

## Supplementary

# Construction of a Miniaturized Detector for Flow Injection Spectrophotometric Analysis

T. Alexandra Ferreira <sup>1</sup>, Mario Ordaz <sup>2</sup>, Jose A. Rodriguez <sup>3</sup>, M. Elena Paez-Hernandez <sup>3</sup> and Evelin Gutierrez <sup>4,\*</sup>

<sup>1</sup> Campus Puebla, Universidad del Valle de Mexico, Camino Real a San Andrés Cholula No. 4002, Emiliano Zapata, San Andres Cholula 72810, Mexico; thania.ferreira@uvmnet.edu

<sup>2</sup> Departamento de Ingenieria Electrica y Electronica, TecNM/Campus Pachuca (IT Pachuca), Pachuca 42080, Mexico; mario.oo@pachuca.tecnm.mx

<sup>3</sup> Area Academica de Quimica, Universidad Autónoma del Estado de Hidalgo, Carretera Pachuca-Tulancingo Km. 4.5, Mineral de la Reforma 42184, Mexico; josear@uaeh.edu.mx (J.A.R.); paezh@uaeh.edu.mx (M.E.P.-H.)

<sup>4</sup> Departamento de Ingenieria Mecatronica, Universidad Politécnica de Pachuca, Ex. Hacienda Santa. Barbara, Zempoala 43830, Mexico

\* Correspondence: evgutierrez@upp.edu.mx

### 1. Characteristics of the 3D piece

The design of the structure that supports the fundamental elements of the spectrophotometer constructed in this experiment was carried out using SolidWorks 2022, a computer-aided design (CAD) software used for 3D modeling of parts and assemblies, as well as 2D drawings. The 3D printing piece was manufactured using the Flashforge Creator Pro 3D printer, whose features are shown in the following Table.

Parameter	Value	Parameter	Value
Extruder quantity	2	Filament compatibility	PLA, TPU95A, ABS, PETG
Nozzle diameter	0.4 mm	Filament diameter	1.75 mm
Maximum temperature extruder	240°C	Print Volume	227×148×150 mm
Print speed	30-100 mm/s	Layer thickness	0.1mm-0.4mm
Maximum temperature platform	120°C	Print precision	±0.2 mm

For the printing, the following specific parameters were used: 1.75mm PLA with supports and without a base, high and low resolutions, 0.4 mm nozzle size, standard slicing profile, extruder temperature of 200 °C, and platform temperature of 60 °C.

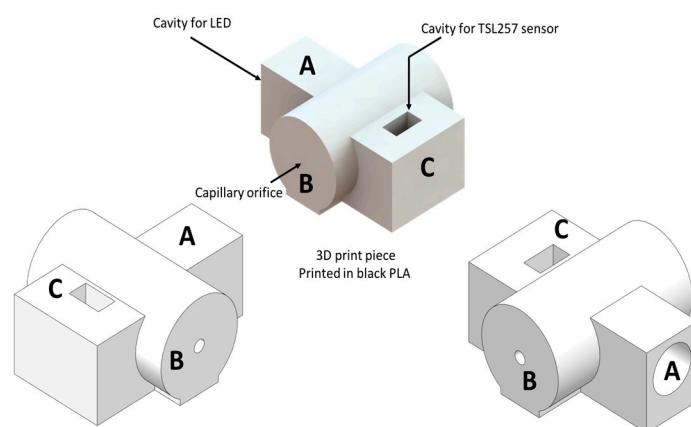
The 3D-printed piece used in this experiment consists of three sections. Figure S1 shows three perspectives of this piece. The function of these three sections is described as follows:

- Cavity to hold the light source with a diameter of 6.0 mm was considered.
- Support for the capillary responsible for transporting the sample. This is the largest section, with a cavity length of 21 mm and a diameter of 1.6 mm. For this experiment, a capillary for hematocrit with a length of 75 mm, an

internal diameter of 1.1-1.2 mm, and an external diameter of 1.5-1.6 mm was used. The variation in the external diameter was taken into account in the 3D design. A small section at both ends of the capillary remains exposed to allow for connection to the hoses used in the experiment.

- C. Cavity to place the light-to-voltage converter TSL257-LF. The dimensions of the 3D printed piece are 2.60 · 4.50 mm with a depth of 5.80 mm. These dimensions were defined considering the TSL257-LF model.

The alignment of the three sections (ABC) facilitates the LED–capillary–photodiode light path. In the 3D design, a spacing of 10 mm was considered for the separation between the LED and the photodiode in the TSL257-LF.

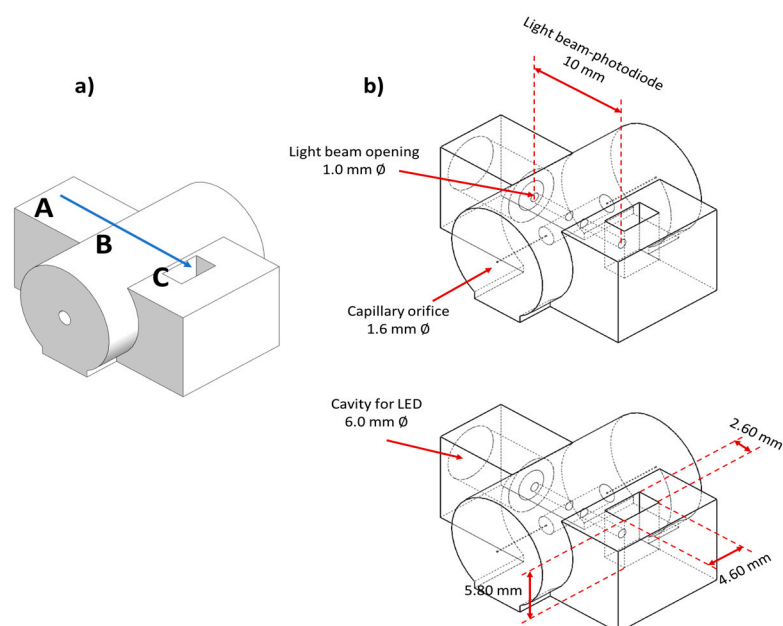


**Figure S1. Three-dimensional-printed piece.**

Considering that the dimensions of the LED used are considerably larger than the diameter of the capillary employed, the diameter of the light beam was controlled in the 3D-printed piece by using a 1.0 mm diameter hole at the inner end of cavity A (Figure S2).

For the 3D-printed piece, the use of matte black Polylactic Acid (PLA) filament is highly recommended to prevent the reflection of the light beam on the inner walls of the piece from creating new wavefronts causing interference in the reading of the TSL257-LF.

Three alternatives are proposed for the development of this experiment. The first alternative involves providing students with the key dimensions of the elements used in the construction of the device, so they can design their own 3D-printed pieces. The second alternative is to provide access to the .sldprt file, allowing users to make modifications according to their needs. The third alternative is to provide the .stl file to facilitate the 3D-printing process exactly as reported in this document.



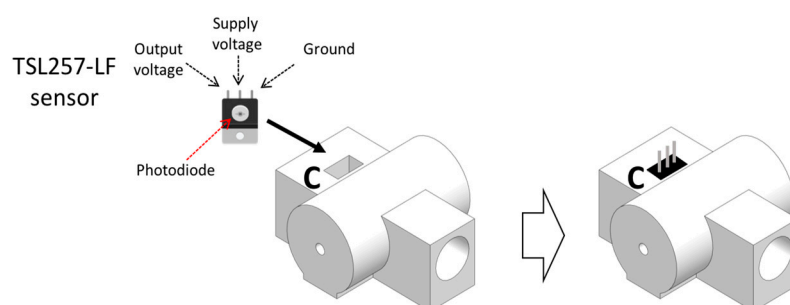
**Figure S2.** a) Alignment for the LED-capillary-photodiode light path. b) Internal structure of the 3D printed piece.

## 2. Circuit construction

For the construction of the electrical circuit, an Arduino UNO development board was used, which was programmed using the Arduino Integrated Development Environment (IDE). The circuit components include a TSL257-LF voltage optical sensor, a blue LED emitter diode (450-500 nm), a 68  $\Omega$  and 0.25 watt resistor, and a mini breadboard. Additionally, the 3D-printed piece described in the previous section was used. The connections were made using male-to-male and male-to-female jumper wires.

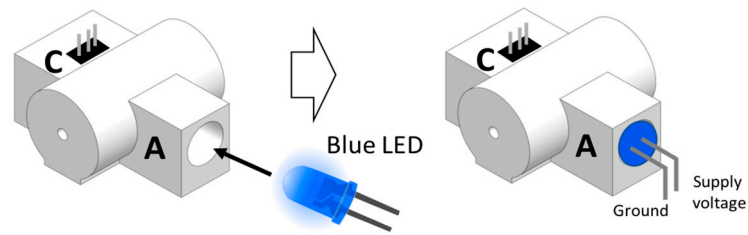
## 3. Construction of the spectrophotometric detector:

1. Place the TSL257-LF in cavity C of the 3D-printed piece. Ensure that the photodiode is directed towards cavity B, which supports the capillary tube for the passage of the sample to be analyzed. Identify the three terminals of the TSL257-LF: ground, supply voltage, and output voltage. These terminals will be later connected to the Arduino UNO. If necessary, you can apply black silicone to prevent interference from external light sources but ensure that the sensor pins remain exposed (Figure S3).



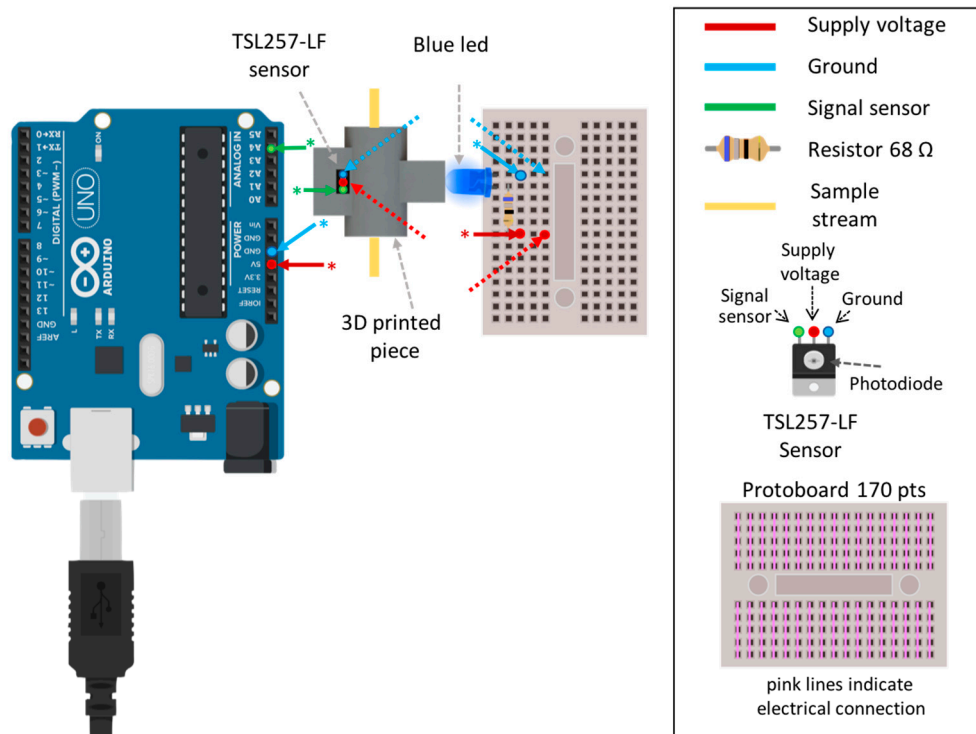
**Figure S3.** Mounting of the TSL257-LF onto the 3D printed piece.

- Place the LED in cavity A of the 3D-printed piece (Figure S4).



**Figure S4.** Alignment of the LED-capillary cavity-TSL257-LF.

- Make the connections as shown in Figure S5.



**Figure S5.** Construction of the circuit.

#### 4. Arduino Integrated Development Environment (IDE)

The process is carried out according to the following steps:

- Connect the Arduino UNO board to your computer using the USB cable.
- The Arduino IDE is a tool that enables writing, compiling, and uploading code to the Arduino board. It is recommended to download and install the Arduino IDE on your computer from the official Arduino website (<https://www.arduino.cc/en/software>).
- Open the Arduino IDE. Upon opening, a window with a blank space will appear where you will write your code.

4. Write the code in the provided text area.
5. Check for any syntax errors in the code by clicking the verification button (tick/checkmark icon).
6. Select the type of board you are using by accessing the "Tools" menu in the IDE and choosing "Board." In this case, select Arduino/Genuino UNO as your board.
7. Select the COM port to which the Arduino UNO board is connected. This option can be found under "Tools" and then "Port" in the IDE menu. If you are unsure about which port to use, it is recommended to disconnect the Arduino, check the available ports, reconnect it, and observe the new port in the list.
8. Once steps 1 to 7 have been completed, you need to upload the code to the Arduino UNO board. To do this, click the upload button in the top left corner of the IDE. As a response, you will see the IDE compiling the code and starting to upload it to the Arduino board. During the process, the activity LED on the Arduino will blink intermittently.
9. The IDE will display a message indicating a successful upload once the process is completed.

#### 5. Program for reading the light to voltage converter (TSL257-LF sensor)

To perform data acquisition, the Arduino UNO board is connected to a computer, which should have the following recommended minimum specifications: an Intel Core i5 10th generation processor running at 4.3 GHz or higher, with 8 GB RAM and the Windows, Mac OS X, or Linux operating system.

The program created in the Arduino IDE for this experiment begins by establishing communication between the Arduino UNO board and the computer through the serial port using a physical USB cable. Communication with any device connected to the RX and TX pins is carried out at a serial speed of 9600 bits per second (baud).

In the program, the analog input A4 is declared to capture the voltage value detected by the TSL257-LF analog sensor. To accomplish this task, it is also necessary to define pin A4 as an input. Subsequently, the sensor reading is initiated within the loop that will run if the Arduino UNO board is powered on. The value read within this loop is binary in nature and ranges from 0 to 1023 (10 bits). This binary value needs to be converted to decimal, which varies between 0 and 4.5 volts. To do so, the binary value is converted to a float format. This float value is then multiplied by 4.5 and divided by 1023, and stored in a float variable that is printed onto the serial monitor every half second. This float value corresponds to the reading from the TSL257-LF sensor and is written in a text file row by row using the CoolTerm software. The code used in the data acquisition process is included in the following lines.

```
// Author: Mario Oscar Ordaz
```

```
const int TSL257Pin = A4; // Define the analog input pin for the sensor
int TSL257value = 0; // Initialize the variable to store the sensor value
unsigned long startTime; // Variable to store the program start time
```

```
void setup() { // Setup function runs once at the beginning
  Serial.begin(9600); // Start serial communication at a baud rate of 9600
  Serial.println(F("Initialize System")); // Print an initialization message to the serial monitor
  pinMode(TSL257Pin, INPUT); // Set the sensor pin as an input
  startTime = millis(); // Save the start time
}
```

```
void loop() { // Main loop function
  // Check if 50 minutes have not passed
  if (millis() - startTime <= 3000000) { // 3000000 milliseconds = 50 minutes
    readSensor(); // Call the readSensor function to read and print sensor values
  } else {
    Serial.println("Program stopped after 50 minutes."); // Print message if 50 minutes have passed
  }
}
```

```

while (true) {
  // Add any additional actions here upon stopping the program
}
}
}

void readSensor() { // Begin of function to read and process sensor values
  TSL257value = analogRead(TSL257Pin); // Read the analog value from the sensor
  Serial.print("Voltage: ");
  Serial.println(SensorRawToPhys(TSL257value)); // Calculate and print the voltage value in Volts
  delay(500); // Add a delay to control the rate of reading
} // End of function to read and process sensor values

float SensorRawToPhys(int raw) { // Begin of function to convert raw analog value to physical voltage in Volts
  float Vout = float(raw) * (4.5 / float(1024)); // Calculate the voltage based on the raw analog value
  return Vout; // Conversion analog to voltage
} // End of function to convert raw analog value to physical voltage in Volts

```

The main sections of the code can be summarized as follows:

The variable "TSL257Pin" is defined as a constant integer with the value A4, which represents the pin to which the TSL257 sensor is connected. The pin is set as an input using the pinMode function.

In the setup function, the serial communication is initialized with a baud rate of 9600. The message "Initialize system" is printed onto the serial monitor using the Serial.println function.

The main loop function repeatedly calls upon the readSensor function, which reads the value from the TSL257-LF sensor and sends the converted voltage to the serial monitor. The analogRead function is used to obtain the analog value from the sensor.

The readSensor function includes a conversion function called SensorRawToPhys, which converts the raw sensor value to a physical voltage value. The converted voltage is then printed onto the serial monitor.

A delay of 500 milliseconds is added after each reading to prevent excessive readings.

The description provided in the above paragraphs outlines the key elements and functionality of the code in reading and converting the voltage from the TSL257-LF sensor.

#### *6. Instructions for storing the data obtained (voltage)*

This section describes the use of CoolTerm software for storing data obtained from the TSL257-LF sensor using the program developed in the Arduino IDE. CoolTerm is a free and open-source serial terminal program for the Windows and macOS operating systems. It is primarily used for serial communication with electronic devices such as microcontrollers, Arduino, GPS modules, among others. The CoolTerm program is widely used in electronic development projects for testing and debugging devices that utilize serial communication. Its functionality and ease of use make it a popular tool. CoolTerm has been chosen since it is free software, easy to use, and allows easy storage of data written to the Arduino UNO development board's serial monitor.

The CoolTerm software installer can be downloaded from the following website where the software is available for different operating systems and processors: <https://freeware.the-meiers.org/>. Since this study was conducted on a personal computer running Windows with a 64 bit Intel processor, the downloaded and executed installer file corresponds to the mentioned system specifications.

Below are the detailed steps to store the data printed onto the Arduino serial monitor on a text file (.txt) using CoolTerm:

1. Connect your Arduino to your computer using the USB cable and ensure that it is correctly configured in CoolTerm. Open CoolTerm on your computer.
2. In CoolTerm, go to the "Options" menu and select "Connections." Make sure the correct COM port is selected for your Arduino.
3. In the "Options" menu, select "Logging." A popup window with options related to data logging will appear.
4. In the "Data logging" section, check the "Log incoming data" box. This will allow the received data from Arduino to be saved in the text file.
5. Click the "Browse" button next to the "Log file" option. A window will open to select the location and name of the text file where the data will be saved. Choose the desired location and assign a name to the file.
6. In the "Logging format" section, select the format in which you want the data to be saved in the file. You can choose between "Raw" or "Formatted." The "Formatted" format will include additional information such as date and time.
7. Adjust any other additional settings you want, such as the delay time between data samples.
8. Click the "OK" button to close the logging configuration window.
9. Now, when Arduino sends data to the serial monitor and CoolTerm is connected to the correct COM port, the data will be automatically saved in the specified text file.
10. To stop logging data, go to the "Options" menu, select "Logging," and uncheck the "Log incoming data" box.

The process is described in a more illustrative and dynamic way in the video at the following link:  
<https://youtube.com/watch?v=xS1DGgCymuQ&feature=share9>