Master Degree in Electrical and Computer Engineering
2018/2019 – Winter Semester

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

PROJECT

Real-Time Cooperative Decentralized Control of a Smart Office Illumination System

Prepared by

Alexandre Bernardino

João Pedro Gomes

Instituto Superior Técnico

Department of Electrical and Computer Engineering

Scientific Area of Systems, Decision and Control

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 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, to experiment and advance on points not requiring lab access. The equipment must be returned by the end of the semester.

Students have to perform two demonstrations of the project: one in the middle of the semester (end of October, granting 25% of the project grade) showing the progress of the first stage, and one at 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 presence of the student in the lab is mandatory unless a justification is provided. Unjustified absences may be penalized.

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. 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 bring to the lab a flash drive** to store the results obtained during 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 Charger (5V) and cable (micro USB B) for raspberry pi 3 power supply.
- 1 SD card for raspberry pi 3
- 1 Ethernet cable
- 2 Resistor 3.3KOhm

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

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

1. Introduction

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 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.

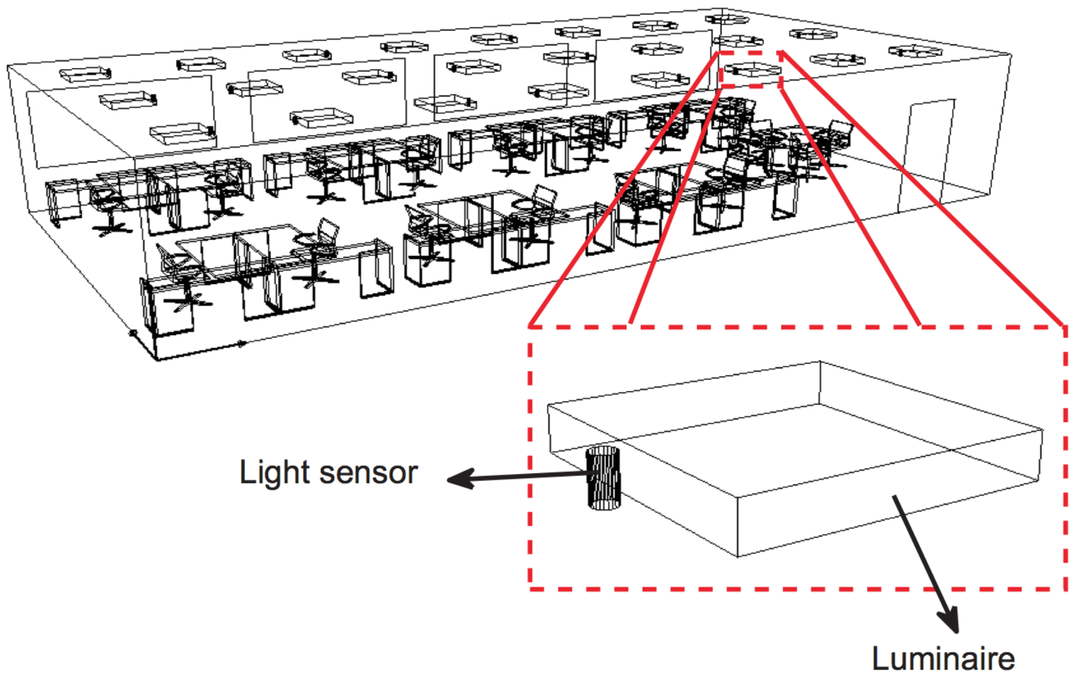


Figure 1. Scenario envisaged in the project. Each desk is equipped with a luminaire, light sensor and presence sensor. The control of the lighting attains fixed levels of illumination at the desk plane (high for occupied desks and low for unoccupied desks) while minimizing the global energy consumption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

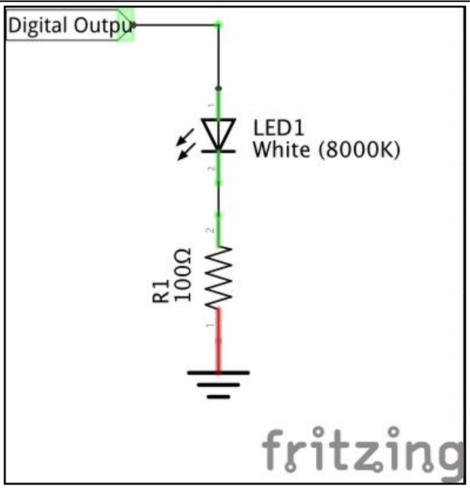
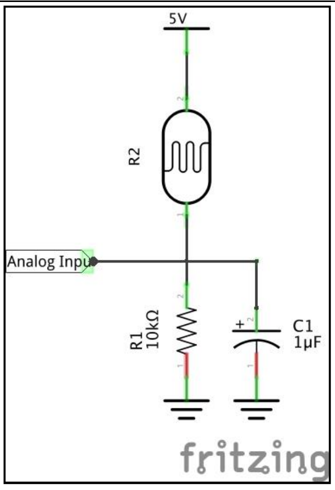
2. Objective

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, a luminance sensor, a presence sensor, as well as computational and communication elements. 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. 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 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, 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

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”.

In addition, a “contact switch” should be implemented using two wires to allow a simple mode of interaction with the user. This “switch” will be used to simulate an occupancy sensor: each contact (touching the ends of the two wires) will toggle the occupancy state of the desk (occupied vs non-occupied).

The distributed 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, if needed. The box should be large enough to contain the 2 luminaires, but not too large. If the box is too large, the LED intensity may 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 reduced scale office illumination control system with two individually controlled luminaires. Each luminaire has an individual local controller that is unaware of the existence of other luminaires. Each luminaire can be decomposed in the following modules: (i) the illuminance measurement system, (ii) the LED actuation system, (iii) the desk illumination system's dynamical model, (iv) the individual luminaire controller, and (v) a simple interface with a PC using the Arduino Serial Monitor. In the individual luminaire controllers, no explicit communication is allowed between different luminaires, and each luminaire only cares about its own desk.

The Illuminance Measurement System

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. In order to have a one-to-one mapping consider the average resistance value at each illuminance level (in logarithmic units).

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, so that the switching does not have a significant influence the luminance measurement signal. Furthermore, the variation of the analog signal should be fast enough to prevent noticeable flickering. A frequency of 100 Hz should be good enough.

The Desk Illumination System's Dynamical Model

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, the desk illumination system's dynamical model reflects how a luminaire's command (LED actuation – Plant input) influences the irradiance at the desk (LDR measurement – Plant output), but in amplitude and time/frequency. 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 desk may have a different dynamical model not only because the luminaires may be different, but also because the light paths travelled in each case are different. Also, 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 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 Individual Luminaire Controller

The individual luminaire controller (a.k.a. local controller) should be implemented as a PID controller with feedforward. 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, keeping in mind that a single MCU may drive more than one luminaire.

Interfacing with PC

To read and write data 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, references for the local controller, and desk occupancies, 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 demonstration 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. However, 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.

- **Session 1 (26 – 28 Sept):** Assemble the luminaire with the provided equipment and write basic ARDUINO programs to (i) repeatedly read a value from the LDR and send to serial output; (ii) read a value from the serial input and set the duty cycle of the LED to this value; (iii) detect changes in the two-wire switch and toggle the state of the LED (ON-OFF).
- **Session 2 (3 – 5 Oct):** Using a shoebox (or similar) simulate an office space at a reduced scale. Calibrate the LDR measurement system to provide measurement in LUX (use the log-log characteristic curve provided in the datasheet). With the box closed, show that, in steady state, identical increments in the LED actuation correspond to identical increments in LUX (the system has a constant DC gain).
- **Session 3 (10 – 12 Oct):** Write a program to perform step changes in the LED actuation and collect the LDR signal (LUX). Assume that the system behaves locally as a first order system $G(s)=K_0/(1+s\tau)$. Estimate the static gain and the time constant. Repeat thesesexperiments at different initial illuminance levels

using both positive and negative steps. What can you conclude about the linearity of the dynamical system? Can you write down a rule that expresses the time constant as a function of the initial and final illuminance?

- **Session 4 (17-19 Oct):** Implement the feedforward controller. Check if the attained illuminance is close to the desired one. Implement the feedback controller. Check how it behaves in the presence of external disturbances.

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. In particular, implement a predictor of your system and compute the average mean squared error between the prediction and the 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.

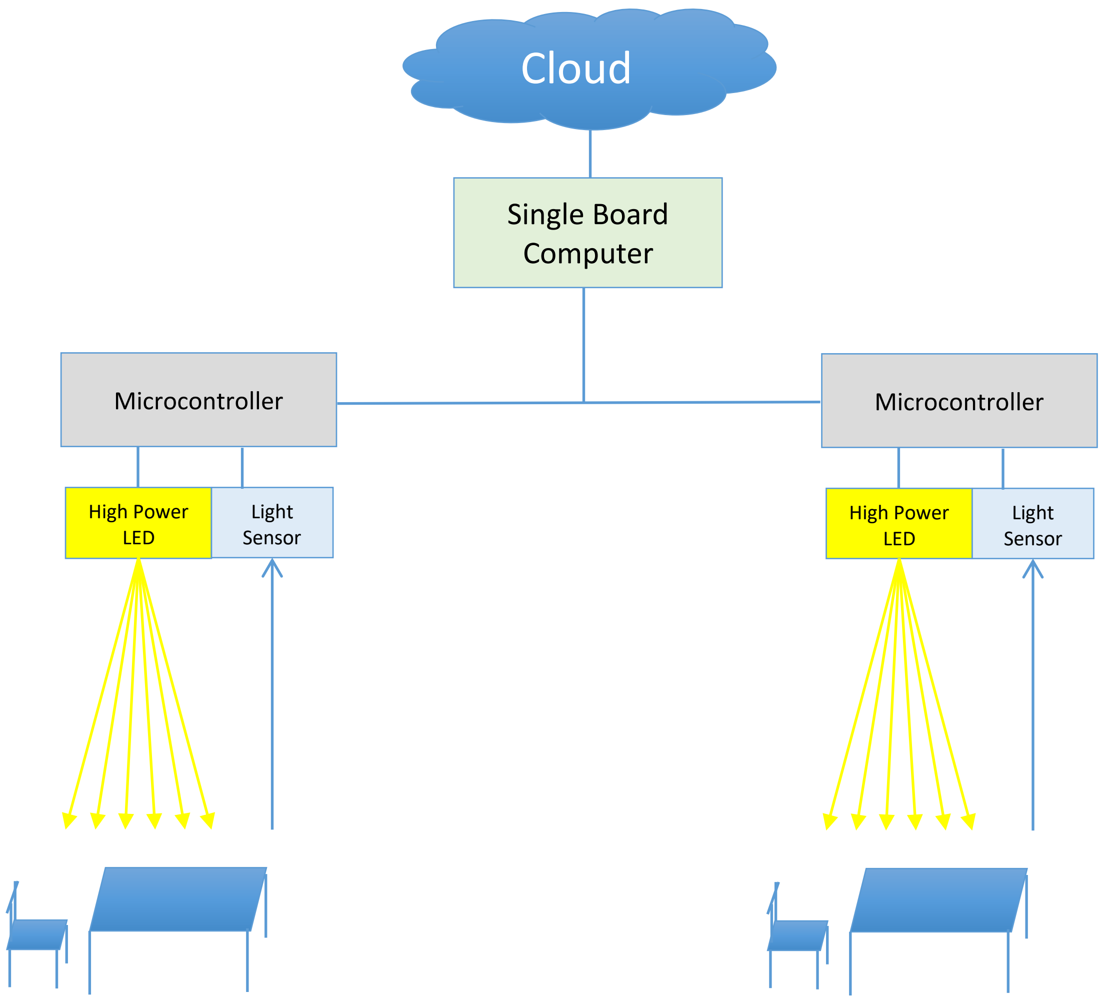
5. Second Stage

5.1. Description

During the second stage the students will implement a cooperative distributed (i.e., non-centralized) control system and an embedded PC interface to the system.

The Arduino nodes (luminaires) will communicate using a control network (I2C). The objective of the distributed controller is to minimize energy consumption and maximize user comfort (luminance always above or equal to the reference set points and low flicker). In this non-centralized cooperative distributed control mode, each luminaire can communicate with its neighbours, and no central master exists.

The embedded PC interface will consist of a Raspberry Pi (RPI) 3B single-board-computer connected to the Arduinos via the I2C link and to the intranet via WiFi. The Raspberry PI will implement a server providing an interface between an application client and the control nodes, to fulfil two main purposes: (i) collect information from the control nodes to store data and compute evaluation metrics; (ii) send information to a client application for monitoring, diagnostics and visualization.



We can decompose the full system into the following modules: (i) inter-node communications; (ii) bus monitoring; (iii) illumination system calibration; (iv) distributed control algorithm; and (v) data server.

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. These messages do not have to be periodic. However, for the purpose of collecting data each node in this project will broadcast its state permanently, i.e., at each sampling period. This will be done using the I2C bus in multi-master mode (all nodes are master and slaves simultaneously). 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.

Bus Monitoring

Due to limitations on its I2C driver, the RPI cannot write to the I2C bus while other nodes are writing — it can only read from the bus. In fact, it does not have I2C collision detection, thus cannot be used in Multi-master configurations. So, the RPI will connect to the I2C bus in read-only mode and collect all messages transmitted in the bus for analysis. This can be done in two ways: (i) use two general purpose digital pins to sample the signals in the bus (sniffing) and convert them to a stream of bytes; or (ii) use the Slave-only I2C interface pins and read the messages on the bus. You can use the PIGPIO library to implement these functions.

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 a change in one 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 background (external) illumination.

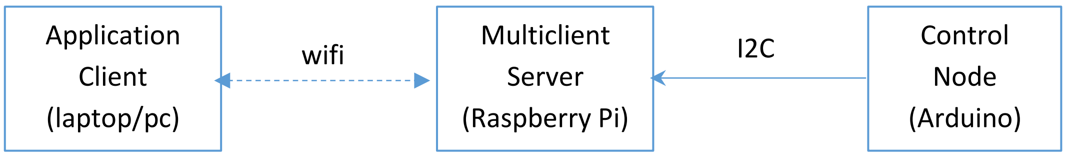
Distributed Control Algorithm

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 pre-specified in a fixed set, and configured using shunts on available digital ports. 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.

Data Server

In addition to collecting data from the illumination network, the Raspberry PI will implement a server providing an interface for an application client. The client application is running on a different machine, connected to the local WiFi network, and can request information to the server.

The client application should be able to read the state of any luminaire (occupation, led dimming level, illuminance level), 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.


Application Client (laptop/pc)
wifi
Multiclient Server (Raspberry Pi)
I2C
Control Node (Arduino)

The RPI server functions and server-client communications will be implemented as a C++ multithreaded application using asynchronous I/O services provided by the Boost ASIO library.

5.1. 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 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_{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 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 value at time t_i as f_i

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

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 **Comfort Flicker as**:

$$\mathrm{C}_{flicker} = \frac{\sum_{i=3}^{N} f_i}{N}$$

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

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

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 week of lectures). This demonstration 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.

5.3. Implementation notes:

Although the default mode of I2C is one-master-multiple-slaves, in this project the bus should be operated as multi master, where any node can take the initiative of communicating with its neighbours whenever necessary. However, the Wire library for I2C provided with the Arduino has some caveats in multi-master mode when the master-receive functions are used. Thus, if using the Wire library, only master-transmit functions should be used.

Exploit all functionalities provided by the communication functions, through proper configuration, 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 C++ server:

To interface the distributed system with the outside world, groups should develop a C++ server using the TCP-IP protocol and asynchronous I/O classes from Boost's ASIO library. 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 s <i>”	“s <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>”	“r <i> <val>”	<val> is floating point number expressing illuminance control reference in lux.
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. Assume each led nominal power = 1W.
Get accumulated comfort error at desk <i> since last system restart.	“g c <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 c T”	“e T <val>”	<val> is floating point number expressing the total Comfort Error in lux. See section

Get accumulated comfort flicker at desk <i> since last system restart.	“g v <i>”	“v <i> <val>”	<val> is floating point number expressing the accumulated Comfort Flicker in lux/s. See section 4
Get total comfort flicker since last system restart.	“g v T”	“v T <val>”	<val> is floating point number expressing the total Comfort Flicker in lux/s. See section
Restart system	“r”	“ack”	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”	Stops the real-time stream of values.

5.5. Milestones for each lab session:

- **Session 5 (31 Oct – 2 Nov):** Assembly of the I2C network. Write and test a program to send and receive simple messages between Arduino nodes. Configuration of the RPI.
- **Session 6 (7 – 9 Nov):** RPI I2C communications. Write a small program on the RPI that reads data from the I2C bus. Write a small program that sends data on the RPI master I2C interface. Connect the GPIO pins for both interfaces and test I2C data communication on the RPI.
- **Session 7 (14 – 16 Nov):** Calibration of the distributed system. Write a distributed 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 distributed system. Connect the RPI on the I2C network and test your code to read and interpret the messages circulating on the bus.

- **Session 8 (21 – 23 Nov):** Implementation of the distributed controller. Deploy your code in the Arduinos and show how the controllers react to changes in desk occupation and external disturbances.
- **Session 9 (28 – 30 Nov):** Data collection server. Implement data structures to store data read from the network and compute some statistics (e.g., the average).
- **Session 10 (5 – 7 Dez):** Client-Server Code. Implement the C++ server. Show that it correctly executes at least 1 command.
- **Session 11 (12 – 14 Nov):** Final Tuning.

5.6. Guidelines to document the second stage:

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

a) (I2C 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 server running on 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 process requests from multiple clients.
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*** –