Buffer, BufferLen);
DAC_value = atoi(Buffer); //convert STRING to INTEGER
if (DAC_value > 4095) DAC_value = 4095;
if (DAC_value < 0) DAC_value = 0;


//DAC setzen
if (CommandSTR.substring(0, 2) == "D1"){
// DAC 1 is the command target
DAC_setVoltage(DAC_value, 1, false);
}
else if (CommandSTR.substring(0, 2) == "D2"){
// DAC 2 is the command target
DAC_setVoltage(DAC_value, 2, false);
}
else if (CommandSTR.substring(0, 2) == "D3"){
// DAC 3 is the command target
DAC_setVoltage(DAC_value, 3, false);
}
else if (CommandSTR.substring(0, 2) == "D4"){
// DAC 4 is the command target
DAC_setVoltage(DAC_value, 4, false);
}

delay(300);
}



//*************************************************************************
****
*****************
```

```
void DoSelecDAC(int Channel){
//Wählt über DAC_Select_A und DAC_Select_B den über Channel übergebenen DAC


//Schritt 1: Reset
digitalWrite(DAC_Select_A, LOW);
digitalWrite(DAC_Select_B, LOW);

switch (Channel) {
case 1:
//DAC 1 wählen
digitalWrite(DAC_Select_A, LOW);
digitalWrite(DAC_Select_B, LOW);
break;
case 2:
//DAC 2 wählen
digitalWrite(DAC_Select_A, LOW);
digitalWrite(DAC_Select_B, HIGH);
break;
case 3:
//DAC 3 wählen
digitalWrite(DAC_Select_A, HIGH);
digitalWrite(DAC_Select_B, LOW);
break;
case 4:
//DAC 4 wählen
digitalWrite(DAC_Select_A, HIGH);
digitalWrite(DAC_Select_B, HIGH);
break;
}
}

```