

From: Caroline Jones, Michael Haley, Sneha Pradhan, Tanatsigwa Hungwe, Zehua Zhao

Team: 1

Date: 04/30/17

Subject: **Engineering Addendum**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Dear Future Engineers,

Lighting systems have failed to take advantage of the opportunities presented by IoT technology - a world of intelligent, interconnected devices. Nowadays, consumers need a lighting system that is more dynamic, intuitive and conscious of our energy consumption. Luminesene’s final deliverable is a lighting system solution that consists of a wearable transceiver (Arduino 101), light fixtures (Particle Photons), an IP enabled central hub (Raspberry Pi 3), and single pixel cameras.

The system engages with the user in two different modes: “gesture mode” and “adaptive mode”. The former allows the user to actively interact with the system and explicitly change the properties of the light fixtures - their colors and brightness - by invoking gestures through the wearable. The central hub analyzes information regarding the selected lights, recognizes the gestures sent from the wearable, and updates the states of the lights. The second mode hosts a passive interaction between the user and the system. The system acts as a traditional IR sensor - turning the lights on in the presence of a user, and turning them off in the absence of a user. The single pixel cameras detect the presence of users, and relays this information to the central hub, where the lights are updated.

In the midst of all the interactions, the system’s energy consumption and performance data is logged into an SQL database and displayed on a Node.js web application. This portal grants the user access to energy usage graphs, where they can track their energy usage over periods of time. Additionally, the application gives the user remote control over the system - allowing them to change gesture preferences and issue commands to luminaires without using the wearable.

The system is currently in its alpha stage, meaning there is plenty of room for improvement e.g. optimization of the wearable, PCB, processing, gesture library etc. As is the case of product development, it is recommended to build upon the product's features iteratively. The system is a sum total of many applications of engineering concepts.

**TLDR; to succeed in this project you will need to be/get familiar with the following:**

* Software Engineering (C, C++, Python, JavaScript)
* Microprocessors (Arduino, Particle Photon)
* Signal Processing (ADC, FFT)
* Web Development (Node.js, SQL, JavaScript, HTML, CSS)
* Electric Circuit Theory (PCB Design, Electronics)
* Computer Aided Design (CAD Tools)

**Software Engineering**

This involves programming the Arduino 101, The Particle Photon and the Raspberry Pi. The Arduino functions as the wearable transceiver - receiving light signals from the luminaires, and sending information to the Raspberry Pi. The Arduino and Photon are programmed in C/C++, whilst the Raspberry Pi is coded in Python and JavaScript. The Python code drives the adaptive procedure of the system e.g. checking for occupancy in room. The JavaScript drives all the data aggregation and processing e.g. the Bluetooth Low Energy information from the Arduino 101, the calls (posts and queries) to the SQL database, and the Photon API calls (send commands to luminaires)

**Microprocessors**

This involves the Arduino 101 and the Particle Photon. These processors are very special\*. The Arduino 101 ditches away the ATMega chip for an Intel chip, allowing for faster high level programming development and IoT interactions. It contains an onboard IMU and Bluetooth Low Energy (BLE) units - two essential components for gesturing and data sending. Programmed in C/C++ just like all (most) Arduinos. **Take care**: the Arduino 101 does not have the ATMega chip, therefore cannot be programmed at same low levels (Assembly Code) as previous Arduinos (due to Intel restrictions).

The Particle Photons are Wi-Fi enabled microprocessors with cloud capabilities i.e. they facilitate API calls and events on the cloud. Photons programs can be built locally or build via the web IDE. Programmed in C/C++ just like the Arduinos. **Take care**: the Photon contains different library definitions for typical data types e.g. the String class has functions different from C++ implementations.

**Signal Processing**

This involves ADC sampling (analog -> digital light signals from luminaires), and FFT processing (changing voltage /time data to amplitude/frequency data). The Arduino 101 is responsible for this data conversions. This is used to identify the luminaires (they are all individually programmed by frequency). Using the FFT functions, the Arduino can take an analog light signal and determine the frequency of the light - and hence, the ID of the luminaire. Programmed in C/C++ on Arduino 101. **Take care**: FFT is very complicated, computationally expensive and unfortunately, the heart of the project. If this doesn’t work; the project doesn’t work. For simplicity’s sake, frequencies of 1-9 kHz are correctly observed by the Arduino. Get this optimized and locked down as soon as you can.

**Web Development**

This involves an application pulling information from a database and rendering it onto the client-side in the form of charts. Additionally, this app makes calls to the Photon API and changes the lights, and subsequently makes calls to the database. The web app is built using Node.js, another special\* tool - a JavaScript web framework. The app utilizes an SQL database, which contains the amount of power utilized by the luminaires and the duration of light usage. This data gets pulled and rendered as a chart on the web app. Additionally, the user can change the lights directly, by making calls to the Photon API (using the JavaScript APK). The HTML is what is rendered and seen by the user, with the CSS (Bootstrap) used to garnish the application. **Take care**: Node.js utilizes modules for functionality. Some of these modules depend on each other (at certain versions), so lock down the module versions to avoid breaking the application. Another consideration is JavaScript’s asynchronous nature. This means that code is evaluated and executed continuously even if certain lines in code haven’t returned specific values unlike C++ which executes and finishes every line before moving to the next (blocking calls). This becomes an issue when waiting for responses e.g. API calls. You will need to make callbacks to fix this.

**Electric Circuit Theory**

This details the circuitry designed for the wearable. This began as a photodiode amplifier circuit (for analog light signal) on a breadboard, which evolved into a PCB design (for minimization and mobility). The current PCB contains a microprocessor, photodiode, and three buttons (for session, selection, and IMU unlock functions). Additionally, the design is powered by a battery supply. **Take care**: the design will boil down to one scenario: Hardware, Software and Size (choose 2). You will have to compromise the hardware complexity for the size/comfort or the software complexity and vice versa.

**Computer Aided Design**

This involves the design of the wearable device. Currently, it is a bulky case housing the Arduino 101, the PCB circuit, and the wearable battery supply. The case contains a strap for wearability purposes. With minimizations to the circuit and the battery solution, the case will have to be refactored. **Take care**: although the case is to cover the circuitry; it needs openings for the photodiode, and the button controls. Get familiar with EPIC - you will be using it for 3D printing and case cutting.

The Luminesense system features a host of capabilities: an accurate gesture recognition system, a comfortable wearable device, an intuitive gesture library, and a state of the art motion capture system that adjusts lights in real time. The project is not exceptional by virtue of its objective, but by the execution of its objective: to create a unified system of communication amongst the various components. In doing so, the system redefines the way humans interact with lights and provides a new energy efficient paradigm for smart lighting systems.

\*special: a major pain