

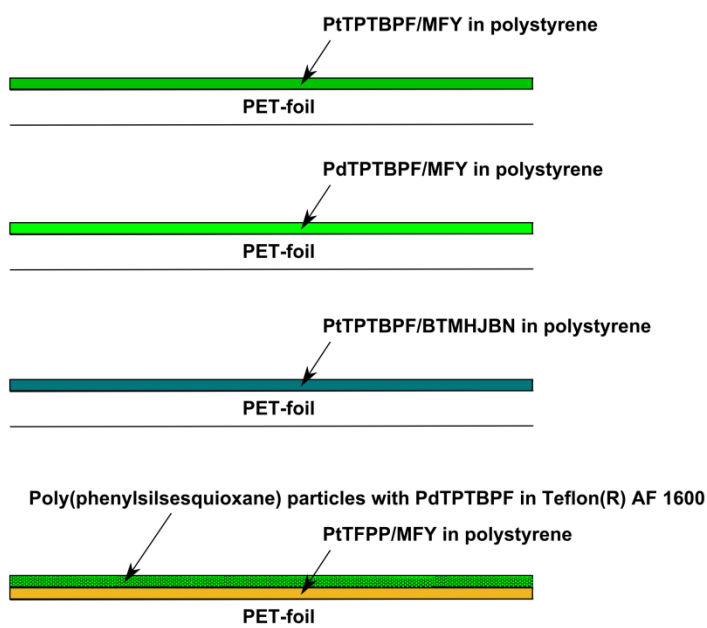
Supplementary information

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

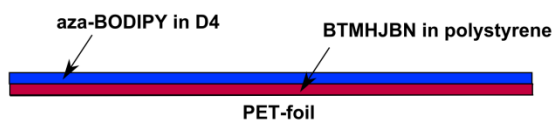
Josef Ehgartner, Helmar Wiltche, Sergey M. Borisov and Torsten Mayr

Supplementary Figures

pO₂-sensors



pH-sensor



pO₂/pH sensor

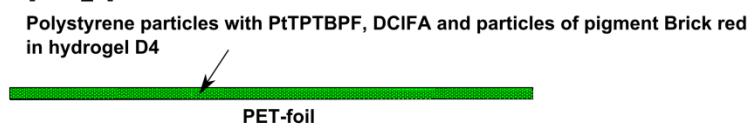
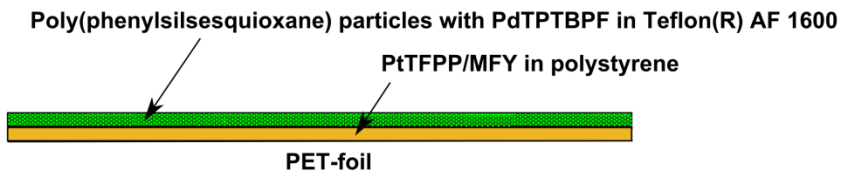
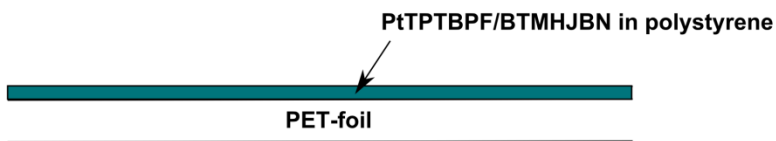
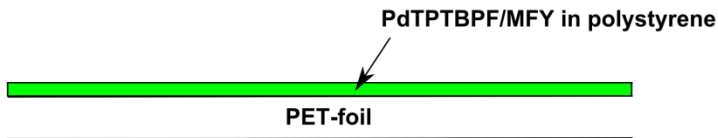
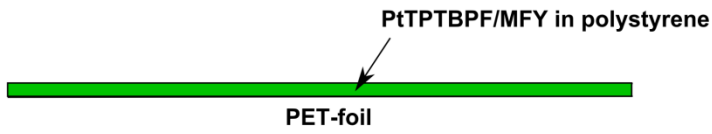
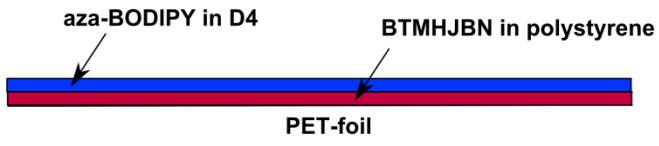


Figure 1 Schematic drawings of the different sensors are shown below.

pO₂-sensors

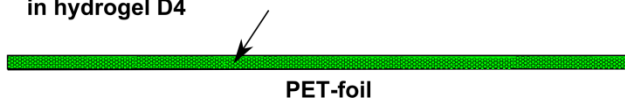


pH-sensor



pO₂/pH sensor

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



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.

```
1 /*
2
3 *****
4 *****
5 Shutter Control
6 0.13 HgW 290913
7
8 *****
9 *****
10
11 RS232 Syntax
12
13 <Mode (SET, GET, RUN, TST)>_<Command (TME, )>_<DEVICE (C1,
14 C2)>_<Parameter>VbCr
15 Example:
16 SET TME C1 02000 Sets the shutter open time of camera 1 to 2000 ms
17 RUN Starts the triggering process
18 SET DAC D1 03000 Sets the D1 DAC to 3000
19 TST Test the communication with the µcontroller
20
21 * LCD RS pin to digital pin 12
22 * LCD Enable pin to digital pin 11
23 * LCD R/W pin to Ground
24 * LCD D4 pin to digital pin 4
25 * LCD D5 pin to digital pin 5
26 * LCD D6 pin to digital pin 6
27 * LCD D7 pin to digital pin 7
28
29 */
30 // include the library code:
31 #include <LiquidCrystal.h>
32 #include <Wire.h>
33
34 // Constants
35 // set pin numbers:
36 const int TTL1Pin = 13; // Camera TTL 1 pin
37 const int TTL2Pin = 8 ; // Camera TTL 2 pin
38 const int DAC_Select_A = 2; // DAC channel select pin A
39 const int DAC_Select_B = 3; // DAC channel select pin B
40
41 // variables will change:
42 int DelayCam1 = 1000;
43 int DelayCam2 = 1000;
44
45 char SerialBuffer[255];
46 int SBufferLength = 255;
47 char vbCr = 13;
48 String SerialBufferSTR = "";
49 String CommandSTR = "";
```

```

49
50 // initialize the library with the numbers of the interface pins
51 LiquidCrystal lcd(12, 11, 4, 5, 6, 7);
52
53
54
55
//*****
****
*****
56 void setup() {
57 // set up the LCD's number of columns and rows:
58 lcd.begin(16, 2);
59 // Print a message to the LCD.
60 DoUpdateLCD();
61
62
63 // initialize the TTL pin as an output:
64 pinMode(TTL1Pin, OUTPUT);
65 pinMode(TTL2Pin, OUTPUT);
66
67 // initialize DAC selection pins as an output:
68 pinMode(DAC_Select_A, OUTPUT);
69 pinMode(DAC_Select_B, OUTPUT);
70
71
72 // initialize the RS232:
73 Serial.begin(9600);
74
75
76 // initialize the I2C communication
77 Wire.begin(); //Pin-Definition is set by the library to A4
(SDA) and A5 (SCL)
78 DAC_setVoltage(0, 1, true); //set the DAC to 0 and put this value in the
EEPROM
79 DAC_setVoltage(0, 2, true); //set the DAC to 0 and put this value in the
EEPROM
80 DAC_setVoltage(0, 3, true); //set the DAC to 0 and put this value in the
EEPROM
81 DAC_setVoltage(0, 4, true); //set the DAC to 0 and put this value in the
EEPROM
82 }
83
84
85
86
//*****
****
*****
87 void loop(){
88
89 digitalWrite(TTL1Pin, LOW);
90 digitalWrite(TTL2Pin, LOW);
91 digitalWrite(DAC_Select_A, LOW);
92 digitalWrite(DAC_Select_B, LOW);
93
94
95 // Step 1: Wait until the RS232 sends a new command
96 while (Serial.available() > 0) {
97 //Read the command
98 Serial.readBytesUntil(vbCr, SerialBuffer, SBufferLength);

```

```

99 if (SBufferLength > 0) {
100 SerialBufferSTR = SerialBuffer;
101 if (SerialBufferSTR.length() > 2) {
102 // Command found - decode it
103 if (SerialBufferSTR.substring(0, 3) == "SET") {
104 // SET Command found - decode it
105 DoDecodeCommand_SET(SerialBufferSTR.substring(4));
106 }
107 else if (SerialBufferSTR.substring(0, 3) == "RUN") {
108 //RUN Command
109 DoDecodeCommand_RUN();
110 }
111 else if (SerialBufferSTR.substring(0, 3) == "TST") {
112 //TST Command
113 Serial.print("OK");
114 }
115 else {
116 //Send via RS232 an ERROR signal
117 Serial.print("ERR");
118 }
119 }
120 }
121 //Update the LCD
122 DoUpdateLCD();
123 }
124 }
125 }
126
127
128
//*****
****
*****
129 void DoUpdateLCD(){
130 lcd.clear();
131 lcd.print("Camera 1: ");
132 lcd.print(DelayCam1);
133 lcd.setCursor(0, 2);
134 lcd.print("Camera 2: ");
135 lcd.print(DelayCam2);
136 /*Serial.print("");
137 Serial.print(DelayCam1);
138 Serial.print("");
139 Serial.print(DelayCam2);
140 Serial.print("");*/
141 }
142
143
144
//*****
****
*****
145 void DoDecodeCommand_SET(String CommandSTR){
146 //Decodiert den übergebenen SET-String
147
148 if (CommandSTR.substring(0, 3) == "TME"){
149 DoDecodeDeviceCommand_TME(CommandSTR.substring(4));
150 //Send via RS232 an OK signal
151 Serial.print("OK");
152 }
153 else if (CommandSTR.substring(0, 3) == "DAC"){

```

```

154 DoDecodeDeviceCommand_DAC(CommandSTR.substring(4));
155 //Send via RS232 an OK signal
156 Serial.print("OK");
157 }
158 else {
159 Serial.print("ERR: DoDecodeCommand_SET");
160 }
161 }
162
163
164
//*****
*****
165 void DoDecodeDeviceCommand_TME(String CommandSTR){
166 //Decodiert den übergebenen TME-String
167
168 char Buffer[6];
169 int BufferLen = 6;
170 String TempSTR = "";
171
172 if (CommandSTR.substring(0, 2) == "C1"){
173 // Camera 1 is the command target: Write to global variable
174 TempSTR = CommandSTR.substring(3);
175 TempSTR.toCharArray(Buffer, BufferLen);
176 DelayCam1 = atoi(Buffer); //convert STRING to INTEGER
177 if (DelayCam1 > 2000) DelayCam1 = 2000;
178 if (DelayCam1 < 50) DelayCam1 = 50;
179 }
180 else if (CommandSTR.substring(0, 2) == "C2"){
181 // Camera 2 is the command target: Write to global variable
182 TempSTR = CommandSTR.substring(3);
183 TempSTR.toCharArray(Buffer, BufferLen);
184 DelayCam2 = atoi(Buffer); //convert STRING to INTEGER
185 if (DelayCam2 > 2000) DelayCam2 = 2000;
186 if (DelayCam2 < 50) DelayCam2 = 50;
187 }
188
189 }
190
191
192
//*****
*****
193 void DoDecodeCommand_RUN(){
194 // RUN Command is processed
195 int TempDelay = 0;
196
197 if (DelayCam1 == DelayCam2) {
198 //Both Cameras have identical delay
199 lcd.clear();
200 lcd.print("Trigger running ...");
201 digitalWrite(TTL1Pin, HIGH); // turn the TTL Pin on (HIGH is the
voltage level)
202 digitalWrite(TTL2Pin, HIGH);
203 delay(DelayCam1); // wait
204 digitalWrite(TTL1Pin, LOW); // turn the TTL Pin off by making the
voltage LOW
205 digitalWrite(TTL2Pin, LOW);
206 DoUpdateLCD;

```

```

207 }
208 else if (DelayCam1 > DelayCam2) {
209 //Camera 1 has a larger delay than Campera2
210 TempDelay = DelayCam1 - DelayCam2;
211 lcd.clear();
212 lcd.print("Trigger running ...");
213 digitalWrite(TTL1Pin, HIGH); // turn the TTL Pin on (HIGH is the
voltage level)
214 digitalWrite(TTL2Pin, HIGH);
215 delay(DelayCam2); // wait
216 digitalWrite(TTL2Pin, LOW);
217 delay(TempDelay);
218 digitalWrite(TTL1Pin, LOW);
219 }
220 else if (DelayCam2 > DelayCam1) {
221 //Camera 1 has a larger delay than Campera2
222 TempDelay = DelayCam2 - DelayCam1;
223 lcd.clear();
224 lcd.print("Trigger running ...");
225 digitalWrite(TTL1Pin, HIGH); // turn the TTL Pin on (HIGH is the
voltage level)
226 digitalWrite(TTL2Pin, HIGH);
227 delay(DelayCam1); // wait
228 digitalWrite(TTL1Pin, LOW);
229 delay(TempDelay);
230 digitalWrite(TTL2Pin, LOW);
231 }
232
233 Serial.print("OK");
234 }
235
236
237
//*****
****
*****
238 void DAC_setVoltage( uint16_t po_voltageDAC, int Channel, bool
write_in_EEPROM
){
239 //Select the DAC
240 DoSelecDAC(Channel);
241
242
243
244 // Now communicate with the DAC
245 uint8_t twbrback = TWBR;
246 uint8_t _i2caddr;
247
248 _i2caddr = 0x60; //Hard-coded - selection is done by the Channel
249 TWBR = 12; // 400 khz
250
251 // First Byte
252 Wire.beginTransmission(_i2caddr);
253
254 // Second byte
255 if (write_in_EEPROM){
256 Wire.write(0x40);
257 }
258 else {
259 Wire.write(0x60);
260 }
}

```

```

261
262 Wire.write(po_voltageDAC / 16); // Upper data
bits (D11.D10.D9.D8.D7.D6.D5.D4)
263 Wire.write((po_voltageDAC % 16) << 4); // Lower data
bits (D3.D2.D1.D0.x.x.x.x)
264 Wire.endTransmission();
265 TWBR = twbrback;
266
267 }
268
269
270
271
//*****
****
*****
272 void DoDecodeDeviceCommand_DAC(String CommandSTR){
273 //Decodiert den übergebenen DAC-String
274
275 char Buffer[6];
276 int BufferLen = 6;
277 String TempSTR = "";
278 int DAC_value = 0;
279
280 lcd.clear();
281 lcd.print("Setting DAC...");
282
283 //DAC Werte berechnen
284 TempSTR = CommandSTR.substring(3);
285 TempSTR.toCharArray(Buffer, BufferLen);
286 DAC_value = atoi(Buffer); //convert STRING to INTEGER
287 if (DAC_value > 4095) DAC_value = 4095;
288 if (DAC_value < 0) DAC_value = 0;
289
290
291 //DAC setzen
292 if (CommandSTR.substring(0, 2) == "D1"){
293 // DAC 1 is the command target
294 DAC_setVoltage(DAC_value, 1, false);
295 }
296 else if (CommandSTR.substring(0, 2) == "D2"){
297 // DAC 2 is the command target
298 DAC_setVoltage(DAC_value, 2, false);
299 }
300 else if (CommandSTR.substring(0, 2) == "D3"){
301 // DAC 3 is the command target
302 DAC_setVoltage(DAC_value, 3, false);
303 }
304 else if (CommandSTR.substring(0, 2) == "D4"){
305 // DAC 4 is the command target
306 DAC_setVoltage(DAC_value, 4, false);
307 }
308
309 delay(300);
310 }
311
312
313
//*****
****
*****

```



```
314 void DoSelecDAC(int Channel){
315 //Wählt über DAC_Select_A und DAC_Select_B den über Channel übergebenen
DAC
316
317
318 //Schritt 1: Reset
319 digitalWrite(DAC_Select_A, LOW);
320 digitalWrite(DAC_Select_B, LOW);
321
322 switch (Channel) {
323 case 1:
324 //DAC 1 wählen
325 digitalWrite(DAC_Select_A, LOW);
326 digitalWrite(DAC_Select_B, LOW);
327 break;
328 case 2:
329 //DAC 2 wählen
330 digitalWrite(DAC_Select_A, LOW);
331 digitalWrite(DAC_Select_B, HIGH);
332 break;
333 case 3:
334 //DAC 3 wählen
335 digitalWrite(DAC_Select_A, HIGH);
336 digitalWrite(DAC_Select_B, LOW);
337 break;
338 case 4:
339 //DAC 4 wählen
340 digitalWrite(DAC_Select_A, HIGH);
341 digitalWrite(DAC_Select_B, HIGH);
342 break;
343 }
344 }
345
```