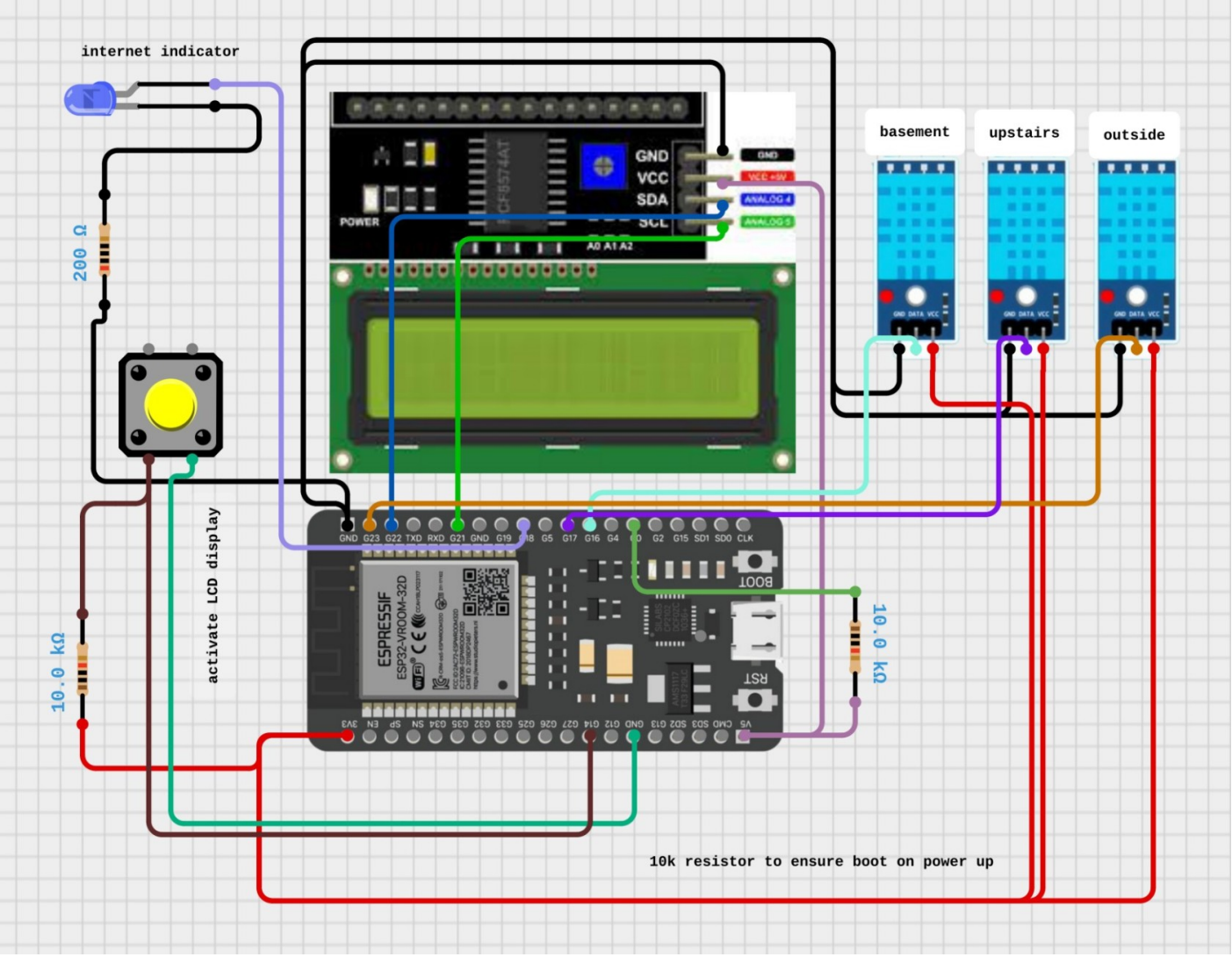
Temperature Monitoring (indoor and outdoor)

This project aims to relay indoor and outdoor temperature (and optionally humidity) to a server log. This project is fairly simple because we can use off-the-shelf DHT11 and DHT22 sensors (DHT22 is for below 0c).

Note to self: avoid non-oem DHT22 modules. You only save a couple of dollars and they are in my experience completely unreliable. The cheap clones either don't work or die after a few hours. I pay about $6 per unit for OEM (original) AM2302 chips, which have been running without issue for months.

The diagram below includes a 10k resistor between VIN (5v) and GPIO 0 because the esp32 unit I was using would not boot on power-up without having to hit the reset button. This is a common flaw with the esp32 units but so far for me only happened with one of the dozens I've used so far.

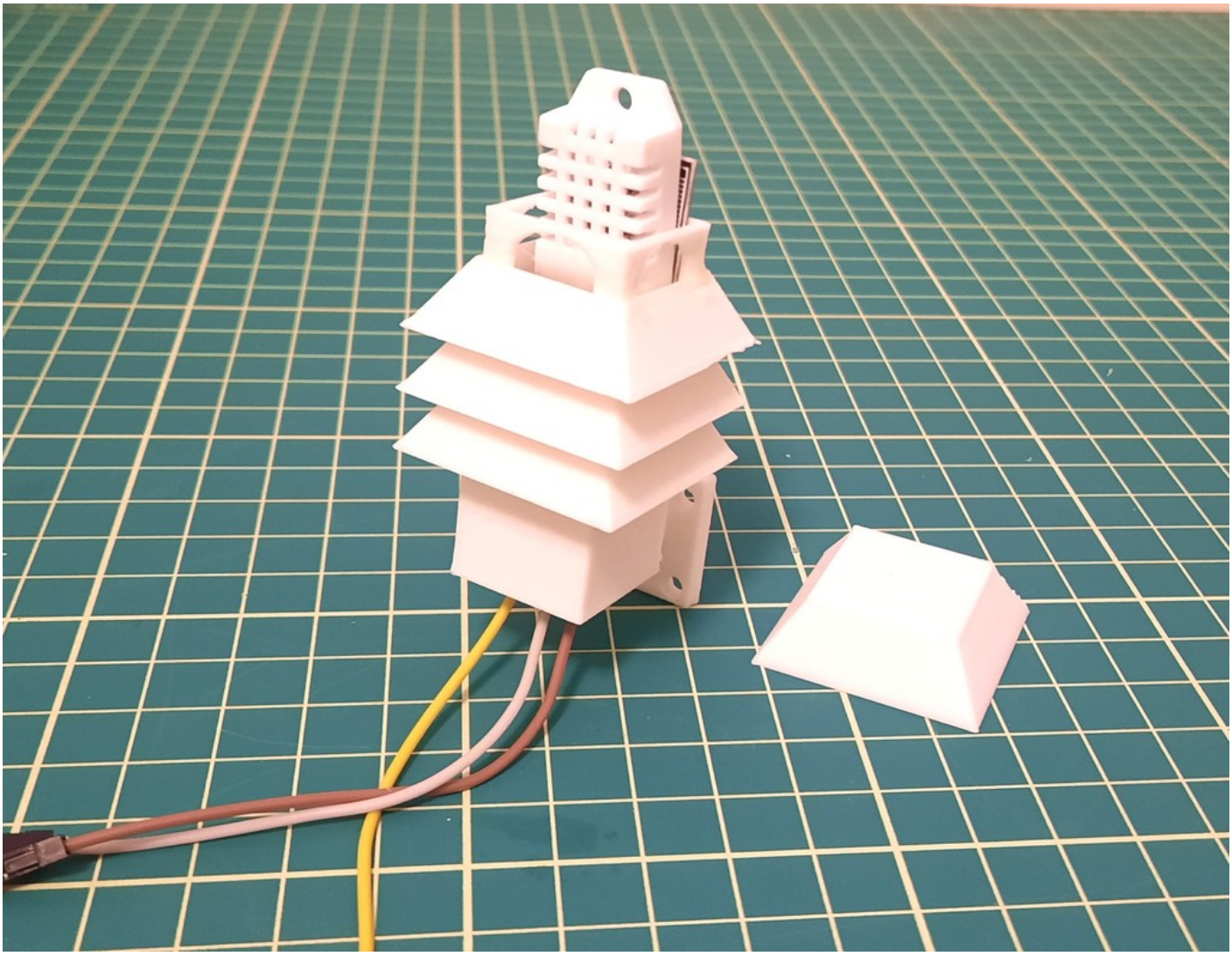


For the indoor sensor I used this case:

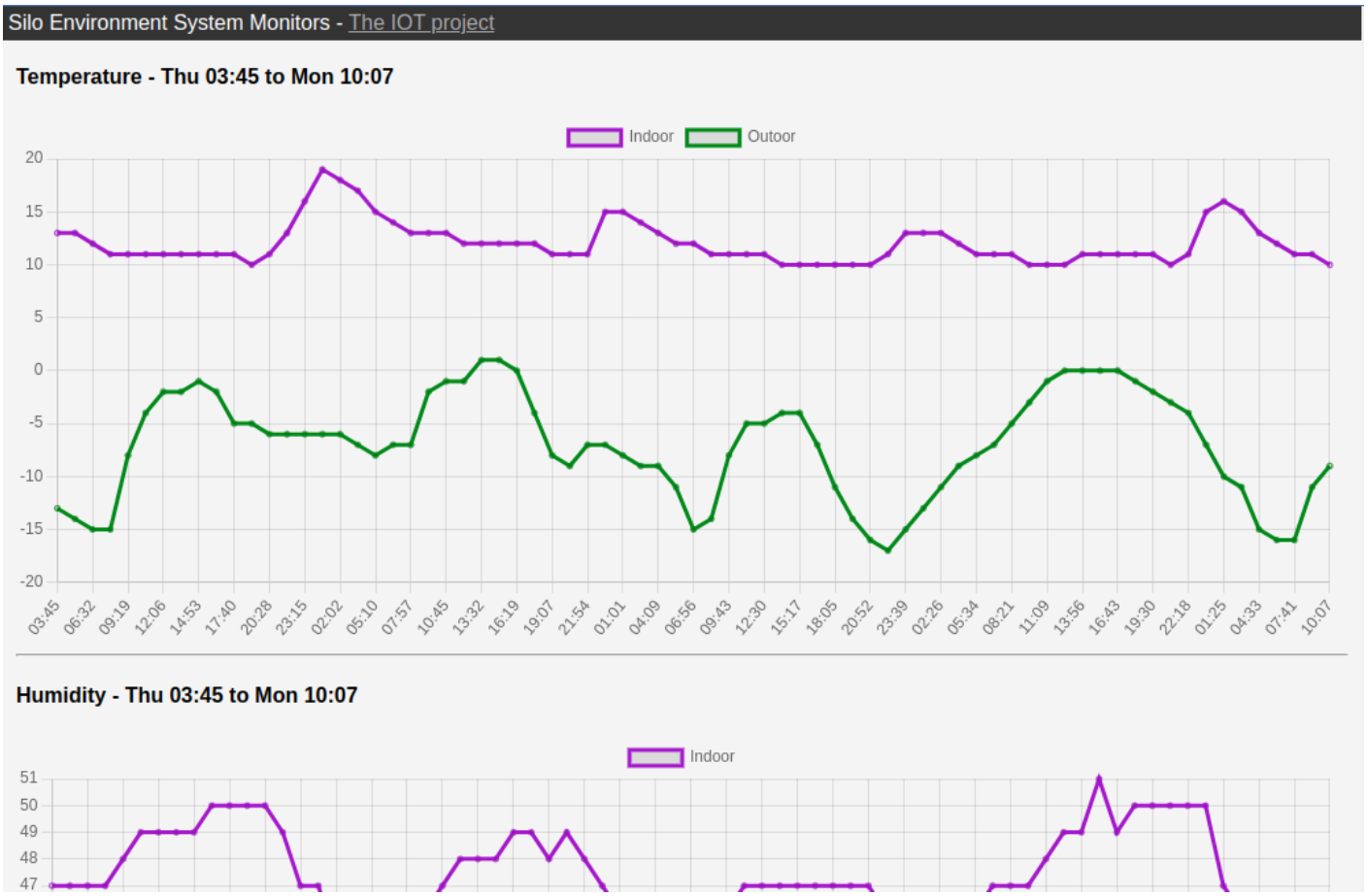
https://www.thingiverse.com/thing:2497711

And for the outdoor sensor I used this case:

https://www.thingiverse.com/thing:6400888



The end result in my case is a chart showing the temperature over a couple of days. If I see the temperature drop significantly (like pipes freezing zone) I at least can get out there to remedy the situation before it becomes an issue.



And the code. It ain't pretty but it works. For the server-side code I am just using free web hosting since the load is almost nothing. For the ESP32 module, the code below is about as minimal as I dare go. I do record the server success/fail rates but if everything seems stable after a few weeks I will remove that too.

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <DHT_U.h>
#include <EEPROM.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"


////////////////////////////////
// EDIT THIS SECTION
// last updated June 23, 2024
```

```
int activeConnection = 1;

// lab
const String ssid1 = "asiuy2G";
const String password1 = "654765";

// other place
const String ssid2 = "cheytr";
const String password2 = "87654";

const String temperatureUrl = "https://homelab.htd.org/set_temp_humidity.php?data=";

// end edit section
//////////////////

#define EEPROM_SIZE 48
int eepromPingsAddress = 0;
float totalServerPings = 0;
int eepromFailedPingsAddress = 1;
float totalServerPingFails = 0;
int eepromActiveConnectionAddress = 1;

String prevTemperatureText = "";
String prevHumidifyText = "";
String prevConnectivityText1 = "";
String prevConnectivityText2 = "";

const int basementTemperaturePin = 16;
const int upstairsTemperaturePin = 17;
const int outdoorTemperaturePin = 23;



const int buttonPin = 14;
int buttonState = 0;
int displayMode = 1;  // 1: temp, humidiy  2: internet

const int pulseRate = 2;  // 2 seconds

const int temperatureServerSendInterval = 1;                                    // 10 minutes between sending a
```

```
temperature update to the server
const int temperatureSamplesPerReading = (60 * temperatureServerSendInterval) / pulseRate;  // every x
minutes send a sample to the server

int temperatureLoopCount = 0;

int lcdBacklightOnCounter = 0;



long secondsOfHour = 0;

const int secondsPerHour = 3600;  // set to 60 for debugging

const int hoursPerDay = 24;









float currentBasementTemperature = 0;
float currentUpstairsTemperature = 0;
float currentOutdoorTemperature = 0;

float currentBasementHumidity = 0;
float currentUpstairsHumidity = 0;
float currentOutdoorHumidity = 0;



bool connectingToWifi = false;
bool wifiConnected = false;
bool wifiPaused = false;
int wifiPausedTick = 0;
bool wifiSleeping = false;
bool debug = false;  // when true server does not update
bool serverFailed = false;
int wifiConnectionAttempts = 0;

HTTPClient http;
```

```
DHT_Unified dhtBasement(basementTemperaturePin, DHT11);
DHT_Unified dhtUpstairs(upstairsTemperaturePin, DHT11);
DHT_Unified dhtOutdoor(outdoorTemperaturePin, DHT11);  //DHT22 is actually in ouse outside

LiquidCrystal_I2C lcd_i2c(0x27, 16, 2);

sensors_event_t basementSensorEvent;
sensor_t basementSensor;
sensors_event_t upstairsSensorEvent;
sensor_t upstairsSensor;
sensors_event_t outdoorSensorEvent;
sensor_t outdoorSensor;

#define LED_PIN 18  // ESP32 pin GPIO18 connected to LED



void setup() {

  // We sometimes run into brownouts due to main hydro line voltage drops. For this situation set BOD to 3 volts
  // WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);  //disable brownout detector
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0x03);  // 3 volts
  //WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0x04); // set brownout detector to 3.2 volts
  // WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0x07); // Set 3.3V as BOD level

  Serial.begin(115200);  // this needs to match the value in the serial monitor

  //Init EEPROM
  EEPROM.begin(EEPROM_SIZE);

  pinMode(buttonPin, INPUT); // button for displaying info on LCD
  pinMode(LED_PIN, OUTPUT); // led to indicate internet connectivity

  delay(2000);

  dhtBasement.begin();
  dhtUpstairs.begin();
  dhtOutdoor.begin();

  dhtBasement.temperature().getSensor(&basementSensor);
```

```
dhtBasement.humidity().getSensor(&basementSensor);

dhtUpstairs.temperature().getSensor(&upstairsSensor);
dhtUpstairs.humidity().getSensor(&upstairsSensor);

dhtOutdoor.temperature().getSensor(&outdoorSensor);
dhtOutdoor.humidity().getSensor(&outdoorSensor);

getSensorReadings();

EEPROM.begin(EEPROM_SIZE);

activeConnection = EEPROM.read(eepromActiveConnectionAddress);

Serial.print("eepromActiveConnection :");
Serial.println(activeConnection);

if (isnan(activeConnection)) {
  activeConnection = 1;
}

if (activeConnection == 0) {
  activeConnection = 1;
}

if (activeConnection > 2) {
  activeConnection = 2;
}
Serial.print("activeConnection: ");
Serial.println(activeConnection);

if (!connectToWiFi()) {
  delay(2000);
  WiFi.disconnect();
  delay(1000);
  if (activeConnection == 1) {
    activeConnection = 2;
  } else {
    activeConnection = 1;
  }
```

```
    connectToWiFi();
  }

  // in case we did a reboot allow remote server a moment
  delay(5000);

  // if wifi connect failed again just reboot
  if (WiFi.status() != WL_CONNECTED) {
    ESP.restart();
  }

  lcd_i2c.init();
  lcd_i2c.backlight();
}


void loop() {

  secondsOfHour += pulseRate;

  // after 24 hours reset this integer
  if (secondsOfHour > 86400) {
    secondsOfHour = 1;
    // Serial.println("restarting :");
    ESP.restart();
  }

  if (lcdBacklightOnCounter == 0) {
    lcd_i2c.noBacklight();
  } else {
    // if lcb backlight counter is over zero it is active so increment
    lcdBacklightOnCounter++;
    // turn off backlight after 300 5 mintes of inactivity
    if (lcdBacklightOnCounter % 300 == 0) {
      lcd_i2c.noBacklight();
      lcdBacklightOnCounter = 0;
    }
  }

  buttonState = digitalRead(buttonPin);
```

```
  if (buttonState == LOW) {
    lcd_i2c.clear();
    lcd_i2c.backlight();

    if (lcdBacklightOnCounter == 0) {
      displayMode = 1;
    } else {
      displayMode++;
      if (displayMode > 2) {
        displayMode = 1;
      }
    }
    lcdBacklightOnCounter = 1;
  }

  sensors_event_t basementSensorEvent;

  ////////////////////////////
  // TEMPERATURE AND HUMIDITY

  // no need to read the temperature every second
  if (temperatureLoopCount % 10 == 0) {
    getSensorReadings();
  }


  String temperatureText = "T U:" + String(currentUpstairsTemperature, 0) + " B:" +
String(currentBasementTemperature, 0) + " E:" + String(currentOutdoorTemperature, 0) + "c";
  String humidifyText = "H U:" + String(currentUpstairsHumidity, 0) + " B:" + String(currentBasementHumidity,
0) + " E:" + String(currentOutdoorHumidity, 0) + "";

  if (wifiPaused) {
    if (wifiPausedTick > 60 && wifiConnectionAttempts == 0) {  // wait 1 minute before attempting to reconnect
      wifiPaused = false;
      wifiPausedTick = 0;
      connectToWiFi();
    } else {
      wifiPausedTick++;
    }
```

```
  }

  //////////////////////
  // SERVER RELAY


  if (!debug && (temperatureLoopCount >= temperatureSamplesPerReading) || serverFailed) {
    String recordedUpstairsTemperature = String(currentUpstairsTemperature, 1);
    String recordedUpstairsHumidity = String(currentUpstairsHumidity, 1);
    String recordedBasementTemperature = String(currentBasementTemperature, 1);
    String recordedBasementHumidity = String(currentBasementHumidity, 1);
    String recordedOutdoorTemperature = String(currentOutdoorTemperature, 1);
    String recordedOutdoorHumidity = String(currentOutdoorHumidity, 1);

    recordedBasementTemperature.trim();
    recordedBasementHumidity.trim();
    recordedOutdoorTemperature.trim();
    recordedOutdoorHumidity.trim();

    temperatureLoopCount = 0;
    setWifiSleepMode(false);
    http.begin(temperatureUrl + recordedBasementTemperature + "," + recordedBasementHumidity + "," +
recordedUpstairsTemperature + "," + recordedUpstairsHumidity + "," + recordedOutdoorTemperature + "," +
recordedOutdoorHumidity);
    int httpCode = http.GET();
    if (httpCode > 0) {
      String payload = http.getString();
      Serial.println("HTTP Response: " + payload);
      recordPingSucces();
    } else {
      recordPingFailure();
    }
    http.end();

    setWifiSleepMode(true);
  }

  temperatureLoopCount++;
```

```
//////////////////////////
// LCD DISPLAY

switch (displayMode) {
  case 1:
    temperatureText.trim();
    humidifyText.trim();

    if (prevTemperatureText != temperatureText || prevHumidifyText != humidifyText) {
      lcd_i2c.clear();
    }

    prevTemperatureText = temperatureText;
    prevHumidifyText = humidifyText;

    lcd_i2c.setCursor(0, 0);
    lcd_i2c.print(temperatureText);
    lcd_i2c.setCursor(0, 1);
    lcd_i2c.print(humidifyText);
    break;

  case 2:
    String connectivityText1 = "";
    if (wifiConnected) {
      connectivityText1 += "WF ON";
    }
    if (wifiConnected && !serverFailed) {
      connectivityText1 += " SERVER ON";
    }

    String serverPings = String(totalServerPings, 0);
    serverPings.trim();

    String serverPingFails = String(totalServerPingFails, 0);
    serverPingFails.trim();

    String connectivityText2 = serverPings + "/" + serverPingFails + " PINGS";

    connectivityText1.trim();
    connectivityText2.trim();
```

```
    if (prevConnectivityText1 != connectivityText1 || prevConnectivityText2 != connectivityText2) {
      lcd_i2c.clear();
    }

    prevConnectivityText1 = connectivityText1;
    prevConnectivityText2 = connectivityText2;

    lcd_i2c.setCursor(0, 0);
    lcd_i2c.print(connectivityText1);
    lcd_i2c.setCursor(0, 1);
    lcd_i2c.print(connectivityText2);
    break;
  }

  //delay(pulseRate * 1000);
  pulseLed(!wifiPaused);

  // analogWrite(LED_PIN, 255);
}




/**
* pulse the led in and out over a span of 2 seconds
*
*/
void pulseLed(bool active) {

  // analogWrite(LED_PIN, 255);

  int dutyCycle = 166;
  int delayMs = pulseRate * 1000 / dutyCycle;

  // Fade in the LED
  for (int brightness = 0; brightness <= dutyCycle; brightness++) {
    if (active) {
      analogWrite(LED_PIN, brightness);
    }
```

```
    delay(delayMs);  // Adjust this value to change the fade duration
  }

  // Fade out the LED
  for (int brightness = dutyCycle; brightness >= 0; brightness--) {
    if (active) {
      analogWrite(LED_PIN, brightness);
    }
    delay(delayMs);  // Adjust this value to change the fade duration
  }

  // delay(1000);
}

bool connectToWiFi() {

  if (connectingToWifi || wifiConnected) {
    return true;
  }

  connectingToWifi = true;

  String activeSsid = "";
  String activePassword = "";

  if (activeConnection == 1) {
    activeSsid = ssid1;
    activePassword = password1;
  } else if (activeConnection == 2) {
    activeSsid = ssid2;
    activePassword = password2;
  }

  Serial.print("Connecting to WiFi: ");
  Serial.println(activeSsid);

  WiFi.begin(activeSsid, activePassword);

  while (WiFi.status() != WL_CONNECTED && wifiConnectionAttempts < 20) {
    delay(500);
```

```
    Serial.print(".");
    wifiConnectionAttempts++;
  }

  wifiConnectionAttempts = 0;

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nConnected to WiFi");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP());

    EEPROM.write(eepromActiveConnectionAddress, activeConnection);
    EEPROM.commit();

    Serial.print("set activeConnection to : ");
    Serial.println(activeConnection);

    // 5 flashes means intenet is connected
    for (int i = 0; i < 5; i++) {
      digitalWrite(LED_PIN, HIGH);
      delay(10);
      digitalWrite(LED_PIN, LOW);
      delay(200);
    }

    wifiConnected = true;
    connectingToWifi = false;
    return true;

  } else {
    Serial.print("Connection to ");
    Serial.print(activeSsid);
    Serial.println(" failed. Trying alternative");

    wifiConnected = false;
    connectingToWifi = false;

    return false;
  }
}
```

```
/**
 * set wifi sleep mode between data relays to conserve energy
 * @param sleepMode - if true set wifi card to sleep to conserve energy
 */
void setWifiSleepMode(bool sleepMode) {

  wifiSleeping = sleepMode;

  if (sleepMode) {
    WiFi.disconnect();
    WiFi.setSleep(true);
    wifiConnected = false;
    delay(1000);
    Serial.print("sleep wifi status: ");
    Serial.println(wl_status_to_string(WiFi.status()));
  } else {
    WiFi.setSleep(false);
    WiFi.reconnect();
    delay(2000);
    Serial.print("awaken wifi status: ");
    Serial.println(wl_status_to_string(WiFi.status()));
    // Check if the connection is still active. if not trigger wait for it to come back online
    if (WiFi.status() != WL_CONNECTED && !wifiPaused) {
      Serial.println("Connection lost. Attempting to reconnect in 1 minute ...");
      WiFi.disconnect();
      wifiPaused = true;
      wifiConnected = false;
      connectToWiFi();
    }
  }
}

/**
 * record server ping success in long term memory
 */
void recordPingSucces() {
  totalServerPings++;
  EEPROM.begin(EEPROM_SIZE);
```

```
  EEPROM.writeFloat(eepromPingsAddress, totalServerPings);
  EEPROM.commit();
  EEPROM.end();
  wifiConnected = true;
  serverFailed = false;
}

/**
 * record server ping fails in long term memory
 */
void recordPingFailure() {
  totalServerPingFails++;
  EEPROM.begin(EEPROM_SIZE);
  EEPROM.writeFloat(eepromFailedPingsAddress, totalServerPingFails);
  EEPROM.commit();
  EEPROM.end();
  wifiConnected = false;
  serverFailed = true;
}

/**
 * ESP32 wifi card statuses
 * @param status
 * @return string
 */
String wl_status_to_string(wl_status_t status) {

  String response = "";

  switch (status) {
    case WL_NO_SHIELD:
      response = "WL_NO_SHIELD";
      break;
    case WL_IDLE_STATUS:
      response = "WL_IDLE_STATUS";
      break;
    case WL_NO_SSID_AVAIL:
      response = "WL_NO_SSID_AVAIL";
      break;
    case WL_SCAN_COMPLETED:
```

```
      response = "WL_SCAN_COMPLETED";
      break;
    case WL_CONNECTED:
      response = "WL_CONNECTED";
      break;
    case WL_CONNECT_FAILED:
      response = "WL_CONNECT_FAILED";
      break;
    case WL_CONNECTION_LOST:
      response = "WL_CONNECTION_LOST";
      break;
    case WL_DISCONNECTED:
      response = "WL_DISCONNECTED";
      break;
  }

  return response;
}



void getSensorReadings() {


  float newTempReading;
  float newHumidityReading;


  // BASEMENT

  dhtBasement.temperature().getEvent(&basementSensorEvent);
  if (isnan(basementSensorEvent.temperature)) {
    Serial.println(F("Error reading basement temperature!"));
  } else {
    newTempReading = basementSensorEvent.temperature;
    if (newTempReading > -10 && newTempReading < 50) {
      currentBasementTemperature = basementSensorEvent.temperature;
    }
    Serial.print("basement temp: ");
    Serial.println(currentBasementTemperature);
```

```
}


dhtBasement.humidity().getEvent(&basementSensorEvent);
if (isnan(basementSensorEvent.relative_humidity)) {
  Serial.println(F("Error reading basement humidity!"));
} else {

  newHumidityReading = basementSensorEvent.relative_humidity;
  if (newHumidityReading > -10 && newHumidityReading < 100) {
    currentBasementHumidity = basementSensorEvent.relative_humidity;
  }

  Serial.print("basement humidity: ");
  Serial.println(currentBasementHumidity);
}


// UPSTAIRS
dhtUpstairs.temperature().getEvent(&upstairsSensorEvent);
if (isnan(upstairsSensorEvent.temperature)) {
  Serial.println(F("Error reading upstairs temperature!"));
} else {
  newTempReading = upstairsSensorEvent.temperature;
  if (newTempReading > -10 && newTempReading < 50) {
    currentUpstairsTemperature = upstairsSensorEvent.temperature;
  }
  Serial.print("upstairs temp: ");
  Serial.println(currentUpstairsTemperature);
}


dhtUpstairs.humidity().getEvent(&upstairsSensorEvent);
if (isnan(upstairsSensorEvent.relative_humidity)) {
  Serial.println(F("Error reading upstairs humidity!"));
} else {

  newHumidityReading = upstairsSensorEvent.relative_humidity;
  if (newHumidityReading > -10 && newHumidityReading < 100) {
    currentUpstairsHumidity = upstairsSensorEvent.relative_humidity;
```

```
    }

    Serial.print("upstairs humidity: ");
    Serial.println(currentUpstairsHumidity);
  }



  // OUTDOOR

  dhtOutdoor.temperature().getEvent(&outdoorSensorEvent);
  if (isnan(outdoorSensorEvent.temperature)) {
    Serial.println(F("Error reading outdoor temperature!"));
  } else {
    newTempReading = outdoorSensorEvent.temperature;
    if (newTempReading > -10 && newTempReading < 50) {
      currentOutdoorTemperature = outdoorSensorEvent.temperature;
    }
    Serial.print("outdoor temp: ");
    Serial.println(currentOutdoorTemperature);
  }



  dhtOutdoor.humidity().getEvent(&outdoorSensorEvent);
  if (isnan(outdoorSensorEvent.relative_humidity)) {
    Serial.println(F("Error reading outdoor humidity!"));
  } else {

    newHumidityReading = outdoorSensorEvent.relative_humidity;
    if (newHumidityReading > -10 && newHumidityReading < 100) {
      currentOutdoorHumidity = outdoorSensorEvent.relative_humidity;
    }

    Serial.print("outdoor humidity: ");
    Serial.println(currentOutdoorHumidity);
  }
}
```

Revision #19
Created 19 February 2024 14:52:26 by peter drinnan

Updated 27 June 2024 03:43:56 by peterd