TÉCNICO LISBOA

Distributed Real Time Control Systems

MEEC

Project Report

Author:

José Pedro da Cunha Rodrigues (113234) jose.p.rodrigues@tecnico.ulisboa.pt

2024/2025 – 2nd Semester, P1

1 Introduction

This project focuses on the development of a distributed real-time control system for smart lighting in an office-like environment. Each luminaire node adjusts its illumination based on desk occupancy and ambient light, aiming to ensure visual comfort while reducing energy consumption. The system uses light sensors, LED drivers, and CAN-BUS communication between nodes.

In stage 1, the focus is on implementing the local control infrastructure: assembling the hardware, calibrating sensors, designing a controller, implementing performance metrics, developing a basic interface and enabling basic communication with other nodes. This stage sets the foundation for the cooperative distributed control to be developed in the second phase of the project.

2 System Overview

The system simulates a smart luminaire controlling the illumination of a desk. It is built around a Raspberry Pi Pico microcontroller, which handles sensing, actuation, control, and communication tasks. An LED acts as the controllable light source, and an LDR (Light Dependent Resistor), integrated in a voltage divider circuit, measures the reflected light from the desk surface. The LED intensity is controlled via PWM (Pulse Width Modulation), and the sensed voltage is converted to illuminance in LUX through a calibration function.

To minimise noise and improve measurement stability, an exponential filter is applied to the sensor readings. Control is executed by a PI controller implemented in C++, featuring anti-windup and feedforward mechanisms. The controller runs periodically at 100 Hz, adjusting the LED duty cycle to keep the illuminance close to a reference value.

The microcontroller also maintains performance metrics, such as energy consumption, flicker, and visibility error, and supports a serial interface for interacting with the system via predefined commands.

In addition, the node is equipped with a CAN-BUS interface (via an MCP2515 driver), allowing it to send or receive messages to/from other luminaires, enabling future cooperation in a distributed network. Communication and control tasks are distributed across the Pico's two cores to ensure real-time performance without blocking critical processes.

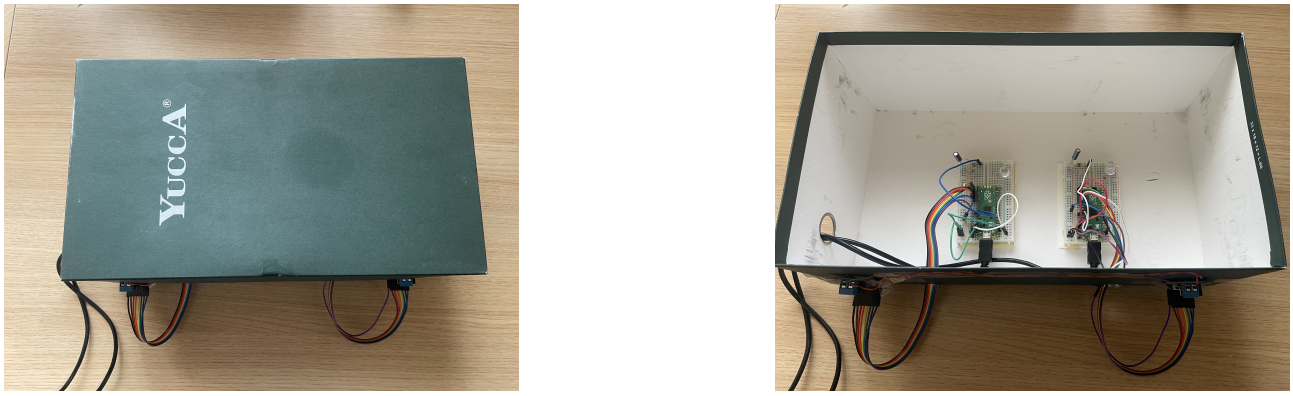
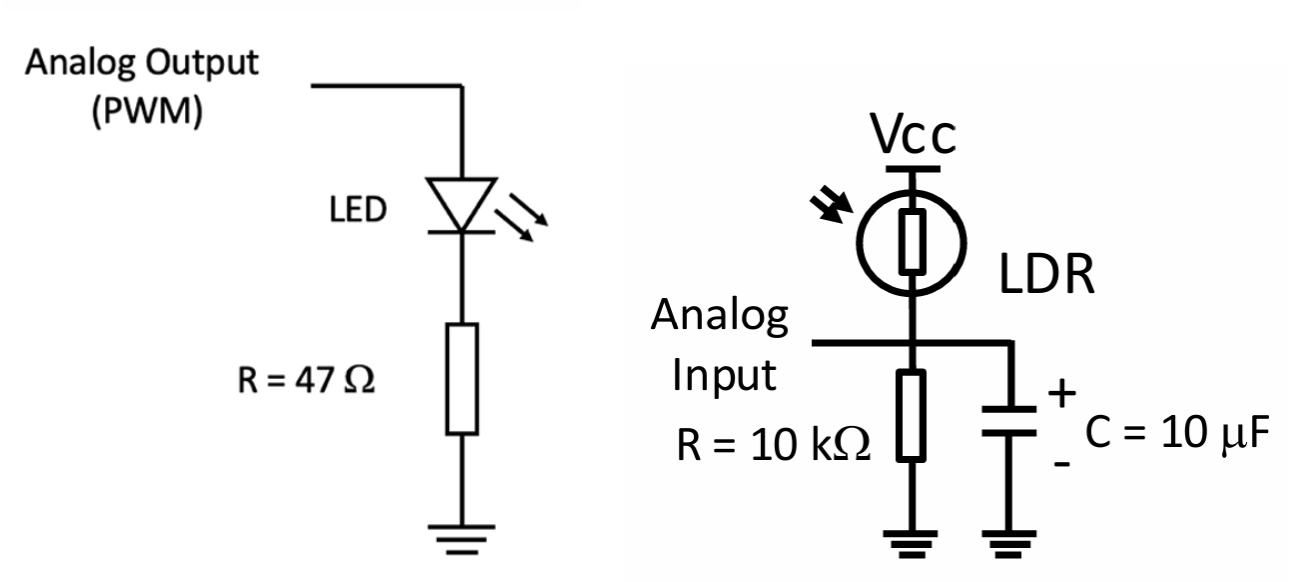


Figure 1: Model of the office

3 System Calibration

To measure the illuminance reflected from the desk surface, the system uses an LDR (Light Dependent Resistor) in a voltage divider configuration. This circuit converts the variation in light intensity into a voltage signal that can be read by the microcontroller's ADC. The LED is driven using PWM, and the LDR responds to the reflected light.



(a) LED actuation circuit **(b)** Luxmeter circuit

Figure 2: Overview of the actuation and sensing circuits used in the luminaire node.

At an early stage of the project, it was necessary to calibrate the sensor system to correctly convert the voltage from the LDR circuit into illuminance values in LUX. For this, a function was implemented to map the analog voltage readings to LUX using the known logarithmic relationship between the LDR resistance and illuminance:

$$\log_{10}(\text{LDR}) = m \cdot \log_{10}(L) + b \tag{1}$$

To express LUX in function of the read voltage input, we can use the voltage divider:

$$\text{LDR} = \frac{R \times (V_{cc} - V_i)}{V_i} \tag{2}$$

$$\text{LUX} = 10^{\left(\frac{\left(\frac{R\times(V_{cc}-V_i)}{V_i}\right)-b}{m}\right)} \tag{3}$$

where $R = 10K$ and $V_{cc} = 3.3V$

To determine m, the datasheet's sensitivity value (slope of the log-log curve) was used. Multiple values of m were tested in the range:

$$m \in [-0.85, -0.70] \tag{4}$$

The value of b was estimated using the nominal resistance at 10 lux (R_{10}), also from the datasheet. We know that

$$\log_{10}(R_{\text{LDR}}) = \log_{10}(R_o) - \gamma \log_{10}(L) \tag{5}$$

, so we can do this

$$\log_{10}(R_{LDR} \times L^{\gamma}) = \log_{10}(R_0) \tag{6}$$

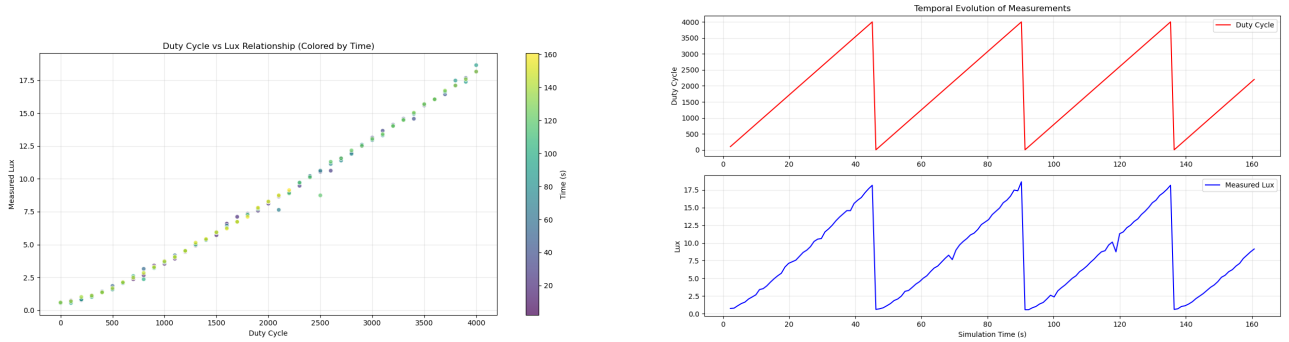
where $b = log_{10}(R_o)$ and $m = -\gamma$. We get this expression:

$$R_{LDR} = R_o \times L^{-\gamma} \tag{7}$$

and then we can calculate our b, after calculating R_o, solving this equation:

$$b = log_{10}(R_o) \tag{8}$$

Once we had a function to compute LUX from voltage, we validated the calibration by plotting illuminance against duty cycle and also visualising LUX and duty cycle over time. This allowed us to verify if the chosen parameters m and b produced a linear and consistent relationship between LED control and measured illumination.



(a) LUX vs Duty Cycle **(b)** LUX and Duty Cycle vs time

Figure 3: LUX and Duty Cycle behaviour

In this plots we can see the linear behaviour between the duty cycle and LUX, so we can conclude that our system is well calibrated.

To estimate the system's static gain G, we initially performed a calibration using **RANSAC** on a scatter plot of LUX vs duty cycle. The illuminance L is related to the control signal u and the external disturbance d through the following equation:

$$L = d + G \cdot u \tag{9}$$

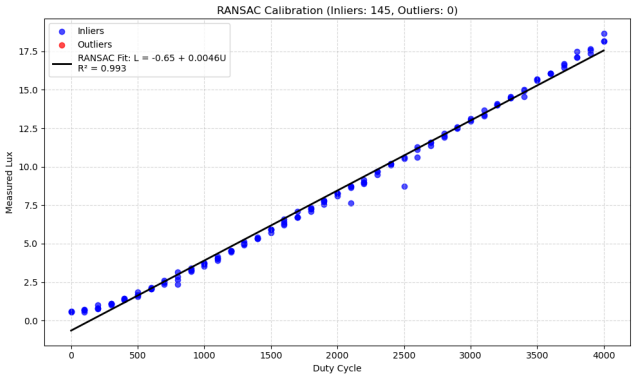


Figure 4: Gain with Ransac

Here we can see that the gain of our box is very low, is always around 0.0046.

RANSAC was applied to this dataset to extract a robust linear fit and estimate the value of G. However, this method was only used at the beginning of the project.

Currently, a simpler and more efficient method is used. Upon each system restart, the gain is recalculated using only two points: the LUX measured at minimum and maximum PWM values (0 and 4095). The gain is then computed as:

$$G = \frac{L_2 - L_1}{4095} \tag{10}$$

where $L1$ is the illuminance at 0 PWM (LED off) and L_2 is the illuminance at full PWM (LED at max brightness).

This method avoids the need for manual calibration or regression each time the system is powered on, while still accounting for variations in component placement or environmental conditions.

4 Local Control

A PI controller was implemented with setpoint weighting, feedforward, and anti-windup mechanisms. The controller runs at a frequency of 100 Hz and adjusts the LED brightness (via PWM) to maintain the desired desk illumination, compensating for external disturbances such as ambient light or reflections. Since the system has no significant inertia or delay, the derivative term is unnecessary and was removed after initial testing.

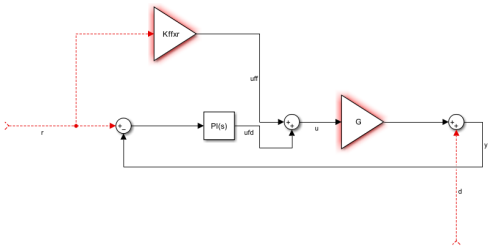


Figure 5: Controller Sketch

Initially, a full PID controller was used, but it exhibited very slow response times and was not effective for the dynamics of our system. After multiple iterations and tests, a PI controller with an added feedforward term was introduced. This updated controller significantly improved response speed and performance.

The Proportional (P) term reacts to the difference between the reference and the measured illuminance, providing an immediate correction based on the current error. The Integral (I) term compensates for steady-state error by accumulating the past error over time, helping the system converge precisely to the desired value. However, due to actuator saturation, an anti-windup mechanism was added to prevent the integral term from accumulating excessively when the actuator is saturated.

A feedforward component was also implemented, which computes a control signal directly from the desired reference, independent of the feedback loop. This helps the controller reach the target more quickly, especially when there's a sudden change in reference. The presence of feedforward is a key factor in the improved performance of the final controller.

In addition, a bumpless transfer mechanism ensures smooth transitions when switching between manual and automatic modes, avoiding abrupt jumps in the control signal that could destabilize the system.

The control signal u is composed of a **feedback** component and a **feedforward** component, represented respectively by u_{fd} and u_{ff}. The complete control law is given by:

$$u = u_{fd} + u_{ff} = P + I + K_{FF} \cdot r \tag{11}$$

The proportional term is defined as:

$$P = K \cdot (b \cdot r - y) \tag{12}$$

where K is the proportional gain, r is the reference illuminance, y is the measured illuminance, and b is the setpoint weighting factor. The integral term I accumulates the control error over time and is updated at each sampling instant as:

$$I = I + b_i \cdot (r - y) + a_o \cdot (u - v) \tag{13}$$

In this expression, $b_i = \frac{K \cdot h}{T_i}$ is the integral coefficient, and $a_o = \frac{h}{T_t}$ is the anti-windup coefficient. The term v represents the unsaturated control output:

$$v = P + I + K_{FF} \cdot r \tag{14}$$

The final output u is obtained by saturating v within the allowed PWM range (0 to 4095). The anti-windup mechanism limits the influence of actuator saturation by applying a corrective term $a_o \cdot (u - v)$ to the integrator.

Finally, to ensure smooth transitions when switching between control modes, a **bumpless transfer** mechanism is used:

$$I_{new} = I_{old} + K_{old} \cdot (b_{old} \cdot r - y) - K_{new} \cdot (b_{new} \cdot r - y) \tag{15}$$

After implementing the feedforward term, it was necessary to calculate its gain. We now that:

$$u = u_{fd} + u_{ff} \tag{16}$$

and

$$y = d + Gu \tag{17}$$

substituting u we get this:

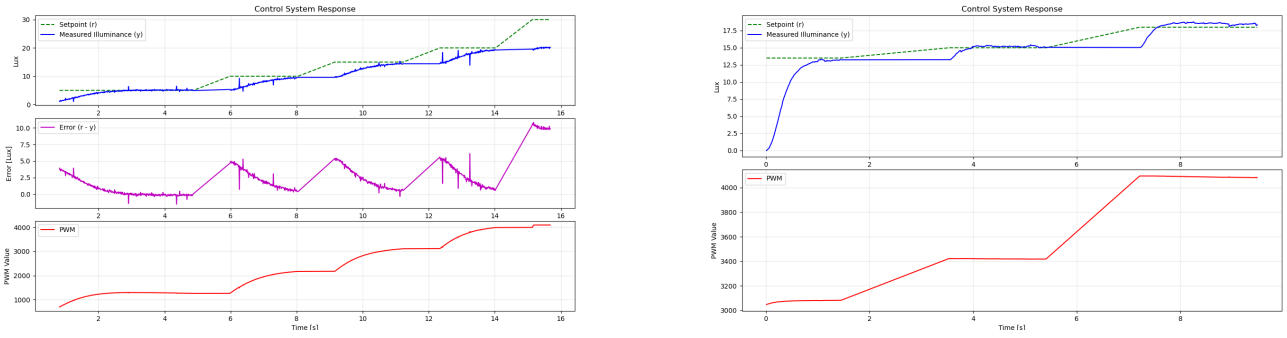
$$y = d + G(u_{fd} + u_{ff}) \tag{18}$$

we know u_{fd} and u_{ff} , so lets solve this equation:

$$\begin{aligned}
&\Leftrightarrow y = d + G(P + I + K_{FF} \cdot r) \\
&\Leftrightarrow y = d + G\left[K(b \cdot r - y) + K_{old} \cdot (b_{old} \cdot r - y) - K \cdot (b \cdot r - y) + K_{FF} \cdot r\right] \\
&\Leftrightarrow y - d = G\left[Kbr - Ky - Kbr + Ky + K_{FF}r\right] \\
&\Leftrightarrow y - d = GK_{FF}r \\
&\Leftrightarrow \frac{y - d}{Gr} = K_{FF} \\
&\Leftrightarrow K_{FF} = \frac{1}{G}
\end{aligned}$$

Note: In the final step, we replace y with r because the control objective is to make the system output track the reference, i.e., $y = r$. The disturbance term d is assumed to be approximately zero and is therefore neglected. The terms K_{old} and b_{old} disappear from the equation because they are both initialized to zero. As a result, their contribution to the expression is null during the first control cycle. This expression corresponds to the bumpless transfer mechanism, which ensures smooth transitions in controller activation. After the first cycle, K_{old} and b_{old} are updated to match the current values of K and b, meaning the bumpless term will evaluate to zero in all subsequent iterations — as intended.

To highlight the improvement, the following figures show a comparison between the initial PID controller and the final PI + feedforward controller. As seen in the plots, the new controller reacts faster and more effectively to changes in reference, reducing both delay and overshoot.

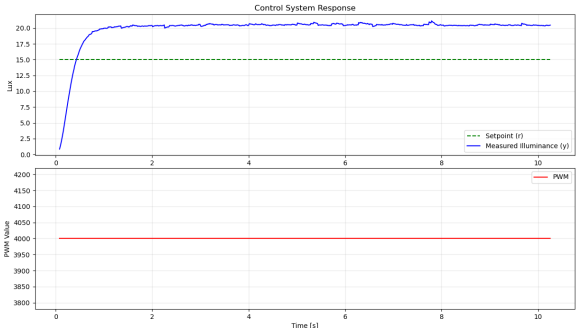


(a) Initial PID controller response

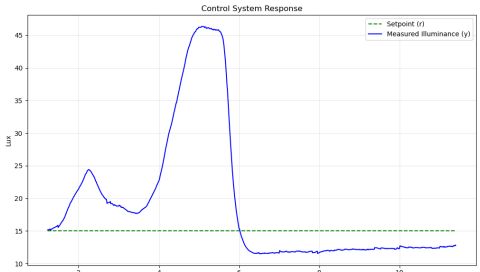
(b) Improved PI controller with feedforward

Figure 6: Comparison between the initial PID controller and the final PI + feedforward controller.

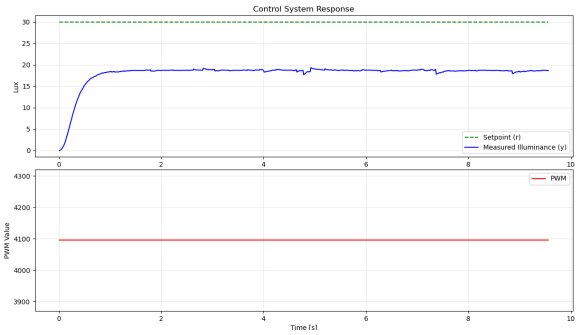
To better demonstrate the performance and robustness of the controller under different configurations and conditions, we present the following plots:



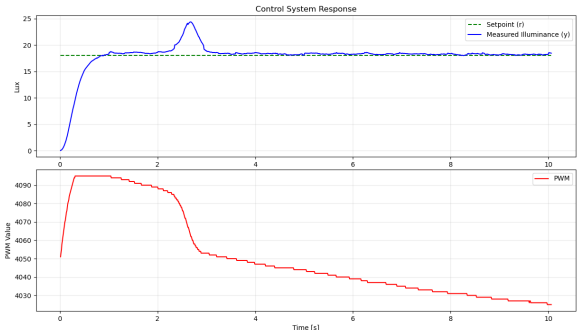
(a) Controller without feedback



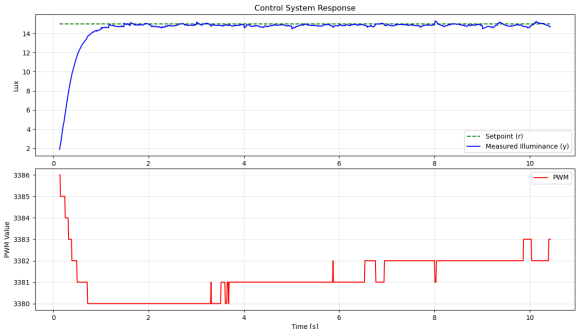
(b) Controller without anti-windup



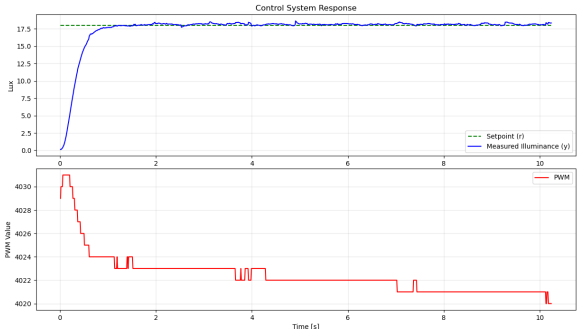
(c) Controller under saturation



(d) Controller with external disturbance



(e) Full controller — reference 15 LUX



(f) Full controller — reference 18 LUX

Figure 7: Controller behaviour under various configurations and conditions.

From these results, we can observe the importance of each controller component.

In the absence of the feedback term, the controller will not change the value of the control signal to reach the reference

When the anti-windup mechanism is disabled, the system may overshoot significantly after exiting saturation, since the integrator continues to accumulate error during the saturated state. This leads to instability or prolonged settling times.

The values choosed for the controller were:

- K = 1
- b = 0.3
- $T_i = 0.08$ and $T_t = \sqrt{T_i}$

4.1 Exponential Filter for Sensor Readings

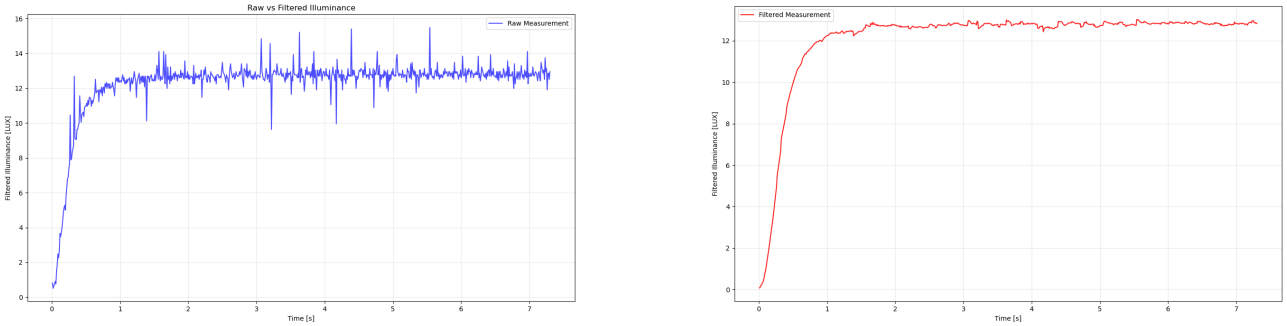
To reduce the noise present in the LUX measurements obtained from the LDR, an exponential moving average filter was applied to the raw sensor readings. This filter smooths out sudden variations in the signal caused by electrical noise or small environmental changes, resulting in a more stable input for the controller.

The filter is applied recursively at each control cycle, and the filtered value is computed using the following formula:

$$y_{filtered} = \text{ALPHA} \cdot y_{\text{raw}} + (1 - \text{ALPHA}) \cdot y_{\text{filtered}} \tag{19}$$

In this equation, y_{raw} represents the unfiltered illuminance measurement obtained from the sensor, and y_{filtered} is the smoothed result. The constant `ALPHA` defines the smoothing factor. `ALPHA` value is 0.1.

The following plots show a comparison between the raw LUX values and the filtered signal, highlighting the effect of the exponential filter:



(a) Raw LUX measurements (no filter)

(b) Filtered LUX measurements

Figure 8: Effect of the exponential filter on LUX sensor readings.

4.2 Jitter Analysis

In real-time embedded systems, especially those controlling physical processes, maintaining consistent timing between control cycles is crucial. The variation in the time interval between consecutive control executions is known as **jitter**.

In our implementation, the controller is scheduled to run every 10 milliseconds (100 Hz). To monitor jitter, we measured the actual interval between two consecutive cycles and compared it to the ideal value $T_s = 10\,\text{ms}$. The jitter at each cycle J_k is calculated as:

$$J_k = |(t_k - t_{k-1}) - T_s| \tag{20}$$

where t_k is the current timestamp (in milliseconds) and t_{k-1} is the timestamp of the previous control cycle.

The following plot shows the jitter measured over time. These measurements confirm that the variation remains minimal and within acceptable limits for real-time control.

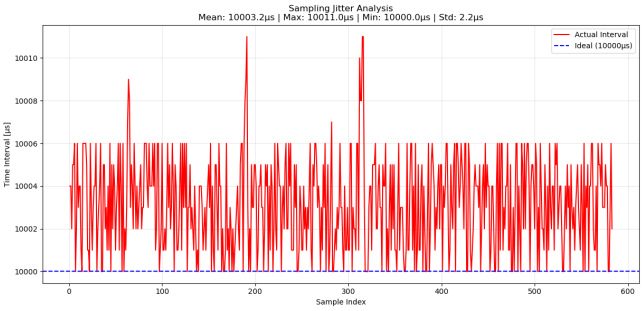


Figure 9: Jitter measured between control cycles during normal operation.

The sampling rate as a standard deviation from the desired one of $2,2\ \mu$s

5 Performance Metrics

To evaluate the performance of our control system, three key metrics were implemented and computed in real-time: energy consumption, visibility error, and flicker. These metrics allow us to quantitatively assess how well the system satisfies the two main objectives: energy efficiency and user comfort.

Before computing the energy metric, we first determined the maximum electrical power consumed by the LED. This was done experimentally using a multimeter to measure the voltage across the LED and the resistance when driven at full PWM.

$$P_{\max} = V_{\text{LED}} \cdot I_{\text{LED}} = V_{\text{LED}} \cdot \frac{V_R}{R} \tag{21}$$

Substituting the measured values, where $V_{\text{LED}} = 2.64$ V, $V_R = 0.3$ V, and $R = 47\ \Omega$, we obtain:

$$P_{\max} = 2.64 \cdot \frac{0.3}{47} \approx 0.00638 \text{ W} \tag{22}$$

This measurement was taken with the LED operating at maximum PWM (4095), ensuring the maximum current flow for accurate estimation.

The **energy consumption** is computed by integrating the instantaneous power over time. Since the PWM duty cycle is proportional to the LED power output, and having a maximum power $P_{\max}$, the energy consumed up to instant t_N is given by:

$$E = P_{\max} \sum_{k=1}^{N} d_{k-1} \cdot (t_k - t_{k-1}) \tag{23}$$

where d_{k-1} is the normalized duty cycle at sample $k-1$, and t_{k-1} is the corresponding timestamp in seconds.

The **visibility error** penalizes situations where the illuminance at the desk is below the reference required by the occupation state. This is computed as the average deficit of light whenever $y(t_k) < L(t_k)$:

$$V = \frac{1}{N} \sum_{k=1}^{N} \max(0, L(t_k) - y(t_k)) \tag{24}$$

where $L(t_k)$ is the reference illuminance and $y(t_k)$ is the measured illuminance.

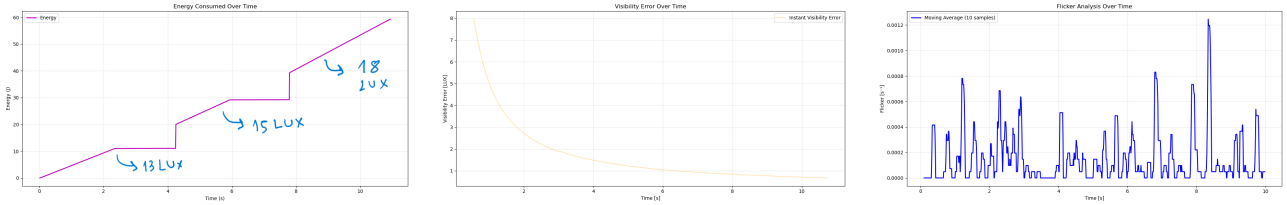
The **flicker** metric captures high-frequency fluctuations in the control signal that can be uncomfortable for users. It is computed by summing the magnitude of second differences in the duty cycle when a sign change in its derivative is detected:

$$f_k = \begin{cases} |d_k - d_{k-1}| + |d_{k-1} - d_{k-2}|, & \text{if } (d_k - d_{k-1})(d_{k-1} - d_{k-2}) < 0 \\ 0, & \text{otherwise} \end{cases} \tag{25}$$

The average flicker is then given by:

$$F = \frac{1}{N} \sum_{k=1}^{N} f_k \tag{26}$$

These metrics are computed at each control cycle (100 Hz sampling rate).



(a) Energy consumption **(b)** Visibility error **(c)** Flicker

Figure 10: Performance metrics computed over time: energy consumption, visibility error, and flicker.

6 User Interface Commands

To allow user interaction with the system in real time, a serial interface was implemented. This interface enables the user to send commands via USB serial monitor, and receive system responses or data streams.

The interface supports a wide range of commands, allowing control and monitoring of different system variables. These include setting and getting the LED control signal, reference illuminance, occupancy status, anti-windup and feedback modes, and retrieving sensor values such as illuminance, voltage, disturbance, and power. Additionally, users can stream live data from the system or request buffered historical data from the last minute. Commands follow a clear and consistent syntax, typically starting with an action identifier (e.g., ‘g‘ for get, ‘s‘ for start stream, ‘u‘ for set PWM) followed by the variable name and the node index.

The following figure shows an example of the interface in use, demonstrating its correct and responsive behaviour.

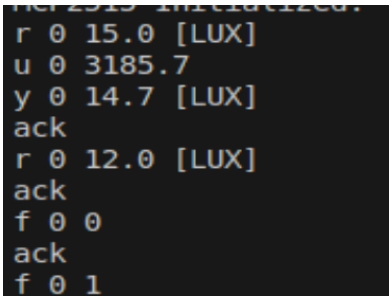


Figure 11: Serial interface responding to user commands in real time.

7 CAN-BUS Communication

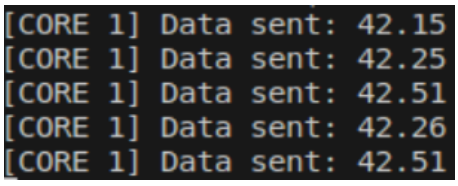
To enable communication between different luminaire nodes in the system, a CAN-BUS interface was implemented using the MCP2515 controller. This allows nodes to exchange data in real time, which will be essential in later stages for cooperative control strategies.

In the current setup, two Raspberry Pi Pico boards were used to simulate two independent nodes. On **Node 1**, the main controller, the CAN-BUS communication is managed in parallel using the multicore capabilities of the microcontroller. The program is split across the two cores as follows:

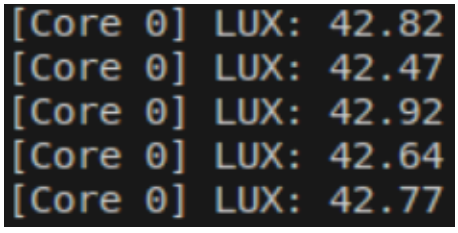
- **Core 0** is responsible for executing the control loop, handling serial commands, computing metrics, and storing data.
- **Core 1** periodically sends CAN messages containing the measured illuminance (LUX) value.

This separation of tasks ensures that time-critical control operations are not delayed by communication routines, effectively avoiding resource conflicts and jitter. Messages are sent every second using a dedicated function launched via `multicore_launch_core1()`.

On **Node 2**, the setup is simpler: it runs on a single core and is dedicated to receiving messages via CAN-BUS. Upon receiving a message with the expected CAN ID, the node decodes the LUX value and prints it directly to its serial terminal.



(a) Node 1 terminal – sending LUX values



(b) Node 2 terminal – receiving LUX values

Figure 12: CAN-BUS communication between two nodes: Node 1 sends LUX values from Core 1, and Node 2 receives and prints them.

8 Timing Analysis of System Components

To assess the efficiency and real-time performance of the system, the execution time of the main components was measured throughout operation. Specifically, we calculated the average time taken by the following:

- Control computation loop (100 Hz)
- Serial communication handling
- CAN-BUS message transmission

The following graphic shows the relative execution time of each component:

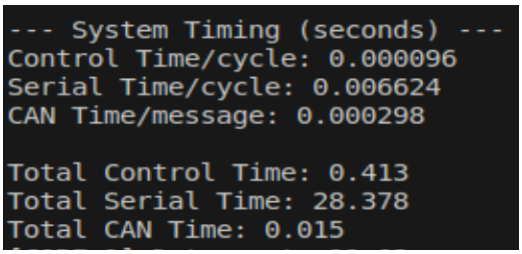


Figure 13: Execution time of control, serial, and CAN operations.

9 Conclusion

In this first stage of the project, we successfully implemented a fully functional local control system for a smart luminaire. The system integrates real-time control, sensor calibration, performance monitoring, a user interface, and CAN-BUS communication. All components were tested and validated under different scenarios.

We fulfilled the objectives defined for Stage 1 of the project, ensuring that the luminaire operates autonomously with stable and efficient control, while laying the groundwork for distributed and cooperative functionality to be addressed in the next phase.