ILLINOIS INSTITUTE OF TECHNOLOGY

# INTELLIGENT LIGHTS AND MESH SENSOR NETWORK

Andreive Giovanni Silva

Breno Martins

Pedro Henrique G. Silva

Prof. Jeremy Hajek

**Contents**

# Intelligent Lights and Mesh Network

Andreive Giovanini Silva, Breno Soares Martins and Pedro Henrique G. Silva

Illinois Institute of Technology

Author Note

Andreive Giovanini Silva, Breno Soares Martins and Pedro Henrique G. Silva, Brazil Scientific Mobility Program.

This research was supported by CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) and IIE (Institute of International Education) as part of the 2016 Brazil Scientific Mobility Program (BSMP).

## Abstract

We are surrounded by lights, lighting practically everywhere we are. Meanwhile, sensors are becoming more present in our lives every day, controlling doors, windows, alarms and even lights. With lights and sensors becoming more present and smarter, why not use technology to integrate them as we want and need? This project proposes to do that by using Arduino and the Hue Lighting System. We used temperature and gas sensors attached to XBee Pro S2B radio modules in a mesh network structure. Data is sent to a coordinator node that passes it to an Arduino Mega through serial communication. The Arduino Mega runs an algorithm that is able to interpret data received from the XBee and change the lamps accordingly – one of the actions taken is to flash lamps in red and orange when gas is detected. Such algorithm also generates a local HTML page that can be accessed through the network allowing direct commands over the lamps. Beyond that, the HTML page allow a user to choose a path in which he or she wants to go and all lamps on that way will guide the person to his/her destination by being colored on a specific color.

*Keywords:* intelligent lights, sensors, mesh network, Arduino, XBee, Philips Hue, path guidance.

## Intelligent Lights and Mesh Network

Technology has evolved fast, becoming smarter, smaller and a powerful tool when it comes to improve our lives and the world we live in. Everyday new devices and software are being developed, creating new possibilities of how to interact and control our surroundings. The automation of houses that allow us to control lights, windows, sound systems and more with our personal computers and smartphones is a good example of where the technology is getting nowadays.

With all these new possibilities at our range, however, an old problem can still be seen: the lack of communication between technologies. The integration of devices and products isn't always possible due differences of design or programming languages for example. This issue doesn't impede a product from working but can limit its potential and new possibilities. Therefore, when a integration of technologies become possible, an opportunity of create something new, creative and useful is shown and lives can be improved one more time.

This project embraces one of these opportunities by working to integrate lights and sensors. Wouldn't be great if the lights of our buildings reacted to the room temperature or the presence of gas in case of a fire or a gas leaking? What if these lights could show the way to a specific room you want to go? We already have signs and maps to guide us but people still get lost and can't find the room they want sometimes. Sensors for temperature and gas also exist and can be seen in many buildings, but without a clear visual warning, like red lights for example, the human response can be slower and people with hearing problems can stay in a dangerous rooms if there's only a loud noise to warn them. With this in mind, this project uses the Philips Hue lights, Arduino and XBee radio modules as main components to create an integrated system of intelligent lights.

## Philips Hue Lighting System

The Philips Hue LED Lighting was the best product of the year of 2012 according to Forbes. This product allows you not only to turn your lights on and off but also set colors and effects and everything on your smartphone or computer. By using the hue system application programming interface (API), effects or configurations desired can be sent to the bridge, which allows you to control all settings of the lights in the system using JSON (JavaScript Object Notation) as data-interchange format. This can also be done using a website or an Arduino board. In this project an Arduino board is used in order to integrate the lights with the temperature and gas sensors which are used as input for the system. Communication with the lamps in this way is made through a Philips Hue bridge connected to a local network – such device can control up to fifty lamps.

*Figure 1. Philips Hue Lamps and Bridge*

## Arduino Platform

This project is based on Arduino platform. This family of microcontroller boards provides sets of digital and analog I/O pins that can interface to various expansion boards (shields) such as Arduino Ethernet Shield R3 which is used to connect the board to the internet. For programming it, the Arduino project provides an integrated development environment (IDE) based on a programming language named *Processing*. This language also supports the languages C and C++, which is used in this project.

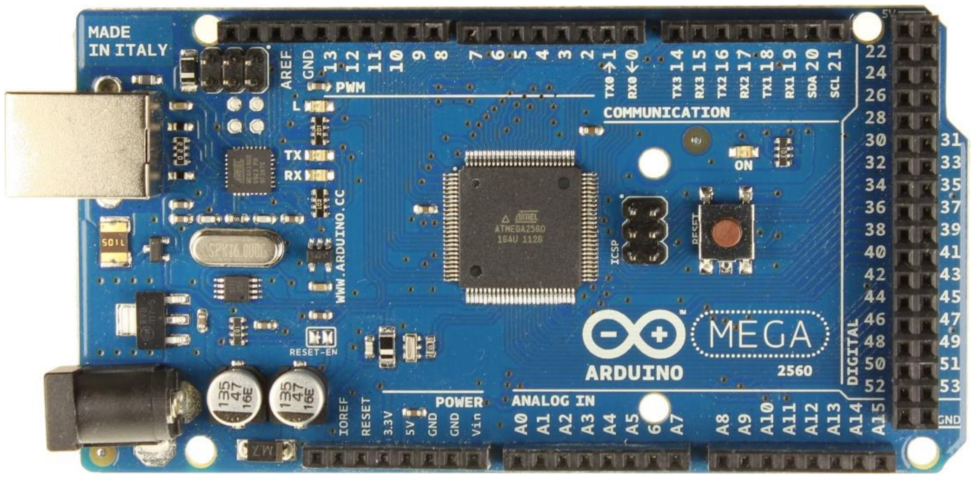


*Figure 2. Arduino MEGA*

**XBee Radio Modules**

The XBee & XBee-PRO DigiMesh 2.4 embedded RF modules utilize the peer-to-peer DigiMesh protocol in 2.4 GHz for global deployments. This mesh protocol offers users added network stability through self-healing, self-discovery, and dense network operation. With support for sleeping routers, DigiMesh is ideal for power sensitive applications relying upon batteries or power harvesting technology for power which is the case of this project. XBee adapters deliver wireless connectivity to electronic devices in wired networks.

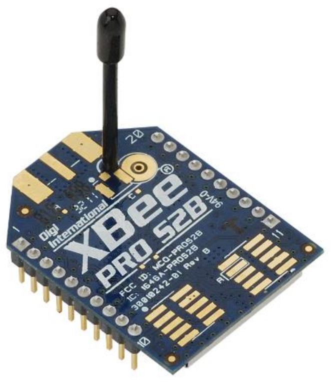


*Figure 3. XBee Pro S2B*

## Temperature and Gas Sensors

Sensors are getting more and more common in our lives. Two of the most useful sensors we can have in our houses are the temperature and gas sensors. If well placed and integrated with the Hue lights, you can know the external temperature or the temperature of a room and if there's gas inside of it, which can prevent accidents such as explosions in the kitchen for example. In this project the temperature sensor TMP36 and the gas sensor MQ-2 are used, both generate an analog signal as output, which can be read and sent by the XBee and read by an Arduino board, and they are cheap if a replacement is necessary (see the Materials and Costs section for prices).

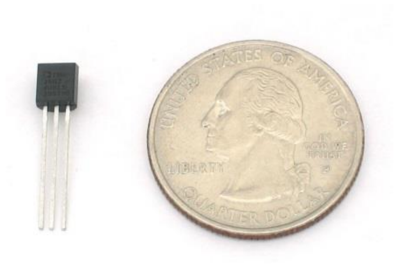


*Figure 4. TMP36 Temperature Sensor*

*Figure 5. MQ-2 Gas Sensor*

## Mesh Network

Communication with both lamps and sensors are based on a mesh network. This network allows each node to be able to reach any other on the same network. This can be accomplished by sending data to an intermediate node, which stores the message and then send it to the final destination or even to another intermediate node. A mesh network is reliable and offers redundancy. If a node fails, the flux of information is redirected and the functionality of the network is not affected.

For a valid mesh network between XBees modules, at least one XBee must be configured as coordinator. It is responsible for selecting an available operating channel, a valid PAN ID, and other parameters necessary to correctly run the network. The coordinator can allow up to 20 devices join the network, giving them a unique address as they first try to connect to the network. Routers are devices that will be permanently on and are able to redirect messages from other nodes. A router can join a network by having the same PAN ID – an address for the network. It performs an active scan searching for such ID and get a response with a local address from a node that allowed it to join. It can also allow another twenty nodes to join the network. End devices join the network just like routers. However, as they are designed for low power and sleep mode, they cannot route messages from other nodes.

## Methods

### Procedure

The development of this project was based on incremental design and testing. Each task to be accomplished was tested individually before being incorporated into the general scope of the project, facilitating debugging. Design was constituted of five main areas: analog data sampling with XBees, communication and transmission of data between XBees modules, commands to control a Philips Hue bridge, exhibition of visual signs via lamps, development of a HTML page, and transmission of sensor data to a remote database.

Two temperature and a gas sensors were used to establish a sensor network integrated via XBees modules. In this area of the project, automated analog data sampling was tested using a connection between the sensors and a XBee module. Each sensor is attached to one XBee. Two different configurations were tested: in one, a XBee was permanently on and took readings in a defined period of one second; In another, a XBee was put to sleep after each reading – the duration of sleep mode was set to one second.

Control over lamps was first tested using simple on/off commands. Later, it was integrated with commands sent via HTTP requests to change colors according to a choice of path. In the final version, a code was deployed on an Arduino to test reception of messages, display of information on the lights and storage of data on the database.

Six lamps were used to create a lighting mesh network attached to a Philips Hue bridge.

## Materials

Table 1 – Materials and costs

| MATERIAL | UNIT COST | QTY | TOTAL COST | SOURCE |
|---|---|---|---|---|
| Arduino Mega | $45.95 | 1 | $45.95 | http://store-usa.arduino.cc |
| Arduino Ethernet Shield R3 | $45.00 | 1 | $45.00 | https://www.adafruit.com |
| XBee DigiMesh 2.4 | $19.00 | 4 | $76.00 | http://www.digikey.com |
| Philips Hue Starter Kit | $199.00 | 2 | $398.00 | http://www2.meethue.com |
| TMP36 Temperature Sensor | $1.50 | 2 | $3.00 | https://www.sparkfun.com |
| MQ-2 GAS Sensor | $3.99 | 1 | $3.99 | https://www.bananarobotics.com |
| Ethernet Cable | $9.49 | 2 | $18.98 | https://www.amazon.com |
| 3.7V Battery | $9.95 | 3 | $29.85 | https://www.adafruit.com |
| | **FINAL COST:** | | $620.77 | |

## Assembly

The final project was constituted by: an Arduino Mega with an Ethernet shield connected to a coordinator XBee Pro S2B and to a local network; two boards with temperature sensors attached to two XBees Pro S2B and a 3.7 V battery; a board with a gas sensor attached to a XBee Pro S2B and a 4.8 V battery; and six Philips Hue lamps linked to a Philips Hue Bridge.

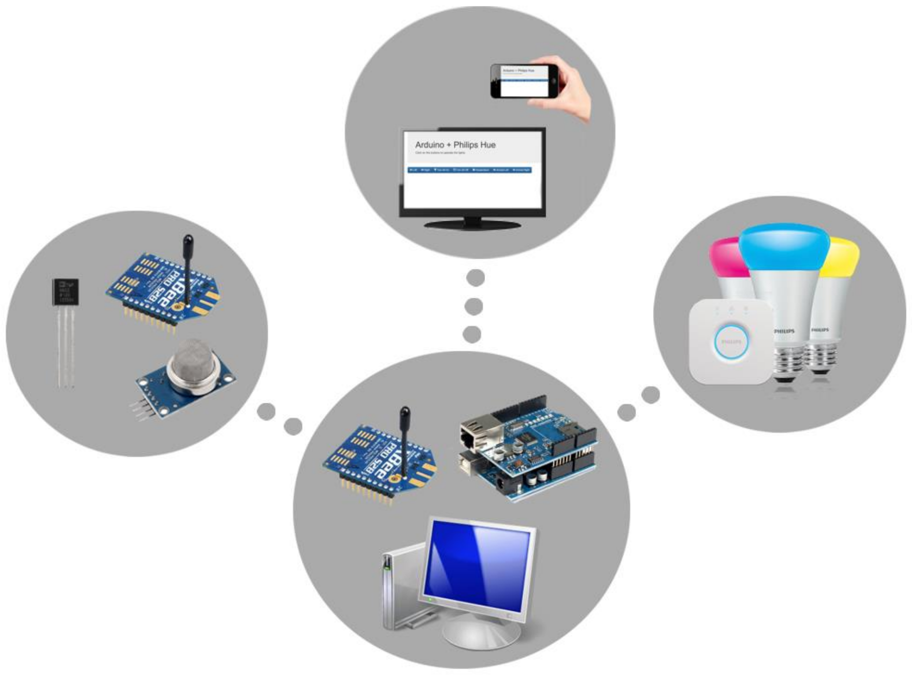


*Figure 6. System's Overview.*

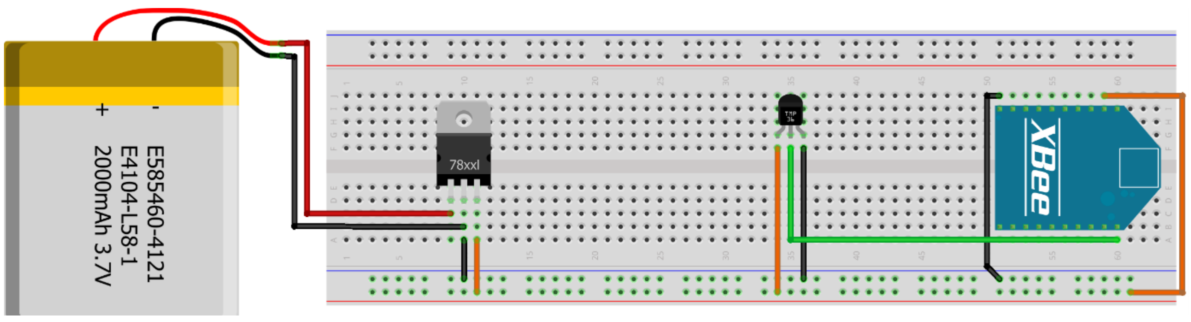


*Figure 7. Board with temperature sensor.*

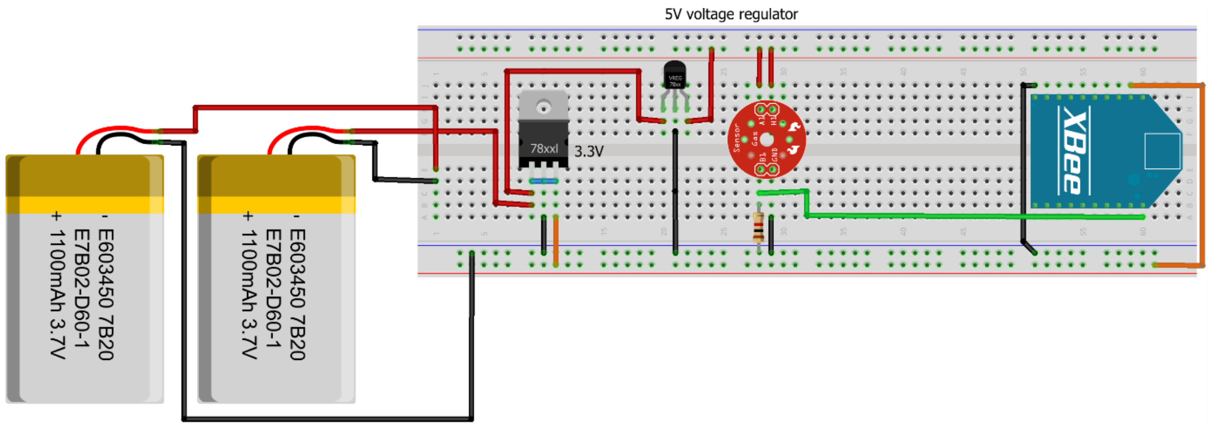


*Figure 8. Board with gas sensor.*

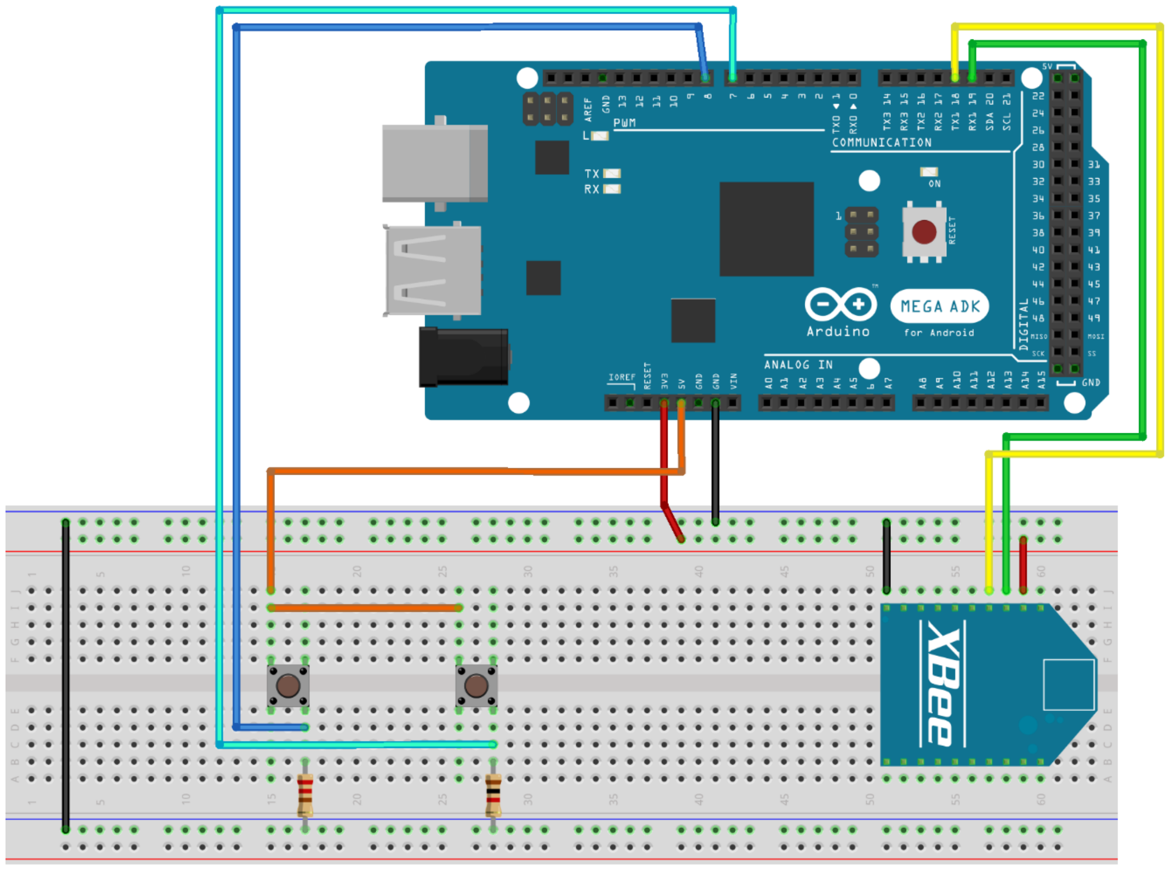


*Figure 9. Arduino Mega and coordinator XBee Pro S2B.*

## Configuration of XBee Pro S2B radio modules

Three different configurations were used in this project: XBee as coordinator, router or end device. The XBee attached to the Arduino Mega was configured as coordinator and is responsible for receiving messages from all the other nodes. Such messages can be passed to the Arduino via serial communication. To configure the XBees modules, one should use the software called XCTU, which is provided by Digi International Corporation. To write on a XBee, USB shields for XBee were used. As you connect a XBee device to a computer, you can identify it on XCTU. The configuration over the default values for each node can be seen below. Each family set is related to the functionality of the node – if it is a coordinator, router or end device:

Table 2 – Configuration for XBee Pro S2B as coordinator with serial communication

| Parameter | Value |
|---|---|
| Product family | XBP24BZ7 |
| Family set | ZigBee Coordinator API |
| Firmware | 21A7 |
| PAN ID | 2016 |
| API Mode | API Mode 1 |

Table 3 – Configuration for XBee Pro S2B as router with analog sampling

| Parameter | Value |
|---|---|
| Product family | XBP24BZ7 |
| Family set | ZigBee Router AT |
| Firmware | 22A7 |
| PAN ID | 2016 |
| Destination address - High | 0 |
| Destination address - Low | 0 |
| JV1 Verification Channel | 1 |
| D0 AD0/DIO0 Configuration | ADC [2] |
| IR IO Sampling Rate | 3E8 (in decimal, it is equivalent to 1000 ms) |

Table 4 – Configuration for XBee Pro S2B as end device with analog sampling

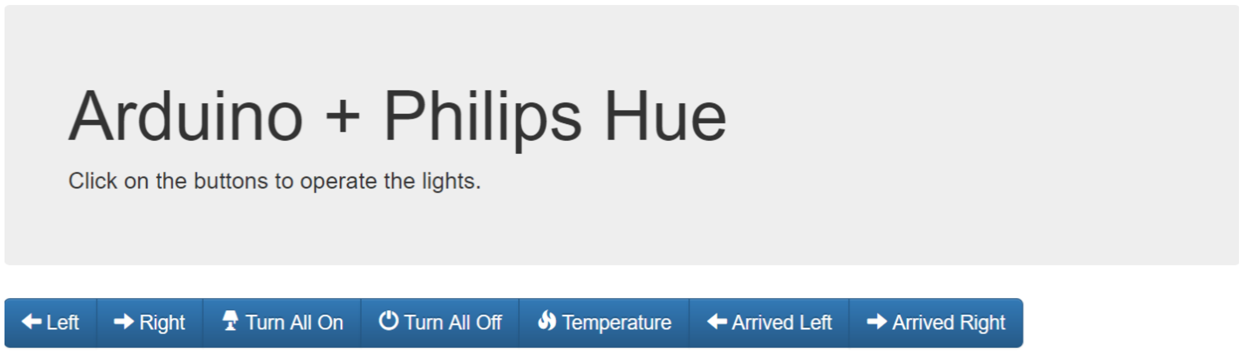
| Parameter | Value |
|---|---|
| Product family | XBP24BZ7 |
| Family set | ZigBee End Device AT |
| Firmware | 28A7 |
| PAN ID | 2016 |
| Destination address – High | 0 |
| Destination address – Low | 0 |
| JV1 Verification Channel | 1 |
| D0 AD0/DIO0 Configuration | ADC [2] |
| IR IO Sampling Rate | 3E8 (in decimal, it is equivalent to 1000 ms) |
| SM Sleep Mode | Cyclic Sleep Pin-Wake [5] |
| SP Cyclic Sleep Period | 64 (equivalent to 1000 ms) |
| ST Time Before Sleep | 14 (equivalent to 20 ms) |

## Path Guidance and Sensor Information on Lamps

Every now and then people face situations which include places they have never been before. It may be a bank, hospital, new school, laboratory or some governmental building. Either way, there’s usually a specific room or desk to find and the path isn’t always obvious and clear. Even with signs and maps some people still face some difficulties trying to find the exact place they have to be. With this in mind, a feature of this project comes to help people to find their destination.

The idea is to use a HTML page with buttons representing the rooms, desks or directions. Each button sends a specific command to Arduino that sends it to the hue lights bridge through a HTTP request, lighting the path to the destination selected with a different color than white, such as green for example. If more than one destination is selected and there are lights being used to indicate more than one path, such lights will alternate between the colors of those paths. As a first stage of development, the HTML page created has only two directions as path options to ensure a good and fast demonstration. However, even in this case these directions can represent the path to a room, an elevator or an exit, depending how the lights are disposed in the building. Once the person arrives at his destination, the same HTML page can be used to stop showing the selected path and set the lights to the white color again.

Information collected on sensors can also be shown on lamps. If one chooses the temperature option, the lamps will flash in a color related to the level of temperature during three seconds, coming back to their previous state later. Such information is displayed on the user interface as well in a numeric scale. The gas sensor works regardless of commands from the user. When it detects that the gas level trespassed a threshold, Arduino will command the lamps to flash in orange and red until the gas level comes back to a normal state.


Arduino + Philips Hue

Click on the buttons to operate the lights.

*Figure 10. HTML Page*

**Algorithm**

The Arduino Mega operates as a state machine, i.e. it changes its current state based on inputs. Such changes can incur in actions and effects over its outputs [Ref: Claudius Ptolemaeus, Editor. “System Design, Modeling and Simulation using Ptolemy II”, Ptolemy.org, 2014. http://ptolemy.org/books/Systems]. The basic diagram for the algorithm running on Arduino Mega can be seen below.

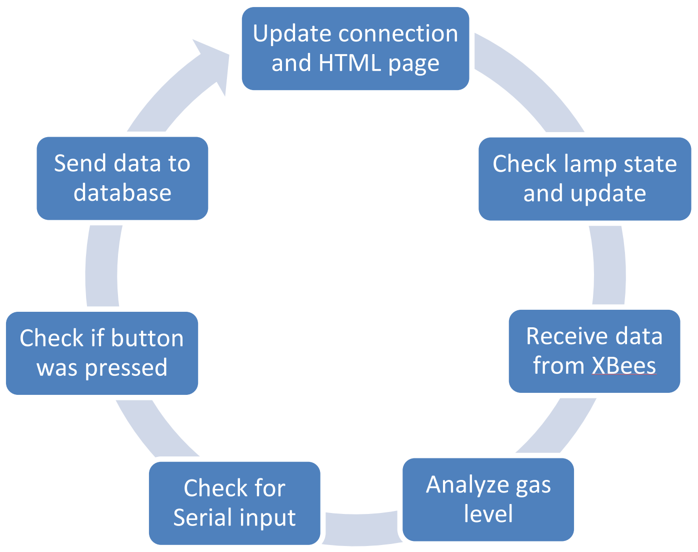

Update connection and HTML page
Check lamp state and update
Receive data from XBees
Analyze gas level
Check for Serial input
Check if button was pressed
Send data to database

*Figure 11. Simplified diagram for algorithm that runs on Arduino Mega*

When it starts for the first time, Arduino sets all global variables that will be used during program execution. Later, it starts an infinite loop that: updates connection with the local network and refreshes the HTML page; check if any lamp needs to have its state changed and updates it; check if there is any new package arriving at the XBee connected in its serial channel and receives data if it's the case; analyze gas level and check if it is below or above a threshold – in this last case, it will change the state of the lamps; checks for serial input coming from the PC; checks if any of the physical buttons were pressed and updates the state of the lamps accordingly; and finally, checks if the current time frame is on the period delimited to send data to a database and does so if the condition is true. The code can be checked on the GitHub website: https://github.com/Illinois-tech-ITM/BSMP-2016-Intelligent-Lights.

## Results

The project was constantly tested during development and demonstrated live for three times. Data sampling taken every second showed stable values. However, wrong readings appeared during debugging without a specific period. Those were either extremely high values or very low values (for example, about 30000 ºC or -15000 ºC when normal temperature is about 25º or 77 ºF).

Connection of XBees End Devices and Routers to the network was efficient. In a few rare cases during development, one module was not able to find the network, but a reset could fix it. Connection of Philips Hue lamps to the bridge never failed. The same happened to the connection of Arduino Mega to local network.

Commands sent to lamps have a noticeable delay of about 0.1 to 0.5 milliseconds. Similarly, the gas sensor also presents a delay of about 1 second to generate an output voltage that is greater than the threshold set. In general, the system was able to reproduce all behaviors predicted and was responsive to commands sent by a user even with delays mentioned before.

## Discussion

Data sampling errors were tested with different times for sleep mode configured on XBee End Devices. As time between consecutive sleeps is reduced, the error rate increases. A period of one second was chosen as reasonable for a low rate of errors and it also allows Arduino to process data more smoothly. That was also other effect of reducing the time between consecutive sleeps – the noticeable delay started to increase. Also, to avoid incorrect readings, only values in a specific range are stored on the Arduino to show the next time it is requested. This works as a filter implemented via software.

Data was chosen to be sent to a database on a fixed period to reduce the delay when sending commands. It is less costly in terms of power processing to check if it is time to send data than establishing a connection and sending it. Finally, the gas sensor sampling rate was also set to one second to avoid too many samples over a period of time and hence making Arduino operation slower. It could be seen that the system still remained responsible to the introduction of inflammable gas in the environment. The lights flash after three seconds at maximum, which is also affected by the responsiveness of the gas sensor – such sensor needs to burn the gas and raise its temperature, making it naturally slow.

**Challenges and Future Steps**

Throughout the development of this project, some challenges appeared. For instance, we started using an Arduino UNO to receive data and send commands to lamps. However, static strings needed to build the HTML page required more memory to run than UNO was able to provide. Because of that, it was decided to switch the board for an Arduino Mega. While an Arduino UNO board has 2KB of SRAM and 32KB of flash memory, an Arduino Mega has 8KB of SRAM and 128KB of flash memory.

Another problem is how to power Arduino and the XBee module. As Arduino and the coordinator XBee need to be powered continuously, it is recommended that they remain connected to a wall socket. The same is applicable to XBees configure as routers. As they need to remain on and store messages from other nodes, they should be connected to a wall socket as well – with a consumption of about 50 mA, any battery wouldn’t last more than a couple of weeks. However, XBee End Devices using sleep mode had a consumption lower than 0.7 mAh. Previous projects used solar panels, which is a good and environmentally friendly solution but since this project is designed for an indoor system such solution wouldn’t be practicable at first. The current solution is to use a 3.3V battery with 2000 mAh, allowing the system to run for roughly 120 days without replacing or recharging the batteries.

A final challenge faced involves cost. The lights used are expensive; costing about fifty to sixty dollars each and that can be a problem if this system is going to be installed in a building with hundreds of lights. Therefore, future projects should aim to develop a cheaper intelligent light that can be used as the Philips Hue lamps. Other steps involve using different sensors and work on the integration of the system with voice controlled devices. If successful, the communication with your lights can happen without moving a finger.

## Conclusion

This project was a start for greater outcomes. Our work was able to apply identification of temperature in different points of a building, detect gas level, and show a path to a specific local through colored lamps. However, there's something beyond such results. The Philips Hue lighting system itself has potential for new creations and, when integrated with sensors and other devices, the possibilities go even further. Sound and presence sensors, light sensor, temperature and radiation sensors are some of the various types of sensors we can find. If integrated with this lighting system, the desired lights can show whenever there's someone in a room or if the radiation level is above a certain limit in a laboratory for example. With enough lights and compatible HTML page, an entire building can have this system to help people find their destination by selecting a button in a web page. From entertainment to public safety, the future is better and more colorful with intelligent lights.

## References

Faludi, Robert. “Building Wireless Sensor Networks”, 1st edition, O’Reilly Media, Inc, 2010

Suresh Chandra Satapathy et all. Proceedings of the Second International Conference on Computer and Communication Technologies: IC3T 2015, Volume 3

Logitech. Harmony experience with Philips wireless lighting. Retrieved 07/2016 from: https://support.myharmony.com/en-us/harmony-experience-with-philips-hue

How to set up a ZigBee mesh network. Retrieved 07/2016 from: https://docs.digi.com/display/XBeeZigBeeMeshKit/How+to+set+up+a+ZigBee+mesh+network

Adafruit. TMP36 Temperature Sensor. Retrieved 07/2016 from: https://learn.adafruit.com/tmp36-temperature-sensor/overview

Sandbox Electronics. MQ-2 Smoke/LPG/CO Gas Sensor Module. Retrieved 07/2016 from: http://sandboxelectronics.com/?product=mq-2-smokelpgco-gas-sensor-module

Arduino. Arduino UNO and Genuino UNO. Retrieved 07/2016 from: https://www.arduino.cc/en/Main/ArduinoBoardUno

Arduino. Arduino Ethernet Shield. Retrieved 07/2016 from: https://www.arduino.cc/en/Main/ArduinoEthernetShield

Arduino. Arduino Mega. Retrieved 07/2016 from: https://www.arduino.cc/en/Main/arduinoBoardMega

Digi. Xbee DigiMesh 2.4. Retrieved 07/2016 from: http://www.digi.com/products/xbee-rf-solutions/modules/xbee-digimesh-2-4