Electronic Supplementary Material (ESI) for Green Chemistry.

Supporting Information

Real-time reaction control for solar production of chemicals under fluctuating irradiance

Fang Zhao, Dario Cambié, Volker Hessel, Michael G. Debije and Timothy Noël*

E-mail: T.Noel@tue.nl

Table of Contents

1. Global Horizontal Radiation data

The irradiance data provided by SEAC (Solar Energy Application Centre) included the minute average of the global horizontal irradiance (GHI), the corresponding direct normal irradiance (DNI) and the ambient temperature as a csv file (an excerpt of their dataset is shown below). To transform the DNI into the direct component of the GHI for Figure 1 in the manuscript, the sun azimuth data with minute resolution was calculated for the measurement day (22/09/2017) with the longitude/latitude position of the measurement site (Eindhoven, the Netherlands) using SunPosition (*http://www.susdesign.com/sunposition*) (details in Figure S2). The DNI value of each minute was then multiplied by the cosine of the azimuth, in radians.

Date	Time	GHI avg	DNI avg	T avg	Azimuth (deg, calculated)	Direct component	Direct percent
22/09/2017	10:00:00	465.415	424.254	15	-45.03	299.836	64.4%
22/09/2017	10:01:00	477.965	415.677	15.1	-44.77	295.106	61.7%
22/09/2017	10:02:00	446.273	323.78	15.2	-44.51	230.897	51.7%
22/09/2017	10:03:00	430.339	268.129	15.2	-44.25	192.061	44.6%
22/09/2017	10:04:00	410.002	215.535	15.3	-43.99	155.069	37.8%
22/09/2017	10:05:00	442.642	276.964	15.4	-43.73	200.136	45.2%
22/09/2017	10:06:00	376.529	148.131	15.4	-43.47	107.504	28.6%
22/09/2017	10:07:00	310.757	39.98	15.4	-43.21	29.139	9.4%
22/09/2017	10:08:00	309.491	58.877	15.4	-42.95	43.095	13.9%
22/09/2017	10:09:00	298.334	57.302	15.4	-42.68	42.126	14.1%
22/09/2017	10:10:00	272.423	24.258	15.4	-42.42	17.908	6.6%
22/09/2017	10:11:00	265.746	19.164	15.4	-42.15	14.208	5.3%
22/09/2017	10:12:00	260.526	12.727	15.4	-41.89	9.474	3.6%
22/09/2017	10:13:00	270.364	25.572	15.4	-41.62	19.117	7.1%
22/09/2017	10:14:00	283.587	41.599	15.4	-41.36	31.223	11.0%
22/09/2017	10:15:00	297.845	65.754	15.3	-41.09	49.557	16.6%

Figure S1 Calculation of the direct component of GHI based on calculated azimuth.

SunPosition output complete
Latitude is 51.26 degrees north
Longitude is -5.29 degrees east
Time zone offset from GMT is 1 hours
Zero azimuth is south
Output angle units are degrees

Figure S2 Settings used in SunPosition for the calculation of the azimuth values.

2. Materials and Method

2.1 Reaction Control System

All the parts used are standard electronics components and were bought from *tinytronics.nl* except the phototransistor, obtained from *www.rs-online.com*. The reaction control system is composed of a phototransistor (TEPT4400, see specifications below), a microcontroller (Arduino UNO) connected to a 12V power supply, a serial to TTL module based on the max232 design (SP3232) and a 10 kΩ resistor connected using a breadboard and jump wires in the circuit represented below in Figure S3. Furthermore, to match the RX/TX pins in the serial module to the corresponding counterpart on the syringe pump (Chemyx Fusion 200) port, a crossover RS-232 adaptor was needed (*Conrad.nl* cat. No 569532–89). No crossover adaptor was needed to connect the reaction control system to a computer via serial connection using a parallel cable. The phototransistor was placed against the reactor edge and kept in position by a 3D-printed holder placed around the LSC-PM.

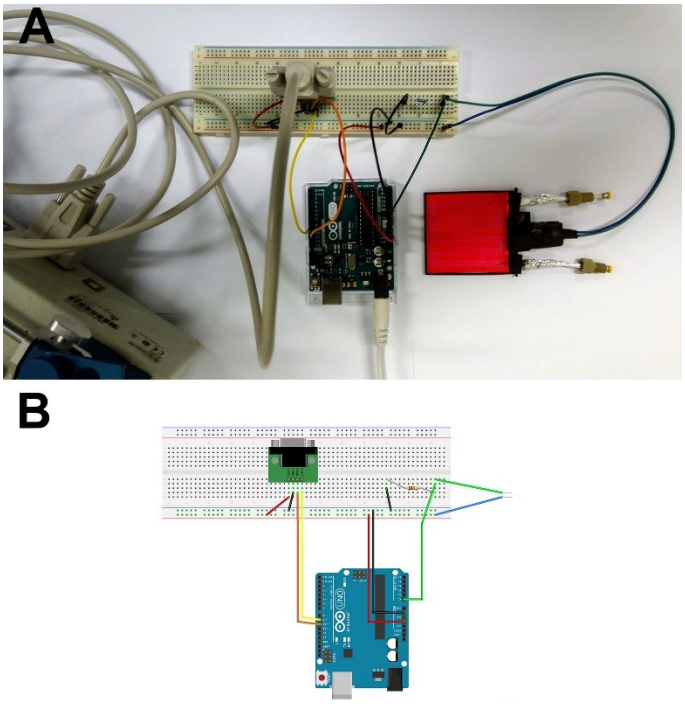

Figure S3 Picture and schematics of the reaction control system.

For convenience and safety, in the experiments the electronical components were all enclosed in 3D-printed box (see Figure S4) featuring an LCD display where the current value of the voltage

drop in the light sensing circuit and the corresponding flow rate were displayed. To take full advantage of the microcontroller capabilities, the box was designed to host two serial connections and 6 phototransistors (Q1-Q6). For the electrical circuit a small breadboard was attached on a ProtoShield connected to the Arduino microcontroller. The overall circuit schematics is presented in Figure S5.

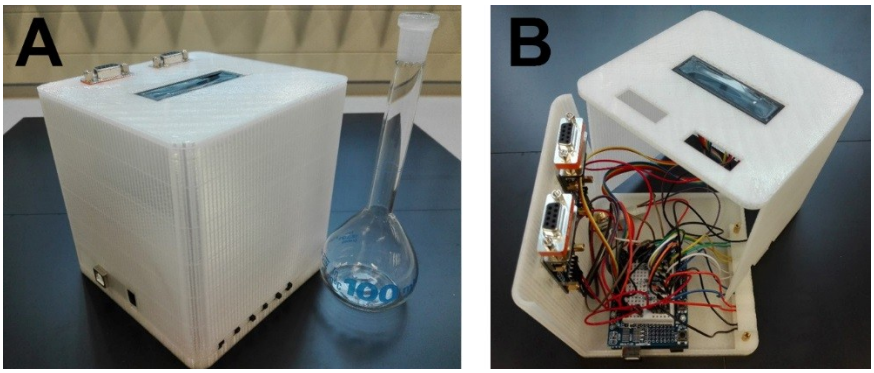

Figure S4 Photographs of the 3D-printed enclosure for the reaction control system. A) Outside view, 100ml volumetric flask for size comparison. B) Inside view with box opened, showing the internal components.

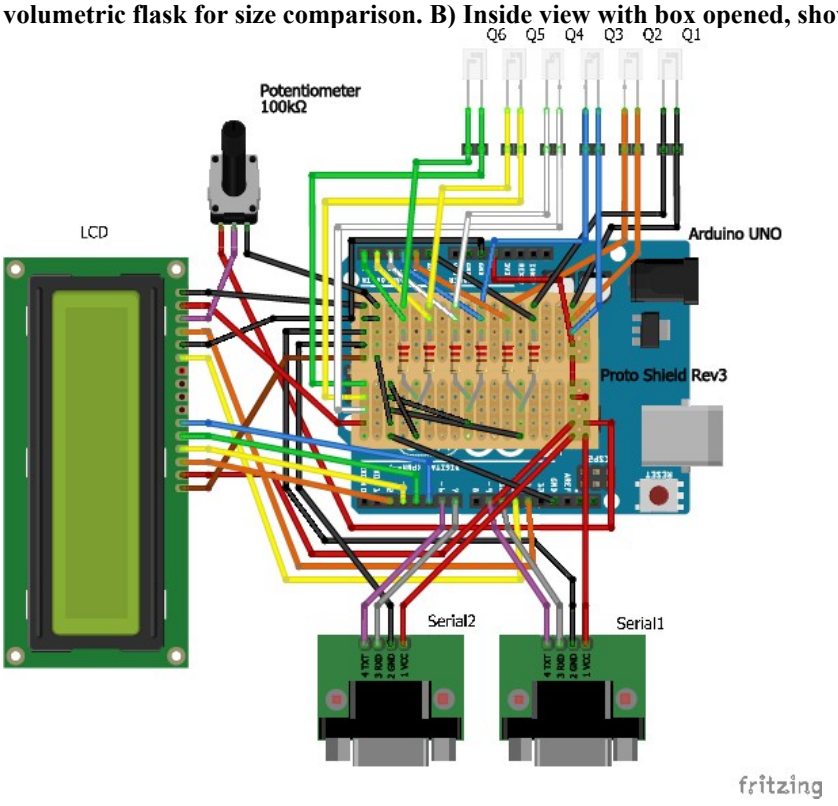

Figure S5 Electrical scheme of the circuit for the reaction control system.

2.2 Phototransistor datasheet

Below are reported the phototransistor specifications as obtained from the manufacturer datasheet.

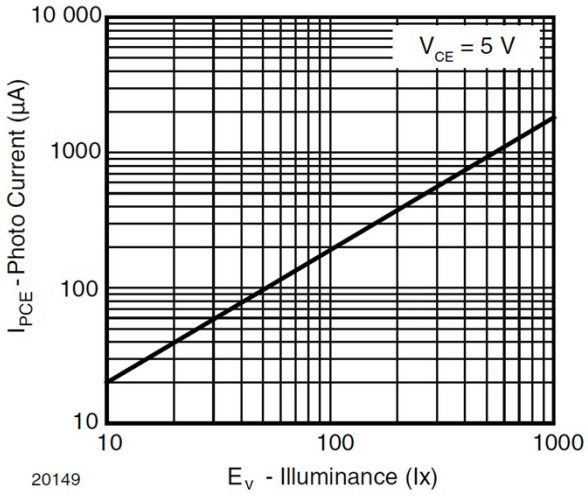

Figure S6 Relationship between photo current and illuminance (log-log plot, adapted from manufacturer datasheet).

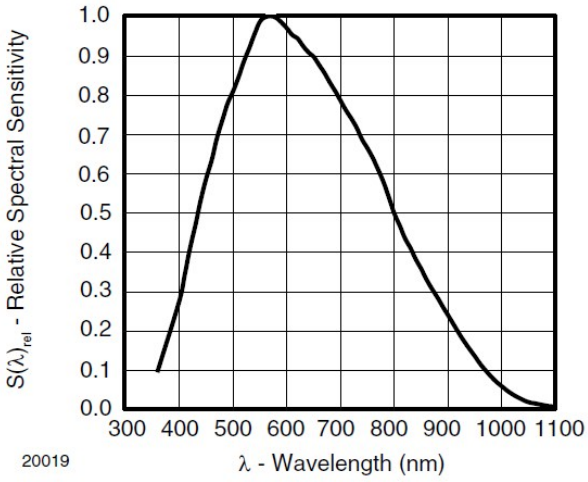

Figure S7 Relative spectral sensitivity of the phototransistor (adapted from manufacturer datasheet). The luminophore emission is centered at about 620nm, therefore matching with the optimal sensitivity range of the detector.

2.3 Light fluctuation experiment

To test the reaction control system with a reproducible light intensity profile, a second microcontroller was connected to the LED strip hosted in the cylindrical box previously described to adjust the light intensity according to a fixed and predefined pattern (see the source code at the end of the SI for details). In particular, a transistor (an IRLB8721 MOSFET) was adopted to change the LED intensity via pulse width modulation as showed in Figure S10.

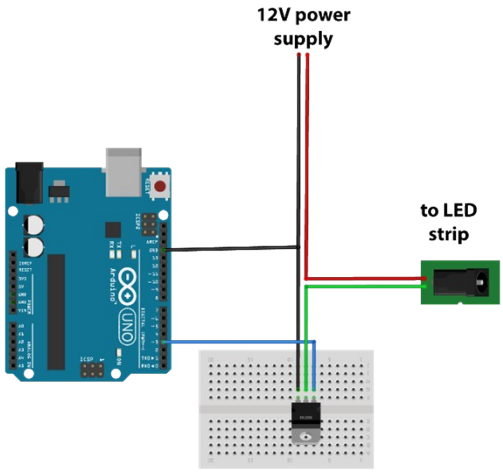

Figure S10 Scheme of the circuit for the LED driver.

2.4 Outdoor experiment

For reaction tests using the natural sunlight on the building roof, two reactors (two dye-doped LSC-PMs with and without reaction control, or a dye-doped LSC-PM with reaction control and a non-doped reactor without reaction control) were tested at the same time, as shown in Figure S11. A four-way valve was used so that the conversion in both reactors could be monitored by switching the valve periodically.

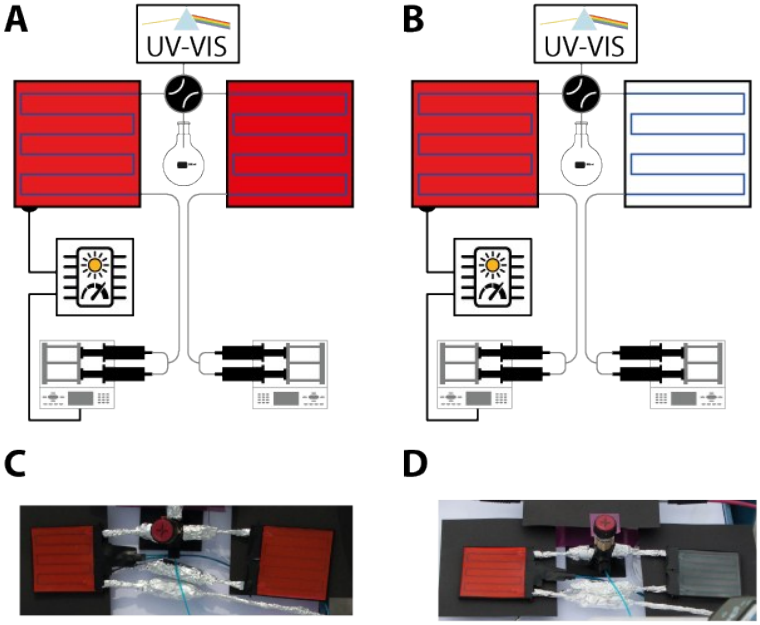

Figure S11 Schematic representation of the setup for the outdoor experiment. A) Comparison between LSC-PM with reaction control and without, B) comparison of the LSC-PM with reaction control and a non-doped reactor, C to E) picture of the setup.

3. Conversion allowance

To reduce the frequency of flow rate commands sent to the pump, the feasibility of employing a conversion allowance was investigated. By prescribing a conversion allowance σ, the control program, after a new voltage value is

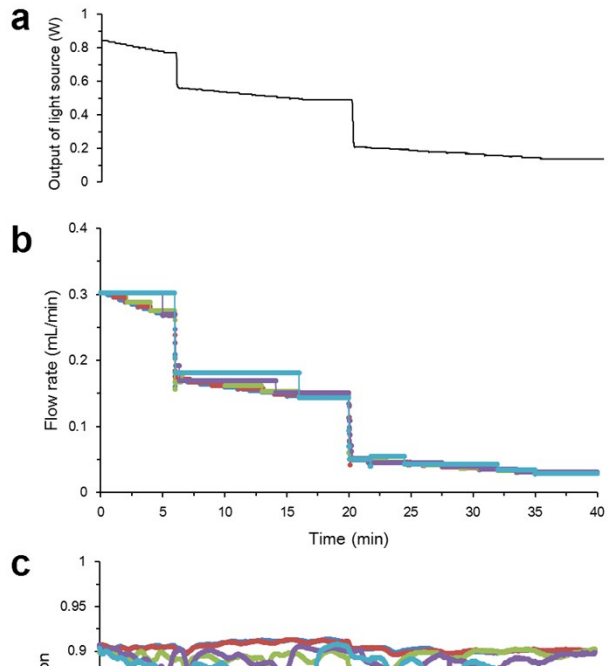

Figure S8 Reaction test for investigation on the effect of conversion allowance σ on control performance. A) The variation in light source intensity along with time. B) The flow rate changes along with time for different values of conversion allowance: 0, 0.5%, 1%, 2.5% and 5%. C) The conversion changes along with time in response to the light intensity variation shown in a for different values of conversion allowance: 0, 0.5%, 1%, 2.5% and 5%.

read, will first estimate the conversion at the current flow rate. If this estimated conversion is still in the range of ($X_T - \sigma$, $X_T + \sigma$, where X_T represents the target conversion), the control program will not send any command to the pump and continue the next loop directly.

A number of σ values were investigated: 0, 0.5 %, 1 %, 2.5 % and 5 %, respectively. In the tests, the light source intensity was decreased step by step from 0.84 to 0.77 W, from 0.56 to 0.49 W and from 0.21 to 0.14 W respectively, as shown by Figure S8 a, with a small step of approximately 0.007 W. The pump flow rate was also monitored for different values of σ and displayed in Figure S8b. It can be seen that the pump flow rate was changed less frequently as the value of σ increased.

4. Kinetic investigations

A schematic figure for the experimental set-up is given in Figure S9. DPA and an oxygen saturated MB solutions (0.2 mM and 0.4 mM in acetonitrile, respectively) were fed by a syringe pump (Chemyx Fusion 200) with the same flow rate. The two solutions were first mixed in a micro T-mixer (1/16″, IDEX Health & Science, Part No P-632) and then entered the LSC-PM which was placed in a 3D-printed cylindrical box (see Figure S9). A White LED strip (4.8W, Paulmann, Cat No 703.18) was wrapped onto the interior wall of the cylindrical box and a power supply (Delta Elektronika, power supply E030-3) was employed to power the LED strip. Pressurized air was blown into the cylindrical box to keep the temperature constant at 25 °C inside the box. The reaction mixture was analyzed with an in-line UV/Vis spectrometer (Avantes, AvaSpec 2048) by monitoring the DPA peak at 372 nm, whose absorbance was then used to calculate the reaction conversion. The syringes, the T-mixer and the connecting tubes were wrapped with aluminum foil to avoid exposure to the light outside the cylindrical box. In the experiments, the output of the power supply was varied to give different light intensity of the LED strip. The flow rate was changed to obtain the kinetic curves under each light intensity. The corresponding voltage value under each light intensity detected by the microcontroller was also recorded.

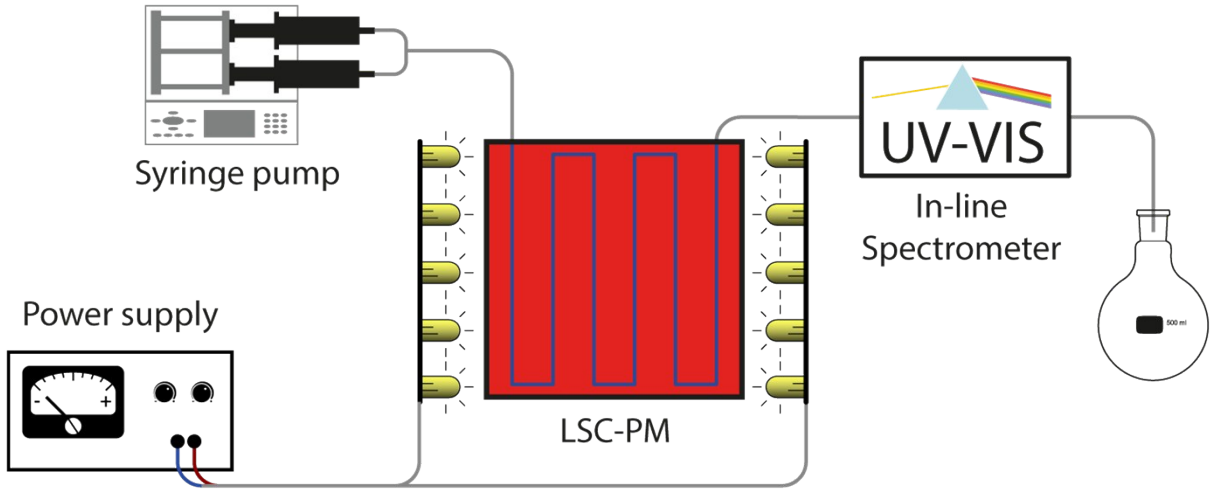

Figure S9 Flow scheme of the setup used for the kinetic investigations. The variable voltage power supply is used to study the reaction kinetics at different light intensity.

4.1 Kinetic investigation result

The experimental data used to generate the light intensity/conversion/residence time surfaces are the following.

The raw data were then interpolated (with Python's spline-fitting function bisplrep) to match the voltages associated with the Arduino light-levels.

Residence time/s	Conversion
0.606 V	
20	5.7%
40	10.9%
80	20.4%
160	35.6%
240	45.8%
0.747 V	
5	7.7%
7	10.6%
11	16.0%
20	25.6%
30	34.6%
45	48.1%
65	60.9%
85	70.6%
105	77.5%
150	86.6%
250	93.7%
0.854 V	
6	15.0%
11	24.8%
20	39.9%
25	46.9%
35	58.6%
45	67.4%
65	78.8%
90	85.7%
130	93.8%
1.001 V	
5	18.9%
6	22.7%
10	35.4%
15	47.7%
20	57.0%
30	70.8%
40	80.1%
55	88.7%
85	95.4%
1.201 V	
5	29.1%
7	38.9%
10	51.4%
14	63.4%
18	71.3%
24	80.8%
34	89.4%
50	95.9%
1.406 V	
4	34.3%
5	40.9%
7	51.9%
10	65.3%
14	76.8%
19	85.1%
30	94.0%
1.712 V	
10	78.1%
13	85.5%
16	90.5%
20	94.3%
25	97.4%
30	98.8%
1.937 V	
9	82.7%
10	85.6%
12	89.8%
15	93.9%
20	97.4%
25	98.9%
2.161 V	
7	81.1%
8	84.7%
10	89.6%
13	93.9%
16	96.1%
20	97.6%
2.343 V	
6	80.7%
8	88.1%
10	92.7%
12	94.8%
16	96.9%
2.536 V	
5	77.6%
6	83.6%
8	90.4%
11	94.7%
14	96.5%

5. Cost evaluation

One notable advantage of our reaction control system is its simple and inexpensive nature. The total price of the components of a reaction control system with one phototransistor and one pump can be calculated as follows:

Table S1 Cost of components for the reaction control system.

Component		Price
Microcontroller	Arduino/Genuino UNO	€ 23.00
Phototransistor	*rs-online.com* TEPT4400	€ 0.36
Resistor	10 kΩ	€ 0.05
Serial port/UART module	SP3232 with mounting screws	€ 5.00
Serial crossover adaptor	*Conrad.nl* 569532-89	€ 12.80
Serial cable	2 male ends	€ 4.79
ProtoShield	Including breadboard	€ 5.00
Jump wires	Pack of 65, various colors	€ 3.00
	TOTAL:	€ 54.00

Remarkably, this total price can be further reduced by about one third by using a generic Arduino clone (≈3€) instead of a genuine one, whose characteristics are identical due to the open hardware nature of the project.

6. Arduino source code

6.1 Relationship between number of voltage readings and the measurement error

By increasing the number of individual calls to the analogRead() function, the error associated with that measurement decreases with the square root of the number of readings per cycle. In Figure S12 the decrease in the CV% of 600 individual measuring cycle in constant irradiation condition is plotted as function of the number of readings averaged per cycle.

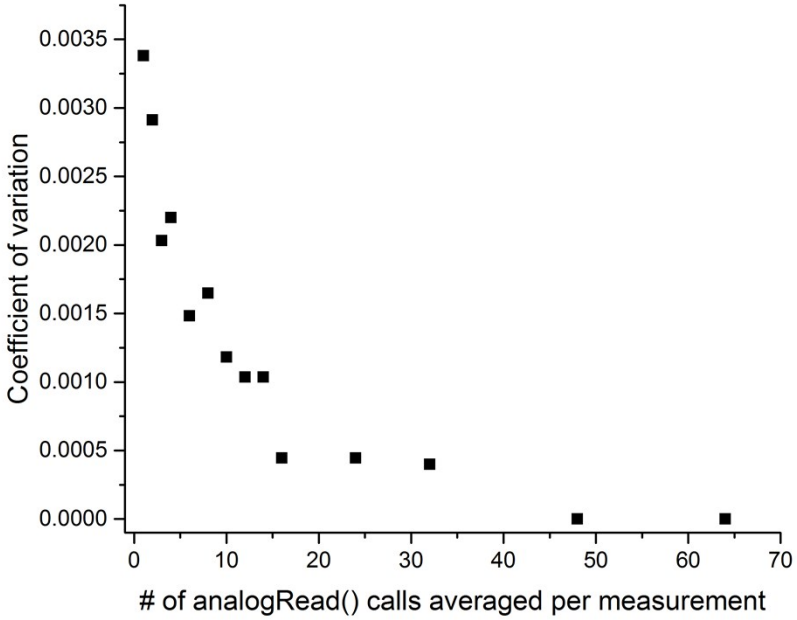

Figure S12 Decrease in the coefficient of variation as function of the number of readings averaged.

6.2 Reaction Control System code

```
/*
  The circuit:
 * LCD RS pin to digital pin 12
 * LCD Enable pin to digital pin 11
 * LCD D4 pin to digital pin 5
 * LCD D5 pin to digital pin 4
 * LCD D6 pin to digital pin 3
 * LCD D7 pin to digital pin 2
 * Serial1 RX pin to digital pin 10
 * Serial1 TX pin to digital pin 9
 * Phototransistor1 to analog pin 0
 */
// Time library
#include <Time.h>

// LCD
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

```
// SERIAL PORTS
#include <SoftwareSerial.h>
SoftwareSerial mySerial1(10, 9); // RX, TX
//SoftwareSerial mySerial2(7, 6); // RX, TX

// Number of iterations (8k is about 1 sec)
int iterations = 4096;
// Phototransistor pin
int analogPin = 0;

// Internal variables
unsigned long sum = 0;
String str_flow_rate = "1.0";
float voltage;
float flow_rate;
// Data from kinetics (below for 50% target
conversion)
```

```
// The data represent the pump flowrate associated
with each 10mV light level in the 0.75-2.53V
interval
float flow_rate_for_voltage[] =
{0.074448329,0.078966952,0.083587801,0.088361292,0.0
93279578,0.098319923,0.103451781,0.108652066,0.11388
3977,0.119132067,0.124380724,0.129650143,0.134964741
,0.14034518,0.145814034,0.151393125,0.15708681,0.162
893859,0.168849678,0.17497382,0.18124301,0.18761583,
0.194174148,0.200797969,0.207542606,0.214416571,0.22
131854,0.228369437,0.235428995,0.242569983,0.2498322
93,0.256965141,0.264337086,0.271560051,0.278790304,0
.286225544,0.29348551,0.300683874,0.308052344,0.3155
79959,0.32266174,0.329881656,0.337254976,0.344799131
,0.352010852,0.359189673,0.366529708,0.374047449,0.3
81756571,0.389291345,0.396666174,0.404189021,0.41186
6799,0.419705589,0.427710669,0.435722731,0.443481493
,0.451380361,0.459419714,0.467596682,0.475910333,0.4
84354701,0.492922623,0.501221764,0.509421507,0.51764
8098,0.525881386,0.534101898,0.542305879,0.55049122,
0.558656968,0.566801884,0.574927492,0.582503529,0.58
9717691,0.596888949,0.604025338,0.611135351,0.618224
245,0.625304381,0.63238512,0.63947591,0.646589334,0.
653740525,0.660941178,0.668208092,0.675556691,0.6830
05761,0.69057308,0.697663471,0.704463103,0.711383594
,0.718440512,0.725657517,0.733055852,0.740656421,0.7
48488132,0.75657091,0.764916534,0.773522167,0.782386
998,0.79151048,0.800889384,0.810523341,0.820407228,0
.830541711,0.84091992,0.851538404,0.862390523,0.8724
95193,0.882013187,0.891605412,0.901254428,0.91094489
3,0.920662176,0.930391529,0.940119689,0.94983278,0.9
59518448,0.96916366,0.978762988,0.988326181,0.997859
264,1.007374926,1.016876693,1.026381071,1.035889014,
1.04541559,1.054964952,1.06455344,1.074184388,1.0838
67871,1.093615781,1.103437743,1.11334515,1.123341832
,1.133441862,1.143659532,1.15399551,1.162703762,1.17
1051525,1.179451426,1.187913291,1.196428929,1.205002
219,1.213621962,1.22228578,1.230987665,1.239724372,1
.24848799,1.257274737,1.266074639,1.274889853,1.2837
0465,1.292518376,1.301324063,1.310112459,1.318876901
,1.327610978,1.336305985,1.344954965,1.353545446,1.3
62066602,1.370503972,1.378847493,1.387080711,1.39519
1658,1.403165214,1.410987557,1.41864454,1.426121127,
1.433402579,1.440479296,1.447325973,1.453936256,1.46
0286154,1.466369277,1.472164502,1.477656202};

void setup() {
  // Serial monitor
  Serial.begin(9600);
  Serial.println("Starting Arduino...");

  pinMode(10, INPUT);
  mySerial1.begin(38400);

  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  lcd.print("Starting controller");
}

float sum_to_voltage(unsigned long sum) {
  // Convert the raw data value (0 - 1023) to
voltage (0.0V - 5.0V):
  return sum/iterations * (5.0 / 1023.0);
}

float Flow_rate_for_voltage(float voltage) {
  int light_level;
  voltage = voltage * 100;
  light_level = (int) voltage;

  // Boundary conditions: edge between 0.75 and 2.53
V
  if (light_level < 75) {
    light_level=75;
  }
  if (light_level > 253) {
    light_level = 253;
  }
  // convert voltage into array indexes (one value
every 10 millivolts, starting from 0.75V)
  return flow_rate_for_voltage[light_level-75];
}

void loop() {
  // Read the voltage for $iterations time
  for (int  i = 0; i < iterations; i++) {
    // read the raw data coming in on analog pin 0:
    sum += analogRead(analogPin);
  }
  // Convert sum of light levels to voltage
  voltage = sum_to_voltage(sum);
  // Get flow rate corresponding to that voltage
value
  str_flow_rate =
String(Flow_rate_for_voltage(voltage), 5);
  // Set sum back to 0
  sum = 0;

  // SET PUMP
  mySerial1.listen();
  // The code below is compatible with Chemyx Fusion
200
  // Verify the command needed for each pump, and
the corresponding connection settings.
  mySerial1.println("set rate " + str_flow_rate);

  // Printing current voltage to LCD
  lcd.setCursor(0, 1);
  lcd.print("Voltage: "+String(voltage, 5));

  // Print data to serial (time, voltage, flowrate)
for logging purposes
  Serial.println(String(millis()) + ", " +
String(voltage, 3) + ", " + str_flow_rate);

  // Wait until pump has set the flow rate and
answered to the previous command. This prevents
occasional pump freeze
  mySerial1.read();
}
```

6.3 LED driver code for the variable irradiation experiment

```
// Time library
#include <Time.h>
// PWM on pin 3
#define WHITEPIN 3

// Seconds in milliseconds, i.e. 1000 (use shorter
to accelerate testing)
#define THOUSAND 1000
// Minute in milliseconds, i.e. 60000 (use shorter
to accelerate testing)
#define MINUTE 60000

float seconds;
unsigned long timedelay_ms;

void setup() {
  // Serial monitor
  Serial.begin(9600);
  Serial.println("Starting Arduino...");
  pinMode(WHITEPIN, OUTPUT);
}

bool Linear_decrease(unsigned int from, unsigned int
to, unsigned int total_time) {
  float time_delay = ( float ) total_time/(from-to);
  time_delay = time_delay * THOUSAND;
  for (int  i = from; i >= to; i--) {
    analogWrite(WHITEPIN, i);
    delay(time_delay);
  }
}

bool Linear_increase(unsigned int from, unsigned int
to, unsigned int total_time) {
  float time_delay = ( float ) total_time/(to-from);
  time_delay = time_delay * THOUSAND;
  for (int  i = from; i <= to; i++) {
    analogWrite(WHITEPIN, i);
    delay(time_delay);
  }
}

bool Squared_decrease_to_fless(unsigned int from,
unsigned int to, unsigned int total_time) {
  float coefficient = (total_time )/sqrt(from-to);
  float previous = (unsigned long) total_time *
THOUSAND;
  for (int  i = (from-to); i >= 0; i--) {
    float current = sqrt(i) * coefficient *
THOUSAND;
    analogWrite(WHITEPIN, i+to);
//    Serial.println(i+to);
    delay(previous-current);
    previous = current;
    }
}

bool Squared_increase_from_fless(unsigned int from,
unsigned int to, unsigned int total_time) {
  float coefficient = (total_time)/sqrt(to-from);
  float previous = 0;
  analogWrite(WHITEPIN, from);
  for (int  i = 1; i <= (to-from); i++) {
    float current = sqrt(i) * coefficient *
THOUSAND;
    analogWrite(WHITEPIN, i+from);
//    Serial.println(i+from);
    delay(current-previous);
    previous = current;
    }
}

bool Squared_increase_to_fless(unsigned int from,
unsigned int to, unsigned int total_time) {
  float coefficient = (total_time )/sqrt(to-from);
  float previous = (unsigned long) total_time *
THOUSAND;
  for (int  i = (to-from); i >= 0; i--) {
    float current = sqrt(i) * coefficient *
THOUSAND;
    analogWrite(WHITEPIN, to-i);
//    Serial.println(to-i);
    delay(previous-current);
    previous = current;
    }
}

bool Squared_decrease_from_fless(unsigned int from,
unsigned int to, unsigned int total_time) {
  float coefficient = (total_time)/sqrt(from-to);
  float previous = 0;
  analogWrite(WHITEPIN, from);
  for (int  i = 1; i <= (from-to); i++) {
    float current = sqrt(i) * coefficient *
THOUSAND;
    analogWrite(WHITEPIN, from-i);
//    Serial.println(from-i);
    delay(current-previous);
    previous = current;
    }
}

void loop() {
seconds = millis()/1000;



  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] -- START
-- Holding at mid");
  analogWrite(WHITEPIN, 158);
  // 1 minute
  timedelay_ms = (unsigned long) 3 * MINUTE;
  delay(timedelay_ms);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Holding at low");
  analogWrite(WHITEPIN, 61);
  // 10 minutes
  timedelay_ms = (unsigned long) 8 * MINUTE;
  delay(timedelay_ms);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Holding at high");
  analogWrite(WHITEPIN, 255);
  // 2 minutes
  timedelay_ms = (unsigned long) 2 * MINUTE;
  delay(timedelay_ms);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Holding at mid");
  analogWrite(WHITEPIN, 158);
  // 3 minutes
  timedelay_ms = (unsigned long) 3 * MINUTE;
  delay(timedelay_ms);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Linear decrease to low");
  Linear_decrease(158, 61, 150);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Linear increase to high");
  Linear_increase(61, 255, 300);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Linear decrease to mid");
  Linear_decrease(255, 158, 150);
```

```
  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Holding at mid");
  analogWrite(WHITEPIN, 158);
  // 3 minutes
  timedelay_ms = (unsigned long) 3 * MINUTE;
  delay(timedelay_ms);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Squared decrease to low");
  Squared_decrease_to_fless(158, 61, 150);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Squared increase to mid");
  Squared_increase_from_fless(61, 158, 150);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Squared increase to high");
  Squared_increase_to_fless(158, 255, 150);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Squared decrease to mid");
  Squared_decrease_from_fless(255, 158, 150);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Holding at mid");
  analogWrite(WHITEPIN, 158);
  // 3 minutes
  timedelay_ms = (unsigned long) 3 * MINUTE;
  delay(timedelay_ms);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Squared decrease to low");
  Squared_decrease_from_fless(158, 61, 150);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Squared increase to mid");
  Squared_increase_to_fless(61, 158, 150);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Squared increase to hi");
  Squared_increase_from_fless(158, 255, 150);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Squared decrease to mid");
  Squared_decrease_to_fless(255, 158, 150);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Holding at mid");
  analogWrite(WHITEPIN, 158);
  // 3 minutes
  timedelay_ms = (unsigned long) 3 * MINUTE;
  delay(timedelay_ms);

  seconds = millis()/1000;
  Serial.println("[" + String(seconds) + "] --
Swithcing off (2h)");
  analogWrite(WHITEPIN, 0);
  // 3 minutes
  timedelay_ms = 120 * MINUTE;
  delay(timedelay_ms);
}
```