Distributed Illumination Control

João Santos 79036 —— Tiago Dias 78435 —— Nuno Brandão 85232

***Abstract*— This project aims to control a small-scale model of an office space (3-LED illumination system) in a distributed manner, maximizing the user (desk) comfort and minimizing the energy spent. A consensus algorithm was applied to a control system, in which an agent network finds a consensus about the optimal solution, trying to respect the system's reference (input). If those references are not achievable, the network should be able to determine the closest best solution. We could not implement some parts of the project: the PC application and the energy consumption control. However, we did manage to obtain the illuminance distributed control which was the main goal.**

***Index Terms*—**

I. INTRODUCTION

The last decade brought a new way of lighting - LEDs (Light Emitting Diodes). They outlast conventional light bulbs and have a much lower energy consumption. Besides, LED's have no danger associated with handling mercury and other metals. They also allow for an easier integration with sensors, computers, controllers and all kinds of other electronics. With that said, there are still some challenges with LED's regarding optimization.

This project aims to tackle those challenges. In order to do so, the system is defined as an agent network in which each agent tries to respect minimum specifications, taking into account its neighbors' interference and trying to achieve the best global situation. Such compromise is managed with a communication system. This network could be centralized, but it would take more processing (and hardware) power since centralized control needs to make all operations in the same place. It would also make the system's response slower overall.

The main goal is to simulate an office floor, where minimum illuminance levels must be guaranteed at each desk. The desk's occupancy is also taken into account, meaning it is not necessary to illuminate an empty desk. Of course, day/night cycles (outside interference) are a factor to consider as well as the other desk's light interferences.

Everything was implemented through a distributed control system with a coordinated agent network (based on the consensus algorithm). The system can be monitored and changes can be made by the user, regarding illuminance levels and interferences. All the code developed is available in the delivered folder.

II. PROBLEM DEFINITION

To simulate the office floor, 3 luminaires (LED's) were set up inside a shoe box. This box was "coated" with white paper on the inside to maximize reflection, and cuts were made to simulate an open/closed window. Each luminaire is attached to a PCB (printed circuit board). Each PCB also has an LDR (Light Detection Resistor) attached, in order to monitor our illuminance levels reflected. The 3 PCBs simulate the 3 desks. In figure 1 is shown a picture of such setup and figure 2 shows the electrical configuration (connections) in each PCB. 2 resistors (100Ω and $10k\Omega$) and 1 capacitor were used per PCB (besides the LED and LDR).

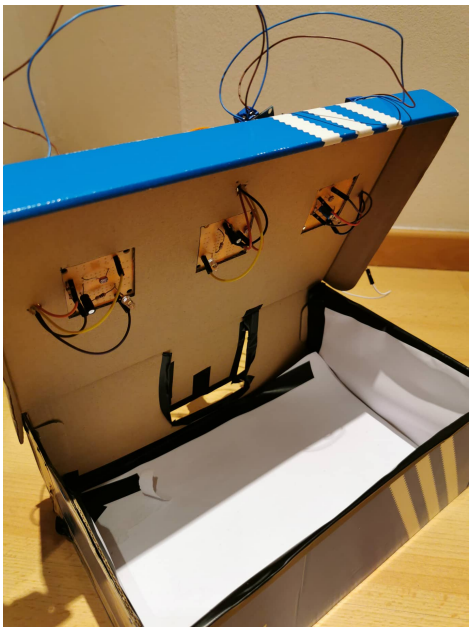

Fig. 1: Luminaire setup

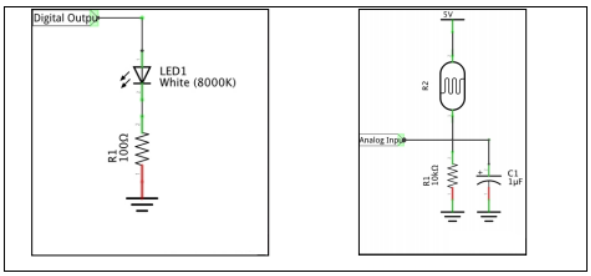

Fig. 2: LED driving circuit (left) and LDR Illuminance Reading circuit

One arduino Uno is connected to each one of the PCB's and all 3 arduinos are connected through a CAN-Bus setup, which is shown in figure 3.

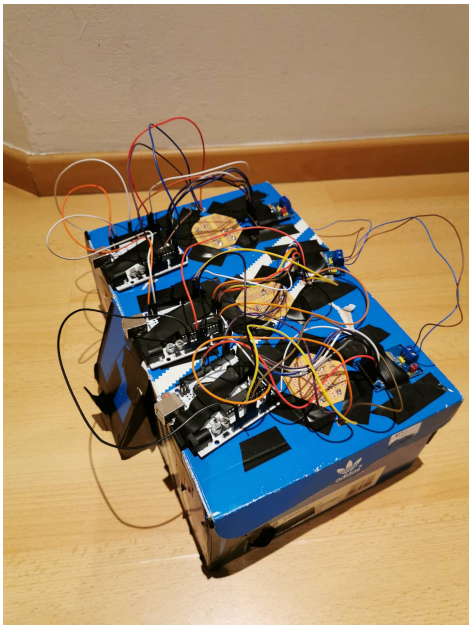

Fig. 3: Arduino communication setup

The LDR is a resistor that changes its resistance values according to the illuminance level, according to figure 4.

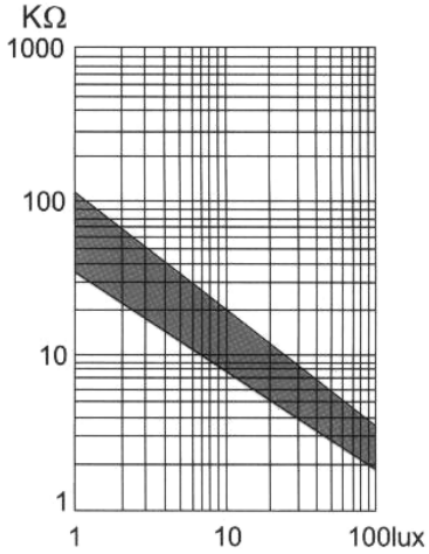

Fig. 4: Resistance values vs Lux values (log scale)

The LED's output illuminance level is determined by it's duty cycle - the bigger the duty cycle, the higher the illuminance level (Lux). The duty cycle levels vary from 0 (min) to 255 (max), and are set by the arduino's PWM (Pulse Width Modulation).

For this problem, a consistent way of correlating the variables [*duty cycle*] and [*illuminance*] is needed, so the following is obtained:

$$[Lux] = K * [DutyCycle]. \quad (1)$$

So, in other words, K must be determined, so illuminance levels are set by adjusting duty cycle levels.

The comunication system between arduinos, with the consensus algorithm alongside individual feedback control for stability, simulates a smart illuminance system for an office space according to illuminance criteria set by the user.

In this case, the illumination system can be seen as a constrained optimization problem, since the illuminance value of each desk is influenced not only by the duty cycle of the LED above it, but also by the illuminance values of the other desks and external interferences. Given the set of variables:

- i - index of the desk/node,
- l_i - illuminance at desk i,
- L_i - lower bound of illuminance at desk i,
- d_i - dimming level (duty cycle) of each LED (mapped from 0 to 100),
- K_{ij} - influence of the illuminance of led j in the irradiance at desk i,
- o_i - effect of external illumination at desk i,
- n - number of nodes,
- c_i - energy "cost" at desk i,

and the objective of minimizing all d_i while guaranteing that all l_i are equal or greater than L_i, this optimization problem can be exposed as:

$$\begin{aligned} \underset{d_1,\dots,d_n}{min} \quad & f(d_1,...,d_n) \\ \text{s.t.} \quad & 0 \leq d_i \leq 100 \\ & l_i = \sum_{j=1}^{n} d_j K_{ij} + o_i \geq L_i, \forall i \end{aligned},$$

with

$$f(d_1,...,d_n) = c_1 d_1 + ... + c_n d_n.$$

So, in this case, all K_{ij} and o_i values must be determined.

III. Proposed Approach

As mentioned, the problem addressed was solved using a decentralized control system in which each arduino acts as a local controller to each luminaire, in order to achieve a global goal. Two different approaches were used: co-operative and non-cooperative control. Considering the set objective of energy minimization for the total system, it was necessary to implement a cooperative mode. Regarding the non-cooperative solution, it was implemented as a means of canceling fluctuations around the first approach solution.

Due to decentralization of the control system, communications had to be implemented to make sure the arduinos shared information, which was critical to achieve the goal. The protocol used between arduinos was CAN-Bus, which uses serial asynchronous communication, offers bit rate up to 1 Mb/s, and gives the possibility of filtering received messages (through buffer filters) in order to decide if they're important to this arduino or not. This protocol also allows for communications without a master or a slave, which is

ideal for decentralized system. This communication protocol is made connecting 7 wires from each CAN-Bus device to each arduino, and 2 wires both between CAN-Bus's and Arduinos, according to figure 5.

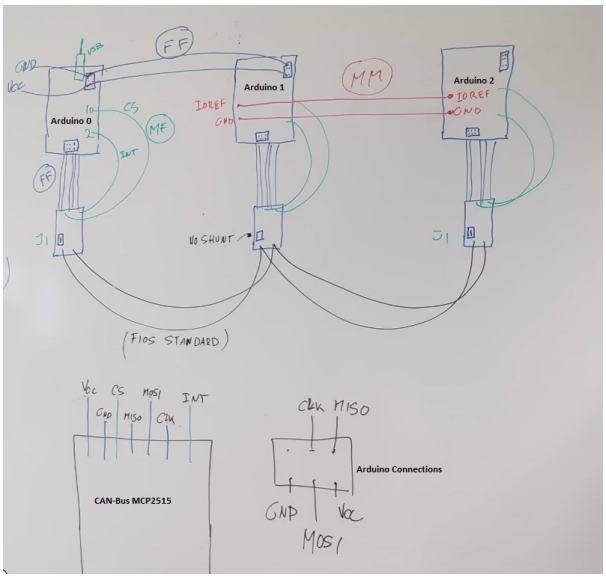

Fig. 5: Serial CAN-Bus Arduino communication configuration

Regarding cooperative control, it was demonstrated that this system can initially be formulated as a constrained global optimization problem. This problem can be manipulated and transformed in an equivalent optimization problem with a quadratic function and a convex set of constraints, as will be discussed later. Besides, it is possible to decompose this problem so that each arduino performs individual computations to reach the global solution. As such, the consensus algorithm becomes a possible and desirable implementation. This algorithm provides a distributed way to compute the solution by the different luminaires so that each one has to respect a smaller set of constraints by intruducing a new global variable that all of them have to "agree". For each variable to be determined (the ducty cycles to apply), each luminaire saves two different values. One which results from its computations and another one that is the average of those computations of all the system's luminaires. The algorithm tries to achieve the minimization solution while converging these two values. If the problem is feasible, it guarantees that all luminaires' variables converge to the same value.

The non-cooperative control system is implemented as a PID control, that follows the reference given by the consensus algorithm. The PID acts in order to achieve its own objective, and thus making the system resistant to external disturbances. Note that the reference that the PID controller follows already takes into consideration the gains K_{ij} regarding other arduinos duty cycle to each LED.

IV. Development

A simplified diagram of the project is presented in figure 6. A detailed discussion of its modules is presented in the next sections, starting from isolated components to the fully integrated form.

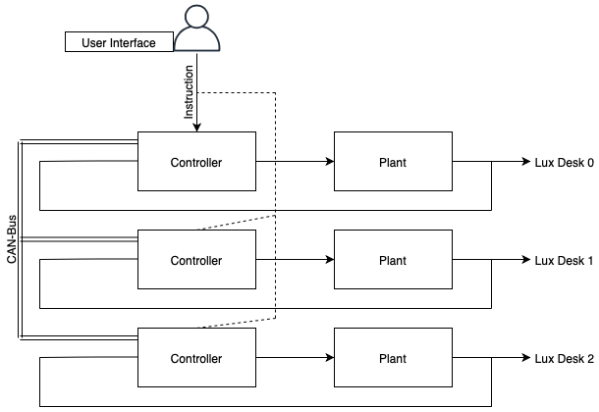

Fig. 6: Overall System Diagram

A. Plant - The luminaire: construction and modeling

The basic functionality of each luminaire is to ensure an illuminance threshold by selecting adequate duty cycles for its LED. However, this illuminance is not read directly from an arduino port. All measurements from the arduinos are digital scaled voltages (between 0 and 1023), therefore, there is a need to determine the relations between the variables of interest and the available measurements.

Let us first consider the LDR component. Figure 7 shows the LDR circuit in analysis:

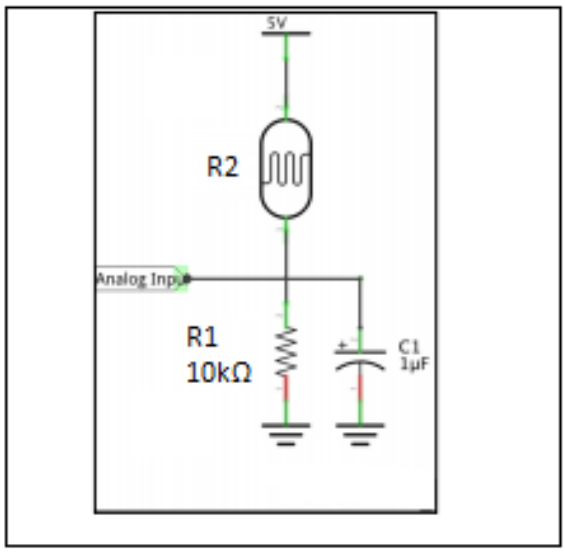

Fig. 7: LDR Illuminance Reading circuit

As previously said, it is possible to calculate the desk's illuminance knowing the LDR resistance value. Analysing figure 4, it is now possible to calculate such resistance considering the voltage divider in steady state:

$$U = 5\frac{R_1}{R_1 + R_2}. \tag{2}$$

The value of U is read by the arduino as a digital voltage and the resistor R_1 is known ($10k\omega$), which leads to

$$R_2 = \frac{5R_1}{U} - R_1. \tag{3}$$

The only thing left is to determine the parameters that characterise the LDR properties expressed in figure 4. The illuminance values can be linearized in the form $y = mx+b$, in which y is the LDR resistance value and x is the illuminance. Applying the logarithmic scale to the equation, we obtain:

$$log_{10}(R) = mlog_{10}(L) + b, \tag{4}$$

which results in

$$L = 10^{\frac{log_{10}(R)-b)}{m}}. \tag{5}$$

Now that there is a clear relation between illuminance and resistance values, the calibration process starts, with the goal of determining values for m and b; b being the illuminance value at PMW = 0.

We put a single luminarie inside a closed box, set the PWM duty cycle to a value and record the LDR voltage that is read after a certain time to ignore the transient state. This process is repeated for duty cycles varying from 0 (minimum value) to 255 (maximum value) with increments of 5. This was done for each luminaires.

Since the objective is to have the illuminance values expressed by $L = K*DutyCycle$, the only relation between them that is acceptable is a linear one. A matlab algorithm was made to test several m and b values, using the Least Square method, allowing it to find the values of m and b that make the relation between Lux and duty cycle the "most linear". It was observed that this linearity depended only on the m parameter. The other one only influenced the gain K. As so, the later parameter was selected such that the three LDR read equal values for equal duty cycles and that for the maximum duty cycle each one read an illuminance of aproximatelly 500 Lux.

With the optimal values of m and b found, we obtained the gain between illuminance and duty cycle K. The matlab algorithm was run through for each of the other LDR. Figure 8 shows the best regression for one of the luminaires, which resulted in K=1.94.

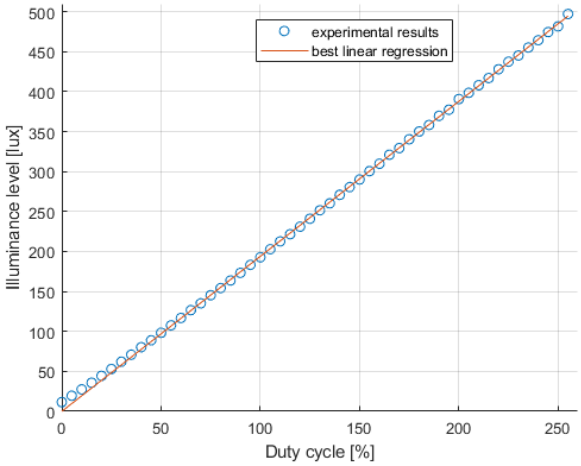

Fig. 8: Illuminance read by one of the luminaires as a function of the PWM duty cycles, for m = -0.630 and b = 2.3150

On a first approach, the information gathered up to now should be sufficient to design a controller that met the requirements. Only the steady state is being considered, but since the transient state is fast compared to the human perception and the system is projected to work over long and continuous periods without sudden variations, there is no major problem. However, for a more rigorous approach, the transient state must be studied. For this, the necessity for changing the system's model arises.

Analysing figure 7, being x the illuminance value obtained in the static state ($Lux = K * DutyCycle$). Now, taking into account the transient state:

$$\dot{v}\tau(x) = -v + V_{cc}\frac{R_1}{R_1 + R_2(x)}, \tag{6}$$

where v is the LDR's voltage, while $\dot{v}$ is its variation in time. τ is the time constant (s) relative to this system, which is obtained from:

$$\tau(x) = R_{eq}(x)C_1, \tag{7}$$

C_1 being the capacitor's capacitance and

$$R_{eq} = \frac{R_1R_2(x)}{R_1 + R_2(x)}, \tag{8}$$

is the circuit's equivalent resistance value, where R_2 is the resistance value of the LDR, described by:

$$R_2(x) = 10^{mlog_{10}(x)+b}. \tag{9}$$

The value of τ was obtained through tests, running an algorithm through the arduino that increased the duty cycle from 10 in steps of 30, while reading the LDR voltage value "continuously". Once it reached a duty cycle of 250, the same steps were decreased with the same frequency, untill it reached the initial duty cycle. A more detailed analysis on how the starting and ending duty cycles affect the time constant can be found in the appendices.

The data was exported and processed through a matlab algorithm, producing this graph of voltage vs time (s), shown in figure 9. Note that the first and last dot (going left to right) in the graph is not considered to the calculations, despite being shown in the graph (dots were implemented as formulas for all intervals in order to illustrate the points used to calculate τ).

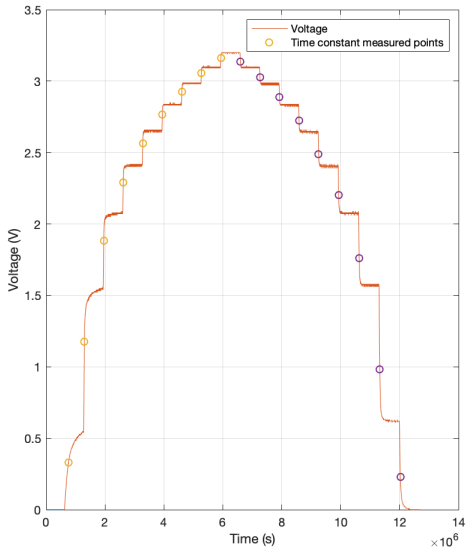

Fig. 9: Voltage of the LDR vs time (step analysis)

From the data structure the value of τ is found:

- For the positive steps, by selecting the time instant that corresponds to $0.66(V_{final} - V_{initial})$ in each step,
- For the negative steps, by selecting the time instant corresponding to $V_{initia} - 0.66(V_{initial} - V_{final})$ in each step,

being $V_{initial}$ and V_{final} the voltage values in the beggining of the step and at the end, respectively.

Those time instants (τ) are then compared to their corresponding illuminance values (obtained through the voltage value), and the following graph is made - figure 10, showing the evolution of τ along the illuminance values.

With the experimental values (red dots) of τ obtained, it's possible to approximate the τ vs Lux relation to equation 6, as the blue line seen in figure 10. The approximation obtained is defined by:

$$\tau(x) = \frac{A \cdot 10^{mlog_{10}(x)+b} \cdot 10^3}{B + 10^{mlog_{10}(x)+b} \cdot 10^3}, \tag{10}$$

where m and b are already known, and:

- $A = 0.0435$,
- $B = 17360.74$.

The multiplication by 10^3 on the numerator and denominator are due to unit adjustments relating resistance values.

There is also a delay associated with the transient state that needs to be included in the model, which is essencially

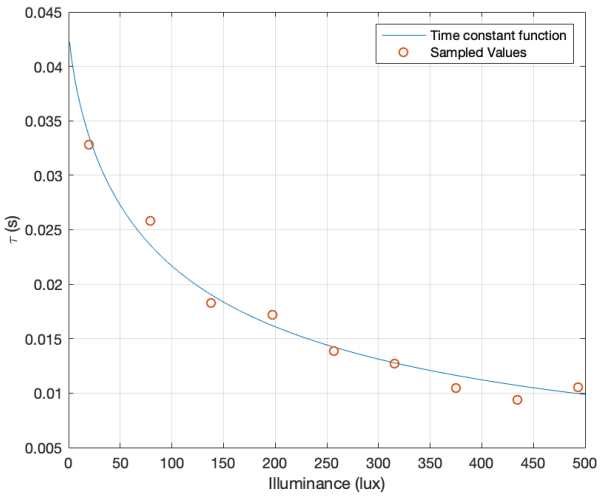

Fig. 10: τ vs Lux Values

the time it takes for the LDR to "enter" the transient state, in other words, the time it takes for the LDR to "react". The illuminance system model for transient state should be in the form:

$$y(t) = f(v(t - \theta)), \tag{11}$$

y being the illuminance according to the model, which is given by a function of voltage(v) through time.

A test was compiled to detect how much time it would take in transient state for the LDR voltage value to change. Once the delay θ was known, it was added to the expression 10, computing:

$$\tau(x) = \frac{A \cdot 10^{mlog_{10}(x)+b} \cdot 10^3}{B + 10^{mlog_{10}(x)+b} \cdot 10^3} - \theta, \tag{12}$$

with a delay $\theta = 660\mu s$

The approximated model obtained can thus be used to further approximate our system's response to the real one.

B. The local controller

This is the feedback control system implemented in which both the PID controller and the reference value are controlled by the arduino. The reference value is the feedfoward's output. The PID controller was implemented in C++ enabling duty cycle changes to correct any overshoot or any outside disturbances. Figure 11 shows the feedback control scheme.

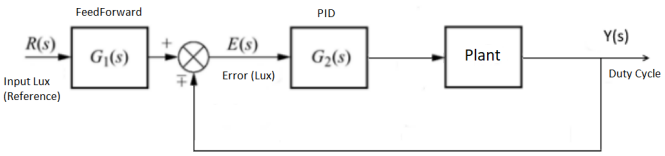

Fig. 11: Feedback control scheme

Figure 12 shows the PID controller configuration scheme.

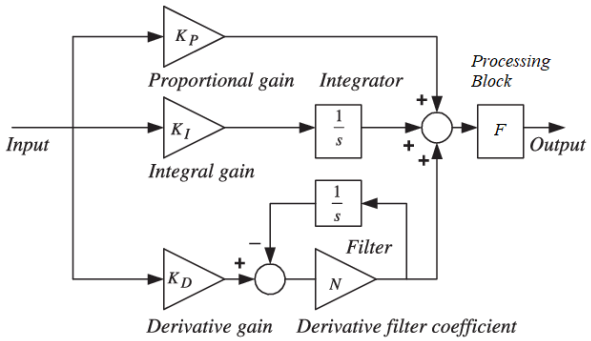

Fig. 12: PID controller scheme

The PID algorithm was implemented as a class in c++. It receives as input the current duty cycle of the LED, reference lux and measured lux, and calculates the proportional, integral and derivative contributions. Then, the processing block will convert the sum of the contributions to duty cycle units, and add or subtract it to the current duty cycle. An anti wind-up and saturation function was then made in c++, that limits the duty cycle to $[0, 255]$ and resets the integral contribution whenever saturation occurs or the integrator term overpasses a limit (established hardcoded).

The gains K_p, K_i and K_d were manually tuned, until a reasonable performance was found.

C. The cooperative controller

As explained previously, the cooperative controller was designed using the consensus algorithm. A detailed explanation on this method can be consulted [4] if needed as this article will only focus on the fundamental aspects that allow the understanding of the discussed topics. In terms of implementation in the project, all variables and computation explained from now are condensed in a C++ class.

The original problem is manipulated and decomposed so that each luminaire only has to focus on its local properties/variables:

- "The local illuminance lower bound (occupancy state) L_i.
- The local external illuminance influence o_i
- The local cost c_i.
- The local actuator bounds (in our case are all the same, 0 and 100).
- The coupling gains from the other luminaires to itself, k_i.
- The global optimization parameter ρ."[4]

Firstly, the problem is decomposed so that there are three different cost functions, one per module. Then, it is used the *ADMM* (alternated direction method of multipliers) algorithm, that forms an Augmented Lagrangean that can also be decomposed in individual terms. Three iterative actions are required to solve the minimization problem:

- the computation of the primal (original) variables to determine, the duty cycles (based on said Lagrangean),
- the computation of the average of those duty cycles of all luminaires,
- the computation of the dual variables (Lagrange multipliers).

In between, it is also necessary to share those duty cycles among the luminaires. As so, a fourth action is required after the first one to ensure the correct transmission of information and its synchronization.

The first computation leads to another formulation, this time to an optimization problem with a quadratic function and the same convex set of constraints. Being a convex quadratic form, if the solution is in the interior of the set, it is guaranteed to be the optimal one. If not, the solution has to be in the boundary. This last situation makes the problem a quadratic program with equality constraints and has five possible solutions. When this happens, it is necessary to determine the cost associated with each one to select the best one.

In the C++ class, that calculation can be made calling a function. It first checks for the solution inside the interior of the set and only checks the other five if it is infeasible.

After choosing the best duty cycles, each luminaire shares them with rest so that it is possible to calculate the averages.

Finally, the dual variables are updated and a new iteration begins. A maximum of 50 iterations is set since it is enough for convergence to happen.

All this process is expressed in the fluxogram of figure 13, in which additional communication protocols are implemented to guarantee synchronism.

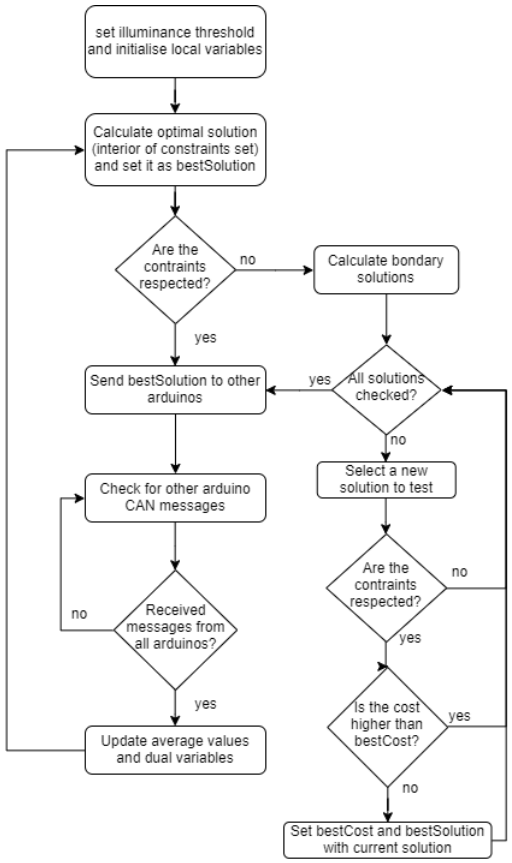

Fig. 13: Fluxogram Consensus

Each luminaire sends three messages via CAN-Bus, one duty cycle at a time. Therefore, each one waits for 6 messages in total before stepping into the next iteration.

D. Initialization and calibration

1) Initialization

During the design of each controller, a sampling rate of $100Hz$ was imposed, using arduino $TIMER1$ interrupts. When an interrupt is triggered, a flag is used to order the system to sample and control. This way, the jitter effect was surpassed. An initialization of $TIMER1$ interrupt had to be done. To do so, the following arduino registers were updated:

- $OCR1A$ (Output compare register A for timer 1) - The interrupt will trigger when $TIMER1$ reaches $OCR1A$ count.
- $TCCR1B$ (Timer 1 counter control register) - Sets operation mode to *Clear on Compare*, and sets prescaler to maximmum in order to minimize flickering.
- $TIMSK1$ (Timer 1 interrupt mask register) - Enables timer1 to compare interrupts.

The information provided by the system modelling of each luminaire was stored in each arduino's EEPROM. Thus, during initialization, a function that reads data from the EEPROM is called, and values are stored as global variables in a data structure.

The network is built by the individual systems described previously. Now, each of the 3 arduino's individual gains K_{ii} is the same, but there are relative gains K_{ij} that need to be determined, so that each desk's illuminance level is described by:

$$\begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} o_1 \\ o_2 \\ o_3 \end{bmatrix}. \quad (13)$$

- u_i - duty cycle applied in desk i,
- L_i - illuminance level in desk i,
- o_i - outside disturbance level (lux) in desk i,

To provide CAN-Bus communications, the mcp2515 was connected to each arduino via serial peripheral interface (SPI) communication. Thus, SPI communications had to be initialized, to allow mcp2515 to trigger an interrupt on the arduino each time a CAN message was received. Then, the used interrupt number was attached to an interrupt handler function, which reads messages from buffers that had the incoming message flag ON, and puts them into the adequate CAN frame stream. Masks were used to separate User Instruction into one CAN frame stream, and messages regarding luminaire crucial information into another CAN frame stream. This way, a clear separation between messages is done, and is easier to process each message received.

The consensus algorithm also needs an initialization to pass values read from arduino EEPROM, gains K_{ij} (provided by the calibration explained in the next section) and the external interference into variables stored inside consensus class.

2) Calibration

The calibration function has to calculate, in each arduino, gains corresponding to its own luminaire and to other arduino luminaires. To simplify the analysis, fluxogram in figure (14) was made to illustrate how the function was implemented.

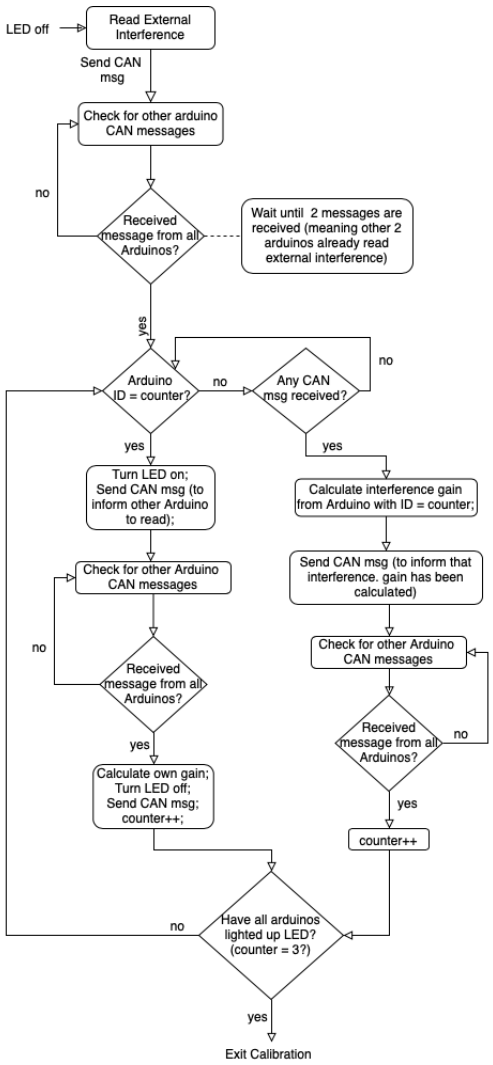

Fig. 14: Fluxogram Calibration

E. CAN-Bus Communication System

The CAN-Bus communication was made with the MCP2515 repository available on *GitHub*. The classes used had the following c++ header files: *can.h*, *can_frame_stream.h*, *mcp2515.h*.

Each CAN message is stored in a structure named *can_frame* from library *can.h*. This structure is composed by variables that store message ID and data. Message data is stored in an array of 8 bytes, and in order to correctly send and read messages, a union type variable must be used. Union type variables point to the same value, but represent it in different ways. In order to achieve communication, the

union used represents data in form of *unsigned long* and as an array of characters *unsigned char*. When a message is read or written, data is passed to input argument of the function that writes or reads as an *unsigned long*, and then the union is used to write data in bytes into CAN frame stream.

As mentioned before, there was a separation between messages regarding user input and messages regarding luminaires' correct functionality. Thus, two CAN frame streams were used. A CAN frame stream consists of a buffer stored in arduino memory, with capacity to 10 CAN messages before losing information. It also has variables to store index about where to read and write next CAN message. When a message is received (function that handles CAN message interrupt is called), the message ID is filtered, and after the controller decides in which CAN frame stream has to write, the function *put(can_frame &frame)* from *can_frame_stream.h* is called to copy the received message into to the desired stream, and returns 1 when message was correctly copied or 0 otherwise. When in main loop, a function tries to read messages from any CAN frame stream, the function *get(can_frame &frame)* from the same library is called, that returns 1 if any message was read or 0 otherwise.

The library *mcp2515.h* was mainly used to initialize CAN communications, attach interrupt handlers (as explained in Initialization section), and analyse error messages.

F. State machine and User Interface

The PC application using *BOOST Asio* library was not implemented. Instead, an interface through the arduino serial monitor was made. Only few options are available as user input, however the communications between arduinos in order to comply with inputs are viable, and fully functional.

The fluxogram in figure (15) illustrates the loop function of each arduino.

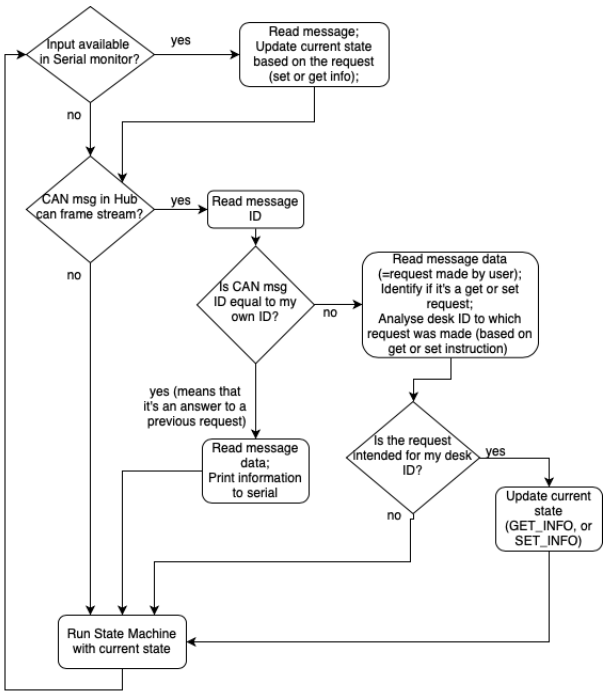

Fig. 15: Fluxogram of the loop for each controller

The state machine runs according to the variable *currentState*. It has 5 possible states: CALIBRATION, CONSENSUS, PID, GET_INFO or SET_INFO.

The calibration state runs the calibration function, and updates *currentState* to CONSENSUS. The consensus state implements the consensus algorithm explained in previous sections, and updates *currentState* to PID. The PID state checks desk occupancy, if it's an occupied desk and if the flag activated by TIMER 1 interrupt is ON, the PID controller is invoked. If the desk is not occupied, it turns the LED off, and if TIMER 1 interrupt flag is not ON it will advance to next loop iteration. Once in PID state, the only way to change state is by user input.

When a request is made by the user, the arduino that is connected to the serial monitor receives a message, and processes it regarding if it's a get or set request. The only difference between the two states is the way message data is interpreted, and answered. In state GET_INFO each request is inserted as "g [a] [i]", being [a] the information requested and [i] the desk ID to which the request is sent. Thus, the response of the system is "[a] [i] [val]", begin val the value related to information requested in [a]. In state SET_INFO requests are made in the form "[a] [i] [val]", being [a] the action requested, and [val] the value to set action [a] with. The response of the system is "ack" if action was correctly implemented or "err" otherwise. To illustrate the algorithm implemented to analyse requests, fluxogram in figure (16) was made. Note that the information read from the serial monitor and/or HUB can frame stream is stored as global variables. Also note that if the arduino connected to the serial monitor receives a request from the user, but the request is intended to another arduino, it sends the whole

message into other arduinos, and a flag will be activated in the arduino to which the request is intended, in order to send back the action output to the initial arduino, and afterwards being printed on the serial monitor.

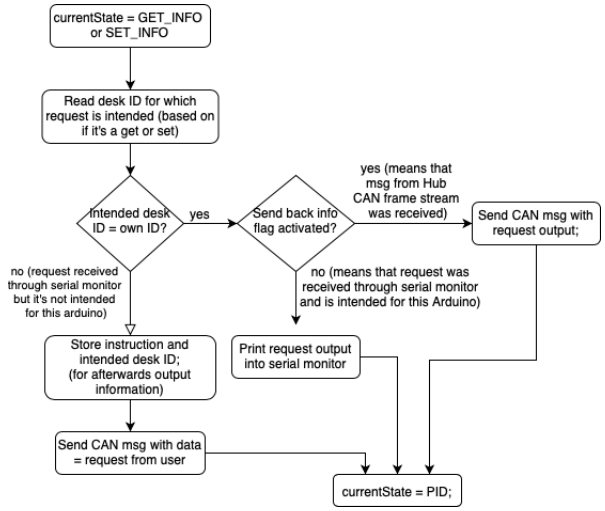

Fig. 16: Get and set info fluxogram according to requested input

V. Experiments

A. Individual feedback + feedforward control system

In order to test our individual control system, an individual luminaire was placed inside the shoe box. The code implemented for this experiment asked the user (serial) to input an integer between 0 and 500, which was our range of values of illuminance in that setting. With no disturbances (closed box), a duty cycle of 0 should produce 0 lux of illuminance, and a duty cycle of 255 should produce around $500Lux$ of illuminance. The experiment was conducted, imposing an operation timer on the arduino in order keep the resuslts consistent, as discussed earlier.

When a valid input was detected, the algorithm applied:

- Conversion from illuminance to duty cycle considering static state - $L = K * DutyCycle$,
- Feedforward control, taking into account the transient state model with the delay, mentioned earlier, in order to simulate the step more accurately,
- Feedback control with anti-windup and deadzone, to:
  - Reduce overshoot,
  - Keep the Lux value "stable" at the reference (input) value,
  - Be able to correct for outside disturbances (when the shoe was opened)

We also tested the systems ability to change its reference value from $200Lux$ to a low reference value of $20Lux$ (no disturbances applied). This allows for a more thourough PID testing.

The results are shown in the next section.

B. Cooperative decentralized control system

This experiment had many goals:

- Verify if the cooperative calibration was in accordance to the theoretical results,
- Test the effectiveness and speed of the communication system,
- Test the consensus algorithm implemented,
- Verify if the local controllers (PID's) were working as intended,
- Test the system's capability of handling user requests while running

The correct calibration can be tested whenever the arduino starts, since it is the first task to be done. The next thing to test after confirming the correct gains (K_{ij}) and interference factors (o_i) is the consensus algorithm. In order to do so, all of the duty cycles of the global system (three per arduino) are printed in the serial monitor during each iteration. This way, it is possible to check if the values are converging. This also allows to test the communication protocols since we can check if the duty cycles are being correctly transmitted by opening another serial monitor.

VI. Results

A. Individual System

After the aforementioned manual PID tuning, the system behaved in a very stable way, showing very little oscillations, but also relatively fast corrections to disturbances with reduced overshoot. Regarding these criteria, the feedforward controller had minimal to no effect, since its purpose is only related to the transient state (which is very fast, almost like an instant step). Every individual arduino behaved more or less the same way, with no differences worth pointing out.

Figure 17 shows a graph exported from the serial plotter showing the system's behaviour, given a reference, with no disturbances.

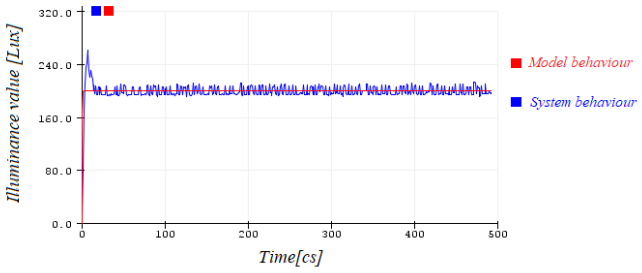

Fig. 17: System behaviour to a given reference value of $200Lux$ with no disturbances

The anti-windup feature is quite valuable here, since it stops most of the overshoot in rising steps (as well as falling steps). We can also see that the dead zone implemented helps the system to oscillate a lot less, and between closer values to the reference.

Figure 18 shows a graph of the same system's behaviour, given the same reference, but with one big disturbance

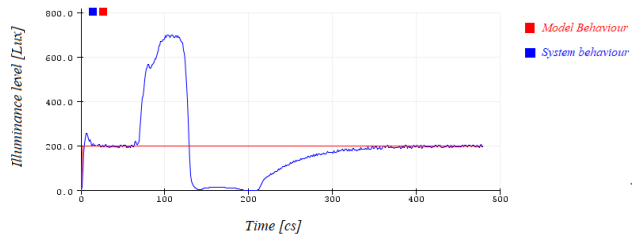

Fig. 18: System behaviour to a given reference value of $200Lux$ with a large disturbance

This helps us see how fast the local controller acts, correcting disturbances, how much it over-corrects, and also if the system stays accurate after the correction. Of course, the bigger the disturbance, the bigger the over-correction and the more time it takes for the system to return to the reference value.

Some individual systems showed some difficulty changing their reference values from a medium/high value to a low value, as showed in figure 19. This was usually solved by repeating the individual calibration process for the luminaire.

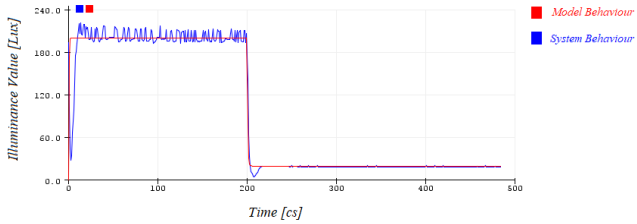

Fig. 19: System behaviour while changing reference values from $200Lux$ to $20Lux$

This case shows the system behaviour after re-calibration, having no trouble adjusting from a higher illuminance value to a lower one. It can also be seen that the higher the illuminance value, the higher the oscillations are.

B. Cooperative System

The cooperative control system is regarding the consensus algorithm. The matlab graphics that foresee the consensus working are represented in figures 20, 21 and 22. The reference illuminance was *120 lux* for each desk.

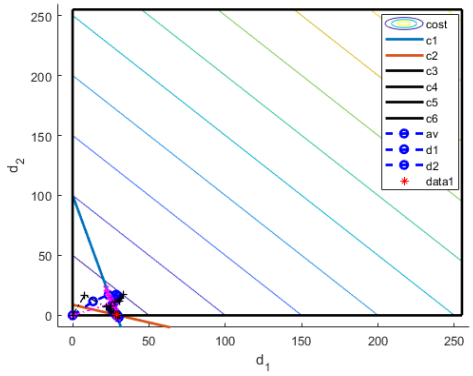

Fig. 20: Solution convergence space for the first and second arduinos

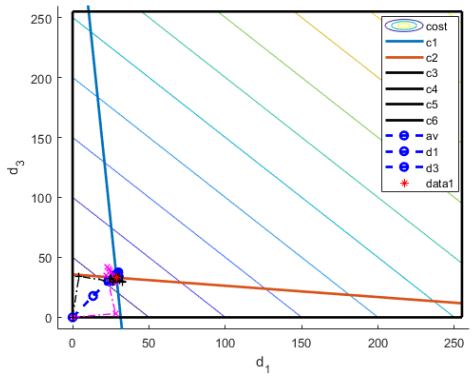

Fig. 21: Solution convergence space for the first and second arduinos

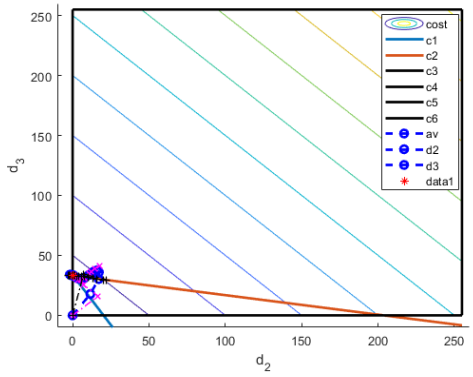

Fig. 22: Solution convergence space for the first and second arduinos

Since this algorithm has 50 iterations, the results from the implementation in c++ that follows only show the output from first and last iteration, in order to confirm convergence. Note that the test was made with the same illuminance reference as the matlab simulations, and it is confirmable that the simulations were accurate. Also note that first duty

represents the duty cycle of the arduino in desk ID = 0, and so on.

Arduino 0:

```
Iteration number: 0
First duty: 3.22  Second duty: 5.96  Third duty: 34.16
Iteration number: 49
First duty: 29.23  Second duty: 7.16  Third duty: 31.60
```

Arduino 1:

```
Iteration number: 0
First duty: 30.93  Second duty: 9.75  Third duty: 2.77
Iteration number: 49
First duty: 29.24  Second duty: 7.17  Third duty: 31.61
```

Arduino 2:

```
Iteration number: 0
First duty: 10.23  Second duty: 6.55  Third duty: 22.61
Iteration number: 49
First duty: 29.25  Second duty: 7.15  Third duty: 31.66
```

Fig. 23: Values of duty cycles for all arduinos at initial and final consensus iterations

In all arduinos final iteration, the duty cycle for each arduino is similar, comproving the algorithm convergence.

VII. Conclusions and Future

This system's study and implementation was not complete, yet some conclusions were still able to be drawn.

The implementation of the anti-windup feature on the PID controller was crucial to the stability of each luminaire. This feature has virtually no downsides, since it allows for a very substantial overshoot reduction, at almost no cost. Implementing a PID with anti-windup would only have donwsides in some cases where the system is designed only to achieve a certain value the fastest way possible and overshooting is not an issue.

The distributed cooperative control allows for smart real-time control of systems. It reduces overshoot and flickering, comparing to non-cooperative control (due to the lack of need to exchange references between agents), and also has a lower energy consumption. This allows for a more efficient system operation.

Unfortunately, it was not possible to setup a message acknowledgement system for the communication network, therefore there is no way of guaranteeing the absence of lost messages.

Regarding communications, there are many aspects to be perfected, as well as a few to be improved. Of course, not implementing the UI Application to establish communications, as well as commands, is the biggest part missing in the system. Although a copy of the UI Application was emulated through code implemented in the Arduino, establishing the commands/responses through the serial monitor, it lacks many optimization options, thread exploration, and efficiency.

VIII. Bibliography

[1] D. Caicedo, A. Pandharipande Distributed Illumination Control With Local Sensing and Actuation in Networked Lighting Systems. IEEE Sensors Journal, Vol. 13, Nº 3, Março 2013.

[2] A. Bernardino, Real-Time Distributed Control Systems 2019/20 Module 22 The consensus algorithm applied to the distributed illumination control problem, November 29, 2019

[3] C.Fernandez, D.R. Distributed smart lighting systems: sensing and control. Techniche Universiteit Eindhoven DOI: 10.6100/IR774336, 2014.

[4] A. Bernardino. Solution of Distributed Optimization Problems: The consensus algorithm, 2017,

[5] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato and Jonathan Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers 2010,

[6] Arduino Playground - MatrixMath http://playground.arduino.cc/Code/MatrixMath

[7] GitHub - Arduino MCP2515 CAN interface library https://github.com/autowp/arduino-mcp2515

IX. Appendices

A. Transient state time constants

Another different transient state model example worth mentioning takes into account not only the final illuminance but also the starting one. This leads to a branched equation defining the time constant instead of a single one. To obtain it, a series of cycles were run in the arduino. Each cycle started on a different and increasing duty cycle. A series of growing steps were applied and the LDR voltage was recorded during each cycle. The same thing was done for decreasing steps starting at higher duty cycles. With the data collected, it is possible to calculate the time constant of each of those cases and start making relations. In figure 24 and 25 this data is represented graphically. A vertical line represents one of the mentioned cycles, while an horizontal one represents the final step value. This is valid for both graphics .

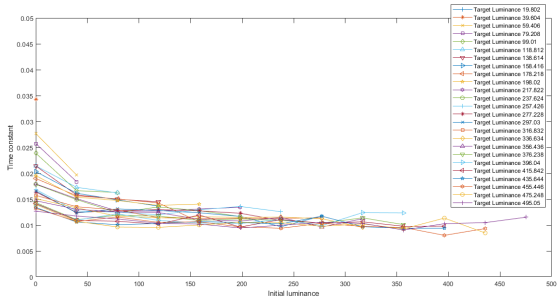

Fig. 24: Time constants of various positive steps from different initial illuminances

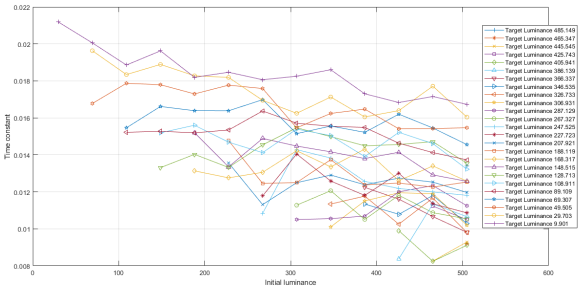

Fig. 25: Time constants of various positive steps from different initial illuminances

It is clear that the descent steps have a greater variation than the climb ones. Besides, it can be observed that the step responses obtained from lower duty cycles are higher than the rest. As so, a new model can be created in which two different best fits must be calculated: one that uses data from low duty cycles and another from the regular cases.

Since the transient state is not that relevent considering our project objectives, the simpler model was considered.