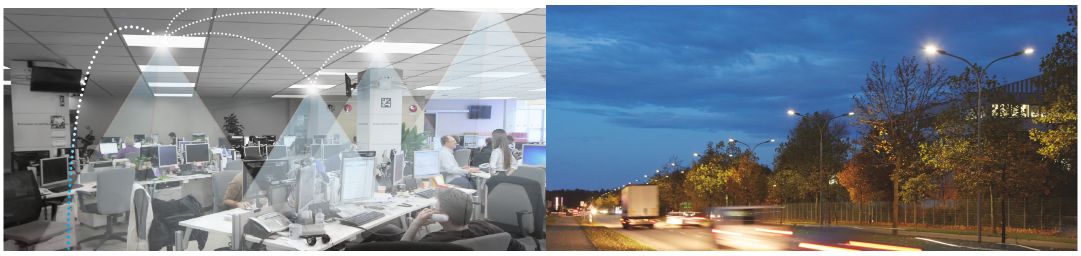
Master Degree in Electrical and Computer Engineering
2020/2021 – Winter Semester

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

PROJECT

Real-Time Cooperative Decentralized Control of Illumination Systems



Prepared by

Alexandre Bernardino, João Pedro Gomes, Ricardo Ribeiro

Instituto Superior Técnico

Department of Electrical and Computer Engineering

Scientific Area of Systems, Decision and Control

Version 1.0

Instructions for the project

Students must form groups of three to execute the project. There are weekly laboratory sessions of 1.5hr each, where students will get the relevant equipment for the execution of the project and receive guidance from the teaching staff. Students can take the equipment home and develop most of the project autonomously. Anyway, the presence in the lab sessions is essential for continuous assessment and learn best practices that are hard do get otherwise. The equipment must be returned by the end of the semester after all assessments have finished.

The project is divided in two phases. One finishing in the **end of October** and the other in the **middle of December**. At the end of each phase, there will be a demonstration:

- In the first phase each student will work individually in the construction of one luminaire and the development of its local controller. In the end of this phase, each student will demonstrate his work individually.
- In the second phase, students will work collaboratively in the group, each one specializing in a particular component of the system: distributed control or communications or, concurrent programming. The final demonstration will be made in group, but each student must take responsibility for a fair part of the work.

After the demonstrations, the group has to write a report to be delivered before the exam period (first week of January). Each student must contribute to the writing of the report and each section must be identified by its main author.

The evaluation of the project has two components:

- Presential component, to access the individual and group performance during lab sessions and in the demonstrations (25% for each stage)
- Written component, to access the quality of each project in the following aspects: definition of the problem, justification and quality of the approaches, experiment planning and execution, relevance and presentation of the results, overall clarity and coherence of the report. (50%).

For the presential component, the assiduity and punctuality of the student in the lab is also considered. Unjustified absences may be penalized.

The report has a hard limit of 20 pages. It 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 must 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, etc. Grades may be individualized if the quality is inhomogeneous throughout the report.

Software, hardware schematics and other material developed to execute the project should be submitted jointly with the report.

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 will be pursued to the full extent of IST regulations and Portuguese law.

Read this carefully!

Safety warnings

- In case of gross misuse of the equipment or violation of its operational limits, you will be requested to replace 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** store in your own storage media the materials developed in the lab computers. Lab computers are used by many students and we cannot assure the persistence of the data stored internally.
- When using the lab computers make sure that you are using a working folder with write permissions (can be an external flash drive), otherwise you may be unable to save data.

Take home material

First Stage (for each student)

- 1 Arduino UNO Rev. 3 or equivalent
- 1 USB cable type A/B
- 1 Breadboard
- 1 Prototype Board Luminaire composed of
    - 1 Light Emitting Diode (LED superbright, 20mA, 3.4V)
    - 1 Light Dependent Resistor (LDR GL5528)
    - 1 Resistor 100 Ohm
    - 1 Resistor 10 KOhm
    - 1 Capacitor 1 microF
- 1 MCP 2515 Can-bus Module
- 1 Shunt for CAN-BUS termination.
- 7 Jumper Wires of type male-female (SPI Arduino->MCP2515)

Second stage (for each group)

- 4 Flexible Wires (for CAN-BUS twisted pairs)
- 1 Raspberry PI
- 1 formatted SD card
- 1 Charger 5V
- 1 Cable micro USB B
- 1 Ethernet Cable

Other material that can be requested (subject to availability)

- Servo Motor HITEC H-S322HD
- Micro DC motors
- Single color LEDs.
- RGB LEDs
- Miscellaneous discrete components (resistors, capacitors, diodes, transistors)
- 10 KOhm Potentiometers
- Push buttons
- Piezo buzzer
- Piezo knock sensor
- Temperature sensors

1. Motivation

With the surge in electricity prices, a large research effort has been devoted during the last decade to the development of efficient illumination systems. Advances in semiconductors have brought us high power LEDs 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 homes, cars, offices, cities and smart building spaces (www.ledmarketresearch.com).

The flexibility of LED lighting is powering another recent trend in energy optimization and comfort control. Recent research is being devoted to adaptive and distributed lighting control systems that account for the occupation status of spaces and the intensity of external illumination [1][2][3]. The power used to achieve the required comfort luminance levels for occupied spaces can be controlled with cheap luminance and presence sensors, while reducing it in unoccupied spaces. Lights in streets, buildings and public spaces are already being turned on and off using motion detection sensors, but more versatile systems can be developed to consider external illumination and jointly coordinate the activation of multiple interacting luminaires. 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.

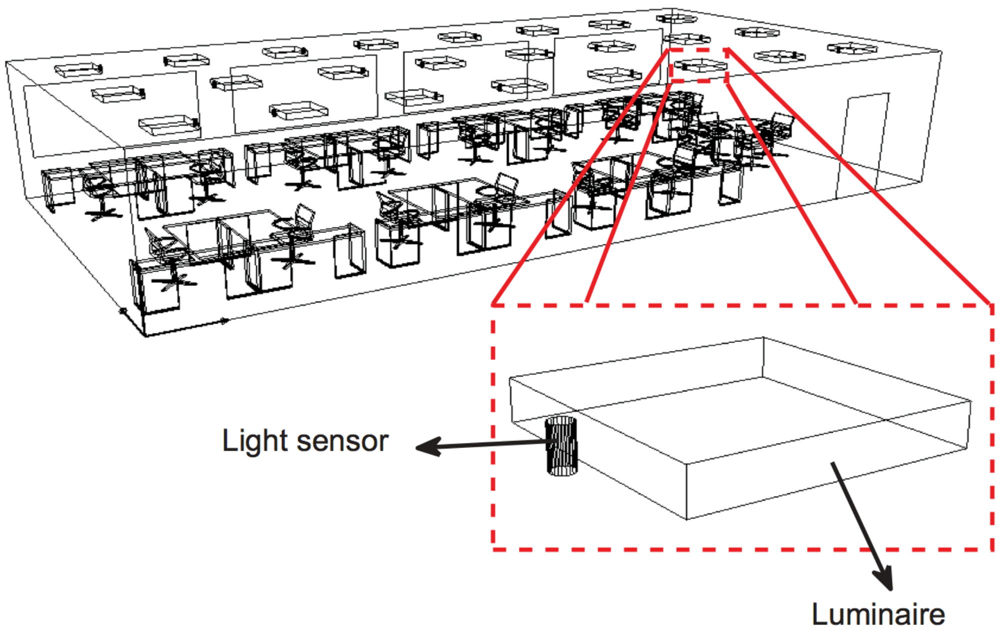


Figure 1. Scenario envisaged in the project. Each desk is served by a luminaire containing a light sensor, presence sensor and communication links to its neighbours. Note that the sensor measures the reflected light in the table and not the direct light from the lamp. The control of the lighting attains 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 neighbouring luminaires.

2. Objective

The objective of this project is to design a real-time control system for a distributed illumination system in a small-scale model of an office space. Conceptually, each desk has a smart luminaire comprising a light emitting device (LED), a luminance sensor, a presence sensor, as well as computational and communication elements. Note that the luminance sensor should measures the reflected light in the table and not the direct light from the led. In the actual project we will simulate the office with a small opaque cardboard box and each luminaire consists of a breadboard with the Arduino, LED and LDR circuits. In the first stage of the projects each student will create a model of a small office with 1 luminaire. In the second stage, each group will extend the system to 3 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. A small window should be simulated by creating an opening in the box, to exploit energy harvesting and simulate external disturbances.

The objective is to minimize the energy consumption and maximize 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 ones have a luminance level 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 (visibility), while minimizing the up-and-down 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 provided equipment. The following diagrams illustrate a possible LED driver circuit and an illuminance reading circuit.

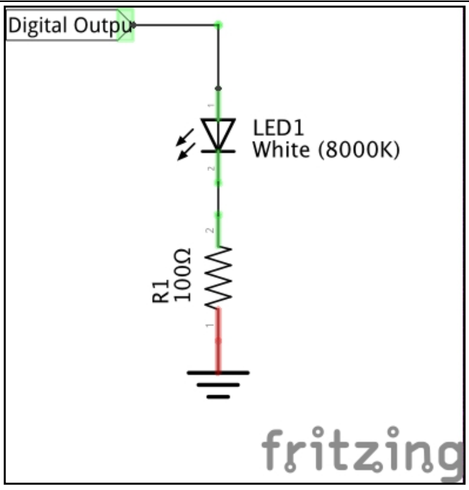
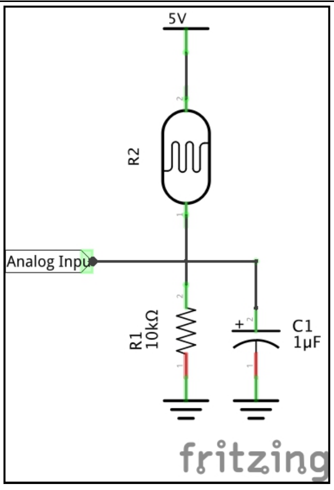
 Digital Outpu LED1 White (8000K) R1 100Ω fritzing a) Led driving circuit	 5V R2 Analog Inpu R1 10kΩ C1 1μF fritzing b) Illuminance reading circuit

Figure 2 – Schematics for the luxmeter (a) and LED driver (b) circuits.

Dimming of the LED 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 helps to reduce the 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 sensor should measure the reflected light in the table and not the direct light from the lamp.

The illumination system should be implemented in a reduced scale model. An opaque box with a cover should be used to completely block 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 be insufficient to properly illuminate the LDR (note that the LDR receives mostly light reflected off the sides of box). To improve light reflection, if needed, the interior of the box may be covered with white paper. Small openings in the box should be made to pass cables and wires. Try to insulate these openings as much as possible to prevent uncontrolled light from entering the box. A window should be cut into the box to test the system under controlled external light.

4. First Stage

4.1. Description

During the first stage of the project, students will implement the luminaire and its local control system, and test it in a reduced scale office model. The local controller is unaware of the existence of other luminaires and has the objective to keep the illumination level as close as possible to the desired value, despite disturbances due to variations in external illumination, cast shadows or changes in the light reflection paths. The controller should both have a fast response to changes in the desired illumination level, and avoid flicker and overshoot in response to disturbances and noise. Because these objectives are often conflicting when using a feedback controller, it is recommended to also use a feedforward component.

Each luminaire can be decomposed in the following modules: (i) the illuminance measurement system (luxmeter), (ii) the LED actuation system (driver), (iii) the individual luminaire controller (local controller), (iv) a calibration procedure to identify the office model system response (controller calibration), and (v) a simple interface with a PC to receive commands and send data for analysis.

The Illuminance Measurement System

The purpose of this system is to measure the illuminance in LUX units. The LDR is a non-linear element, i.e., its gain (ratio of the variation of the illuminance to variation of the measured voltage or current) varies with the operating point. However, its relationship to the standard illuminance unit — the LUX — is known. The LDR readings should be converted to LUX units with the help of the LDR characteristic response in the datasheet. Note that the LDR datasheet indicates a range of resistance values for each illumination intensity (see Fig. 3). In order to compute a one-to-one mapping between resistance and illuminance, consider the middle of the range specified in the data sheet.

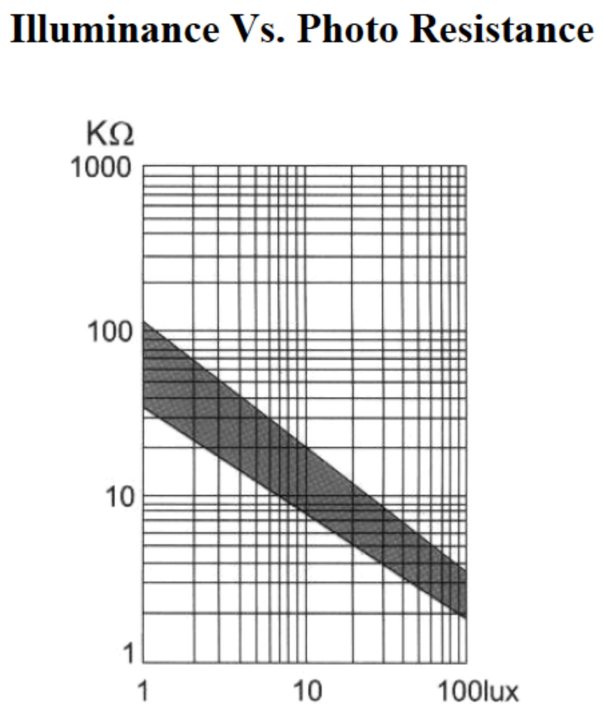


Figure 3 – Illuminance vs Resistance characteristic curve of the light sensor. Note that axes are in logarithmic scales.

The LED Actuation System

The ARDUINO does not have a pure analog output. Instead, it emulates it using a switching digital signal with Pulse Width Modulation (PWM), whose ratio of the duration of 1's to the duration of 0's (duty cycle) is proportional to the analog voltage required. The frequency of the PWM signal should be at least 10x higher than the cut-off frequency of the input filter (implemented by the input capacitor), so that the switching does not have a significant influence the luminance measurement signal. Note that the human eye is a low pass filter that is insensitive to flicker at frequencies above around 100Hz.

System Calibration

Although not mandatory in the design of a PID controller, a dynamical model of a physical system (Plant) is always an important component in a control system. In our problem, this model represents how much light emitted by the LED is received at the LDR (gain), and how fast variations in illuminance can be read by the measurement system (time constant). Having a good dynamical model allows us to predict the values of the Plant after some actuation and compare this with the current measurements to detect external disturbances. Note that each system may have a different dynamical model not only because the luminaires may be different, but also because the light paths travelled in each office model are different (different size, different reflectivity of internal surfaces, different arrangement of the elements inside the box). Every time the luminaire is moved or the configuration of items in the office changes, the dynamical model will change (in this case the amplitude is more affected than the temporal characteristics). To identify the model, a set of experiments should be planned to feed the Plant with different actuations and characterize its response. These tests should be carried out carefully, to prevent changes in the operating conditions during the process (e.g., external illumination). Also check that the PWM frequency for the LED is high enough to prevent unnecessary noise in LDR measurements (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 a time interval and compute the average (or the median) of the values. The analog input resolution can also be increased by using the AREF pin.

The Individual Luminaire Controller

The individual luminaire controller (local controller) should be implemented as a PID controller with feedforward. The feedforward term has the role of producing fast changes of the LED in response to changes in the desired reference values at the desks. The feedback term has the role of responding to external disturbances. Start with the development of the feedforward term, whose objective is to drive 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 feedback controller to cope with disturbances and modelling errors. Do not forget to implement adequate integrator anti-windup functions, to cope with actuator limits. **Use a sampling rate of 100Hz**. Write a C++ class to implement your controller, such that a single MCU gets the ability to drive more than one luminaire.

Interfacing with PC

To read and write data from/to the Arduino, a simple PC interface can be implemented using the program “serial monitor” in the Arduino IDE. The students should implement a simple character-based protocol to read LDR values in LUX, set LED PWM values, set upper (occupied) and lower (unoccupied) reference values for the local controller, set desk occupancy, turn on/off the feedforward term, turn on/off the anti-windup term, or to perform any other operations that you may find useful for testing/debugging or reporting.

4.2. Evaluation of the first stage:

The first stage will be evaluated through a demonstration of the local illumination control in the middle of the semester (end of October). This stage accounts for 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, and demonstrate the performance of the control system in step changes in the reference set point and under external disturbances.

4.3. Implementation notes:

1. No report is needed for the first stage but the code may be requested for delivery at this stage. Also, it is highly recommended that the information required to write the final report be collected at every stage. In particular, it is very important to collect data from your system to make plots of the different signals in the control system (references, control values, measurements) and compute metrics that show the correct operation of the system. Implement functions in the PC interface that allow you to collect this data.
2. Note that serial communications use precious microprocessor time. Choose messages with short size and a high baud rate. Compute communication delays and verify that the communication time can be accommodated within the available control loop period.
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. Also keep in mind that the serial monitor interface can display data graphically.
4. Add functionalities that facilitate development, testing, debugging, and demonstration of the applications. You may request additional electronics components for such purposes.
5. Always use SI units for pertinent quantities. Check the datasheets to verify the conversions from electrical to physical units.

4.4. Milestones for each lab session:

Although it is not mandatory to follow a strict agenda in the execution of the project, there are some minimum objectives that should be met each week, to ensure a timely execution of the project. But note that objectives are quite hard to achieve during the 90 min of the session, so some items must be prepared at home before the session, and others completed autonomously after the session. Especially for session 3, the students should bring the objectives almost finished because they will have to demonstrate their progress during the session.

Session 1 (Weeks of 28 Sept and 5 Oct) – Intro to the Hardware and Software:

- Reception of the material needed for the project.
- Assembly of the luminaire with the provided equipment.
- Implementation of a function that converts the voltage read at the analog input port to LUX (you must use the expressions of the voltage divider and the log-log characteristic curve provided in the datasheet).

- Development of basic ARDUINO programs to (i) repeatedly read values from the LDR, convert to LUX, and send those values to serial output; (ii) read values from the serial input and set the duty cycle of the LED to this value).

Session 2 (Weeks of 12 and 19 October) – System Assembly and Identification:

- Use a showbox (or similar box) to create a physical model of an office space at a reduced scale. It is important that the box can isolate well the external light.
- Write a program to perform step changes in the LED actuation and collect the LDR signal in LUX and Volt at a high sampling rate.
- Use the steady state values of the curve in LUX to compute the static gain of the system (LUX/Input Units). Note that, theoretically, this gain should be constant.
- Due to the sensibility of the static gain to changes in the box structure, devise and implement a simple method to calibrate the static gain everytime the systems boots. This way the system is always working with calibrated gains, even if the elements inside the box moved due to transportation.
- Use the transient values of the curve in Volt to estimate the time constant $\tau(x)$. Note that, theoretically, this value should depend on the current illuminance. For a better precision in the computation of $\tau(x)$ perform experiments at different illuminance levels using both positive and negative steps.
- Write down a rule that expresses the time constant as a function of the initial and final illuminance.
- Create a class for a system simulator, that can predict the values read by the sensor, given the actuation commands. Evaluate the quality of your simulator by comparing simulated with real responses.

Session 3 (Weeks of 26 Oct and 2 Nov) – Local Controller and Demonstration

- Feedforward Control. Using the knowledge of the static gain computed in the previous session, write and test a feedforward controller for your system. Check the properties of the feedforward controller in terms of accuracy and response time to step changes in the references.
- Feedback control. Implement the feedback controller with anti-windup. Check how it behaves in the presence of external disturbances, in terms of accuracy, overshoot and oscillations (flicker).
- Integrate the feedback controller with the feedforward controller developed in the previous session. Generate appropriate references to the feedback controller to prevent overshoot in response to step changes in the desired values.

During session 3 students must demonstrate the results of the first stage. Points that may be checked:

- Ability to set desired upper and lower reference values for the illuminance, and set/reset the occupation of the desk via the PC interface.
- Ability to control the illuminance to the correct values in the occupied and non-occupied states.
- Ability to keep the desired illumination at the correct level in case of external disturbances (inject light through the window of the box).

Be prepared to show the implemented code an answer questions about the implementation.

4.5. Guidelines to document the first stage:

a) Take pictures of the interior and of the exterior of the box enclosing the luminaires. Make sure that you illustrate the position of the LED, the LDR and the emission / reflection path.

b) Show plots of the steady state characteristic of the system. Show step responses of the system in different illuminance conditions.

c) Characterize the jitter in your control system. How much does the sampling rate deviate from the desired one?

d) Characterize the error in your feedforward controller. Implement a simulator of your system and compute the average mean squared error between the simulation and the actual measurements.

e) Characterize the dynamic characteristics of the feedback controller, in different illuminance conditions (overshoot, damping factor).

f) Illustrate any improvements that you make to the basic feedback controller (feedforward term, anti-windup, etc) with plots of the time responses.

g) Comment the Arduino code for your controller.

h) Characterize the processing time taken by the control computation, serial communications and other computations.

5. Second Stage

5.1. Description

During the second stage the students will collaborate within a group to implement a cooperative distributed (i.e., non-centralized) control system and a multi-task embedded PC-based interface to the system, implemented in a Raspberry PI module.

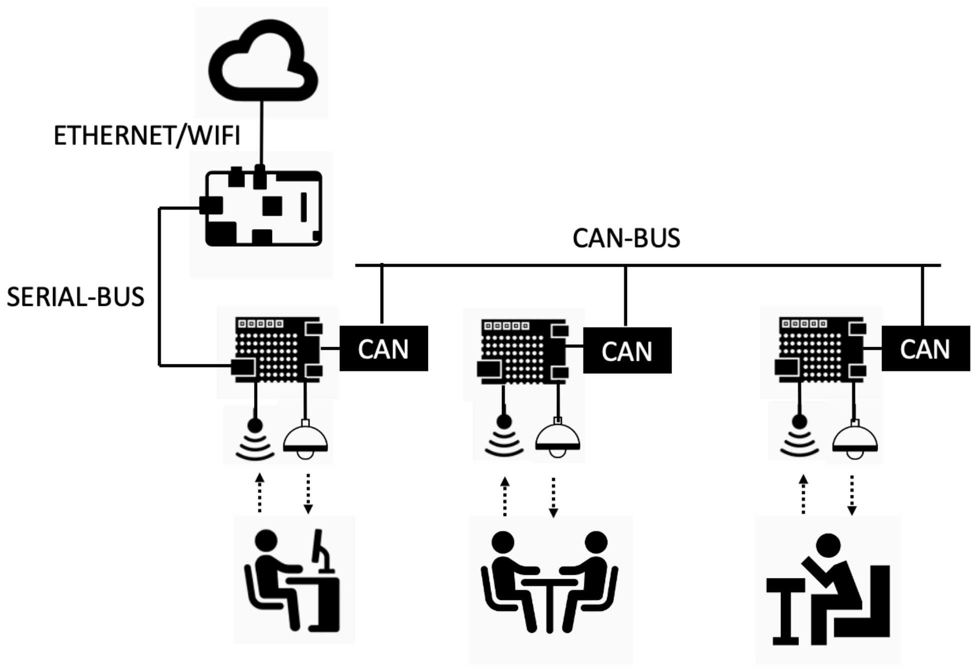


Figure 4 – Diagram of the distributed illumination network.

Three Arduino nodes (luminaires) will be installed in the office model, one for each desk. The luminaires can communicate using a control network (CAN-BUS). The luminaires should exchange messages whenever necessary to jointly minimize energy consumption and maximize user comfort. User comfort is defined as having a luminance on its desk always above or equal to the reference set points (visibility criteria) and low variations of high frequency noticeable to the human eye (flicker criteria). Later will be given metrics to compute these quantities. In this non-centralized cooperative distributed control mode, there is no central node. Ideally all luminaires should have the same code and implement "boot" mechanisms to recognize each other and be able to operate in a network with a variable number of nodes.

The embedded PC interface will connect to one of the luminaires via serial communications. This luminaire will operate as a hub to interchange information between the network and the embedded PC. The embedded PC application has two main purposes: (i) collect information from the control nodes to store data and compute evaluation metrics; (ii) send commands to the control nodes (set points, working modes, etc). Later is given a list of all commands that should be implemented.

We can identify the following main functionalities of the system: (i) inter-node communications; (ii) hub function; (iii) illumination system calibration; (iv) distributed control algorithm; and (v) PC application.

Inter-Node Communications

In cooperative distributed control systems each node typically takes the initiative to send messages to its neighbours and is permanently listening for incoming messages from other nodes. Messages can be directed to any particular node or they can be broadcasted to all nodes. Some messages can indicate a change of state that requires control actions by the neighbouring nodes, while others may be transmitted for logging purposes. Each node's address can be set through EEPROM registers.

Hub Function

A node, if connected to a PC, should enter in hub function mode. This function adds, to the normal operation of a node, the role to communicate with the PC. The communication with PC is made with serial link, whereas the communication with the other nodes is made with CAN-BUS. The hub node is responsible to route information from the other nodes to the PC and vice-versa.

Illumination System Calibration

When a luminaire changes its actuation, it impacts neighbouring luminaires (coupling effect). To realize the global controller, it is necessary to model the effects of the intensity of a luminaire in the measured illuminance of the other(s). 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. This effect is almost linear when LUX units are used for the illuminance, and only a small number of measurements is required to model it. This method should also allow the estimation of the current background (external) illumination.

Distributed Control Algorithm

The distributed controller should be of the "non-centralized" type, i.e., no central master exists. This improves the overall reliability because the whole system can still operate even if one node stops working. Each node will be an independent decision-making unit with a shared "goal": to minimize a global cost function (energy consumption), while satisfying the minimal illuminance levels according to the desks' occupation. Two cost functions will be considered: one with luminaires with identical power consumption, and another with luminaires with different powers. The coefficients of the cost function for each luminaire, which are proportional to its power consumption, can be set via the PC application. A few distributed optimization algorithms will be given in the course lectures and the groups will have to choose which ones can be used and discuss the pros and cons of each alternative.

Embedded PC application

The embedded PC can be seen as a tool for the maintenance of the distributed illumination system and/or for its access from a remote location. It should log the information flowing in the system for diagnosis purposes and allow interaction between a local or remote user and the illumination network, for instance to read the state of any luminaire (occupation, led dimming level, illuminance level), and access/compute system statistics (energy consumption, comfort metrics). The information can be requested by the user in three different modes: actual values, last minute history or real-time stream.

The interaction with the user can both be local (via a console interface) or remote (web). A web server task will provide the user interface functions to a remote client running on a different machine, connected to the internet.

The embedded PC application will be implemented as a C++ multitask application using asynchronous I/O services provided by the Boost ASIO library.

5.2. Performance metrics:

To properly validate an engineering solution, it is fundamental to define appropriate performance metrics, expressing in a quantitative way the requirements addressed in its formulation. For this project, we aim to minimize the energy spent in illumination while providing comfort to the users. The energy is the accumulation (integral) of the instantaneous power along time. Suppose the maximum power of luminaire *j* is denoted as P_j. Then, a formula to compute the energy consumed at each desk is:

$$E_j = P_j \sum_{i=1}^{N} d_{i-1}(t_i - t_{i-1})$$

where *i* is the index of the control samples, t_i is the time in seconds of the *i*-th sample, 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, comfort criteria are 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*) and the measured illuminance (*l*) for the periods when the measured illuminance is below the reference. Let us call this quantity the **Visibility Error**:

$$V = \frac{1}{N}\sum_{i=1}^{N} \max\Big(0, L(t_i) - l(t_i)\Big)$$

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 frequent ups-and-downs of illuminance (flickering) while the reference is at a constant value. A metric to assess this criterion can be defined as the average magnitude of the signal derivatives when it changes sign, during periods of constant occupation. Let us first define the flicker at time t_i as f_i

$$\mathrm{f}_i = \begin{cases} \left(|l_i - l_{i-1}| + |l_{i-1} - l_{i-2}|\right) / \left(2T_s\right) & \text{if} \quad (l_i - l_{i-1}) \times (l_{i-1} - l_{i-2}) < 0 \\ 0 & \text{otherwise} \end{cases}$$

where T_s is the sampling period, l_i is the measured illuminance at time t_i, and $|A|$ denotes the absolute value of *A*. Transient periods due to explicit variation of the reference should be excluded from the formula.

We can now define the **Flicker Error** as:

$$F = \frac{1}{N}\sum_{i=1}^{N} f_i$$

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

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

The computations should be done at all control cycles, i.e. with a frequency of 100Hz.

5.3. Evaluation of the second stage:

The second stage will be evaluated through a demonstration of the final distributed control system by the end of the semester (last weeks of lectures). This stage accounts for 25% to the project grade.

A written report containing both parts, and associated software, shall be delivered two weeks after the final demonstration (just before exams season). The report accounts for 50% of the final grade. All members of the groups must contribute to the report writing. All sections of the report must indicate its main contributor.

5.3. Implementation notes:

Take caution in defining the messages to exchange in the network in order to minimize the communication overhead (e.g., select appropriate baud rates for the existing cable lengths and define short messages). The final report should include an analysis of the times spent on communications. Use C++ classes to implement the distributed control and calibration code.

5.4. The PC Application:

To interface the distributed system with the outside world, groups should develop a C++ application using asynchronous I/O classes from Boost's ASIO library. The application should create a console to allow user interaction via the keyboard and string commands. A specification of the user commands to implement 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 – unoccupied, 1 – occupied.
Set current occupancy state at desk <i>	"o <i> <val>"	"ack" or "err"	<val> is a Boolean flag: 0 – unoccupied, 1 – occupied.
Get lower bound on illuminance for Occupied state at desk <i>	"g O <i>"	"O <i> <val>"	<val> is floating point number expressing illuminance lower bound for Occupied state in desk <i> in lux.
Set lower bound on illuminance for	"O <i> <val>"	"ack" or "err"	<val> is floating point number expressing illuminance lower bound for Occupied

Occupied state at desk <i>			state in desk <i> in lux.
Get lower bound on illuminance for Unoccupied state at desk <i>	"g U <i>"	"U <i> <val>"	<val> is floating point number expressing illuminance lower bound for Unoccupied state in desk <i> in lux.
Set lower bound on illuminance for Unccupied state at desk <i>	"U <i> <val>"	"ack" or "err"	<val> is floating point number expressing illuminance lower bound for Unoccupied state in desk <i> in lux.
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 x <i>"	"x <i> <val>"	<val> is floating point number expressing external illuminance in lux.
Get current illuminance control reference at desk <i>	"g r <i>"	"r <i> <val>"	<val> is floating point number expressing illuminance control reference in lux.
Get current energy cost at desk <i>	"g c <i>"	"c <i> <val>"	<val> is floating point number expressing the current cost of energy
Set current energy cost at desk <i>	"c <i> <val>"	"ack" or "err"	<val> is floating point number expressing the current cost of energy
Get instantaneous power consumption 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 elapsed time since last restart	"g t <i>"	"t <i> <val>"	<val> is floating point number expressing elapsed time in seconds.
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.
Get accumulated visibility error at desk	"g v <i>"	"v <i> <val>"	<val> is floating point number expressing the accumulated visibility error in lux.

<i> since last system restart.			See section 4 – Evaluation Metrics
Get total visibility error since last system restart.	"g v T"	"v T <val>"	<val> is floating point number expressing the total visibility error in lux. See section 4 – Evaluation Metrics
Get accumulated flicker error at desk <i> since last system restart.	"g f <i>"	"f <i> <val>"	<val> is floating point number expressing the accumulated flicker error in lux/s. See section 4 – Evaluation Metrics
Get total flicker error since last system restart.	"g f T"	"f T <val>"	<val> is floating point number expressing the total flicker error in lux/s. See section 4 – Evaluation Metrics
Restart system	"r"	"ack" or "err"	Reset all values and recalibrate.

Get last minute buffer of variable <x> of desk <i>. <x> can be "l" or "d".	"b <x> <i>"	"b <x> <i> <val1>, <val2>, …<val_n>"	Values are returned in a string of comma separated numbers. The string is terminated with the newline character.
Start stream of real-time variable <x> of desk <i>. <x> can be "l" or "d".	"s <x> <i>"	"s <x> <i> <val> <time>"	Initiates a real-time stream of values. Every time a new sample of a certain variable is available, it is sent to the client in a string with the format indicated. <time> is an increasing timestamp in milliseconds.
Stop stream of real-time variable <x> of desk <i>. <x> can be "l" or "d".	"s <x> <i>"	"ack" or "err"	Stops the real-time stream of values.

5.5. Milestones for each lab session:

The milestones of the second phase can be made in parallel by the different elements of the group or made in a different order as the suggested here. The order given is the one most synchronized with the contents given in the theory lectures, but if the group wants to progress faster, the relevant information will be given per request. Again, the objectives are too challenging to complete during session time (90 min) and autonomous work will be most likely required.

Session 4 (Weeks of 9 and 16 Nov) – Implementation of the Distributed Network

Place all luminaires in the same box (e.g. choose the "best" on your group). Test the performance of the local controllers in the shared environment (non-cooperative decentralized control). Can they achieve the desired performance?

Assembly of the CAN-BUS network. Connect all luminaires in a CAN-BUS network. Write and test a program to send and receive simple messages between Arduino nodes.

Develop network "wake-up" procedures so that all nodes are aware of each-other and boot properly.

Implement the Hub function. Write and test a program on the Arduino to detect if it is connected to a PC and, if true, route messages between the PC and other Arduino nodes. Write a test program in the PC to send/receive data to/from the Arduino hub via serial communications.

Session 5 (Weeks of 23 and 30 Nov) – System Calibration and Distributed Control

Calibration of the distributed system. Write a program to be deployed in all Arduinos to make them synchronously turn on and off the luminaires. This code then should be used to calibrate the self- and cross-coupling gains between the several nodes in the distributed system. This procedure should be done every time the system wakes up to prevent errors in calibration due to movement of the elements inside the box.

Implement the distributed controller given in the theory lectures using C++ classes. Define appropriate communication protocol to exchange the required messages between the nodes. Deploy and test your code in the Arduinos. Show how the controllers react to changes in desk occupation and external disturbances. Consider both cases of identical or different cost in the energy minimization criteria. Compare with the non-coordinated controller tested in the previous session.

- **Session 6 (Weeks of 7 and 14 Dec) – PC Interface and Demonstrations**

PC Interface. Write the code to accept commands from the user and show the results of the operations in a text-based console (see Table in 5.4).

Data collection and statistics. Implement data structures to store, process data and compute some statistics (e.g., the average). Implement the last-minute buffer functionality. Test and decide where to implement these functions: in each Arduino node or in the PC?

Implement the streaming functionality. Devise ways to store collected data to files in order to create graphics that illustrate the performance of the cooperative vs the non-cooperative distributed controller.

Demonstrate the ability of the PC application to interface with the network. Did you implement all functions? Demonstrate the ability of system to properly wake up and calibrate the system. Demonstrate the ability of the system to properly control the illuminance of each desk in a cooperative way. Demonstrate the ability of the system to compute energy saving solutions both for identical and different costs of energy in the luminaires. Demonstrate ability of the system to react properly to external illumination. Be prepared to show the implemented code an answer questions about the implementation.

5.6. Guidelines to document the second stage:

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

a) (CAN-BUS Communications) Indicate the average time required to send a request message and receive an answer.
b) (C++ Classes, Sockets, Parallelism) Indicate the C++ classes that are most relevant for coding the PC interfacing to the Arduino(s).
c) (Tasks) List the number of tasks used to implement the PC application. Describe the functions of each task.
d) (Concurrency) Indicate the methodologies used in the PC application to implement parallelism in order to execute the different tasks.
e) (Distributed Control) Describe the implementation of the cooperative distributed controller.
f) Perform experiments to compare 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 create a table with their performance metrics.

6. Report

Write a report describing the problem, the adopted solutions, the obtained results, and discussing the pros and cons of your design choices. The report must be complete but succinct, with less than 20 pages including graphics and references. Avoid redundancies and overly technical content. Try to summarize as much as possible but do not leave out the important issues. Graphics should be self-contained, i.e., fully labelled and with complete captions.

– ***Enjoy the project*** –