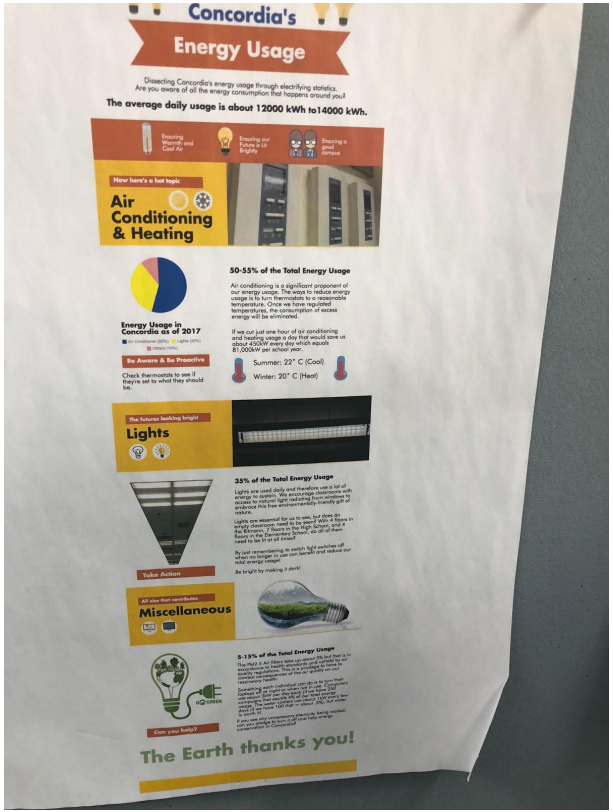
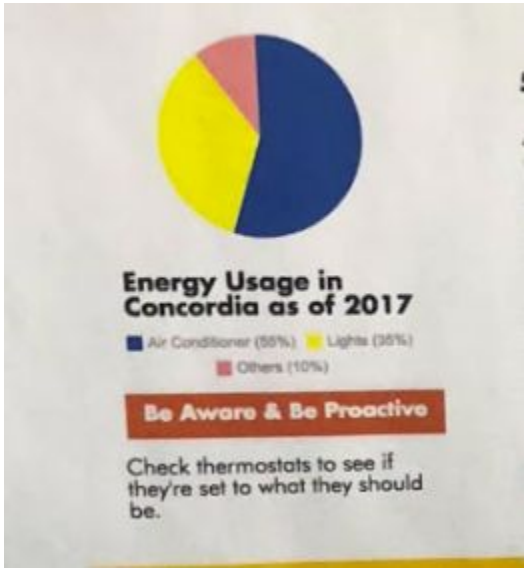
Energy and Light Conservation

Zi Xuan Teh and Jeremy Jung

Problems Addressed





Lights: 35% of Total Energy Usage

CISS Light Automator 9000

Materials used:

-ESP8266

-TSL2591 infrared/visible light sensor

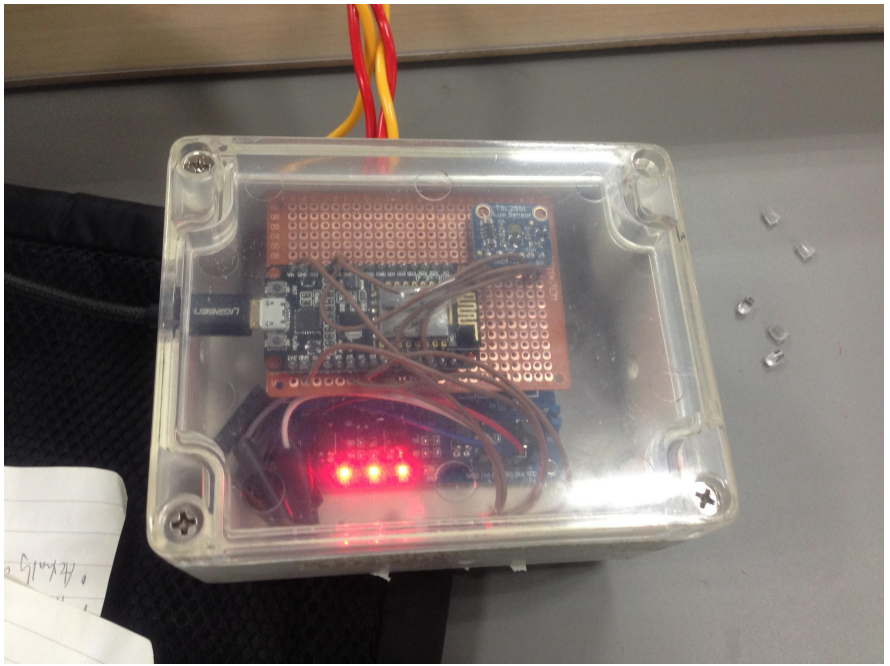
-Micro AC adapter

-Electrical cables, socket, and plug

-SPDT relay

-Jumper wires and solder

-Plastic Box



Software (Setup)

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPClient.h>
#include <ESP8266httpUpdate.h>
#include <DNSServer.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_TSL2591.h"
#include <WiFiUdp.h>

//Values for GPIO pins on the ESP8266
static const uint8_t D0   = 16;
static const uint8_t D1   = 5;
static const uint8_t D2   = 4;
static const uint8_t D3   = 0;
static const uint8_t D4   = 2;
static const uint8_t D5   = 14;
static const uint8_t D6   = 12;
static const uint8_t D7   = 13;
static const uint8_t D8   = 15;
static const uint8_t D9   = 3;
static const uint8_t D10  = 1;
```

Libraries and ESP pins

Software (Setup)

```
void setup() {
  // Sets up pins D6, D5, and D7 to output and sets them to on. The connections are working if all three LEDs on the relay are on.
  pinMode(D6, OUTPUT);
  digitalWrite(D6, LOW);
  pinMode(D7, OUTPUT);
  digitalWrite(D7, LOW);
  pinMode(D5, OUTPUT);
  digitalWrite(D5, LOW);

  Serial.println(F("Starting Adafruit TSL2591 Test!"));

  if (tsl.begin())
  {
    Serial.println(F("Found a TSL2591 sensor"));
  }
  else
  {
    Serial.println(F("No sensor found ... check your wiring?"));
    while (1);
  }

  /* Display some basic information on this sensor */
  displaySensorDetails();

  /* Configure the sensor */
  configureSensor();

  Serial.begin(9600);
  delay(10);
  Serial.println("Starting up...");
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  IPAddress ip(172, 18, 255, 24); // Assigned IP Address. Can be changed.
  IPAddress gateway(172, 18, 255, 254); // Network gateway for CISS_Visitors
  Serial.print(F("Setting static ip to : "));
  Serial.println(ip);
  IPAddress subnet(255, 255, 240, 0); // Subnet mask for CISS_Visitors
  WiFi.config(ip, gateway, subnet);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) //Waits to get connected.
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.println("WiFi connected");
  server.begin();
  Serial.println("Server started");
  Serial.print("MAC Address: ");
  Serial.println(WiFi.macAddress());
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
  Serial.println("Starting UDP");
  udp.begin(localPort);
  Serial.print("Local port: ");
  Serial.println(udp.localPort());
}
```

-Setup

-Connects to WiFi and hosts web server

-Web server hosted at desired IP address

Software (Webpage)

-HTML page

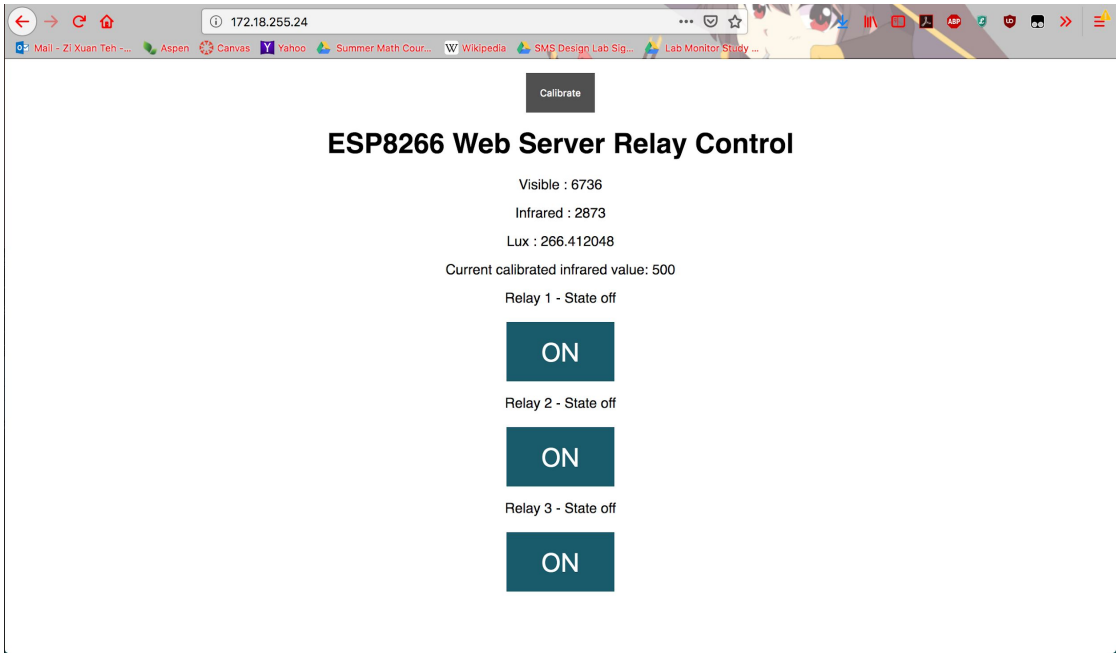
-Data display

-Buttons

```
void loop() {
  delay(1000);
  getTime();
  uint32_t lum = tsl.getFullLuminosity();
  uint16_t ir, full;
  ir = lum >> 16;
  full = lum & 0xFFFF;
  Serial.print(F("[ ")); Serial.print(millis()); Serial.print(F(" ms ] "));
  Serial.print(F("IR: ")); Serial.print(ir);  Serial.print(F("  "));
  Serial.print(F("Full: ")); Serial.print(full); Serial.print(F("  "));
  Serial.print(F("Visible: ")); Serial.print(full - ir); Serial.print(F("  "));
  Serial.print(F("Lux: ")); Serial.println(tsl.calculateLux(full, ir), 6);
  ir5 = ir4;
  ir4 = ir3;
  ir3 = ir2;
  ir2 = ir1;
  ir1 = ir;

 WiFiClient client = server.available();   // Listen for incoming clients
 if (client) {                             // If a new client connects,
   Serial.println("New Client.");          // print a message out in the serial port
   String currentLine = "";                // make a String to hold incoming data from the client
   while (client.connected()) {            // loop while the client's connected
     if (client.available()) {             // if there's bytes to read from the client,
       char c = client.read();             // read a byte, then
       Serial.write(c);                    // print it out the serial monitor
       header += c;
       if (c == '\n') {                    // if the byte is a newline character
         // if the current line is blank, you got two newline characters in a row.
         // that's the end of the client HTTP request, so send a response:
         if (currentLine.length() == 0) {
           // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
           // and a content-type so the client knows what's coming, then a blank line:
           client.println("HTTP/1.1 200 OK");
           client.println("Content-type:text/html");
           client.println("Connection: close");
           client.println("Refresh: 3");     //Refreshes the page every 2 seconds to update lux values on the website.
           client.println();

         else if (header.indexOf("GET /2/on") >= 0) {
           Serial.println("Relay 2 on");
           D7State = "on";
           digitalWrite(D7, HIGH);
         } else if (header.indexOf("GET /2/off") >= 0) {
           Serial.println("Relay 2 off");
           D7State = "off";
           digitalWrite(D7, LOW);
         }

         else if (header.indexOf("GET /3/on") >= 0) {
           Serial.println("Relay 3 on");
           D5State = "on";
           digitalWrite(D5, HIGH);
         } else if (header.indexOf("GET /3/off") >= 0) {
           Serial.println("Relay 3 off");
           D5State = "off";
           digitalWrite(D5, LOW);
         }
         else if (header.indexOf("GET /calibrate") >= 0) {
           Serial.println("Calibrating...");
           irThreshold = calibrateSensor();
           Serial.println("Calibration complete.");
         }
         // Display the HTML web page
         client.println("<!DOCTYPE html><html>");
         client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
         client.println("<link rel=\"icon\" href=\"data:,\">");
         // CSS to style the on/off buttons
         // Feel free to change the background-color and font-size attributes to fit your preferences
         client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");
         client.println(".button {background-color: #1950A4; border: none; color: white; padding: 16px 40px;");
         client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
         client.println(".button2 {background-color: #77878A;}");
         client.println(".button3 {background-color: #505050; border: none; color: white; padding: 16px; text-align: left;}</style></head>");
```

```
         // Web Page Heading
         client.println("<p><a href=\"/calibrate\"><button class=\"button3\">Calibrate</button></a></p>");
         client.println("</body></html>");
         client.println("<body><h1>ESP8266 Web Server Relay Control</h1>");
         client.print("<p>Visible : "); client.print(full - ir); client.println("</p>");
         client.print("<p>Infrared : "); client.print(ir); client.println("</p>");
         client.print("<p>Lux : "); client.print(tsl.calculateLux(full, ir), 6); client.println("</p>");
         client.print("<p>Current calibratated infrared value: " ); client.print(irThreshold / 5); client.println("</p>");
         // Display current state, and ON/OFF buttons for GPIO 4


         // e.g if the D6State is off, it displays the ON button
         // The "on" button will print /(relay number)/on, and the off will print /(relay number)/off
         client.println("<p>Relay 1 - State " + D6State + "</p>");
         if (D6State == "off") {
           client.println("<p><a href=\"/1/on\"><button class=\"button\">ON</button></a></p>");
         } else {
           client.println("<p><a href=\"/1/off\"><button class=\"button button2\">OFF</button></a></p>");
         }

         client.println("<p>Relay 2 - State " + D7State + "</p>");
         if (D7State == "off") {
           client.println("<p><a href=\"/2/on\"><button class=\"button\">ON</button></a></p>");
         } else {
           client.println("<p><a href=\"/2/off\"><button class=\"button button2\">OFF</button></a></p>");
         }

         client.println("<p>Relay 3 - State " + D5State + "</p>");
         if (D5State == "off") {
           client.println("<p><a href=\"/3/on\"><button class=\"button\">ON</button></a></p>");
         } else {
           client.println("<p><a href=\"/3/off\"><button class=\"button button2\">OFF</button></a></p>");
         }
         // The HTTP response ends with another blank line
         client.println();
         // Break out of the while loop
         break;
       } else { // if you got a newline, then clear currentLine
         currentLine = "";
               }
             } else if (c != '\r') {  // if you got anything else but a carriage return character,
               currentLine += c;      // add it to the end of the currentLine
             }
           }
         }
         // Clear the header variable

       }
       else {
         if (ir1 + ir2 + ir3 + ir4 + ir5 > irThreshold) {
           digitalWrite(D6, LOW);
           digitalWrite(D5, LOW);
           digitalWrite(D7, LOW);
           D6State = "off";
           D5State = "off";
           D7State = "off";
           delay(500);
         }
         else {
           digitalWrite(D6, HIGH);
           digitalWrite(D7, HIGH);
           digitalWrite(D5, HIGH);
           D6State = "on";
           D5State = "on";
           D7State = "on";
           delay(500);
         }
       }
       header = "";
       // Close the connection
       Serial.flush();
     }
```

Webpage


172.18.255.24
Mail - ZI Xuan Teh -...
Aspen
Canvas
Yahoo
Summer Math Cour...
Wikipedia
SMS Design Lab Sig...
Lab Monitor Study ...
Calibrate
ESP8266 Web Server Relay Control
Visible : 6736
Infrared : 2873
Lux : 266.412048
Current calibrated infrared value: 500
Relay 1 - State off
ON
Relay 2 - State off
ON
Relay 3 - State off
ON

Software (Time)

```
void getTime() {
  //IPAddress timeServer(129, 6, 15, 28); // time.nist.gov NTP server
  //get a random server from the pool
  WiFi.hostByName(ntpServerName, timeServerIP);
  sendNTPpacket(timeServerIP); // send an NTP packet to a time server
  // wait to see if a reply is available
  delay(1000);
  int cb = udp.parsePacket();
  if (!cb) {
    Serial.println("no packet yet");
  }
  else {
    Serial.print("packet received, length=");
    Serial.println(cb);
    // We've received a packet, read the data from it
    udp.read(packetBuffer, NTP_PACKET_SIZE); // read the packet into the buffer
    //the timestamp starts at byte 40 of the received packet and is four bytes,
    // or two words, long. First, extract the two words:
    unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
    unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
    // combine the four bytes (two words) into a long integer
    // this is NTP time (seconds since Jan 1 1900):
    unsigned long secsSince1900 = highWord << 16 | lowWord;
    Serial.print("Seconds since Jan 1 1900 = " );
    Serial.println(secsSince1900);
    // now convert NTP time into everyday time:
    Serial.print("Unix time = ");
    // Unix time starts on Jan 1 1970. In seconds, that's 2208988800:
    const unsigned long seventyYears = 2208988800UL;
    // subtract seventy years:
    unsigned long epoch = secsSince1900 - seventyYears;
    // print Unix time:
    Serial.println(epoch);
    // print the hour, minute and second:
    Serial.print("The UTC+8 time is ");       // UTC is the time at Greenwich Meridian (GMT)
    Serial.print((((epoch  % 86400L) / 3600) + 8) % 24); // print the hour (86400 equals secs per day, also converted to UTC+8 time)
    Serial.print(':');
    hr = (((epoch  % 86400L) / 3600) + 8) % 24;
    if ( ((epoch % 3600) / 60) < 10 ) {
      // In the first 10 minutes of each hour, we'll want a leading '0'
      Serial.print('0');
    }
    mint = (epoch  % 3600) / 60;
    Serial.print(mint); // print the minute (3600 equals secs per minute)
    Serial.print(':');
    if ( (epoch % 60) < 10 ) {
      // In the first 10 seconds of each minute, we'll want a leading '0'
      Serial.print('0');
    }
    Serial.println(epoch % 60); // print the second
  }
  // wait ten seconds before asking for the time again
  delay(1000);

  // send an NTP request to the time server at the given address

  /* Don't hardwire the IP address or we won't get the benefits of the pool.
      Lookup the IP address for the host name instead */
}
```

-NTP request

-Time limit for automation

```
unsigned long sendNTPpacket(IPAddress & address) {
  Serial.println("sending NTP packet...");
  // set all bytes in the buffer to 0
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  // Initialize values needed to form NTP request
  // (see URL above for details on the packets)
  packetBuffer[0] = 0b11100011;   // LI, Version, Mode
  packetBuffer[1] = 0;     // Stratum, or type of clock
  packetBuffer[2] = 6;     // Polling Interval
  packetBuffer[3] = 0xEC;  // Peer Clock Precision
  // 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12]  = 49;
  packetBuffer[13]  = 0x4E;
  packetBuffer[14]  = 49;
  packetBuffer[15]  = 52;

  // all NTP fields have been given values, now
  // you can send a packet requesting a timestamp:
  udp.beginPacket(address, 123); //NTP requests are to port 123
  udp.write(packetBuffer, NTP_PACKET_SIZE);
  udp.endPacket();
}
```

```
else {
  if (hr <= 19 || hr >= 7) {
    if (ir1 + ir2 + ir3 + ir4 + ir5 > irThreshold) {
      digitalWrite(D6, LOW);
      digitalWrite(D5, LOW);
      digitalWrite(D7, LOW);
      D6State = "off";
      D5State = "off";
      D7State = "off";
      delay(500);
    }
    else {
      digitalWrite(D6, HIGH);
      digitalWrite(D7, HIGH);
      digitalWrite(D5, HIGH);
      D6State = "on";
      D5State = "on";
      D7State = "on";
      delay(500);
    }
  }
  else {
    digitalWrite(D6, LOW);
    digitalWrite(D7, LOW);
    digitalWrite(D5, LOW);
    D6State = "low";
    D5State = "low";
    D7State = "low";
  }
}
```

Software (TSL Setup)

-TSL Initialization

-Data Reading

```
void configureSensor(void)
{
  // You can change the gain on the fly, to adapt to brighter/dimmer light situations
  //tsl.setGain(TSL2591_GAIN_LOW);    // 1x gain (bright light)
  tsl.setGain(TSL2591_GAIN_MED);      // 25x gain
  //tsl.setGain(TSL2591_GAIN_HIGH);   // 428x gain

  // Changing the integration time gives you a longer time over which to sense light
  // longer timelines are slower, but are good in very low light situtations!
  //tsl.setTiming(TSL2591_INTEGRATIONTIME_100MS);  // shortest integration time (bright light)
  // tsl.setTiming(TSL2591_INTEGRATIONTIME_200MS);
  tsl.setTiming(TSL2591_INTEGRATIONTIME_300MS);
  // tsl.setTiming(TSL2591_INTEGRATIONTIME_400MS);
  // tsl.setTiming(TSL2591_INTEGRATIONTIME_500MS);
  // tsl.setTiming(TSL2591_INTEGRATIONTIME_600MS);  // longest integration time (dim light)
}
```

```
uint32_t lum = tsl.getFullLuminosity();
uint16_t ir, full;
ir = lum >> 16;
full = lum & 0xFFFF;
Serial.print(F("[ ")); Serial.print(millis()); Serial.print(F(" ms ] "));
Serial.print(F("IR: ")); Serial.print(ir);  Serial.print(F("  "));
Serial.print(F("Full: ")); Serial.print(full); Serial.print(F("  "));
Serial.print(F("Visible: ")); Serial.print(full - ir); Serial.print(F("  "));
Serial.print(F("Lux: ")); Serial.println(tsl.calculateLux(full, ir), 6);
```

Software (Calibration)

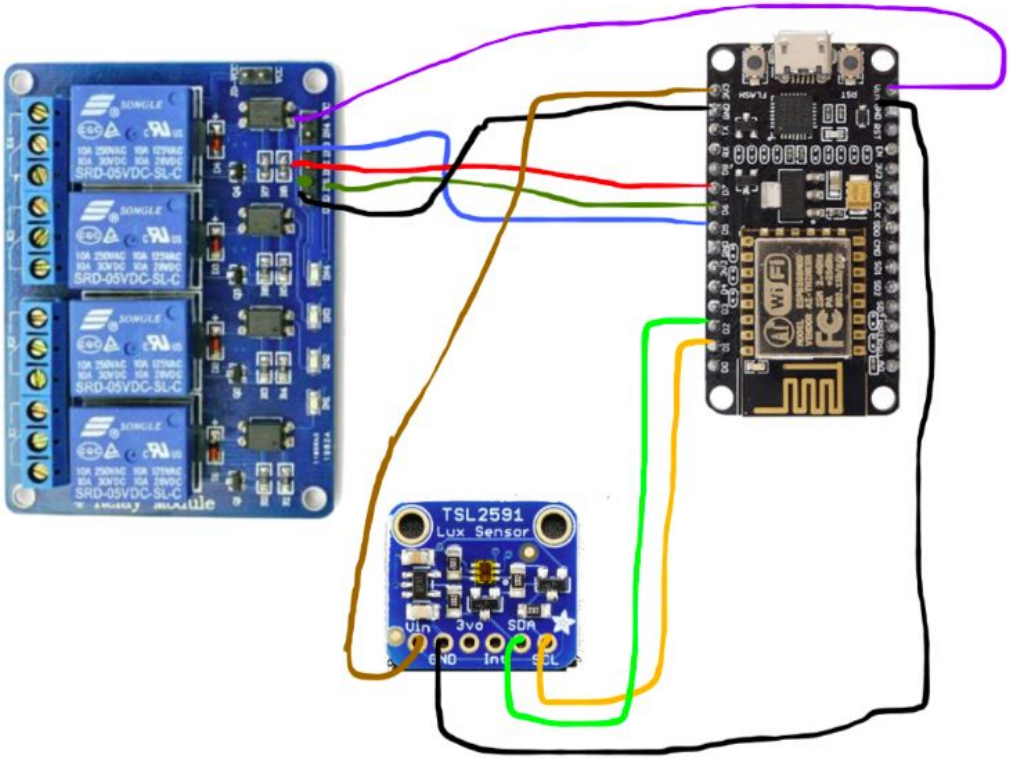
```
uint16_t totalIR = 0;
int calibrateSensor() {
  int n = 0;
  while (n != 5) {
    uint32_t lum = tsl.getFullLuminosity();
    uint16_t ir, full;
    ir = lum >> 16;
    full = lum & 0xFFFF;
    totalIR += ir;
    delay(500);
    Serial.print(".");
    Serial.print(totalIR);
    n++;
  }
  Serial.println(totalIR);
  return totalIR;
}
```

```
  else {
    if (ir1 + ir2 + ir3 + ir4 + ir5 > irThreshold) {
      digitalWrite(D6, LOW);
      digitalWrite(D5, LOW);
      digitalWrite(D7, LOW);
      D6State = "off";
      D5State = "off";
      D7State = "off";
      delay(500);
    }
    else {
      digitalWrite(D6, HIGH);
      digitalWrite(D7, HIGH);
      digitalWrite(D5, HIGH);
      D6State = "on";
      D5State = "on";
      D7State = "on";
      delay(500);
    }
  }
  header = "";
  // Close the connection
  Serial.flush();
}
```

```
  else if (header.indexOf("GET /calibrate") >= 0) {
    Serial.println("Calibrating...");
    irThreshold = calibrateSensor();
    Serial.println("Calibration complete.");
  }
```

Explanation:

Hardware: Wiring (without load)

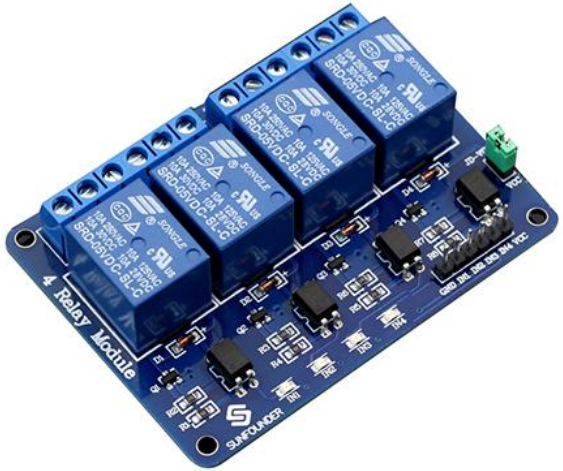


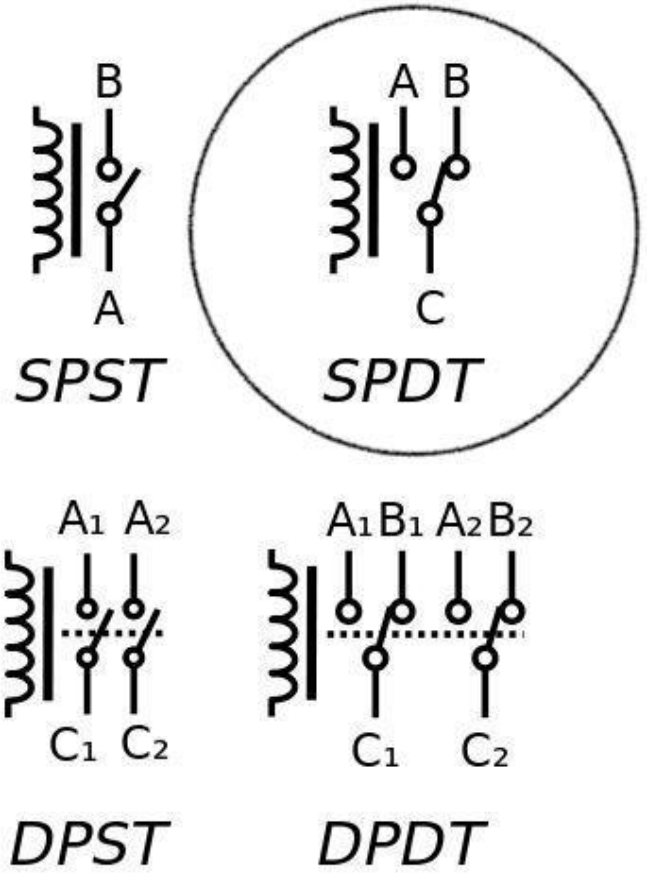
Hardware: Relay

Single Pole Double Throw Relay

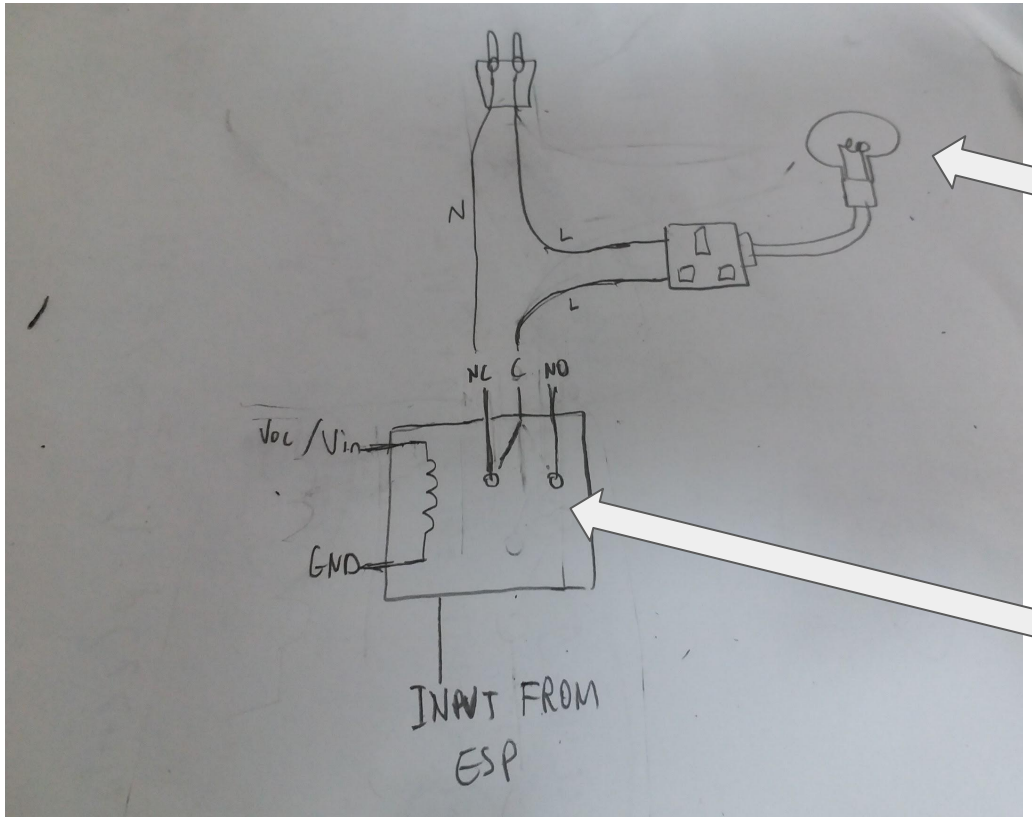
Normally Closed

-Disconnection when relay is activated





Hardware: Relay-Lights



Light ON when NC-C
Light OFF when NO-C

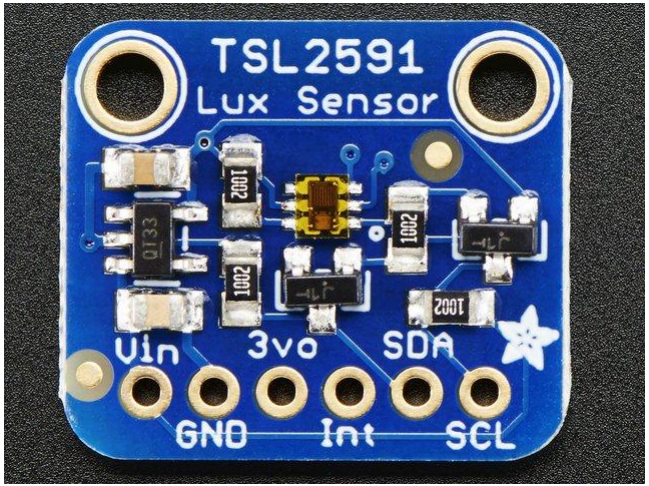
NC-C when digitalWrite(LOW)
NO-C when digitalWrite(HIGH)

Hardware: TSL2591

Luminosity Sensor

Measures Infrared, Full-spectrum, or human-visible light

-Infrared: Sunlight



Guide

1) Connect the plugs
2) Press RST button with a long object
3) Watch in awe and wonder

Struggles: Software & Hardware

Software:

-Getting real-time data (refresh)

-upload_mem_failed (Hold flash, plug ESP in, release Flash)

Hardware:

-Faulty wires, inconsistent photoresistor (additional solder, wire organization)

-Unresponsive Relay (changed model completely)

Project Outline

What do we need?

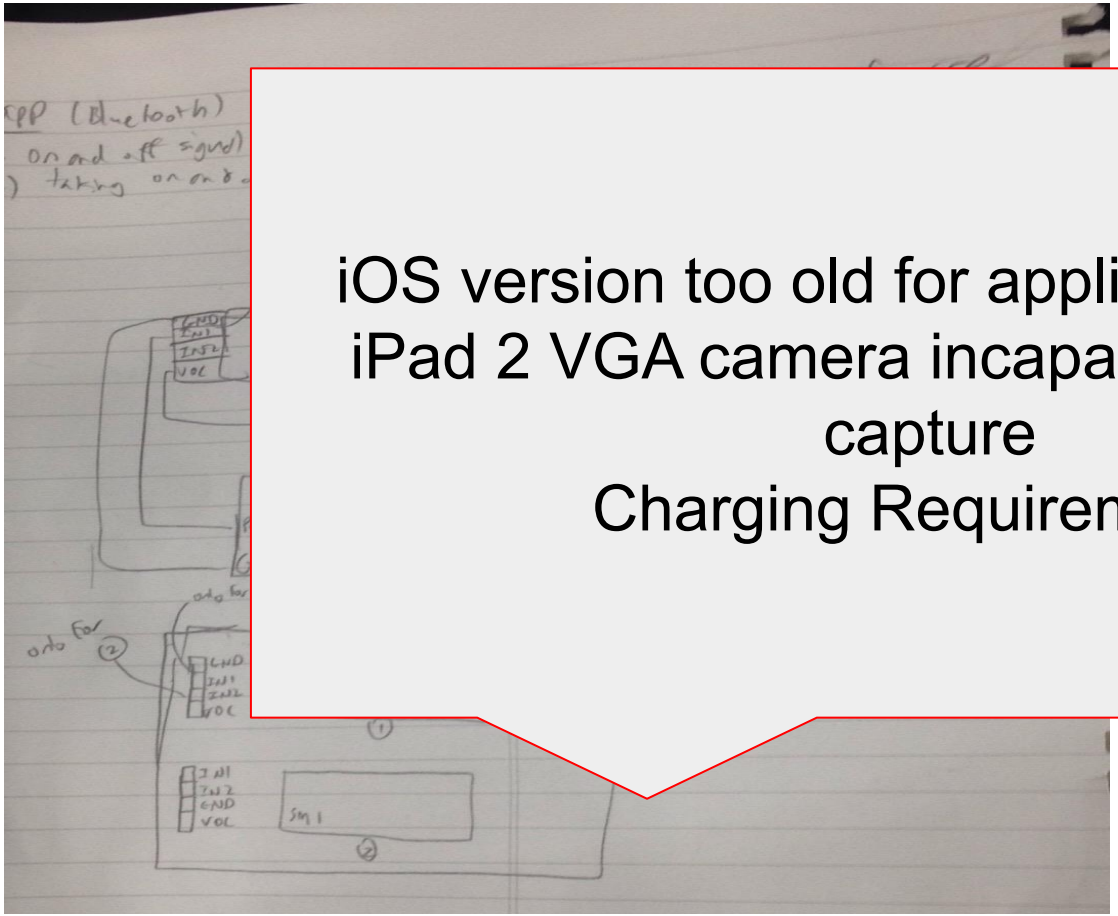
-Light sensing Capability

-Means of communication between the client and the main apparatus.

-Way to turn any AC current on and off.

-A user interface

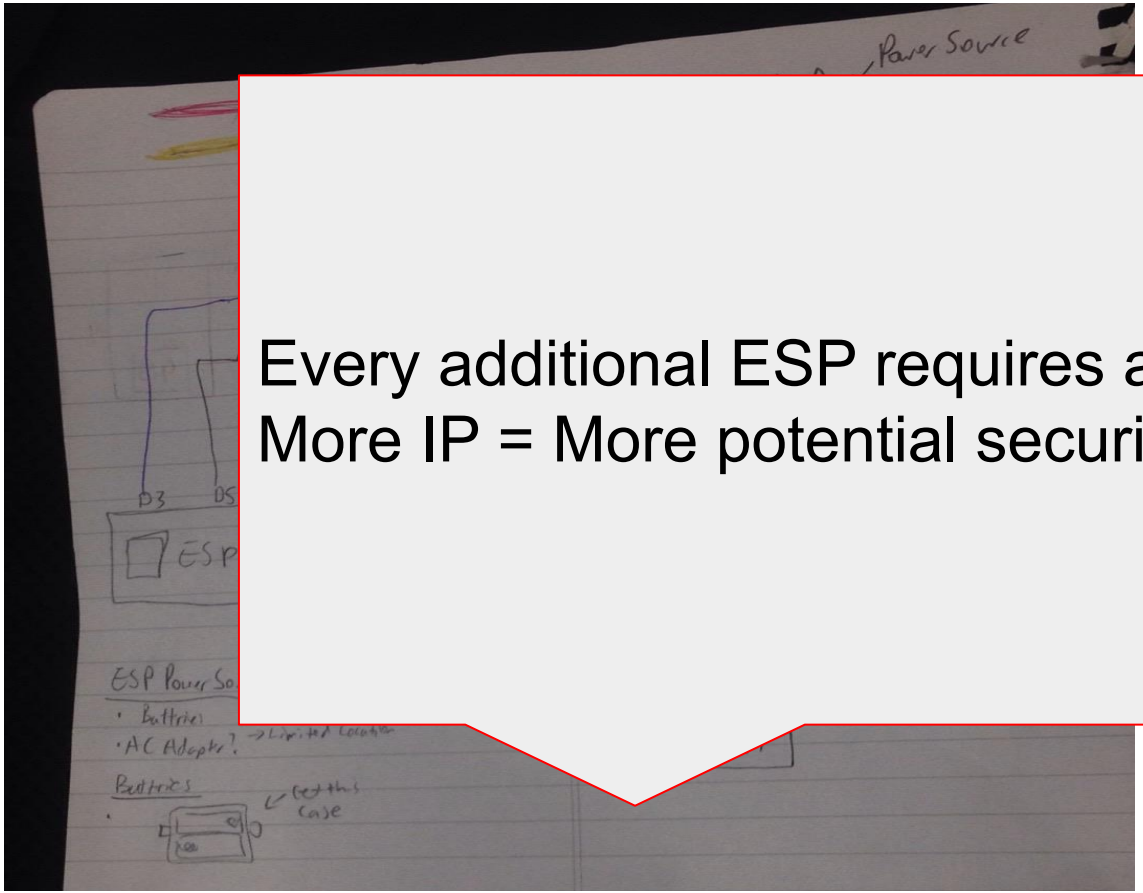
Version 1.0: Arduino and iPad



iOS version too old for application support
iPad 2 VGA camera incapable of infrared capture
Charging Requirements

tion UI
receive
ata,
tion)
t Sensor

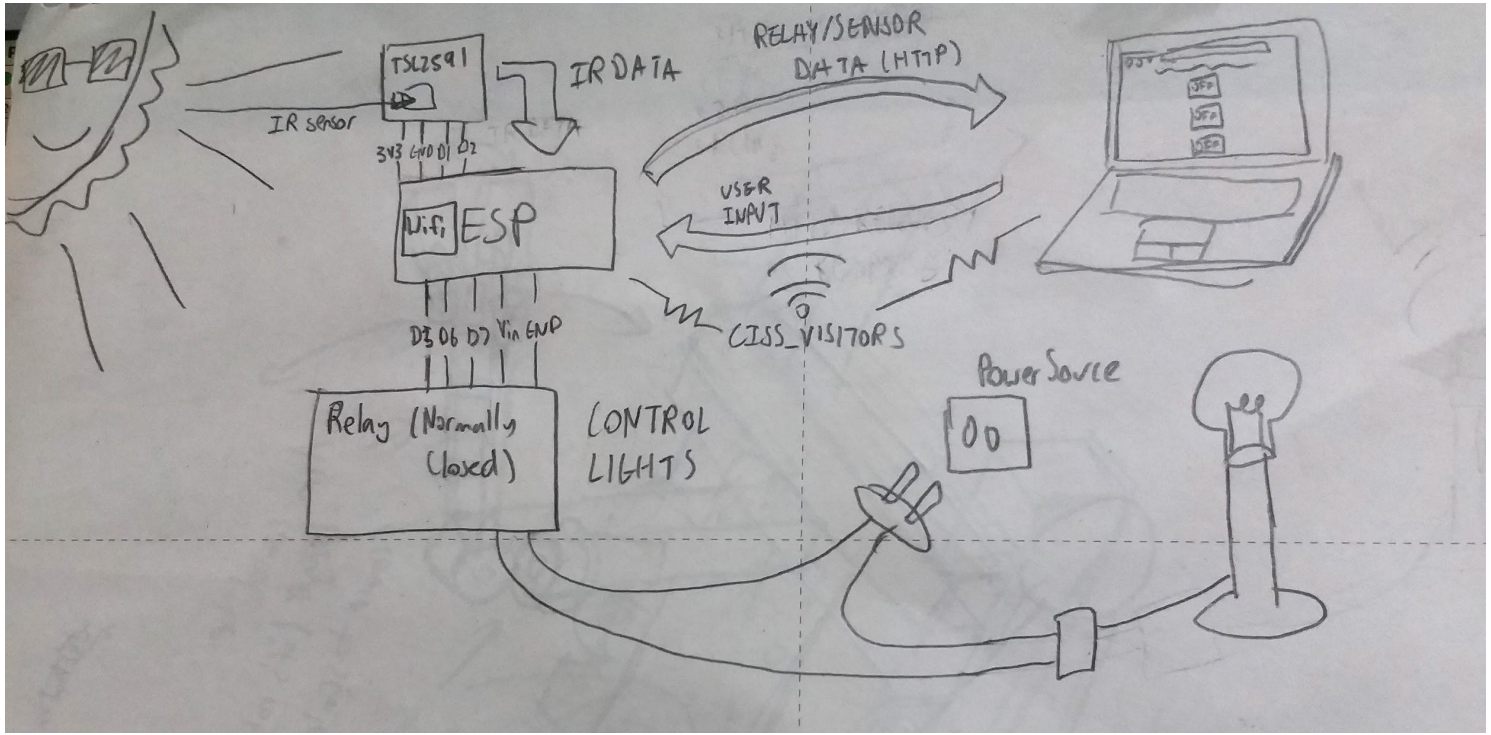
Version 1.1: 2 ESP and web UI



Every additional ESP requires a power source
More IP = More potential security breach

e
equests
tes,

Final Version: 1 ESP and web UI



Final thoughts