**Memo**

To: Professor Alan Pisano, Professor Osama Alshaykh, Fulya Ekiz Kanik

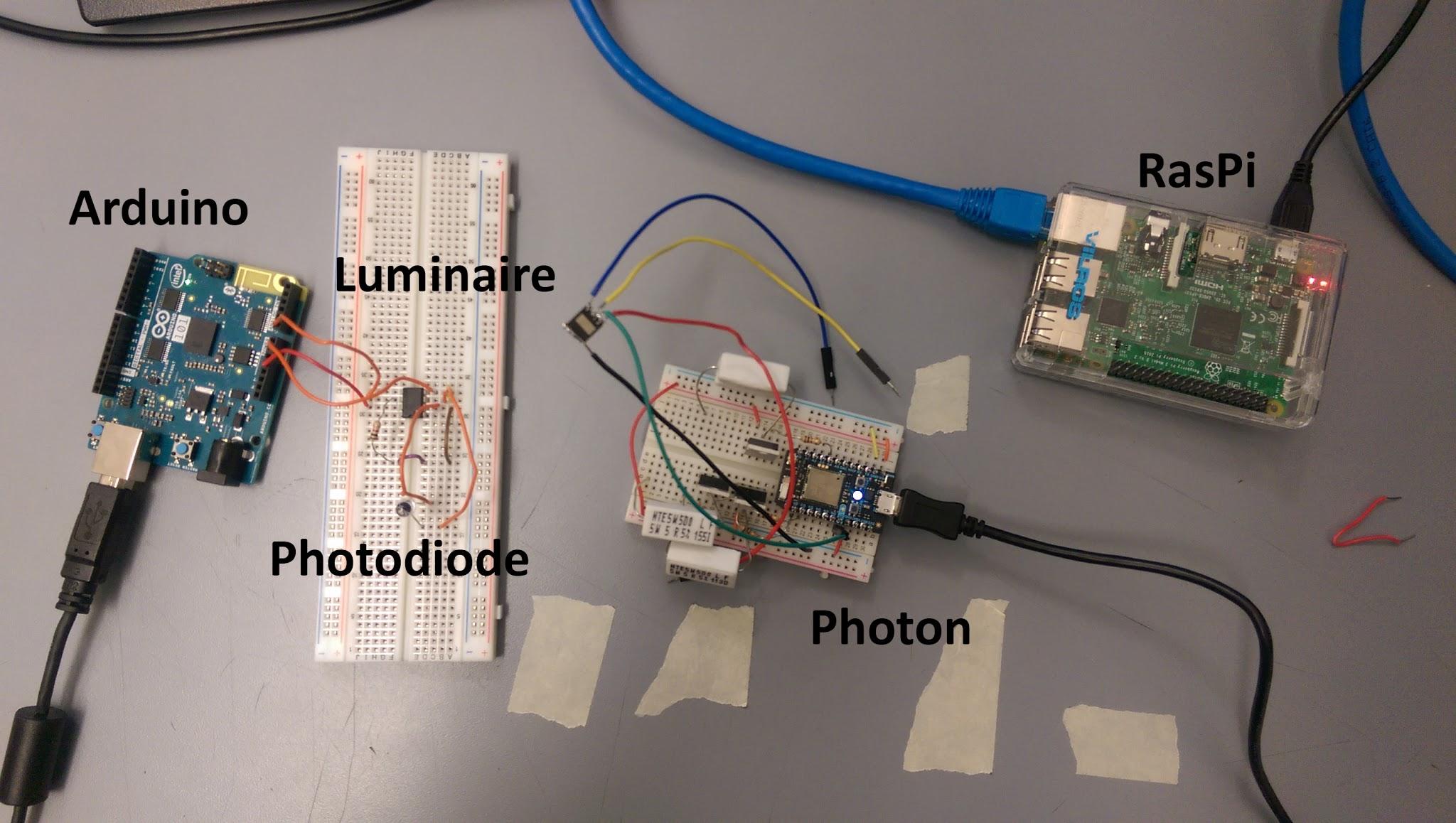
From: Caroline Jones, Michael Haley, Sneha Pradhan, Tanatsigwa Hungwe, Zehua Zhao

Team: 1

Date: 02/16/17

Subject: **Luminesense System Second Deliverable Test Plan**

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



1. **Light Modulation**

**1.1 Description and Goal:**

The “Luminaire” is a device provided by the client. It is a powerful light emitting diode which emits a light code that is uniquely identifiable by its frequency. The luminaire is driven by a Particle Photon. In order to interact with the luminaire prototype, the Luminesense system must be able to sample and decode these light codes. For this purpose, a photodiode amplifier circuit was attached to the Arduino 101. The following arrangement models the basic functionality of the wearable device: The photodiode samples analog light signals, and subsequently, Arduino determines the frequency of the light signals and hence, determines the light chosen.

**1.2 Procedure:**

Utilizing the Arduino IDE, flash the *Luminaire\_FFT.ino* code onto the Arduino 101. This code that reads analog input from the photodiode connected to the Arduino’s A0 port. The readings from the photodiode are then aggregated to determine the frequency of the light source using Fast Fourier Transform (FFT). This involves dividing the sample rate of the ADC into 128 bins/bands of frequency to pinpoint the amplitudes of sampled light (The highest amplitude indicating the light source being observed by the system). Using the oscilloscope, the frequency of the light source is then monitored for verification.

**1.3 Verifiable Result:**

A main requirement of the client is for the Luminesense system to wirelessly and seamlessly integrate with their luminaire technology. The most important aspect of that integration is being able to distinguish amongst any of the lights in the room. Showcasing successful reception and transmission of correct luminaire frequencies across the system guarantees that the user has more precise control over the light fixtures in the room - allowing for more effective communication amongst system components, advanced system control, and a better user experience.

**2.0 Arduino Gestures**

**2.1 Description and Goal:**

One of the key features of the Luminesense product is the ability to correctly recognize gestures the user performs to control the luminaires accordingly. There are a lot of gestures that can be added in the new world of Internet Of Things as everything around us is becoming smart. For this deliverable, we would like to add the feature of changing the preset and color of the lighting. In order to do this, we plan to add more gestures to be able to switch between the different colors of the lights.

**2.2 Procedure:**

Similar to the on and off gestures, we can twist our right hand rightwards to change the color to red and twist it leftwards to change the color to blue. As we are using a RGB LED for the testing, we have a lot of choices for our colors that we can choose from and create the related gestures.

**2.3 Verifiable Result:**

The result can be verified by performing the gestures and seeing the change in the color in the LED that is being powered by the Photon. We used a RGB LED for the purposes of this deliverable, however, the integration into the client’s setup in PHO 207 will be the next step.

**3.0 Web Application - Gestures**

**3.1 Description and Goal:**

One of the features that we would like to add is the option to choose the preset and color of the luminaries via our web application. It was one of the client’s deliverable as well that we have a web interface to interact with our system. With clear UI/UX, web applications become a great tool for a system. In the future, the user would be able to choose the gesture that he or she prefers via the web application.

**3.2 Procedure:**

The web application has already been setup for the necessary gestures. Cloud integration has also been done, so a user can access the gestures via the web URL. The user can control the lights via the web application, so this allows another platform for the user to control the lights, if anything is to happen to the wearable.

**3.3 Verifiable Result:**

To verify the results of the procedure, one can click on the buttons for the respective gesture and see the corresponding response in the LED.

**4.0 Web Application - Energy Savings**

**4.1 Description and Goal:**

One of the requirements of the customer was to provide an interface for a user to track their energy consumption and energy savings. The method that we have chosen to accomplish this is a web application. The web application should serve as a hub for the user to be able to monitor their system and show them important information. The web application was developed using Node.js and is deployed to the cloud using Heroku. A PostgreSQL database is used to track energy consumption.The first iteration of the web application is live at <https://luminesense-test.herokuapp.com/>. It currently shows a graph of Watt consumption per day, a box indicating money saved, and a pie chart to view hourly statistics of energy use for the day. The data for the graph is pulled from the PostgreSQL database.

**4.2 Procedure:**

Push the code for the web application to Heroku using the following series of commands in the project directory:

$Git add .

$Git commit -m “commit for testing purpose”

$Git push heroku master

Then open the live web page using the following command:

$Heroku open

Opening the web page triggers entries into the heroku log files, including console prints. Open the heroku logs using the following command:

$Heroku logs

The logs show a successful connection to the PostgreSQL database and prints the power and time data that is used to populate the area chart on the index.

**4.3 Verifiable Result:**

The live web application shows the successful use of the cloud for hosting as well as for database hosting, which can be seen by the data that is pulled from the database for the purpose of display. This web application is one of the main ways that the user will interact with the system, and as such it should display important information quickly and intuitively. The hosting is the first major hurdle to accomplishing this task, and successive deliverables will build upon this foundation.

**5.0 Wearable Design**

**5.1 Description and Goal:**

The goal of wearable design is to make the Arduino 101 portable so that it will meet the function as a watch. The choice of battery determines the operating life of the wearable and the safety of use for users in the future. We have selected a battery for the wearable that will allow a user to wear it as a watch.

**5.2 Procedure:**

Upload the code onto Arduino 101 first. Connect the battery shield to Arduino board then turn on the switch of the battery. Then it will allow the battery to operate independently from the laptop cable.

**5.3 Verifiable Result:**

With fully charged battery, the Arduino can be supported up to twenty hours without other battery support.

**6.0 Adaptive Mode**

**6.1 Description and Goal:**

When a wearable is not present in the room, the system should operate automatically so that the lights are controlled at all times. The mechanism that makes this possible is our adaptive mode control. The driving force of this mode are “single pixel sensors” which record the thermal status in the room as RGB data. In order for the Luminesense system to respond to occupants and other changes in the environment, these sensors are crucial.

**6.2 Procedure:**

Connect laptop to SLURP network via ethernet and thunderbolt cable.

In terminal, enter ssh pi@192.168.1.202

Once connected to Raspberry Pi, enter these commands:

$Cd doug/UROP\_SinglePixelLocalization/raspberryPi-drivers

$MQTTMux.py -- host 192.168.1.220 -- time 100 -- gain 60

On local machine, move to “recorder” subdirectory of singlepixellocalization-master direct

To begin streaming sensor data in text file, run the command:

$java -jar recorder.jar config.yml /Users/michaelhaley/Desktop CSV

A .txt file will appear in the selected directory (e.g. Desktop) containing each of the 12 single pixel sensors recorded blue, red, green, white values.

**6.3 Verifiable Result:**

The verifiable result will be the RGB values displayed in the .txt file, showing the data that each single pixel sensor is capturing at any given moment. These values will be relatively steady when the room is unchanged and spike when stimulated (motion, changing lights, etc).

**7.0 Luminaire Controls**

**7.1 Description and Goal:**

An additional important deliverable is to demonstrate advanced control over the luminaires. In previous deliverables, only two actions could be performed on the luminaire: on and off. The client desires more features such as brilliance control (brighten, dimmen) as well as color temperature control (red, blue, green light component manipulation). The luminaire is driven by a Particle Photon. The Photon is uniquely identified via device ID and an access token; these attributes are also used in the authentication process. Particle API function calls are used to communicate with and send instructions to the Photon. The Photon currently holds the code *luminaire\_control.ino* that receives an input and switches the desired luminaires’ states depending on the command.

**7.2 Procedure:**

Power the Particle Photon and open the IDE and flash the code *luminesense\_desired\_control.ino* to the Photon. Communication with the Photon will take place with a PC over the Particle Cloud API. The *luminaire\_control.ino* code enables the cloud connection from the PC and also contains the *luminaire\_control* function that parses the input from the PC. The input command is received is the gesture command. After parsing the data, the *luminaire\_desired\_control* function sets the TX (Blue LED), WKP(Green LED), A4(Red LED) and the A5(White LED) pins to HIGH or LOW depending on the gesture command. The instructed commands from desired\_control would be disco (show off Red, Green, Blue manipulation), brighten and dimmer.

**7.3 Verifiable Result:**

Another major facet of the deliverable is the ability to change the color temperature of the lights and change states of multiple lights simultaneously . This test confirms the correct transmission of command data from the PC to the Luminaire. As a result, the Photon consequently changes the state of the luminaire. The luminaire accurately changes state with depending on the desired command. The test provided showcases mastery of luminaire control but also highlights a positive response from the system to newly integrated features and functionality.

**8.0 Latency Determination**

**8.1 Description and Goal:**

Since the Gesture-control mode requires the Luminaires respond to gesture commands immediately, there should be little latency for a luminaire to react when a gesture is performed. Since last semester, the latency was a crucial feature that has to be terminated. In order to achieve the goal the cause of delay has to be determined.

**8.2 Procedure:**

To determine the latency, the test is divided into three parts: IMU data collection; Bluetooth Low Energy data transportation; Wi-Fi data transportation. At first the latency was assumed to be caused by the “print” commands, but when those commands were commented out there was still significant delay. To test the first part, IMU data collection, “CurieTime” library is used. The Arduino Curie will print out time in a form of “hour:minute:second” every time there is a change in IMU reading.

**8.3 Verifiable Result:**

The test results show that the latency is caused by the IMU data collection based on the time that the IMU updates on the Arduino Serial Monitor. The when the position of the Arduino board is changed, it will take five seconds on average. But once the IMU reading is updated, the Raspberry Pi will receive the IMU data immediately and also the luminaire will respond to that command.