TÉCNICO LISBOA

Instituto Superior Técnico

Master Degree in Electrical and Computer Engineering

Distributed Real Time Control Systems

Project Report - Part 1

João Paiva de Castelo-Branco

ist1111019

Prof. Alexandre Bernardino

2023/2024

1 - Introduction

This report covers the first phase of the Distributed Real-Time Control Systems project for the academic year 2023/2024. The document begins with a brief and straightforward overview of the project goals and requirements. Following this introduction, it provides a comprehensive explanation of how the specified requirements have been met.

2 - Project Overview and Model

The project's aim is to create a networked real-time control system for a simulated small-scale office lighting environment. Each workspace in the model is equipped with an intelligent luminaire, featuring a light source, a luminance sensor to gauge reflected light, a presence detector, and elements for system control and communication. The simulation uses an opaque white shoe box with adequate dimensions to house the system, ensuring controlled light conditions and reflection.

In this setup, luminaires are interconnected via a CAN-BUS system for data exchange, focusing on adjusting LED brightness to balance user comfort and energy efficiency. The technical setup includes a microcontroller-driven LED, a light-dependent resistor for illuminance measurement, and a setup to minimize noise and ensure accurate light readings.

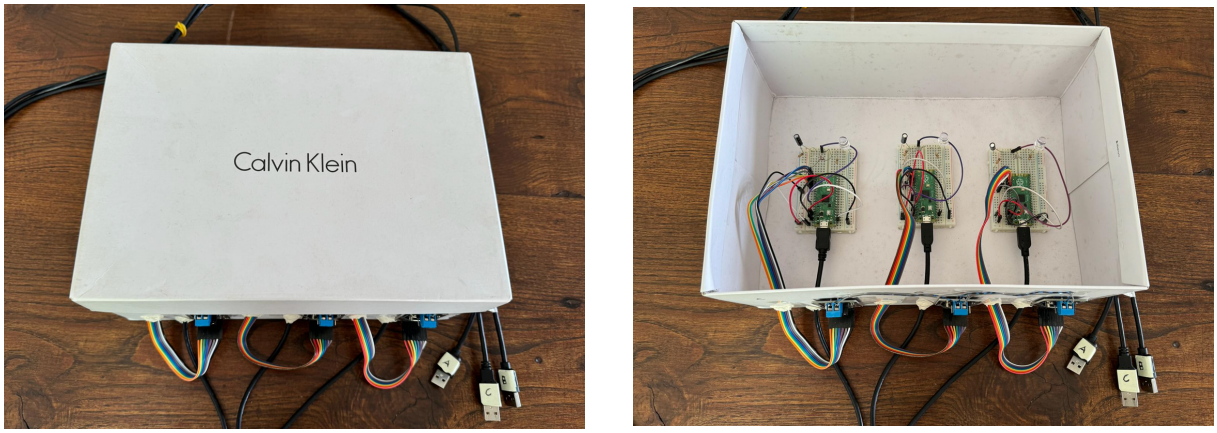

Figure 1: Small scale model of the office space

3 - System Development and Configuration

This section details the methodologies and approaches applied in the project, along with the performance outcomes achieved.

In the initial phase, the project involves setting up the luminaires, establishing the control network, and configuring the microcontroller. The tasks encompass developing local control (for a sampling rate of **100Hz**), creating a basic interface with the PC through Arduino IDE, and evaluating CAN-BUS communication among the nodes. At this juncture, each local controller operates independently, striving to maintain the illumination level as close to the set value as possible amidst external disturbances. The system is designed to respond swiftly to changes in the desired illumination level while minimizing flicker and overshooting.

3.1 Illuminance Measurement System

As mentioned previously, the system must accurately measure the environment illuminance. Figure 2 represents the electrical circuit of the illuminance measurement subsystem.

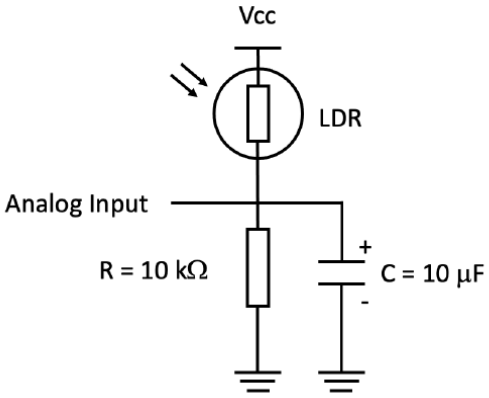

Figure 2: Schematics of the luxmeter circuit

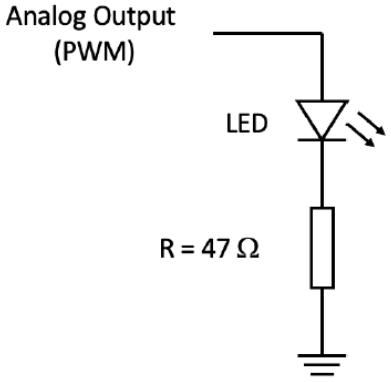

Figure 3: Schematics of the LED actuation driver circuit

Therefore, a function is needed to convert the voltage read at the analog input port to LUX. According to the LDR datasheet, the relationship between LUX and the LDR resistor value is given by:

$$\log_{10}(\text{LDR}) = m \log_{10}(\text{LUX}) + b \tag{3.1}$$

To express LUX as a function of the read voltage input, we can use the voltage divider equation and combine it with the previous expression:

$$\text{LDR} = \frac{R \cdot V_{cc}}{V_i} - R \tag{3.2}$$

$$\text{LUX} = 10^{\left(\frac{\log_{10}\left(\frac{R \cdot V_{cc}}{V_i} - R\right) - b}{m}\right)} \tag{3.3}$$

Note that $R = 10k\,\Omega$ and $V_{cc} = 3.3V$. According to the LDR datasheet, $m = -0.8 \pm 0.1$ and $b \in [5.976, 6.277]$. To evaluate differences among the three used LDRs, they were placed inside the box one at a time, in the same position and orientation, with the box closed, using the same LED and allowing enough stabilization delay for each measurement. Then, different PWM values were applied to change LED intensity (more details about this in the next section), and the corresponding computed LUX values were measured, using $m = -0.8$ and $b = 6.127$. Due to the LDR's large tolerance, the obtained curves slightly differed for the same lighting conditions, as seen in figure 4.

However, the goal is to achieve consistent measured LUX values for the three LDRs under the same lighting conditions, rather than measuring the exact physical LUX. Therefore, the m and b parameters for luminaires B and C were adjusted until the three obtained curves were sufficiently similar (figure 5). The adjustments were $b_B = 6.480$, $m_B = -0.81$, $b_C = 5.820$, and $m_C = -0.77$.

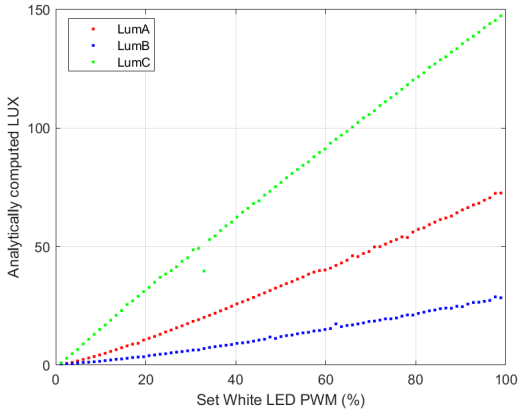

Figure 4: PWM - LUX functions under the same lighting conditions

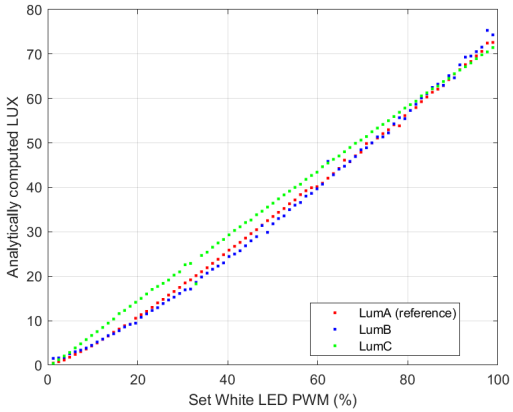

Figure 5: Adjusted PWM - LUX functions under the same lighting conditions

3.2 LED Actuation System

The electrical circuit for the LED actuation driver is depicted in figure 3. The utilized microcontroller (Raspberry Pi Pico) lacks a pure DAC (Digital-to-Analog Converter) output; thus, a square wave at **60kHz** is employed to drive the LED and the light intensity is linearly proportional to the set duty cycle. This frequency is sufficiently high to prevent the human eye, which acts as a low-pass filter for frequencies above 100Hz, from detecting flicker. Consequently, the LED pulsing is not perceivable. Additionally, the chosen frequency is at least ten times greater than the cutoff frequency of the input filter (enabled by the input capacitor), ensuring that the switching does not significantly impact the illuminance measurement signal.

3.3 Individual Luminaire Controller

The overall project system is illustrated in Figure 6. For effective controller implementation, determining internal parameters such as gains and time constants is essential.

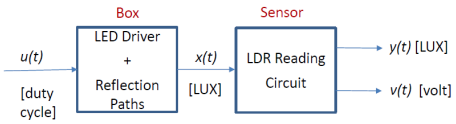

Figure 6: Schematics of the project system

3.3.1 Subsystem Parameters

Assuming external illumination is negligible, the box subsystem can be modeled as a zeroth-order system with a static gain G:

$$x(t) = Gu(t) \tag{3.4}$$

For each luminaire, plotting the PWM - LUX function allows G to be determined as the slope of this plot, as presented in figure 8. Nevertheless, since this gain is highly influenced by factors such as the orientation of the LED and LDR, and the reflective characteristics of the box, it is advisable to calibrate G at the start of each working session. Given the linear relationship between PWM and LUX, **only two points are needed to determine the gain**, as it is equivalent

to the slope. This was the method adopted in the project's code.

The sensor subsystem, on the other hand, can be modeled as a first-order system, described by the following equation:

$$\dot{v}\tau(x) = -v + V_{cc}\frac{R_1}{R_1 + R_2(x)} \tag{3.5}$$

Considering the linear approximation of figure 7, the time constant $\tau(x)$ and gain $H(x)$ must be computed as a function of the set x_{ref}.

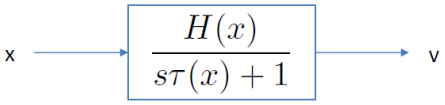

Figure 7: Sensor System Linear Approximation

Considering $C = 10\mu F$, $R_1 = 10^4\Omega$, $R_2 = LDR = 10^{m\log_{10}(x)+b}$ and $R_{eq} = \left(\frac{1}{R_1} + \frac{1}{R_2}\right)^{-1}$:

$$\tau(x) = R_{eq}(x)C = 10^{-1}\frac{10^{m\log_{10}(x)+b}}{10^{m\log_{10}(x)+b} + 10^4} \tag{3.6}$$

$$H(x) = \frac{V_{ss}}{x} = \frac{33000}{10000 + 10^{m\log_{10}(x)+b}}\frac{1}{x} \tag{3.7}$$

To validate the theoretical expressions, each luminaire was placed inside the box, and voltage values were recorded around a step change in the PWM input, as shown in figure 9.

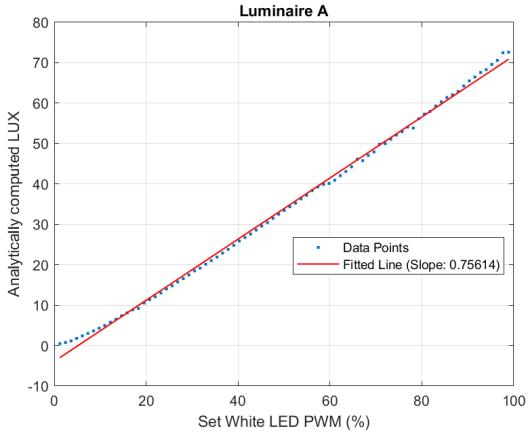

Figure 8: Computation of static gain for Luminaire A

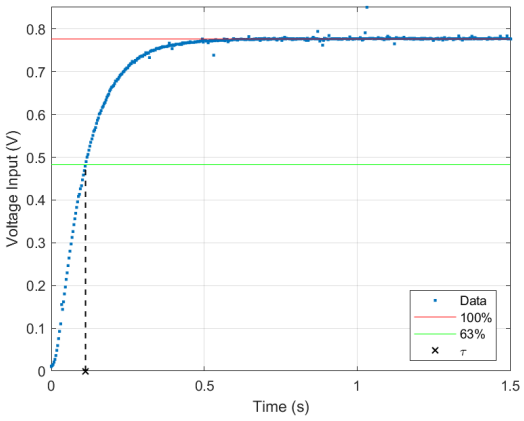

Figure 9: General procedure of τ experimental computation

Following the represented procedure, τ and H were computed for several x_{ref} [LUX] values. Note that the τ value may depend not only on the final value, so the procedure was performed for several initial conditions. The experimentally obtained values for H were reasonably similar to the theoretical computation as seen in figure 11. However, the theoretical τ values were lower than the experimental values. Therefore, a constant of $0.070[s]$ was added to the theoretical formula to minimize that difference, and the used approximation became reasonable enough to be used in every case, as seen in figure 10.

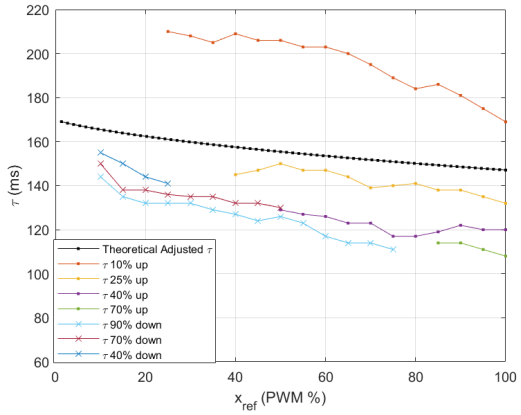

Figure 10: Luminaire A - experimental computation of τ for several references and from different starting points

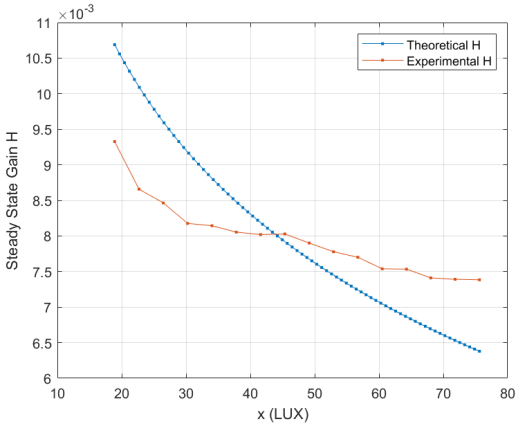

Figure 11: Luminaire A - experimental computation of steady state gain H for several references

3.3.2 PI Controller

For the linear approximation of the lab system, the set point weighting PI controller represented in Figure 12 was used. Note that the system has no inertia, so the derivative term is not needed.

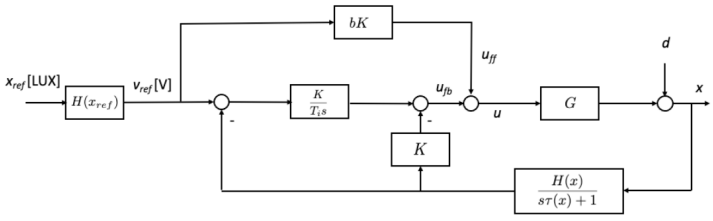

Figure 12: PI controller for the linear approximation of the project system

The set-point transfer function, provided in the course classes, is given by:

$$\frac{X}{X_{\text{ref}}} = \frac{(bT_i s + 1)(s\tau + 1)}{(s\tau + 1)\frac{T_i s}{GHK} + T_i s + 1} = \frac{bT_i \tau s^2 + (bT_i + \tau)s + 1}{\frac{T_i \tau}{GHK} s^2 + (\frac{T_i}{GHK} + T_i)s + 1} \tag{3.8}$$

For the LED to follow the reference very quickly, without overshoots or oscillations, an ideal transfer function is the identity. In order to achieve that goal, the values of b, K, and T_i must follow the following equations:

$$bK = \frac{1}{GH} \wedge T_i = \tau \tag{3.9}$$

Note that there is still ambiguity in the values of b and K. This can be resolved by computing the disturbance transfer function and setting K such that the system reacts slowly to external disturbances, as specified in the project system. This transfer function can be given by:

$$X = D - G\left(\frac{KH}{\tau s + 1} + \frac{KH}{(\tau s + 1)(T_i s)}\right) X \tag{3.10}$$

$$\frac{X}{D} = \frac{\frac{T_i^2}{GHK}s^2 + \frac{T_i}{GHK}s}{\frac{T_i^2}{GHK}s^2 + \left(\frac{T_i}{GHK} + T_i\right)s + 1} \quad \text{(setting } \tau = T_i\text{)} \tag{3.11}$$

$$\frac{X}{D} = \frac{\alpha T_i s(T_i s + 1)}{(\alpha T_i s + 1)(T_i s + 1)} \quad \text{(setting } \alpha = \frac{1}{GHK}\text{)} \tag{3.12}$$

$$\frac{X}{D} = \frac{\alpha T_i s}{\alpha T_i s + 1} \tag{3.13}$$

It is clear that in the steady state, the disturbance transfer function becomes null as desired (when $s \to 0$). To set the respective time constant to a value of x, we can do:

$$\alpha T_i = x \Leftrightarrow \frac{1}{GHK} T_i = x \Leftrightarrow K = \frac{T_i}{GHx} \tag{3.14}$$

$$b = \frac{1}{GHK} \Leftrightarrow b = \frac{x}{T_i} \tag{3.15}$$

Finally, the PI controller was tested experimentally to fine-tune these parameters, and the final expressions are presented below, using $x = 20$ (slow response to disturbances). This guarantees optimal system behaviour for each set reference, as parameters are dynamically computed. Note that the expression for b had to be slightly modified by dividing its theoretical expression by x to obtain the desired system behaviour. The system performance is shown in the following report sections.

$$T_i = \tau \wedge K = \frac{T_i}{20GH} \wedge b = \frac{1}{T_i} \tag{3.16}$$

3.3.3 Anti-windup and Bump-less Transfer Function

The actuator signal varies between 0% and 100%, which corresponds to 0 and 4095, as a 12-bit range is used for the DAC. When the control signal saturates, the integral part continues to accumulate, leading to a phenomenon known as integrator windup or reset windup. To mitigate this effect, a back-calculation strategy is implemented in the previously described PI controller with discharge integrator constant set to $T_t = 1$.

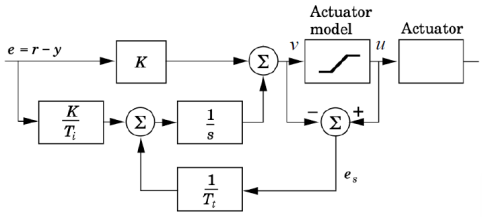

Figure 13: Project PI controller with back-calculation

On the other hand, dynamically changing the controller's parameters can create "bumps" in the output. Nonetheless, it is possible to ensure no changes in the output when the error is zero and the system is in a steady state, by making the following update to the integral term. These upgrades ensure a more robust and efficient controller for the project system.

$$I_{new} = I_{old} + K_{old}(b_{old}y_{sp} - y) - K_{new}(b_{new}y_{sp} - y) \tag{3.17}$$

3.3.4 Performance testing

In this section, the performance of the assembled luminaire controller is presented in different scenarios.

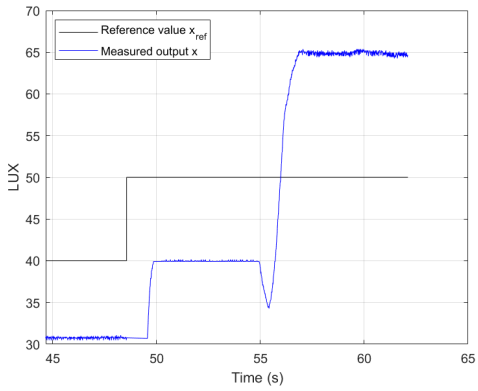

Figure 14: Controller without feedback

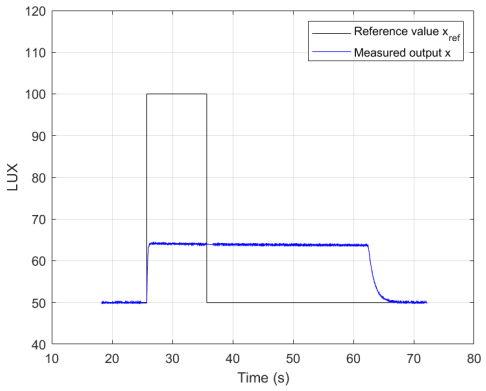

Figure 15: Controller without anti-windup

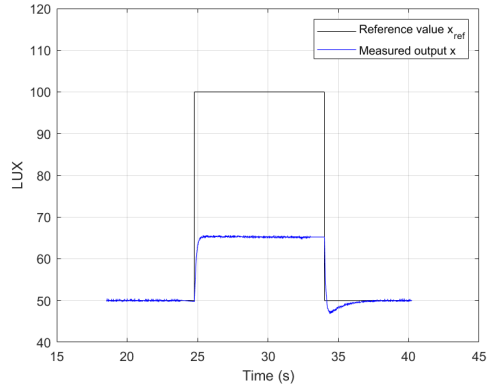

Figure 16: Complete controller - saturation scenario

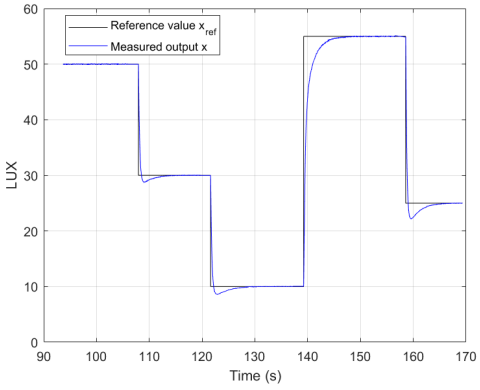

Figure 17: Complete controller - reference changes

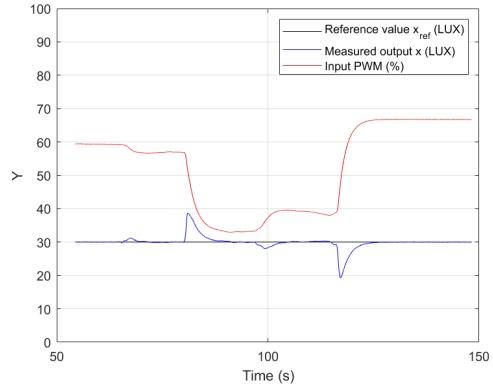

Figure 18: Complete controller - Disturbance reaction

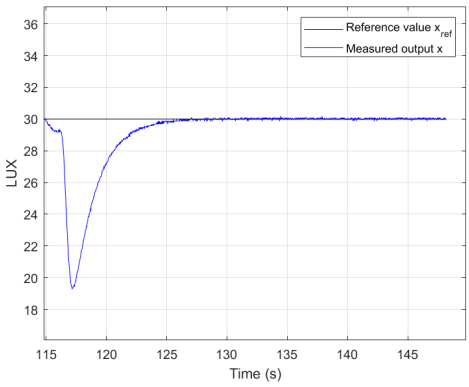

Figure 19: Complete controller - Disturbance reaction close up

3.4 Sampling and Filtering Strategy

To ensure proper sampling intervals of $10ms$, an interrupt service routine (ISR) is employed, which sets the value of a Boolean variable to `True`. In the main loop, this variable is checked, and the control computation is executed if indicated. This strategy ensures an accurate sampling interval and, consequently, low jitter, as illustrated in Figure 20.

Additionally, a low-pass digital filter is implemented in the LDR readings to reduce noise. During each sampling interval, 10 samples are acquired, and the median of these values is computed. The improvement obtained through this filtering approach is demonstrated in Figure 21.

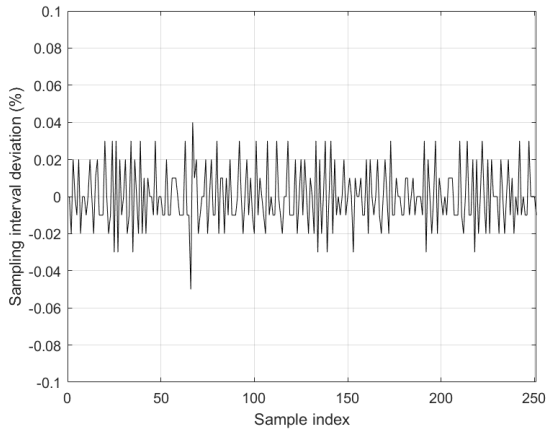

Figure 20: Relative measured jitter

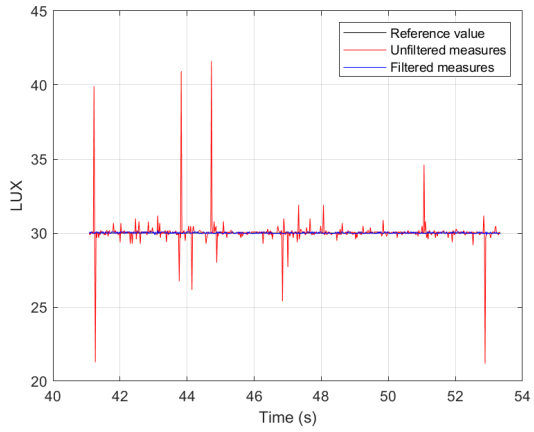

Figure 21: Filtering effect on analog readings

It is also important to note that to ensure control computations and CAN-BUS routines do not interfere with the accurate sampling interval, their execution time is monitored. If these exceed a predefined time threshold, a warning is dispatched to the user. According to measurements made, on average, control computations take 0.400 ms, and serial port communications with 8 bytes, for example, take 0.650 ms, which is much lower than the sampling time.

3.5 Data Storage and Metrics Computation

As described in the project statement, different routines were implemented to store data in the microcontroller about performance metrics since each restart of the run program. Every sampling interval of $10ms$, energy, visibility error, and flicker are updated using the new system measures and following the expressions below. These metrics can be displayed at any point by using the corresponding user interface commands, specified later in the report.

$$E = P_{\text{max}} \sum_{k=1}^{N} d_{k-1}(t_k - t_{k-1}) \tag{3.18}$$

$$V = \frac{1}{N} \sum_{k=1}^{N} \max(0, L(t_k) - l(t_k)) \tag{3.19}$$

$$F = \frac{1}{N} \sum_{k=1}^{N} f_k \tag{3.20}$$

$$f_k = \begin{cases} |d_k - d_{k-1}| + |d_{k-1} - d_{k-2}| & \text{if } (d_k - d_{k-1})(d_{k-1} - d_{k-2}) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.21}$$

It is also crucial to accurately compute the P_{max} parameter of the used white LED. To do this, the LED duty cycle is set to 100%, and two voltages are measured using a multimeter: one across the LED and one across the resistor. Then, applying basic electrical circuit concepts (Figure 3), P_{max} is calculated as follows:

$$P_{\text{max}} = V_{\text{LED}} I = V_{\text{LED}} \frac{V_R}{R} = 3.0V \times \frac{0.3V}{47\Omega} \approx 19.15\,\text{mW} \tag{3.22}$$

The following graphs were obtained by displaying these computed metrics in real time. As intended, the consumed energy shows a slope proportional to the applied PWM. The visibility error is low and tends to decrease, while the flicker is extremely low and mostly constant.

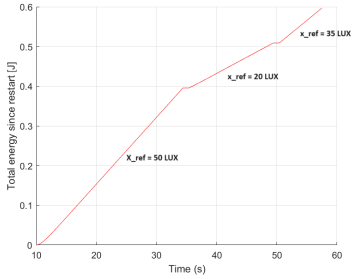

Figure 22: Total consumed energy

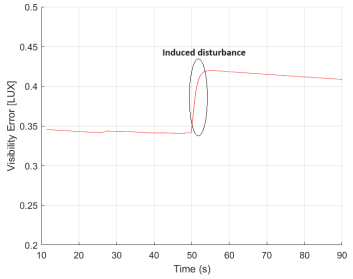

Figure 23: Average Visibility Error

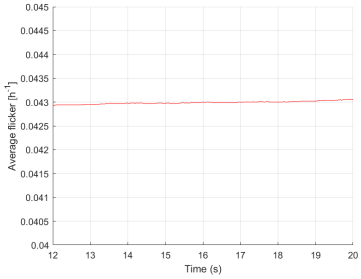

Figure 24: Average Measured Flicker

3.6 CAN-BUS Assembly and Functioning

The CAN-BUS network was assembled by connecting all luminaires to a CAN-BUS network, as illustrated in Figure 1. Terminator jumpers in the CAN-BUS controllers were employed at the extremities of the bus. A program to send and receive simple messages between an Arduino node and the microcontroller was implemented. It uses a non-blocking state machine to efficiently handle the multiple tasks running on the microcontroller, such as user interface and local control. The subsequent figure demonstrates a basic interaction between two nodes within this distributed decentralized cooperative control system.

```
Sending message 5 A : 00000005
Received message number 2 from node C : 00000000
Sending message 6 A : 00000006
Received message number 3 from node C : 00000001
Sending message 7 A : 00000007
Received message number 4 from node C : 00000002
```

```
Sending message 2 C : 00000002
Received message number 2 from node A : 00000007
Sending message 3 C : 00000003
Received message number 3 from node A : 00000008
Sending message 4 C : 00000004
Received message number 4 from node A : 00000009
Sending message 5 C : 00000005
```

Figure 25: Examples of node communication using CAN-BUS.

It is noteworthy that these interactions among nodes will be further enhanced during the second stage of the project.

3.7 User Interface

The user interface is designed as a straightforward mechanism to read from and write data to the microcontroller. It was implemented using a simple PC interface through the ”serial monitor” program, utilizing a character-based protocol. The commands suggested in Tables 1 and 2 of the project statement appendix have been successfully implemented. These include both basic commands and performance-related commands for the specified luminaire. The following figures illustrate some of the possible commands and the system's responses.

Figure 26: Examples of user interface commands and respective response.

It is important to note that, in the initial phase of the project, these commands were implemented for a single luminaire; i.e., the <i> variable must be of the char type representing the microcontroller we are communicating with (can be 'A', 'B', or 'C').

4 - Conclusion

The project's first stage objectives have been met. The individual luminaire controller is fully operational, and the assembly of the distributed system has started. These steps lay the groundwork for the the subsequent project goals.