

Engineering an Internet of Things-based data collection system for Machine Learning algorithm use

Arthur Kho Caayon

Bachelor's Thesis
Degree Programme in Business ICT
2018



Author(s) Arthur Kho Caayon	
Degree programme BITE	
Report/thesis title Engineering an Internet of Things-based data collection system for Machine Learning algorithm use	Number of pages and appendix pages 20 + 14
<p>Every physical or virtual activity can be turned into data. And the challenge is two-fold. First, in identifying and capturing events as they happen. Second, in converting them into a manipulable form.</p> <p>This project is an attempt in tackling the challenge posed by a specific scenario. That of how to come up with ways and means of identifying and capturing the artifact of the passive interaction between a group of students and a computerized “intelligent system” where the context for the passive interaction is inside a classroom.</p> <p>The solution is through the use of an array of electronic sensors that is controlled by a smart system. The idea is to use the sensors to capture events of interest. The smart system then converts those sensor readings into something that a Machine Learning algorithm can process.</p> <p>A proof-of-concept was produced. The scale and scope of the solution was deliberately constrained to that of only the hardware side.</p>	
Keywords Data Science, Machine Learning, Internet of Things, electronic sensors, microcontrollers, emdedded systems	
Author(s) Arthur Kho Caayon	
Degree programme BITE	

<p>Report/thesis title Engineering an Internet of Things-based data collection system for Machine Learning algorithm use</p>	<p>Number of pages and appendix pages 20 + 14</p>
<p>Every physical or virtual activity can be turned into data. And the challenge is two-fold. First, in identifying and capturing events as they happen. Second, in converting them into a manipulable form.</p> <p>This project is an attempt in tackling the challenge posed by a specific scenario. That of how to come up with ways and means of identifying and capturing the artifact of the passive interaction between a group of students and a computerized “intelligent system” where the context for the passive interaction is inside a classroom.</p> <p>The solution is through the use of an array of electronic sensors that is controlled by a smart system. The idea is to use the sensors to capture events of interest. The smart system then converts those sensor readings into something that a Machine Learning algorithm can process.</p> <p>A proof-of-concept was produced. The scale and scope of the solution was deliberately constrained to that of only the hardware side.</p>	
<p>Keywords Data Science, Machine Learning, Internet of Things, electronic sensors, microcontrollers, emdedded systems</p>	

Table of contents

1. Introduction	1
1.1. Deliverables and objectives	1
1.2. Scope and delimitations.....	2
2. Related Research.....	2
2.1. The Internet of Things	2
2.2. Data	2
2.3. Machine Learning	3
3. Implementation	3
3.1. Visualizing the project using a physical model.....	3
1. Arguments and motivation for the sensor placements	5
3.2. Electronic sensors.....	5
3.3. Microcontrollers.....	8
3.4. Assembling the collector units	10
4. Results.....	13
5. Discussions.....	15
References	18
Appendices	19
Appendix 1. Intelligent System Classroom Assistant For Instructors (ISCAFI)	19
1. Overview	19
2. Class models	19
3. Use case diagrams	21
4. Sequence diagrams.....	25
5. Storyboard	29

1. Introduction

The 'Intelligent system classroom assistant for instructors' (ISCAFI) is my interpretation of a subset of Amir Dirin's vision of the 'classroom of the future', specifically the part where intelligent systems assist instructors in the classroom and facilitate their task of giving lectures/classes.

As an example, when a class is scheduled to begin shortly, this intelligent system will unlock the classroom door, switch on the lights—to let students in—and then set the thermostat to 20°C. When the instructor arrives, this intelligent system will prepare the desktop computer, warm up the projector, initialize the recording system and the livestream system, and so on. (*See Appendix 1 for the documentation*)

As far as we're concerned, we do not care that this system is controlled and directed by an *intelligent system*; or that its "intelligence" comes from a rules-based AI; or that the automatic switching on and off of equipment in the classroom is because of smart objects. All we care about is that when it's cold outside, the classroom is warm; or when the projector is in use, the front lights automatically switch off so that we can see the presentation, etc. And that's how ISCAFI is envisioned to be: efficient, useful, and ubiquitous.

But in spite all these, ISCAFI lacks one important thing--it is not truly intelligent. It's because it has no ability to learn from its environment. It has no data collection and feedback mechanism. It may act like it's intelligent but in truth it is merely following a recipe, blindly acting upon what its program is telling it to do.

So this is where this project comes in. The goal is for the ISCAFI to enable for itself to gain the capability to sense and notice historical patterns in its environment. This project is aimed at visualizing a path for taking the "intelligent" in the intelligent system to a higher level by replacing its intelligence module, from a static one, to that of an adaptive one (i.e., a system that learns) through machine learning.

1.1. Deliverables and objectives

The project spans two systems, the deliverables are:

- Charts and diagrams of the ISCAFI.
- The documentation of this project.

The objective is to be able to have a grasp of the big picture, to see how to fit these two systems together; and to produce a physical proof-of-concept for this project (not the ISCAFI).

The proof-of-concept, in turn, has three objectives:

1. To demonstrate, either through simulation or Wizard of Oz prototyping (for the IS-CAFI), that the scenario is viable through physical interaction with actual IoT devices.
2. To test that the system's sensor components do collect data.
3. To test that the collected data are transmitted via the network and onto the cloud.

1.2. Scope and delimitations

For the ISCAFI portion, only the planning and system documentation will be included. There are no physical artifacts or implementations.

For the thesis project, only a subset of the functionalities will be implemented (described in Chapter 5. Results); and since this is a proof-of-concept it will be barebones, have no polish, no robustness, nor any notion of security—these considerations will, perhaps, be relegated to some project sometime in the future.

Because the focus of this project is the data collection part, only a cursory treatment of machine learning theories and algorithms will be tackled, although actual data collection will be via actual IoT-enabled sensors though.

2. Related Research

2.1. The Internet of Things

The term IoT refers to network that is composed of physical objects connected to the Internet. These objects are identified and recognized just like all the other everyday devices and computers that we use. (Saleh & Bouhai 2017). This "Thing" receives input from our physical world and transform them into data; then sends those data via the networks for storage and processing. The reverse path is also true, it can produce outputs into our physical world in the form of actions through actuators; some of these outputs could be triggered by data that has also been collected and processed off the Internet. (McEwen & Cassimally 2014).

Because the maturation of sensor technologies over the past decades have accelerated, it has made all of the Internet of Things even much more possible; and in parallel, we see the tremendous progress in the areas of wireless networks, data processing, and machine learning at the same time. These advances have shifted the focus from low level data collection to high-level data collection, to inference, and even to "cognicity (Dormehl 2017)".

2.2. Data

Data is what drives our modern digital universe forward. It is the reason why giant corporations like Facebook or Google offer their services for free because our personal data

has become the commodity. Simply put, this currency—the datum—is a piece of information that is an abstraction of a real-world entity (Kelleher and Tierney 2018), that in the same way our Facebook profile is an abstraction of ourselves from the real world into the digital world.

2.3. Machine Learning

Machine learning is the field of study that develops the algorithms enabling computers to identify and extract patterns from data. It involves a broad two-step process. First, an ML algorithm is applied to a data set to identify useful patterns in the data. These patterns can be represented in a number of different ways such as decision trees, regression, or neural networks. These representation of patterns are known as models. In essence, machine learning algorithms create models from data, and each algorithm is designed to create models using a particular representation like those three examples mentioned above.

Second, once a model has been created, it is then used for analysis. In some cases, the structure of the model is what is important, because, for example, it can reveal what the important attributes are in a given domain. In other cases, a model is used to label or classify new examples. A common example is of a spam-filter model. Its main purpose is to label new email as either spam or not spam—rather than to reveal the defining attributes of a spam email.

But machine learning is not always smooth sailing. Practitioners face real challenges, foremost among them is in finding the algorithm whose learning bias is the best match for a particular data set.

(Kelleher and Tierney 2018)

3. Implementation

3.1. Visualizing the project using a physical model

My child's toy barn was repurposed as a classroom model so that the placements of the fixtures and equipment could be visualized, which in turn, would allow the planning of sensor placement accordingly.

The classroom's layout is imagined to be like so: that the main door is at the front; the projector slightly behind the row of the front lights; and the HVAC vent at the back of the classroom.

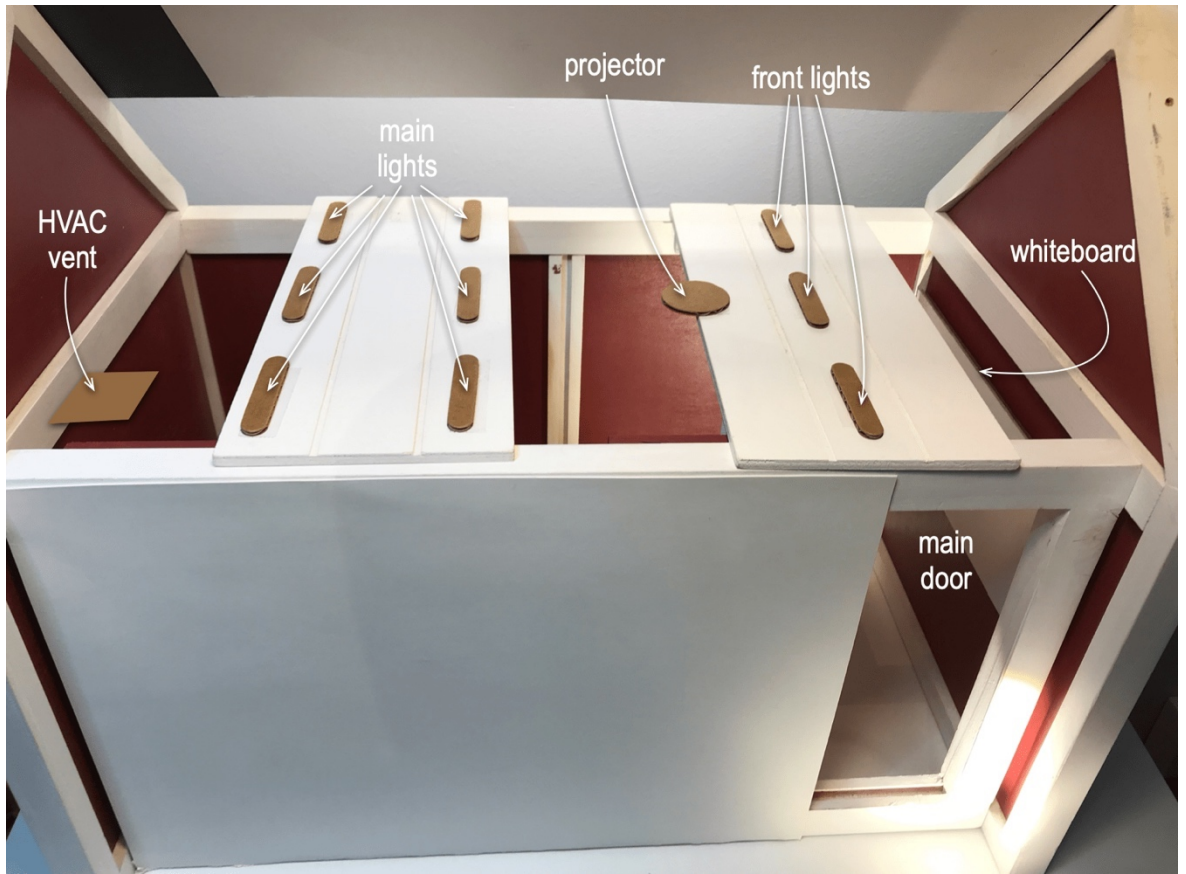


Figure 1. Layout of fixtures and equipment whose data will be collected by the system

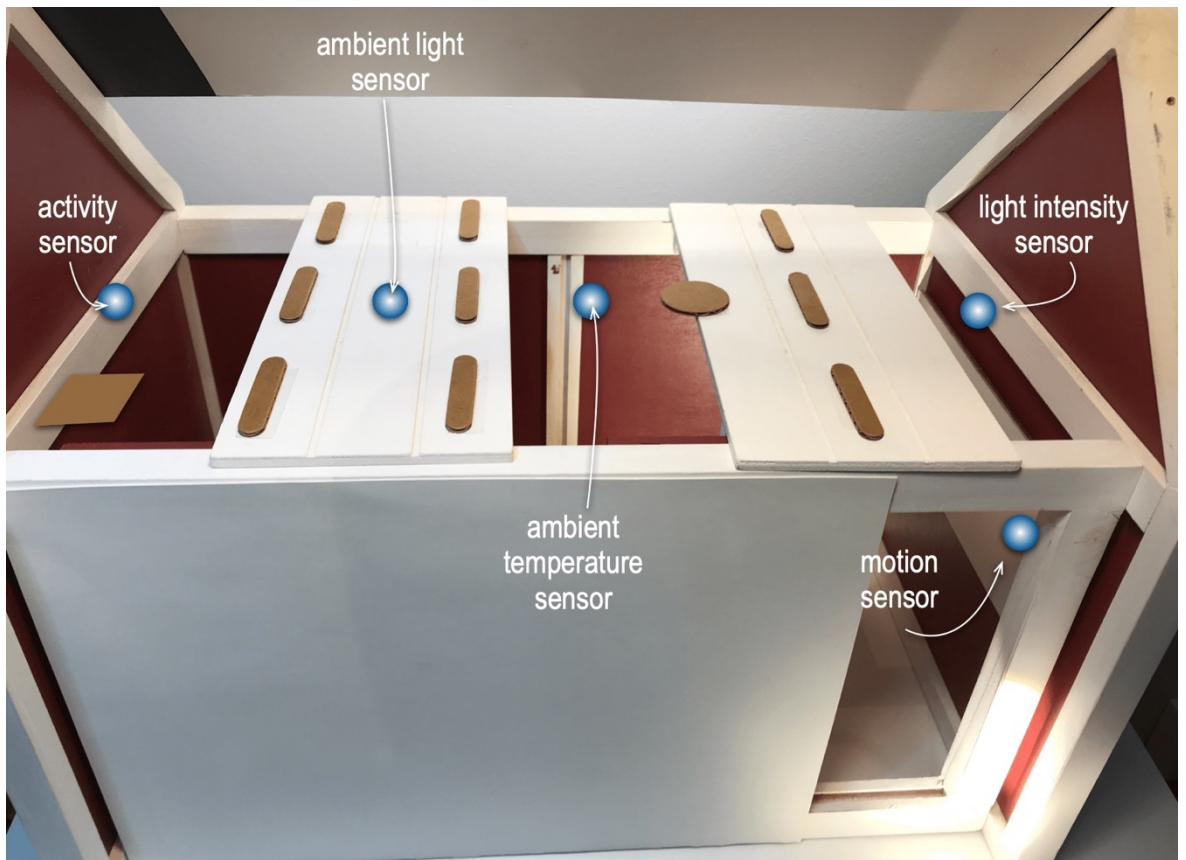


Figure 2. Sensors and their deployment placements

Having done so, mapping the placement of the sensors came next. Common sense was followed in their respective placements, and not according by professional or regulatory standards. So there is no way of having a measurable certainty if these placements are optimal. But that is beside the point.

1. Arguments and motivation for the sensor placements

Motion sensor is by the door so that the following can be logged:

- Ingress and egress. (To determine the frequency of people coming in and out).
- Opening and closing of the door. (To determine the frequency at which they are closed or open at any given lecture period).
- Activities outside the classroom. (To determine how long students wait outside before the door is unlocked by the ISCAFI to let them in, for example, before a lecture).

Ambient temperature sensor is in the middle of the classroom to measure temperature variation across a 24-hour period; or temperature variation across a lecture period. (Probably to forecast energy savings).

Ambient light sensor is a third of the way from the back. To measure ambient lighting conditions across a lecture period. (Probably to determine if more lighting fixtures are needed in that classroom or not).

Light intensity sensor is directly in front of the projector's beam. To record the frequency and length of projector usage. (Probably to forecast maintenance schedule of the device or to project the end of life of the projector's lamp).

Activity sensor is at the back of the room facing to the front. To record general movement of the room occupants over a lecture period. (Probably to see an emerging pattern over time to determine how chairs and tables should be optimally arranged).

3.2. Electronic sensors

An electronic sensor is an electronic device that detects input from the physical environment. It can also respond to the same. The response (or the output) is a signal that is either processed right away or is sent via the network for storage or for further processing. This project utilizes the latter.

Below are the list and the types of electronic sensors used for the implementation:

- LinkerKit Passive Infrared motion detector (PIR motion sensor)
- SparkFun TEMA6000 Breakout (ambient light sensor)

- ThingSee One (ambient temperature sensor)
- Photoresistor (light intensity sensor)
- Raspberry Pi camera module (activity sensor)

LinkerKit Passive Infrared (PIR) motion detector

Passive infrared sensors are motion-detecting devices. They detect motion by sensing light in the infrared spectrum. So when a human or otherwise comes into a PIR's field of view, it senses the subtle temperature changes on that figure. It is those changing patterns that are interpreted by the PIR sensor as a movement.

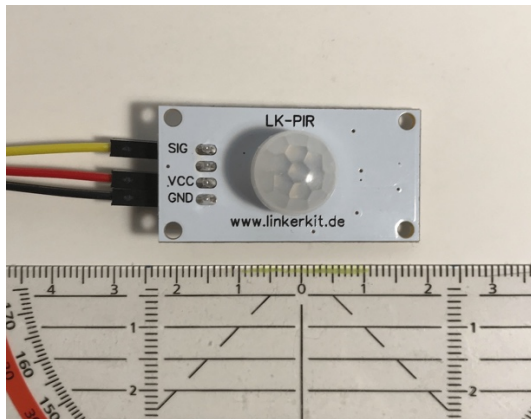


Figure 3. PIR motion detector

SparkFun TEMT6000 Breakout

Ambient light sensors detect and measure the amount of ambient light in its surroundings. It only detects light in the visible spectrum, unlike that of the photoresistor.

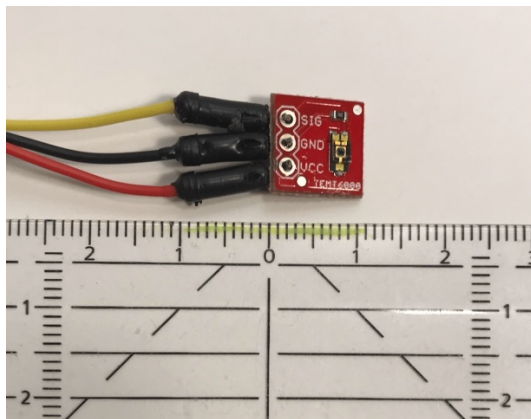


Figure 4. TEMT6000 ambient light sensor

ThingSee One

ThingSee One is an IoT developer device created by a Finnish company called Haltian based in Oulu. As a developer device, it comes built-in with a wide range of sensors including GPS, 3D accelerometer, a magnetometer, a gyroscope, a humidity sensor, a pres-

sure sensor, an ambient light sensor, and a temperature sensor. It can connect to the network via Wi-Fi or via cellular network.



Figure 5. ThingSee One IoT developer device

Photoresistor

This sensor is called by many names: photodetector, photocell, or photoconductive cell; but it is a type of resistor. Its resistance changes when exposed to light, so the amount of resistance depends on the amount of light hitting it.

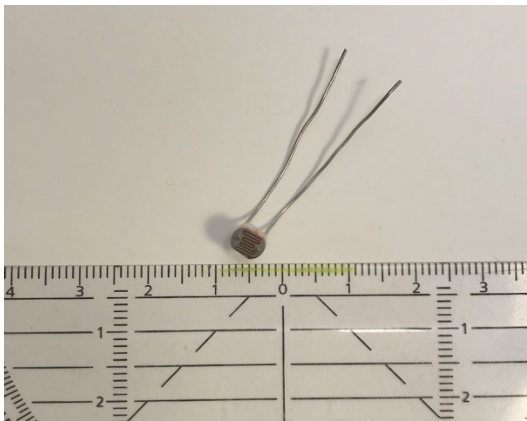


Figure 6. Photoresistor light sensor

Raspberry Pi camera module

This small HD camera module only works with the Raspberry Pi.

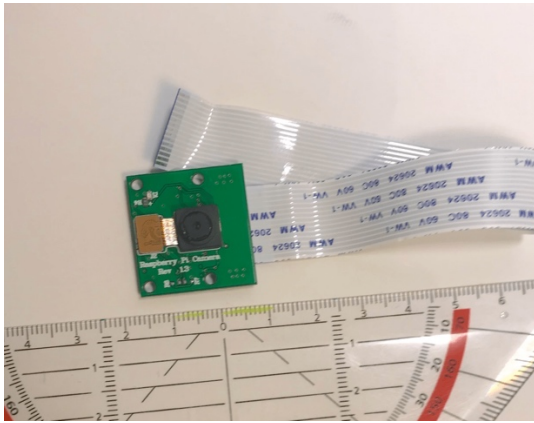


Figure 7. Raspberry Pi HD camera module

3.3. Microcontrollers

A microcontroller is a simple computer with a very small form factor. Because it is a computer, it has a CPU (which executes the programs), a RAM (which stores bits), and input and output capabilities. But unlike the normal computer—able to run multiple programs at a time, a microcontroller can only run one program at a time (Di Justo 2015). Which is why the former is called a *general purpose* computer and the latter a *special purpose* computer.

But before one can get started developing with a microcontroller, one needs a development board and an IDE. A development board is a printed circuit board (PCB) with circuitry and hardware designed to facilitate experimentation with the microcontroller board's features. The development boards come with a processor, memory, chipset, and on-board peripherals. These saves time from messing with jumper wire connections. (“Different Microcontroller Boards And Its Applications”, n.d.)

Below are the different types of microcontrollers used for the implementation:

- Arduino Uno
- ESP8266
- Particle Photon
- Raspberry Pi

The reason why I used several different platforms is because I wanted to simulate a system with as much variety as possible. And because each of these microcontrollers have their own specific ways of doing things, it also serves as a practical experience for myself handling these different platforms.

Arduino Uno

The Arduino Uno is a microcontroller board with no built-in Wi-Fi capability. But it is the easiest to use.



Figure 8. Arduino Uno microcontroller board

ESP8266 Adafruit Huzzah Breakout

The ESP8266 is a microcontroller with built-in Wi-Fi capability. The Adafruit Huzzah ESP8266 is a development board which uses the ESP8266 as its microcontroller. This particular board doesn't come with headers, which means that soldering is needed to attach the pins.



Figure 9. Adafruit Huzzah Breakout ESP8266

Particle Photon

The Particle Photon is a microcontroller with built-in Wi-Fi capability. Like the ESP8266 Adafruit Huzzah, it is designed to be an IoT device. This board happens to have the pin headers already attached.

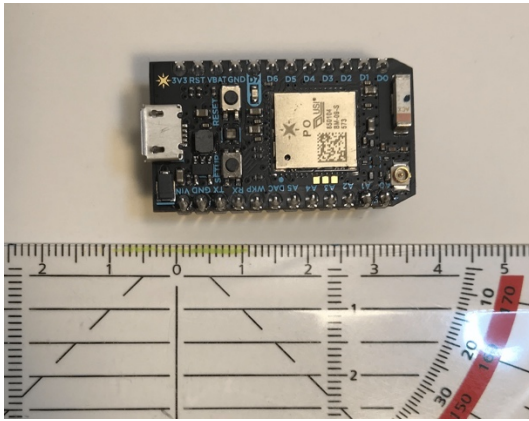


Figure 10. Particle Photon IoT development board

Raspberry Pi

Although the Raspberry Pi is not a microcontroller, I included it here nonetheless. The Raspberry Pi is actually a general purpose computer in that it can run several programs at the same time. It even runs Linux as its OS. But its size is so small—credit-card size, that it is usually lumped along with the microcontrollers.

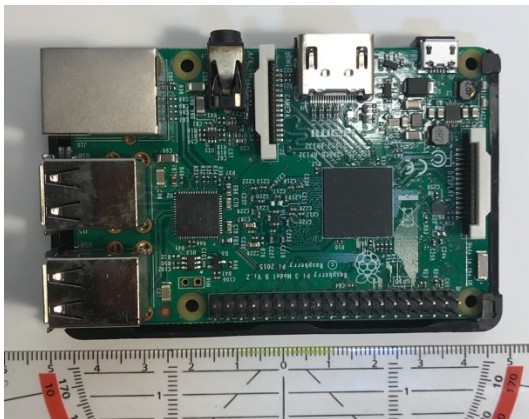


Figure 11. Raspberry Pi

3.4. Assembling the collector units

With the electronic components in place, I set to assemble each one of them to form the data collector units. This consists of connecting them to the correct pins and writing the programming code (called the sketch). So below are their circuit schematics and/or connection points. For the schematics, although I depict the Arduino Uno as the reference microcontroller, but in actuality, I could be using other microcontroller platforms (e.g. ES-P8266, Photon Particle, ThingSee One, or Raspberry Pi).

Motion sensor

The PIR sensor needs to connect its output to the microcontroller's digital input.

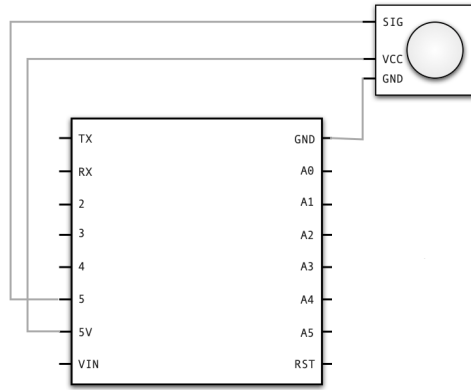


Figure 12. Schematics of the PIR sensor

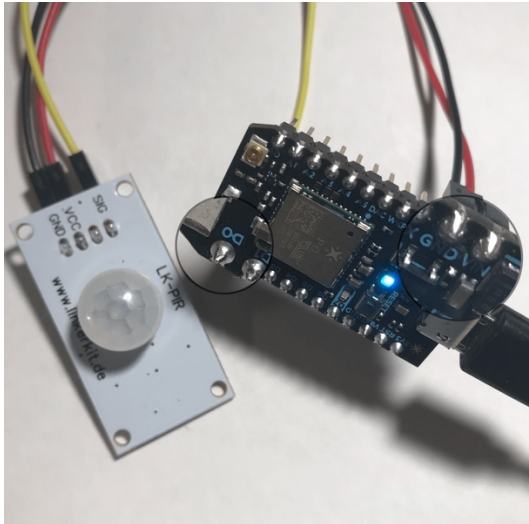


Figure 13. PIR sensor assembly on a Particle Photon microcontroller

Ambient temperature sensor

There was no need to assemble this circuitry here because the *ThingSee One's* temperature sensor was used.

Ambient light sensor

The photoresistor needs to connect to an analog input pin.

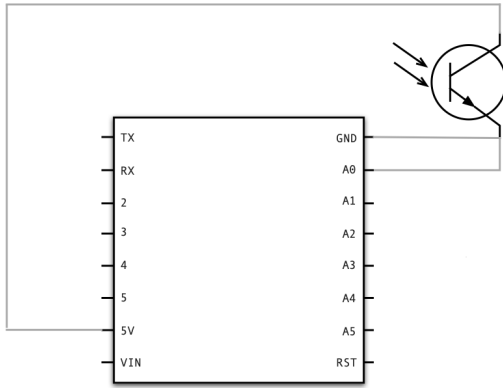


Figure 14. Schematics of the phototransistor

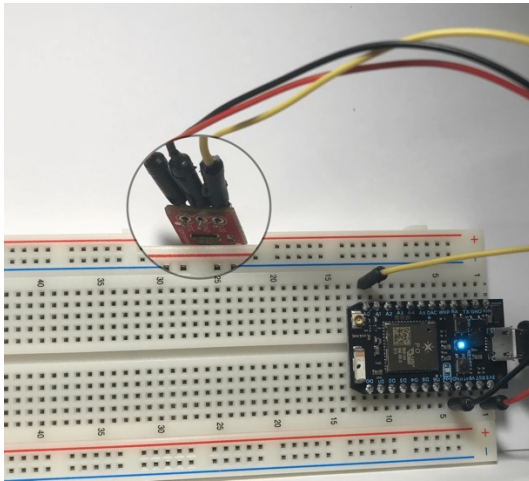


Figure 15. Phototransistor assembly on a Particle Photon microcontroller

Light intensity sensor

The photoresistor needs to connect to an analog input pin.

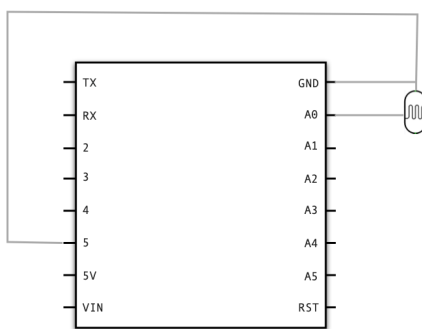


Figure 16. Schematics of the photoresistor

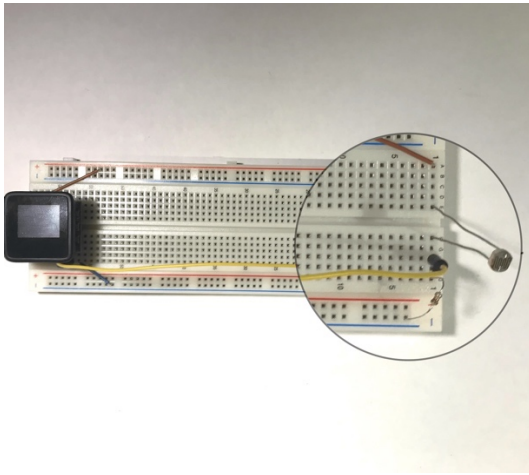


Figure 17. Mounting the photoresistor on a MicroView Arduino module

Activity sensor

There is no wiring schematic for the camera module because it is just plugged straight into the Raspberry Pi's camera port.



Figure 18. Mounting the camera module to the Raspberry Pi

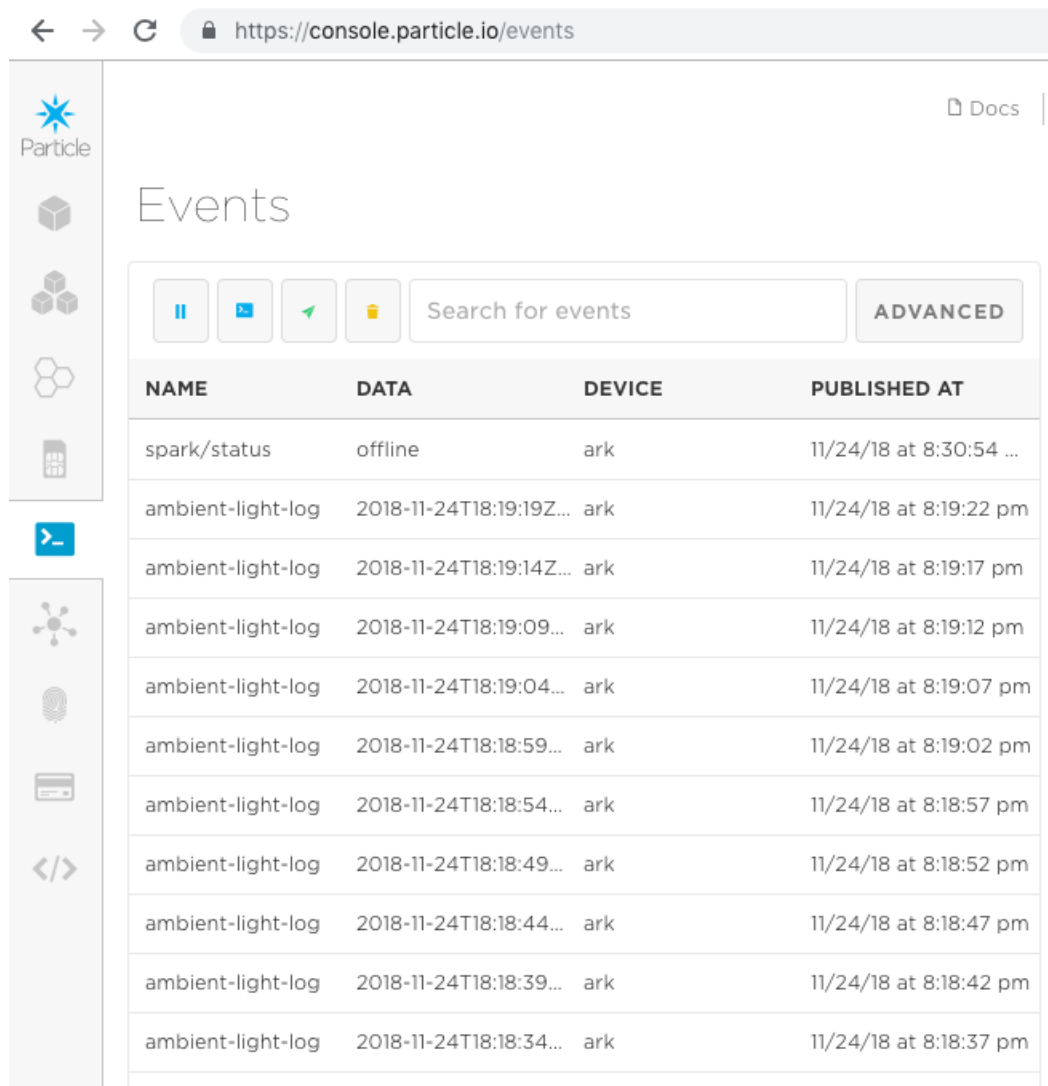
4. Results

A background information/documentation is provided for the ISCAFI system (see Appendix 1). Included with it are UML charts consisting of:

- Class diagram (mapping the static relationship between the participating actors/entities)
- Use case diagram (describing the actions that the participating actors can perform within the system)
- Sequence diagram (showing the interactions between the actors/entities as a sequence of events)
- Object diagram (showing a freeze framed instance of a class diagram)

Also included is a storyboard (helping explore scenarios through visualization in the form of a sequence of illustrations). These documentations help in setting the necessary background and context for the project.

For the project proper. Three sensor collector packages were assembled using the electronic components and the corresponding Photon Particle microcontroller, and they are the following: *PIR motion sensor*, *ambient light sensor*, and *light intensity sensor*. First, they were checked see if they were able to collect data, for example by shining a flashlight onto the light sensor, and then looking at the output on a serial console. Having verified that they did, the sensor was checked to see if they can transmit those same data to the cloud (via Internet). Here is a screenshot of one of the results:



The screenshot shows the Particle console interface at <https://console.particle.io/events>. The page displays a list of events under the heading "Events". The interface includes a search bar and a table with columns for NAME, DATA, DEVICE, and PUBLISHED AT. The events listed are:

NAME	DATA	DEVICE	PUBLISHED AT
spark/status	offline	ark	11/24/18 at 8:30:54 ...
ambient-light-log	2018-11-24T18:19:19Z...	ark	11/24/18 at 8:19:22 pm
ambient-light-log	2018-11-24T18:19:14Z...	ark	11/24/18 at 8:19:17 pm
ambient-light-log	2018-11-24T18:19:09...	ark	11/24/18 at 8:19:12 pm
ambient-light-log	2018-11-24T18:19:04...	ark	11/24/18 at 8:19:07 pm
ambient-light-log	2018-11-24T18:18:59...	ark	11/24/18 at 8:19:02 pm
ambient-light-log	2018-11-24T18:18:54...	ark	11/24/18 at 8:18:57 pm
ambient-light-log	2018-11-24T18:18:49...	ark	11/24/18 at 8:18:52 pm
ambient-light-log	2018-11-24T18:18:44...	ark	11/24/18 at 8:18:47 pm
ambient-light-log	2018-11-24T18:18:39...	ark	11/24/18 at 8:18:42 pm
ambient-light-log	2018-11-24T18:18:34...	ark	11/24/18 at 8:18:37 pm

Figure 19. Console output of my IoT sensor transmitting data to the cloud

The *data* field shows the data that the sensor transmitted (consisting of the timestamp and the value of the reading). The *device* field shows the IoT-enabled sensor unit used. The

name field is the name of the event trigger, which was set to trigger every 5 seconds (to comply with Particle System's rules).

The same process were performed on the *PIR motion sensor* and the *light intensity sensor*. Each of them exhibited the same behavior like that of the ambient light sensor, both locally and on the cloud.

To recap, the three objectives were:

- 1) To demonstrate that the scenario is viable through physical interaction with actual IoT devices.
- 2) To test that the sensors do collect data.
- 3) To test that the collected data can be transmitted via the networks and into the cloud.

Those objectives were met.

5. Discussions

The plan was to implement the system with as much variety of microcontroller platforms as possible, namely, with the Arduino, ESP8266, Particle Photon, Raspberry Pi, and ThingSee One. Thus far, three out five of those platforms were implemented (at a rudimentary level). The Raspberry Pi performed double duty in that it was used to implement the activity sensor (using the camera module and the Python OpenCV library) as well as performing the role of the local gateway. It was set up in such a way so that the ThingSee One send its temperature readings to the Raspberry Pi and, in turn, get the data forwarded for storage to Dropbox (dropbox.com). In the beginning, it was originally envisioned to go with this route all the way (i.e., set up the Raspberry Pi as a local IoT gateway) with the sensors only permitted to communicate with the gateway and then have it transmit all data from its local sensor network to the real cloud for storage. But as the process of prototyping the sensors paired with the other microcontrollers began, for example, with the ESP8266, hurdles were experienced. It wasn't as straightforward as originally envisioned. This particular hurdle is detailed further down below in this section. Next hurdle was with the Particle Photon. Because its designed to be an all-in-one IoT solution, it came with its own particular quirks. In the context of what was tried to be achieved, this quirk was a hindrance because it can only communicate with its own cloud ecosystem which located in the real Internet cloud. For example, if reading from the sensor that is connected to the Photon needed to be sent, there is only one place it can be sent, and that is to the Particle System's own dashboard in the cloud. Therefore, it has to bypass the local Raspberry Pi IoT gateway.

So all in all, the current system implementation is not a cohesive system of sensors that is managed by a local IoT gateway. But instead, it is a patchwork of a sensors with their own different ways of connecting to the cloud directly.

False starts and dead ends

When the circuit prototyping for the sensors was started, a *MicroView Arduino* module was used, and the sensors worked just fine; i.e., detecting the signals that they were supposed to detect, and outputting those readings to the *MicroView's* OLED display. But then when it came time to add other data, for example, adding a timestamp together with the floating point value of the detected ambient light, the *MicroView* (and it goes the same for the *Arduino*) was is not able to tell the current time. It has no mechanism for knowing what the current time is because 1) it has no real-time clock, and 2) no built-in network connectivity from which to query a time service, for example. The closest to getting a timestamp from the *Arduinos* is a relative timestamp, meaning, one gets the elapsed time since it has been switched on. But the countdown always begin from the UNIX epoch (Jan 1, 1970 00:00:00). So, say the *Arduino* has been on for 15 minutes, the timestamp it will output is Jan 1, 1970 00:15:00. So if one wants the *Arduino* to have that specific functionality, a real-time clock (RTC) module is needed because not only does an RTC have its own time keeping but it can do that indefinitely (as long as the 3V lithium cell battery power lasts).

So in order to get around the timestamp limitation, it was decided that switching to another platform—the *ESP8266*, was a good recourse. The reckoning was, since the *ESP8266* has a built-in Wi-Fi connectivity — that it even comes with a complete TCP/IP stack —it can query an NTP server and return the current time. But, it turned out that the *Adafruit Huzzah ESP8266 Breakout* comes pre-loaded with *NodeMCU*, a Lua language interpreter (I don't have Lua experience). To flash it with the stock firmware, a USB console cable would be needed. But the USB console cables on *Partco* (partco.fi) were too expensive. So for the time being, the *ESP8266* was a dead end.

Limitations

The current system isn't able to assemble raw sensor data into a format that can be used for predictive ML algorithms. Neither has it a planned data structure type yet, nor a target predictive machine learning algorithm. Regarding practical matter limitations, the “lab” where the research and development/implementation is done isn't really conducive for these kinds of activities (it's at my home). For one, there's not enough a large workspace area, and second, it's difficult to manage and control distractions, especially when one has a 2 year old child. This might seem arbitrary, but I truly believe that these ancillary things contribute a great deal to one's frame of mind in doing about the work.

Going forward

Although the ISCAFI system is an idea that only exists on paper right now, it is nevertheless important for this project because it is the structure from which this thesis project is built on top of. This fleshes out the abstract ideas into providing concrete demonstration of the viability of an idea by Amir Dirin, which is to create a classroom of the future.

I set out to provide a proof-of-concept of a subset of that idea, which is, a system that can (eventually) collect data and provide those same data to some Machine Learning algorithm—whether they be used to create a predictive model or be used as a feedback mechanism for the system's own adaptive learning. With further research and more fleshing out, the result of this project may potentially be used as a source of idea for anyone tackling the same problem scenario.

References

- Di Justo, P. (2015). Raspberry Pi or Arduino Uno? One Simple Rule to Choose the Right Board. Retrieved November 29, 2018, from <https://makezine.com/2015/12/04/admittedly-simplistic-guide-raspberry-pi-vs-arduino/>
- Different Microcontroller Boards And Its Applications. (n.d.). Retrieved November 29, 2018, from <https://www.elprocus.com/different-types-of-microcontroller-boards/>
- Dormehl, L. (2017). Thinking machines ± the quest for artificial intelligence - and where it's taking us next. New York: TarcherPerigee.
- Hanes, D. (2017). IoT fundamentals: Networking technologies, protocols, and use cases for the Internet of Things.
- Kelleher, J. D., & Tierney, B. (2018). Data science.
- McEwen, A., & Cassimally, H. (2014). Designing the internet of things.

Appendices

Appendix 1. Intelligent System Classroom Assistant For Instructors (ISCAFI)

1. Overview

Below is the system flow consisting of the interactions between a *classroom* entity and an *instructor* entity; and the interaction of a *class* or lecture entity. Together, they make up ISCAFI.

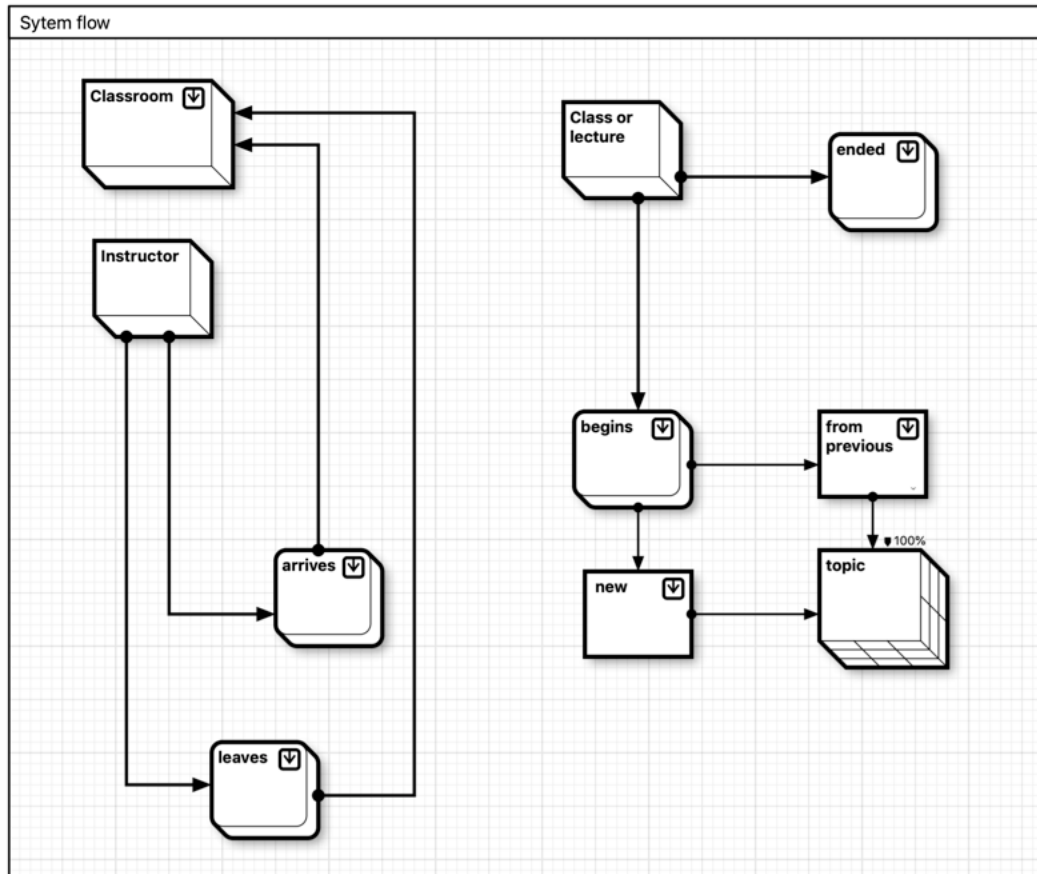


Figure 20. Diagram of the ISCAFI showing the big picture

Following are the events that trigger the intelligent system to act:

- when the class is about to start,
- when the instructor arrives,
- when the lecture begins,
- when the lecture ends,
- when the instructor leaves.

Below is a run down of the participating entities/objects, expressed in UML notation.

2. Class models

A look at the static structure of the main actors in the system.

Classroom

Shows the link to the important objects that can be controllable by the intelligent system.

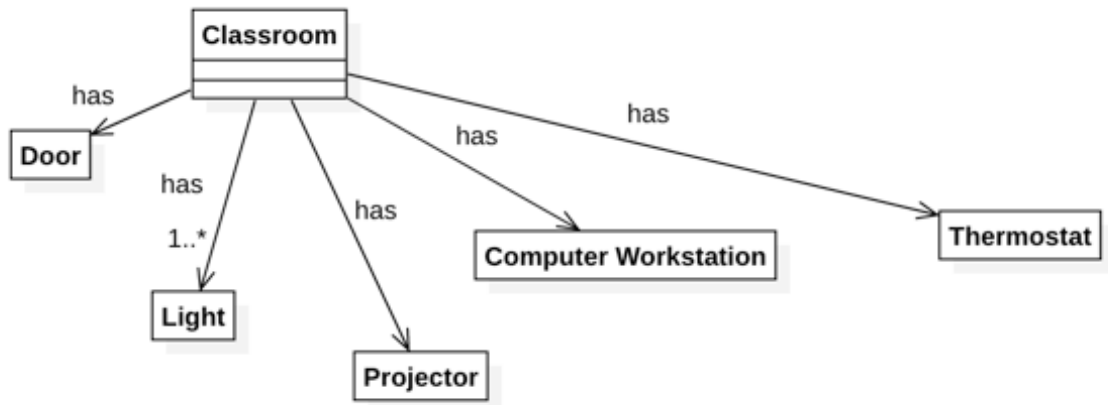


Figure 21. Shows what the classroom object exposes to the intelligent system and to the instructor object

Instructor

Shows what the two actions can do, and to which objects.

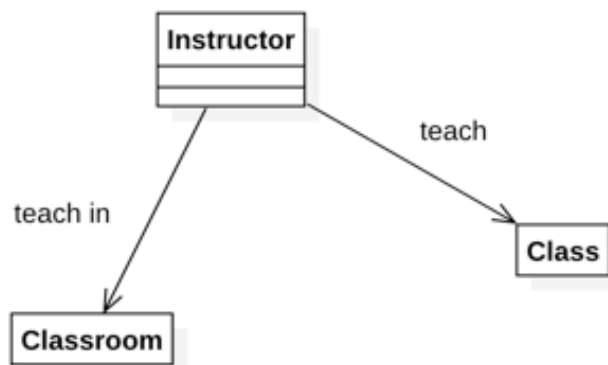


Figure 22. An instructor can teach class (lecture) in a classroom

Class (or lecture)

Shows the single object (Topic) it exposes to the intelligent system.

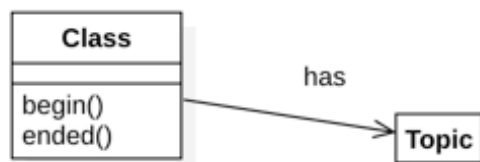


Figure 23. A class (lecture) has a single property and two actions.

3. Use case diagrams

This perspective looks at the actions that

- can be acted upon, and that
- can happen between entities.

Overview

The two main actors and the explicit events they trigger:

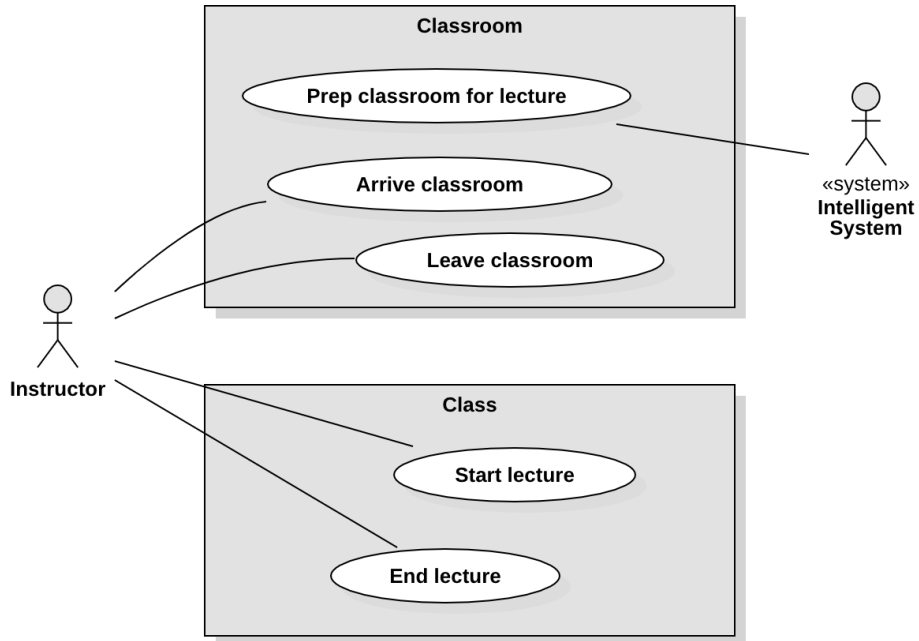


Figure 24. Use case diagram of the instructor and the intelligent system

This is the same as above but just visualized differently:

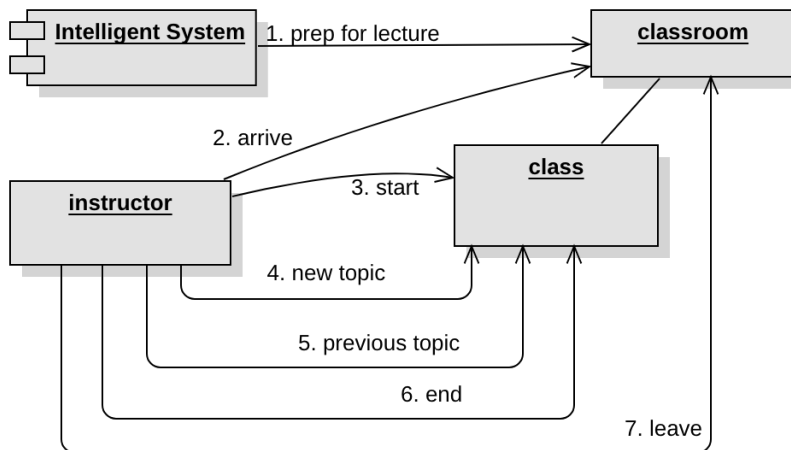


Figure 25. An object diagram of the instructor and the intelligent system showing the sequence of events

Use cases

Prep classroom for lecture

This is how the *intelligent system* assists the instructor (an implicit assistance) when a *class* is scheduled to begin shortly and the instructor is not in the classroom yet.

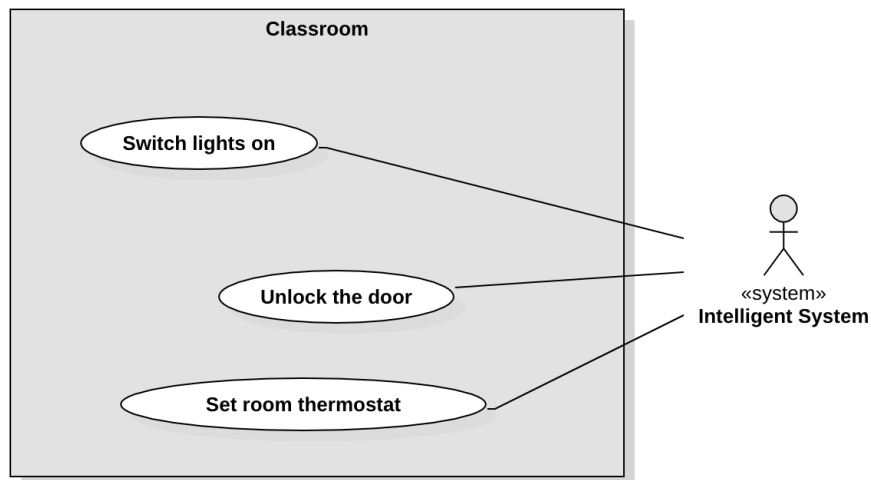


Figure 26. The actions that the intelligent system facilitate when preparing the classroom for a lecture

Instructor arrives

This is how the *intelligent system* explicitly assists when the instructor gets inside the classroom.

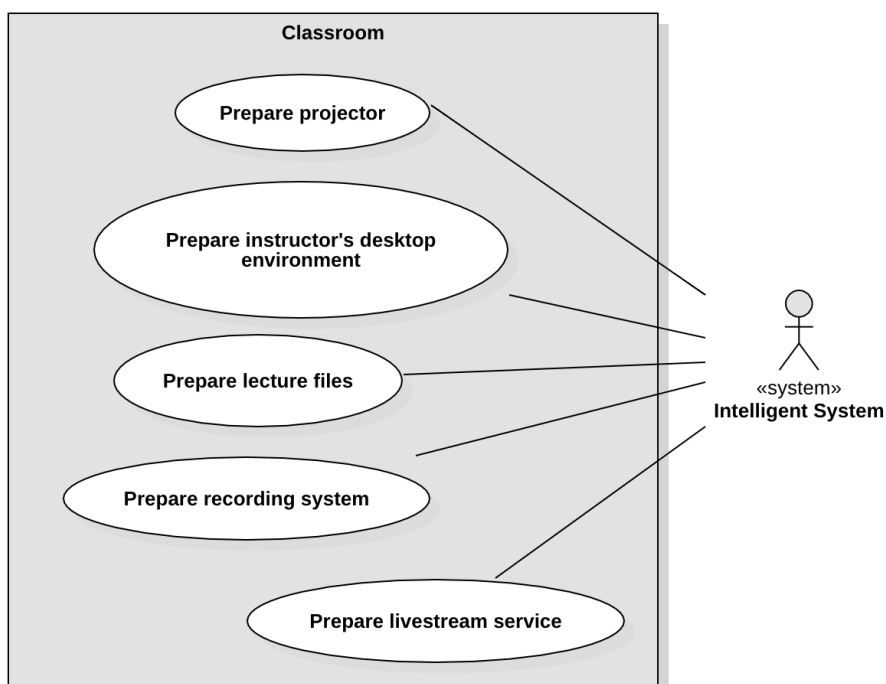


Figure 27. The actions the intelligent system facilitate when an instructor gets inside the classroom

Class (lecture) begins

This is how the *instructor* explicitly triggers events.

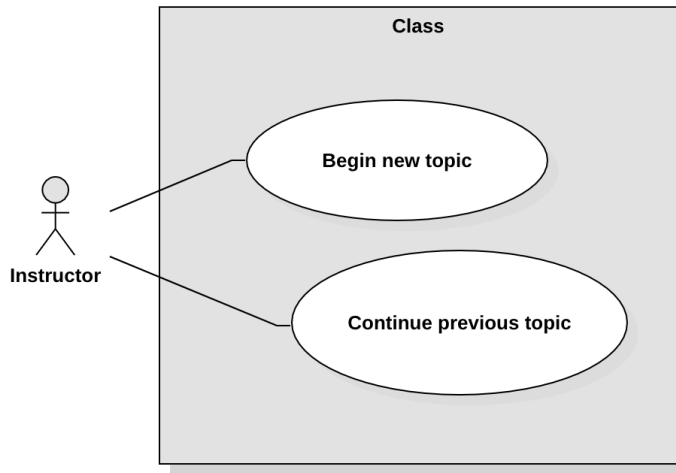


Figure 28. The actions an instructor does when they begin or resume their lecture

Lecture begins with new topic

This is how the *intelligent system* assists the *instructor* behind the scenes (implicit assistance).

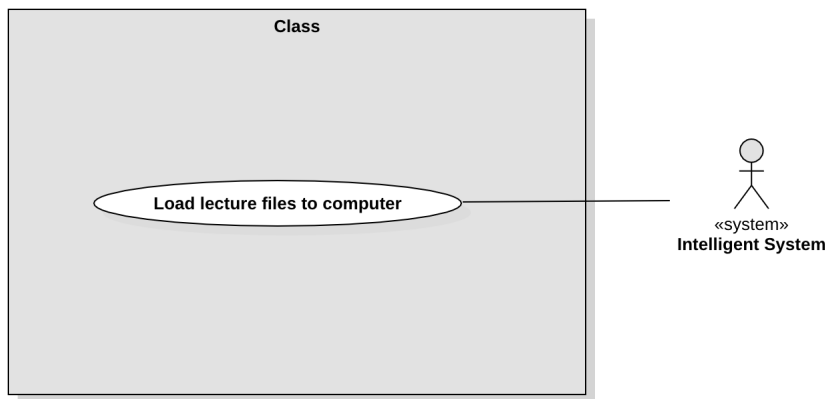


Figure 29. The action the intelligent system facilitate when the instructor begins their lecture

Lecture continues with previous topic

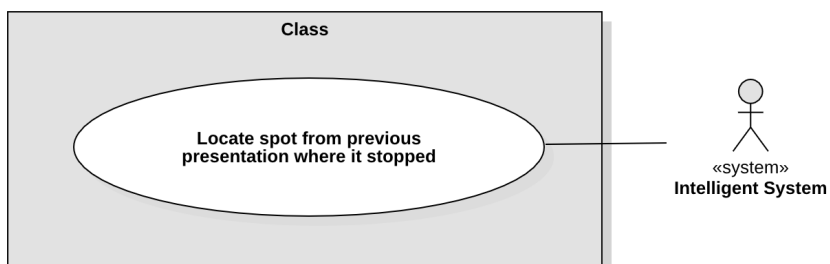


Figure 30. The action that the intelligent system facilitate when the instructor resumes their lecture

Class (lecture) ended

This is how the *intelligent system* assists the *instructor* when the class has ended.

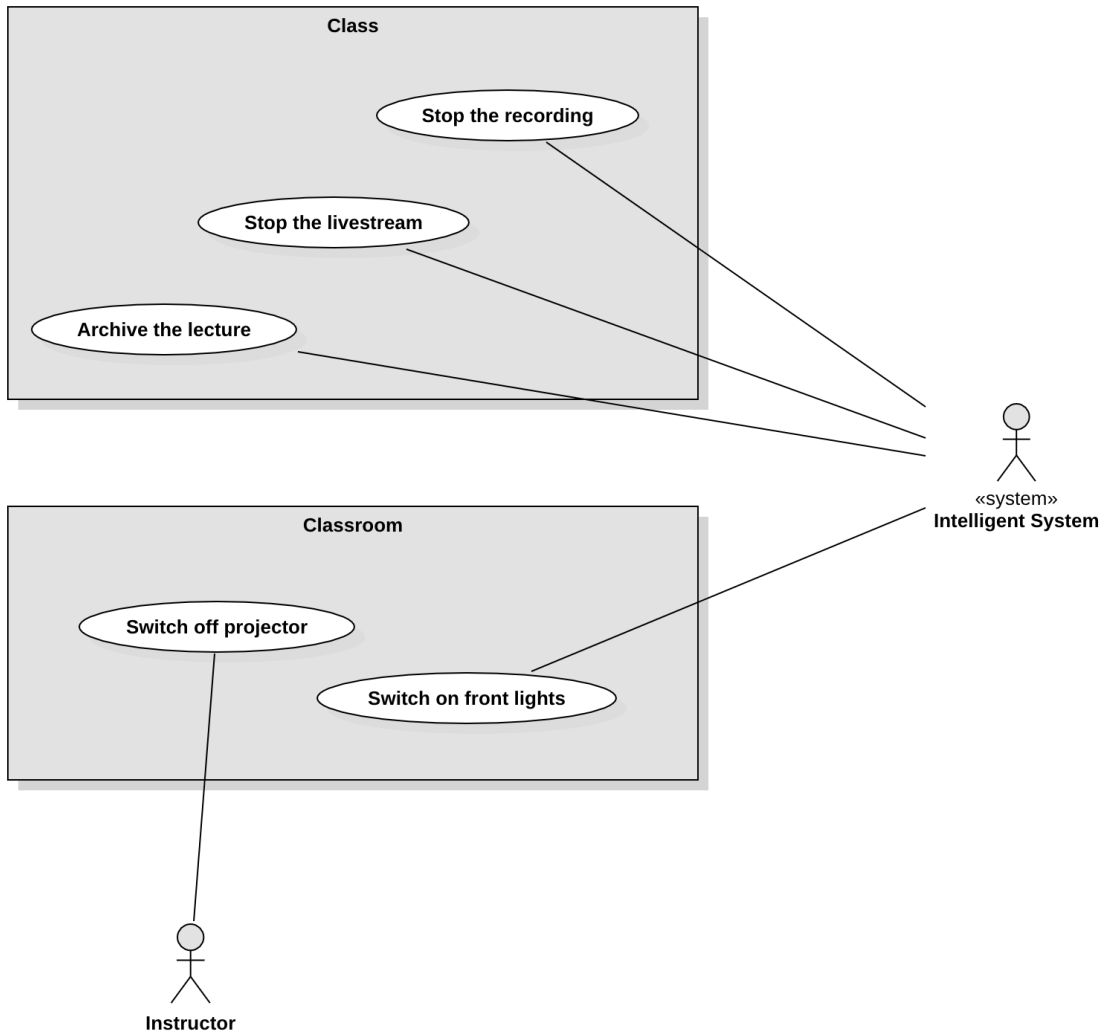


Figure 31. The actions that the intelligent system facilitate and an instructor does when a class ends

Instructor leaves

This is how the *intelligent system* assist the current as well as the next instructor when the current one exits the classroom.

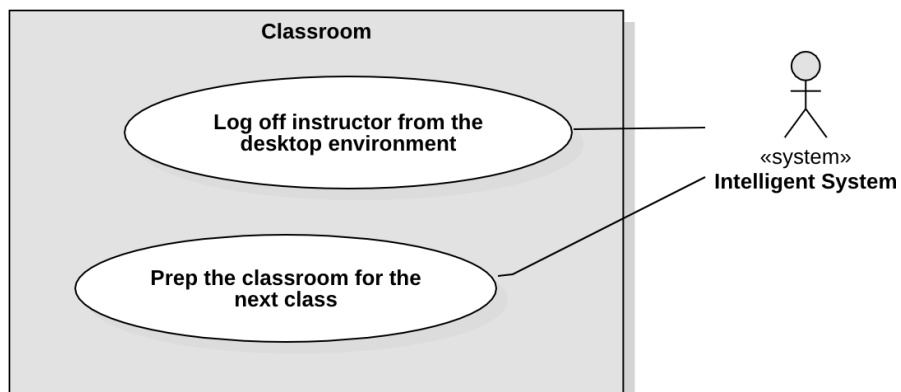


Figure 32. The actions that the intelligent system facilitate when an instructor leaves

4. Sequence diagrams

Overview

Below is the major sequence of events that happen when the intelligent system receives a signal from an outside entity. It's this event that triggers the start of the cascade of events. It ends when the instructor leaves the classroom as scheduled (i.e., nothing out of the ordinary happened).

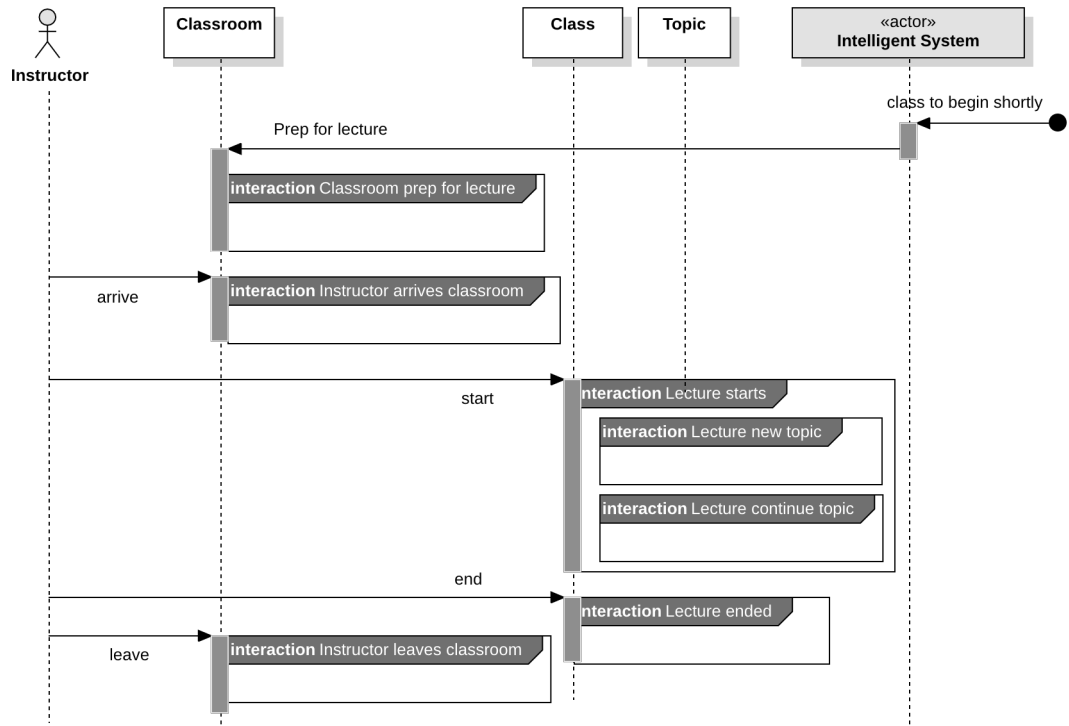


Figure 33. Sequence diagram of the overview

Interactions

Classroom prepped for lecture

The intelligent system switches on the lights, unlocks the door, and regulate the thermostat to 20°C.

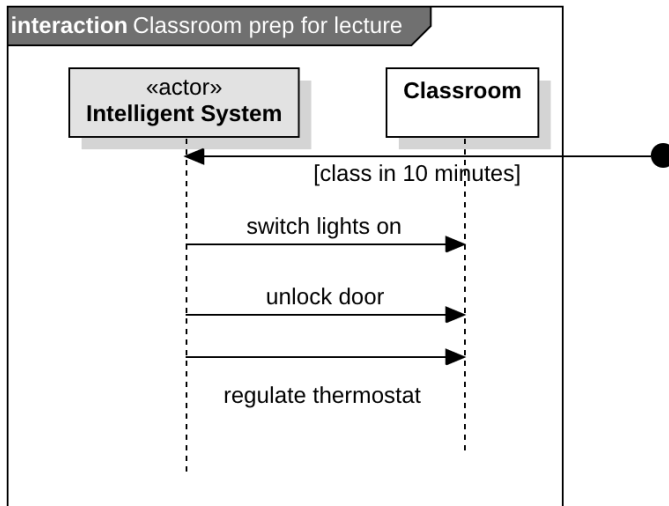


Figure 34. Sequence diagram of interaction: Classroom prepped for lecture

Instructor arrives to classroom

The intelligent system preloads the instructor's desktop environment so that when the instructor is authenticated, the computer loads in an instant.

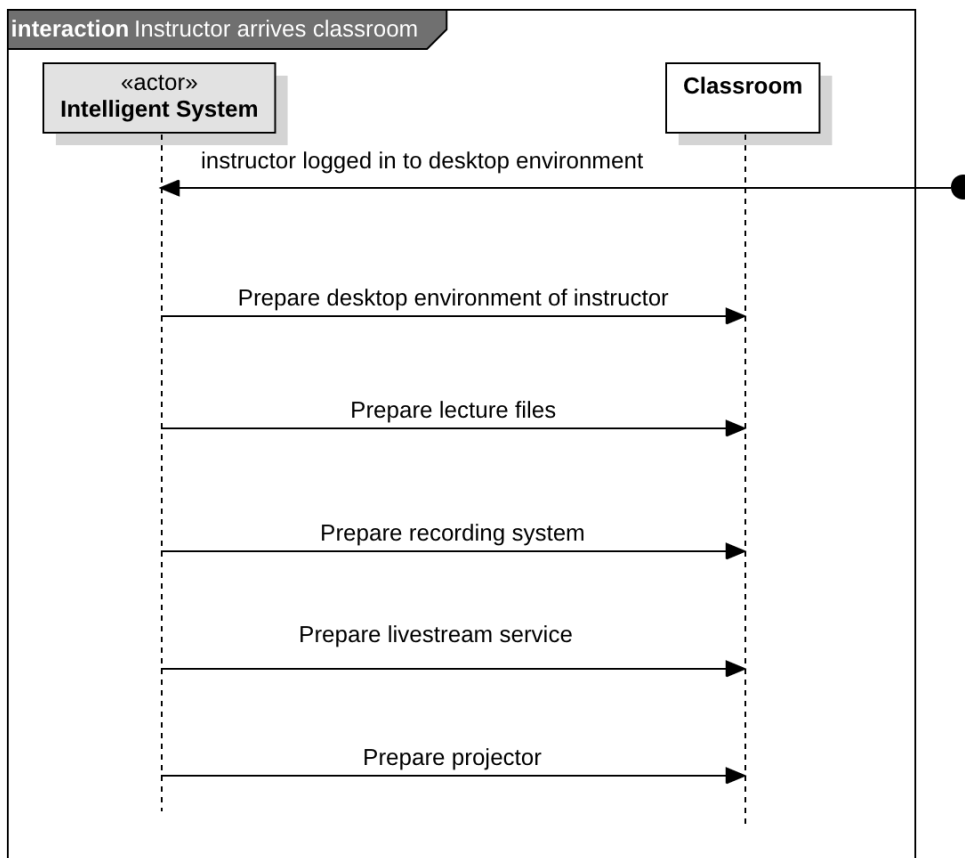


Figure 35. Sequence diagram interaction: Instructor arrives in the classroom

Lecture starts

The intelligent system records the lecture as well as streaming them live online.

When the intelligent system detects the projector light was switched on, it switches off the front lights.

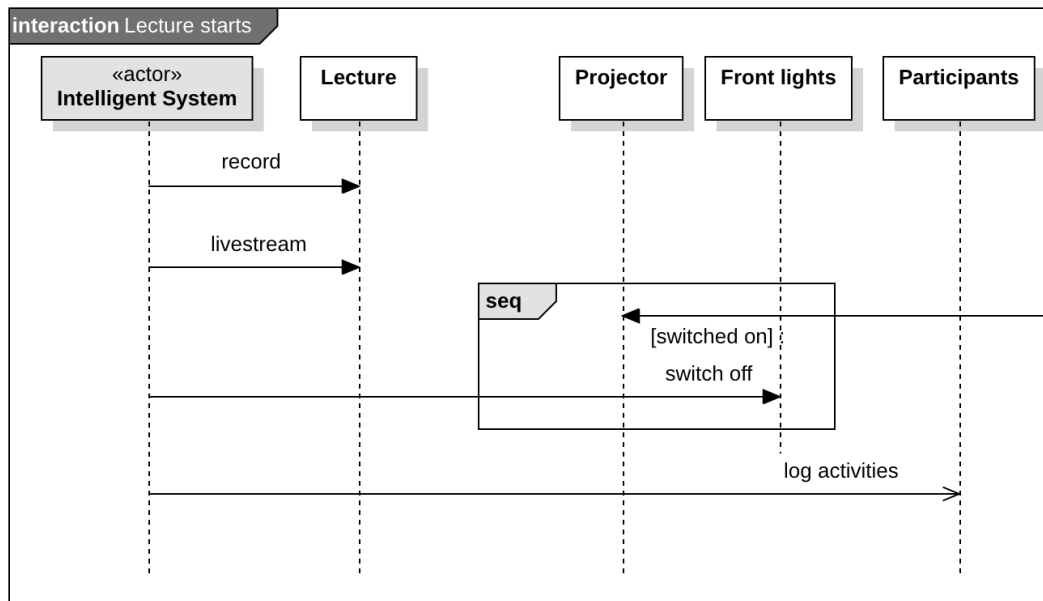


Figure 36. Sequence diagram: Lecture starts

Lecture new topic

The intelligent system loads the presentation file on the instructor's computer.

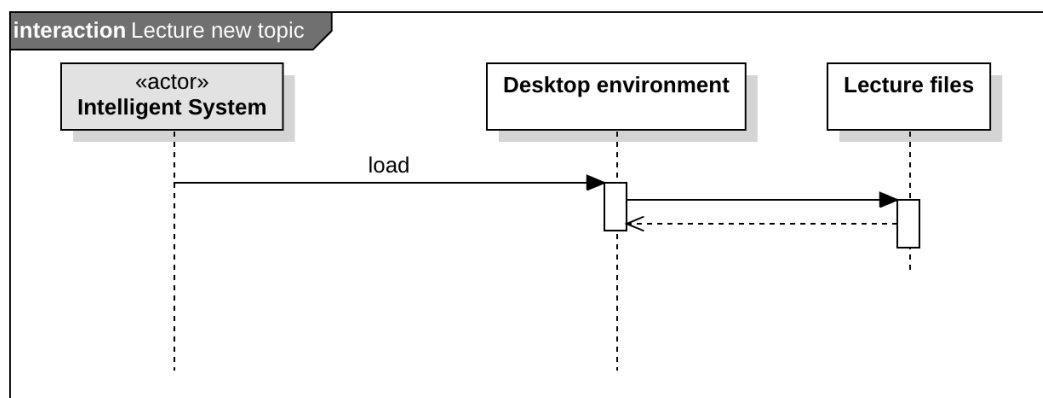


Figure 37. Sequence diagram interaction: Lecture has new topic

Lecture continue topic

When the instructor resumes from a previous topic (for example, because it was interrupted), the intelligent system not only locates the previous presentation file but also the location at the point where the lecture stopped.

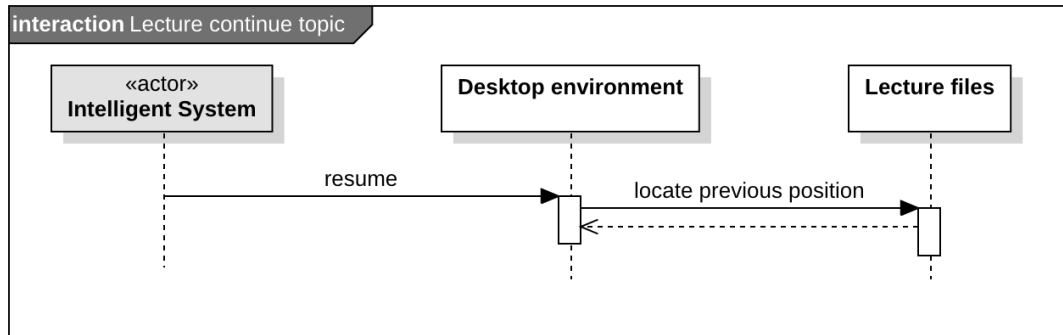


Figure 38. Sequence diagram interaction: Lecture continue topic

Lecture ended

When the instructor ends the lecture, the intelligent system stops the video recording and the live streaming. It then archives the video so that it can be replayed anytime.

When the intelligent system detects the projector light is switched off, it switches on the front lights.

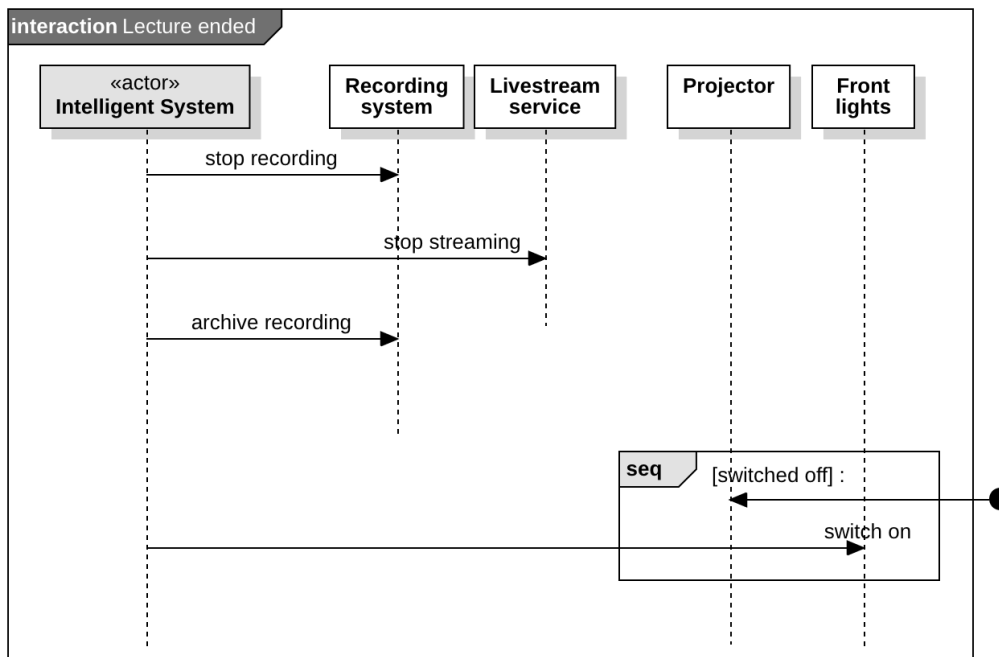


Figure 39. Sequence diagram interaction: Lecture has ended

Instructor leaves classroom

When the instructor exits the classroom, the intelligent system automatically logs off the instructor's computer environment. It then prepares for the next class.

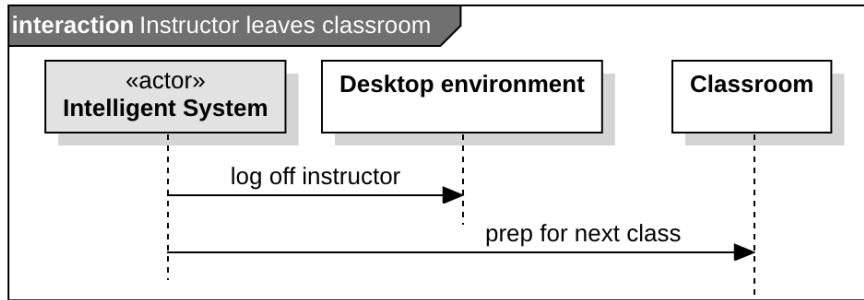


Figure 40. Sequence diagram interaction: Instructor leaves classroom

5. Storyboard

Scenario: An instructor holding a lecture held in an ISCAFI-equipped classroom

The instructor arrived to a brightly lit and comfortably warm classroom.



She greeted her students as she settled at her desk.



She then logged in using face recognition.

She then began her class by recapping the main points of her previous lecture, before continuing from where she left off previously.



When her presentation ended she asked the class if there were any questions.



As she glanced at her computer, she acknowledged a question from a student joining the lecture via livestream.



As she wrapped up her class, she thanked the students in the classroom and on the livestream for their participation.



Later that night somewhere in Denmark, a Helia exchange student viewed the lecture from her web browser, catching up on the lecture and assignments.



All photos by rawpixel on Unsplash
(<https://unsplash.com/@rawpixel>)

(Photos by [rawpixel](#) on [Unsplash](#))

[https://unsplash.com/photos/JjsAirtKGGs?
utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText](https://unsplash.com/photos/JjsAirtKGGs?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

[https://unsplash.com/search/photos/business?utm_source=unsplash&utm_medium=refer-
ral&utm_content=creditCopyText](https://unsplash.com/search/photos/business?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)