TÉCNICO LISBOA

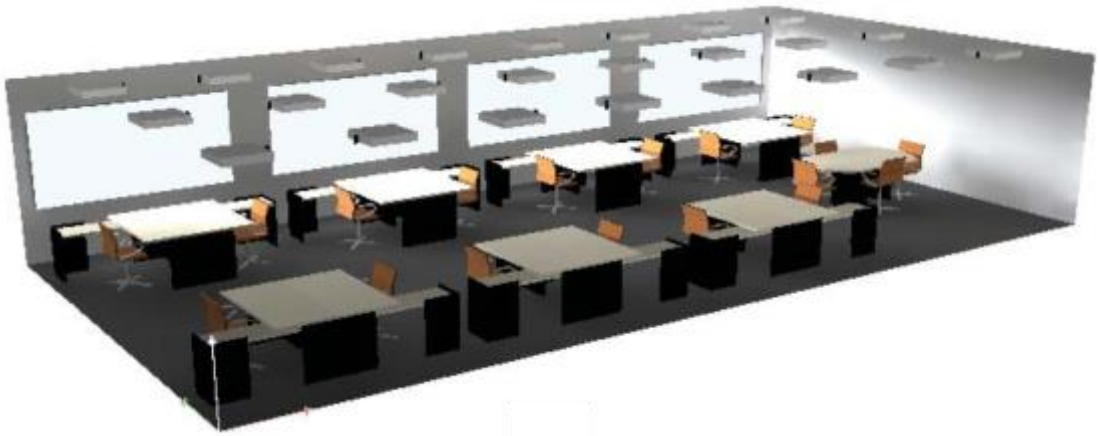
Master Degree in Electrical and Computer Engineering

2016/2017 – Winter Semester

Distributed Real-Time Control Systems (Sistemas de Controlo Distribuído em Tempo-Real)

PROJECT

Distributed Lighting Control



Prepared by

Alexandre Bernardino

Instituto Superior Técnico

Department of Electrical and Computer Engineering

Scientific Area of Systems, Decision and Control

Instructions for the project

The students must form groups of three or four to execute the project. There are weekly laboratory sessions of 1.5hr each where the students will access the relevant equipment to progress in the execution of the project and receive guidance from the teaching staff. Some equipment is available for students to take home in order to experiment and advance on points not requiring lab access. The equipment must be returned in the end of the semester.

The students have to perform two demonstrations of the project: one in the middle of the semester (last week of October or first week of November, granting 25% of the project grade) showing the progress of the first stage, and one in the end of the semester to show the full project (last week of lectures in December, granting 25% of the project grade). Then, the group has to write a report to be delivered before the exam period (first week of January 2017, granting 50% of the project grade).

The report must be direct, concise and short but insightful. Experiments should be designed to highlight the important components of the project and properly illustrated with graphs and tables with all units and axes identified.

Each student in the group should take responsibility for the reporting of one or more project components: system modelling and identification, PID control, distributed control, hardware interfaces, communications, concurrency, microcontroller programming, C++ programming. The report evaluation may be individualized if there are large quality differences in different report sections.

Together with the report, the software, hardware schematics and other material developed to execute the project should be delivered.

Both the report and the software must be sent to the professor in charge of the laboratory within the prescribed time limits.

The reports and the software developed must be original. All forms of plagiarism or copy detected will be punished without contempt according to IST regulations and the Portuguese Law.

Read this with attention!

Safety warnings

- In case of gross misuse of the equipment or violation of its operational limits, the students will be liable for the replacement of the damaged equipment.

Notice also:

- You must **always bring to the lab the equipment taken home** since this is required for the lab sessions.
- You must **always bring to the lab a flash drive** to store the results obtained during the class;
- When using the lab computers make sure that you are using a working folder with write permissions (can be an external flash drive).

Take home material for each group

Basic Kit

- 2 Arduino UNO Rev. 3 or equivalent
- 2 USB cable type A/B
- 2 Breadboard
- 1 Set of multicolored jumper wires.
- 2 Light Emitting Diode (LED superbright, 20mA, 3.4V)
- 2 Light Dependent Resistor (LDR GL5528)
- 2 Resistor 100 Ohm
- 2 Resistor 10 KOhm
- 2 Capacitor 1 microF

After first demo

- 1 Raspberry Pi 3B
- 1 Power supply and cable for raspberry pi 3
- 1 SD card for raspberry pi 3
- 1 Ethernet cable

Other material that can be requested by email (if available)

- Servo Motor HITEC H-S322HD
- Micro DC motors
- Single color LED's.
- RGB LEDs
- Discrete components of varied characteristics and values (resistors, capacitors, diodes, transistors)
- 330 Ohm resistors
- 10 KOhm Potentiometers
- Push buttons
- Piezo buzzer
- Piezo knock sensor
- Temperature sensors

1. Introduction

With the increasing prices on electricity, large research efforts have been put in the search for efficient illumination systems during the last decade. Advances in semiconductors has brought us high power LED's that allow up to 85% saving in energy consumption and high versatility of use due to their small size and dimming ability. Additionally, improvements in technology and reductions in price are making LED the favourite illumination devices for home, automotive, office and smart building spaces (www.ledmarketresearch.com).

The flexibility of LED lighting is allowing for another recent trend in energy optimization and comfort control. Recent research is being devoted to adaptive and distributed lighting control taking into account the occupation status of spaces and the intensity of external illumination [1][2][3]. Cheap luminance and presence sensors allow controlling the power used to achieve the required comfort luminance levels for occupied spaces, and reduce the power in unoccupied spaces. Already in streets, buildings and public spaces lights are turned on and off as a function of motion detection sensors, but more versatile systems can be developed to take into account external illumination and jointly coordinate the activation of multiple luminaires that interact. In this project we will consider an office-like scenario where desks have presence and luminance sensors, and luminaires have light dimming and communication abilities to synchronize with their neighbours.

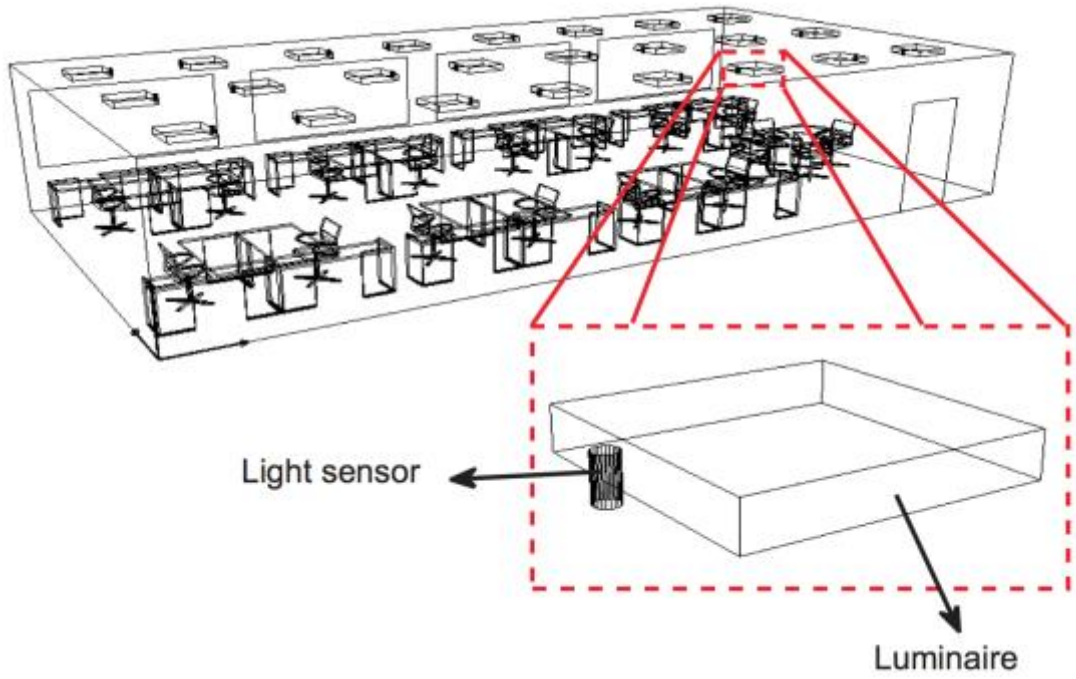


Figure 1. Scenario envisaged in the project. Each desk is equipped with a luminaire, light sensor and presence sensor. The control of the lighting is made to keep fixed levels of illumination at the desk plane (high for occupied desks and low for unoccupied desks) while minimizing the global energy consumption and taking into account the daylight illumination and disturbances from the neighbour luminaires.

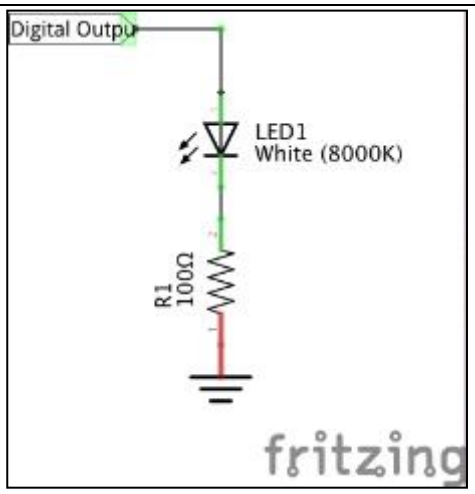
2. Objective

The objective of this project consists on the real-time control of a distributed illumination system in a small scale model of an office space. Consider each desk has a smart luminaire composed of a light emitting device, a luminance sensor, a presence sensor, with computational and communication elements. In the project we will simulate the office with a small opaque car box and each luminaire consists of a breadboard with the Arduino, LED and LDR circuits. Each group will, thus, create a model of a small office with 2 luminaires. For practical reasons, the presence sensors will not be physically implemented, but instead simulated by setting variables in the microcontrollers through a computer interface or push buttons. A small window should be simulated by creating an opening in the card box, to exploit energy harvesting and simulate external disturbances.

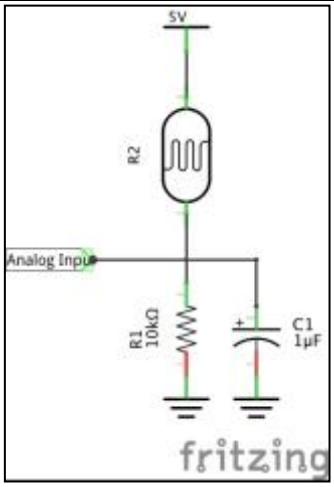
The objective is to minimize the energy consumption and user comfort. Energy minimization will be achieved by controlling the dimming level of each led such that occupied desks have luminance levels above a certain value (HIGH) and unoccupied desks have a luminance levels above a lower value (LOW). According to the European standard EN 12464-1 *"Lighting of indoor workplaces"*, April 2013, in typical office spaces during working hours occupied desks should have a minimum illuminance value of 500 lux (HIGH), whereas non-occupied desks should have illuminance above 200 lux (LOW). With the available equipment and experimental setup, these values are not achievable. Adequate values for HIGH and LOW thresholds should be defined by each group, since it will depend on the dimensions of the reduced scale model and the position of the luminaires. User comfort should be maximized by keeping the illumination always above or equal to the minimum levels, and minimizing the variations of the illuminance (flicker) during desk occupation. These variations may be due to noise, external disturbances, or interference caused by the other luminaires in the shared space. Noise and external disturbances can be compensated by a local feedback control loop at each luminaire, but internal disturbances due to interference from other desks can be predicted and compensated through proper communication and synchronization between luminaires (global control).

3.Plant Description

Each luminaire will be simulated with the equipment supplied to the groups. The following diagrams illustrate a possible LED driver circuit and and an illuminance reading circuit.



a) Led driving circuit



b) Illuminance reading circuit

The LED dimming can be implemented via PWM in one of the digital output ports. The illuminance can be measured via the LDR in a voltage divider circuit. The capacitor in the voltage divider serves to reduce noise in the sensing circuit. The PWM frequency should be configured to further reduce the noise on the analog input. Both the LED and the LDR should be pointing "vertically".

The distributed illumination system should be implemented in a reduced scale model. An opaque box with a cover should be used to allow complete isolation from the external illumination. The box should be large enough to contain the 3 luminaires, but not too large. If the box is too large, the LED intensity may not be enough to provide enough power to illuminate the LDR (note that the LDR receives mostly light reflected in the box walls). To improve light reflection, if needed, the interior of the box may be covered with white paper. Small openings in the box must be made to pass necessary cables. Try to insulate these opening as much as possible to prevent uncontrolled light entrance from the outside. A window should be simulated in the card box to test the system under controlled external light.

4. First Stage

4.1. Description

During the first stage of the project, the students will implement the reduced scale office model, the luminaire actuation and sensing circuits, the actuation and sensing microcontroller code, the local feedback control, and a simple interface with a PC using the Arduino Serial Monitor. In local feedback control no explicit communication is allowed between different luminaires, and each luminaire only cares about its own desk.

The groups should start by creating a model of each luminaire, i.e. the how its input (LED actuation) and output (LDR measurement) are related. To identify the model, a set of experiments should be planned to feed the system with different actuations and characterize its response. This stage should be made carefully, to prevent changes in the operating conditions (e.g. external illumination), during the process. Also check that the LED actuation PWM frequency is high enough to prevent noise in the LDR measurement (check TIMER1 functionality). It may also be useful to implement a digital filter in the LDR readings to reduce noise e.g. acquire many samples during one sample time and compute the average (or the median) of the values. The analog input resolution can also be increased by using the AREF pin.

The LDR is a non-linear element, i.e. its gain (ratio of the variation of the illuminance to variation of the measurement) varies with the operating point. To implement the local controller, its characteristic should be linearized, so that the local controller has similar response in different operating points. It is suggested that the LDR readings are converted to LUX units with the help of the LDR characteristic response in the datasheet. Note that the LDR datasheet indicates a range of resistances for each illumination intensity. In order to have a one-to-one mapping, it is suggested that for each luminance is considered just the middle resistance value (in logarithmic units).

The local controller should be implemented as a PID controller with feedforward. Start with the development of the feedforward term, with the objective of driving the LED in open-loop to achieve some illuminance reference. Of course the obtained value will not be exactly the desired one due to external disturbances and model errors, but it is important to speed up the response of the system. On top of that, implement the PID controller to cope with the disturbances and modelling errors. Do not forget to implement adequate integrator anti-windup functions, to cope with actuator limits.

The PC interface via the serial link should allow setting the LED PWM values, setting references for the local controller, read the LDR values in LUX, setting the occupation state of the luminaires, and other functions you find useful.

This part will be evaluated through a demonstration of the local illumination control in the middle of the semester (end of October or beginning of November). This demonstration contributes with 25% of the project grade. Students should demonstrate the ability to set LED dimming values, read illuminance values in LUX, define setpoints for local control, demonstrate the performance of the control system in step changes in the reference setpoint and under external disturbances.

4.2. Implementation Notes:

1. Using the program "serial monitor" in the Arduino IDE, information can be received from and sent to the Arduino program. Groups should program the Arduino to send relevant information to the PC (for data plotting, diagnostics,

debug, etc.) and receive from the PC relevant commands (set desk occupation, set control parameters, set program configuration modes, etc.).

2. Take into account that serial communications use precious microprocessor time. Choose messages with short size and a high baud rate. Compute communication delays and verify that communication time do not exceed available control loop time.
3. You can copy text from the serial monitor. Format your messages so that you can use the copied text to graphically visualize the data in Matlab or Excel.
4. Add any functionalities that facilitate development, testing, debugging, and demonstration of the applications. There are additional electronics components that you can request for such purposes.
5. Always use SI units to represent quantities. Check the electronic components datasheets to verify the conversions from electrical to physical units.

4.3. Guidelines to document the first stage:

It is not required to deliver a report of the first stage. However, it is highly recommended that the information required to write the final report is collected at every stage. Note that the information to collect, listed in the following paragraphs, is mostly documenting intermediate steps necessary to complete the project.

a) Take pictures of the interior and of the exterior of the box enclosing the luminaires. Make sure you illustrate the position of the LED, the LDR and the emission / reflection path.
b) Show in a plot the voltage of the LDR for a stair like input in the PWM values of the actuation (e.g. steps with amplitude 10 going from 0 to 250, with a duration of 0.1sec at each level. Note the non-linearity of the response.
c) Show in a plot the steady state voltage at the LDR, as measured by the ADC (values 0..1023), as a functions of the various PWM references (values 0..255). In the data acquisition process you may need to wait a few milliseconds to reach steady state before changing the PWM reference.
d) Convert the previous plots to LUX units. Comment on the linearity of the response.
e) Assuming that local step responses observed as intensity illumination changes (Lux) can be modelled as first order systems $G(s)=K_0/(1+s\tau)$, indicate an estimate of the static gain and the time constant. Comment the use of illumination intensity (LUX) vs voltages observed at the capacitor.
f) Demonstrate the performance of the feedforward controller with plots showing reference illumination values and the obtained responses (LUX). Include step changes in the reference set points and external illumination disturbances.
g) Indicate the parameters of the PID controller designed for the system.

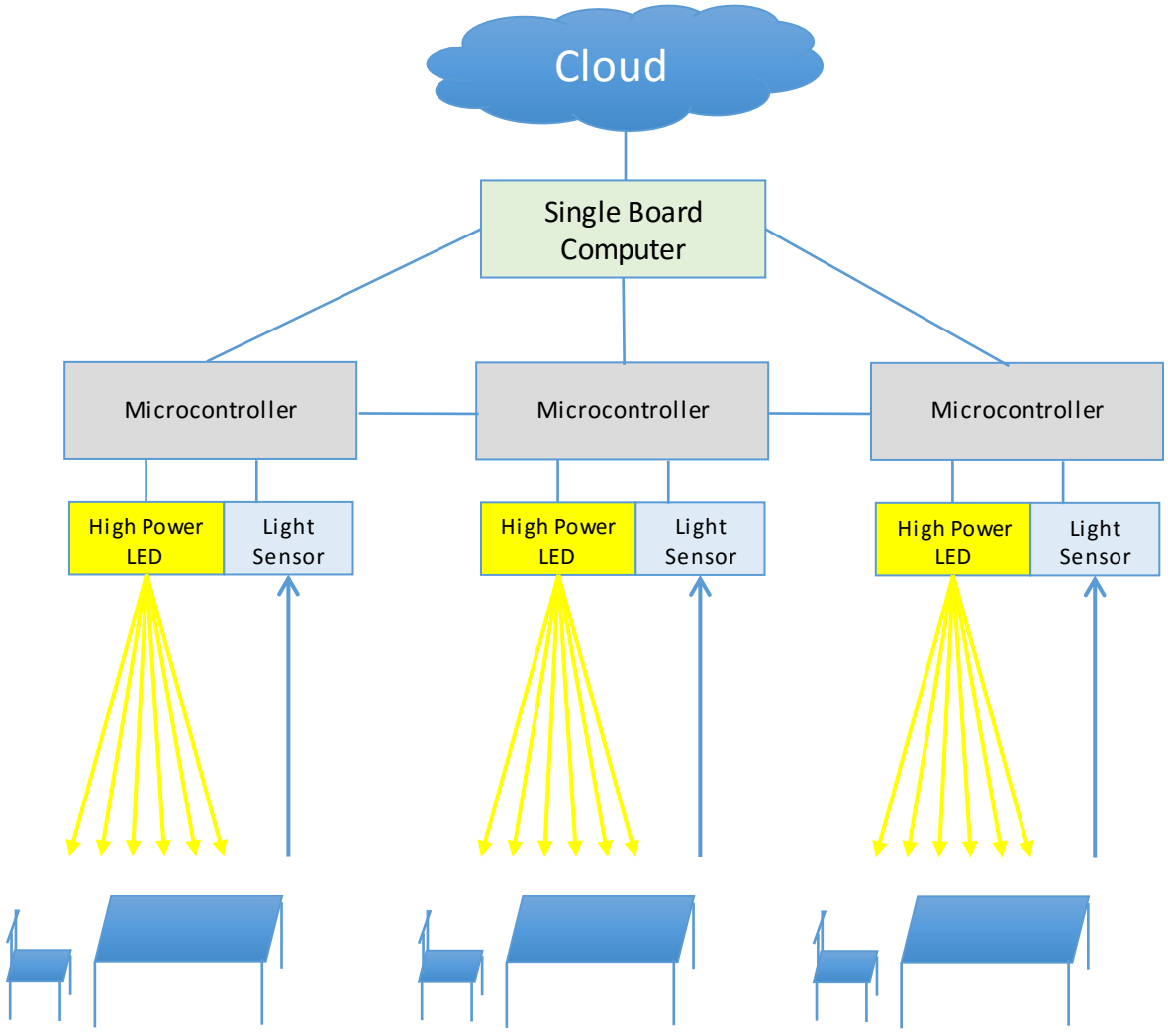
h) Demonstrate the performance of the PID controller, with and without feedforward term. Plot the reference illumination values vs the obtained responses (LUX). Include step changes in the reference set points and external illumination disturbances.

i) Print the Arduino code implementing the PID controller.

5. Second Stage

5.1. Description

During the second stage the students will implement a cooperative distributed control system and a wireless communications hub to interface with the distributed control. The Arduino controllers will be connected using a control network (I2C) to allow communication between the luminaires. A communications hub will be implemented with a Raspberry Pi 3B single-board-computer connected to the Arduinos via the serial link and to the intranet via wifi.



The objective of the distributed controller is to optimize the following global metrics: maximize energy consumption and maximize user comfort (luminance always above or equal to the reference set points and flicker reduced to minimum).

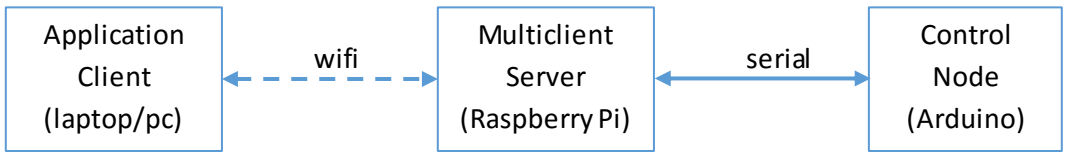
When a luminaire changes its actuation, it has an effect in the neighbour luminaires (coupling effect). Each group must develop a distributed control algorithm to coordinate the several luminaires so the effects of this coupling effect is minimized. To realize the compensation, it is necessary to model the effects of a change in one luminaire in the measured illuminance of the other. Because this coupling between luminaires depends on many factors, a calibration procedure should be made at system startup. For example, turn on one luminaire at each time and measure the illuminance in all other luminaires.

The distributed controller should be of the "non-centralized" type, each luminaire can communicate with its neighbours but no central master exists. Despite in this work only two luminaires are available, the developed algorithm should be designed for easy extensibility to a larger number of luminaires.

The Raspberry PI will implement a server providing an interface between an application client and the control nodes, to fulfil three main purposes:

a) send commands from the application client to the control nodes (set points, occupancy, etc);
b) collect information from the control nodes for storage, and compute the evaluation metrics;
c) send information for the application client, for monitoring, diagnostics and visualization.

These functions will be implemented as a C++ multithreaded server using serial and socket communications. The client application should be able to read the state of any luminaire (occupation, led dimming level, illuminance level), set values to the system (occupancy, led dimming level, redefine the reference levels HIGH and LOW), and access system statistics (energy consumption, comfort metrics). The information can be requested in three different modes: actual values, last minute history or real-time stream.



This part will be evaluated through a demonstration of the final distributed control system in the end of the semester (last week of lectures). This demonstration contributes 25% to the project grade.

A written report containing both parts, and associated software, will be delivered two weeks after the final demonstration (just before exams season). The report represents 50% of the final grade.

5.2. Evaluation Metrics

To properly validate an engineering solution, it is fundamental to define appropriate evaluation metrics, expressing in a quantitative way the requirements addressed in its formulation. For our system we target at minimizing the energy spent in illumination while providing comfort to the users. The energy is the accumulation (integral) of the instantaneous power along time. Let us assume a nominal (fake) value for the maximum power of the luminaire as 1W. Then, a formula to compute the energy consumed at each desk is:

$$E = \sum_{i=2}^{N} d_{i-1} \left(t_{i-1} - t_i \right)$$

where *i* is the index of the control samples, t_i are the sample times and d_i is the led duty cycle value (between 0 and 1) at sample time t_i. The units of this metric are Joule[J].

While energy minimization is simple to formulate, the comfort criteria is more subjective. We can consider the following rules:

a) The system should prevent periods of illumination below the minimum settings defined by an occupation state. A metric to assess this criterion can be defined as the average error between the reference illuminance (l_{ref}) and the measured illuminance (l_{meas}) for the periods when the measured illuminance is below the reference. Let us call this quantity the **Comfort Error**:

$$C_{error} = \frac{1}{N} \sum_{i=1}^{N} \max(l_{ref}(t_i) - l_{meas}(t_i), 0)$$

where N is the total number of samples used to compute the metric and t_i are the sampling times.

The previous expression refers to a single desk. The total average error should be computed as the sum of the average errors at each desk. The units of this metric are [LUX].

b) The system should prevent sudden variations in the illuminance (flickering) while the reference is at a constant value. Variations of illuminance at a desk should only happen during alterations of the occupation state of that desk. Therefore, a metric to assess this criterion can be defined as the average variation of the illuminance during periods of constant occupation computed by and approximation to the second derivative of the illuminance. Let us call this quantity the **Comfort Variance**:

$$V_{flicker} = \frac{1}{N} \frac{\sum_{i=3}^{N} \mid l_{meas}(t_i) - 2l_{meas}(t_{i-1}) + l_{meas}(t_{i-2}) \mid}{T_s^2}$$

where T_s is the sampling period and $\mid A \mid$ represents the absolute value of A. The transient periods due to explicit variation of the reference, should be excluded from the formula. Again, the previous expression refers to a single desk. The total average variation should be computed as the sum of the average variations at each desk. The units of this metric are [LUX/s^2].

In the report, indicate factors that can influence this metric.

5.3. Implementation Notes

Despite the default use of I2C is one-master-multiple-slaves, it can be also used as multi master, where any node can take the initiative of communicating to its neighbours whenever necessary. However, the Wire library for I2C provided with the Arduino has some caveats, in particular it blocks in the writing stage, which stops all other ongoing processing. This is a problem of the Wire library and not of the Arduino itself since the ATMega328 I2C hardware implementation is non-blocking and based on interrupts. There are several alternatives to address this problem:

1- Reduce the PID control loop sampling frequency so that I2C communications have enough time to send the data.

2- Stop the PID control during I2C communications. This is feasible in our system since it is a stable system, so holding the control value is not prejudicial in terms of stability.

3- Implement the I2C communications without the Wire library.

Any of these alternatives, or other proposed by the students, are accepted provided that they are properly justified, identifying its advantages and limitations.

It is recommended to exploit all functionalities provided by the communication functions through proper configuration, in order to minimize the communication times, e.g. by selecting appropriate baud rates for the existing cable lengths and define messages with short lengths. The final report should include an analysis of the times taken in the communications.

5.4. The C++ server

To interface the distributed system with the outer world, groups should develop a C++ server using TCP-IP protocol and asynchronous I/O classes from Boost library ASIO. These classes also provide functionality to read/write data in the serial port, for data exchange with the Arduino. The TCP-IP server should listen for connections at port 17000 and be able to serve multiple clients. A sample client will be made available in the course web page and will be used to test the project during the final demonstration.

The server should be able to send and receive data from the client in string format. A specification of the communication protocol between client and server is provided in the following table.

Command	**Client Request**	**Server Response**	**Observation**
Get current measured illuminance at desk <i>.	"g l <i>"	"l <i> <val>"	<val> is floating point number expressing measured illuminance in lux.
Get current duty cycle at luminaire i	"g d <i>"	"d <i> <val>"	<val> is floating point number expressing duty cycle in percentage.
Get current occupancy state at desk <i>	"g o <i>"	"o <i> <val>"	<val> is a Boolean flag: 0 – non-occupied, 1 – occupied.
Get current illuminance lower bound at desk <i>	"g L <i>"	"L <i> <val>"	<val> is floating point number expressing illuminance lower bound in lux.
Get current external illuminance at desk <i>	"g O <i>"	"O <i> <val>"	<val> is floating point number expressing background illuminance in lux.
Get current illuminance control reference at desk <i>	"g r <i>"	"L <i> <val>"	<val> is floating point number expressing illuminance control reference in lux.
Get instantaneous power consumtion at desk <i>	"g p <i>"	"p <i> <val>"	<val> is floating point number expressing instantaneous power at

			desk <i> in Watt. Assume each led nominal power = 1W.
Get instantaneous total power consumption in the system.	"g p T"	"p T <val>"	<val> is floating point number expressing total instantaneous power in Watt. Assume each led nominal power = 1W.
Get accumulated energy consumption at desk <i> since the last system restart.	"g e <i>"	"e <i> <val>"	<val> is floating point number expressing accumulated energy consumption at desk <i> in Joule. Assume each led nominal power = 1W.
Get total accumulated energy consumption since last system restart.	"g e T"	"e T <val>"	<val> is floating point number expressing total accumulated energy consumption in Joule. Assume each led nominal power = 1W.
Get accumulated comfort error at desk <i> since last system restart.	"g e <i>"	"e <i> <val>"	<val> is floating point number expressing the accumulated Comfort Error in lux. See section 4 – Evaluation Metrics
Get total comfort error since last system restart.	"g e T"	"e T <val>"	<val> is floating point number expressing the total Comfort Error in lux. See section
Get accumulated comfort variance at desk <i> since last system restart.	"g v <i>"	"v <i> <val>"	<val> is floating point number expressing the accumulated Comfort Variance in lux/s^2. See section 4
Get total comfort variance since last system restart.	"g v T"	"v T <val>"	<val> is floating point number expressing the total Comfort Variance in lux/s^2. See section
Set occupancy state at desk <i>	"s <i> <val>"	"ack"	<val> is a Boolean flag: 0 – non-occupied, 1 – occupied.
Restart system	"r"	"ack"	Reset all values and recalibrate.

5.5. Guidelines to document the second stage

In your report, please do not forget to address the following points:

a) (i2c Communications) Indicate the average time required to send a request message and receiving an answer.

b) (C++ Classes, Sockets, Parallelism) Indicate the C++ classes most relevant for coding the server running in the PC interfacing to the Arduino(s).
c) (Sockets) List the number of sockets used to implement the server. Describe the functions of the sockets.
d) (Concurrency) Indicate the methodologies used in the server to implement parallelism in order to answer the various clients.
e) (Distributed Control) Describe the implementation of the cooperative distributed controller.
f) Perform experiments that allow comparison of results between the cooperative distributed controller and the non-cooperative controller (independent local controllers). Plot the time response of both controllers in similar situations and present a table with their performance metrics.

6. Report

Write a report describing the problem, the designed solutions and the obtained results, mentioning the positive and negative points of the applied techniques. The report must be complete but succinct, with less than 20 pages including graphics and references. Avoid redundancies and exaggerated technical content. Try to summarize as much as possible but do not miss to write the important things. The graphics must be self-contained, i.e. fully labelled and with complete captions.

– Enjoy the project –