ECE5436 Advanced Microprocessor Systems Final Project

Atmospheric Condition Monitor

Documentation

December 2, 2019

Created by: Mubashar Khan, Lanny Dao, Samantha Chan

Original Project Proposal:

A web-based data acquisition system. For example, collecting temperature and humidity data then pass the data to the web for displaying, but modified to include an NTP clock and maybe an RTC clock. The goal of this project is to gather environmental data using sensors that transmit data to a website for displaying. Sensors that will be used for data collection will measure temperature and humidity periodically using a real-time clock. That data will then be transmitted to a website which will update and display that data gathered along with the current system time according to the network time protocol. Milestones for this project are divided based on hardware and software components. We expect the website design to take only a small portion of the total design time and most of the time will be spent ensuring that the data is collected and accurately transmitted to the website. The hardware design portion which includes writing code associated with the sensors will be divided into 3 parts. The first part is to implement the temperature sensors. We already have a Maxim 18b20 digital temperature sensor. The second part is to implement the humidity sensor. We will source a humidity sensor before October 15. The third part of the hardware is the Ds3231 real time clock. These hardware components will be run using FreeRTOS. Thread scheduling for data collection will follow a periodic round robin system.

Changes made to original project plan:

* Rather than using the Maxim 18b20 digital temperature sensor, we used the DHT22 temperature and humidity sensor. The reason for this is that while searching for an ESP32 compatible humidity sensor, we found that the DHT line of sensors was the most readily available.

Parts needed for this project:

* ESP32 ESP-32S Development Board 2.4GHz
* DS3231 RTC
* DHT22 Temperature and Humidity Sensor
* 1 x 10 [µF] Capacitor
* 1 x 10 [KΩ] Resistor

Resources:

FreeRTOS documentation:

<https://www.freertos.org/Documentation/RTOS_book.html>

<http://web.ist.utl.pt/~ist11993/FRTOS-API/index.html>

Creating a Web Server:

<https://www.electronics-lab.com/project/esp32-webserver-tutorial/>

<https://randomnerdtutorials.com/esp32-web-server-arduino-ide/>

Using the DS3231:

<https://datasheets.maximintegrated.com/en/ds/DS3231.pdf>

<http://www.esp32learning.com/code/esp32-and-ds3231-rtc-example.php>

Using the DHT22:

<https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf>

<https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/>

Pseudo Code for the Atmospheric Condition Monitor

Import all required libraries

Define username and password to be able to access the internet network

Initialize hardware components

Set up time delay using Freertos

readDHTTemperature()

get temperature from dht22

if reading temperature failed, return a blank

otherwise return the temperature as a string.

readDHTHumidty()

get humidty from dht22

if reading humidity failed, return a blank

otherwise return the humidity as a string.

HTML Javascript Webpage setup:

Head:

Set up styles, fonts, and paragraph styles for the webpage

Body:

Website Title: Atmospheric Condition Monitor

Paragraph 1: Temperature Display

Import thermometer graphic

Create space for temperature to be displayed as text

Paragraph 2: Humidity Display

Import water droplet graphic

Create space for humidity to be displayed as text

Paragraph 3: Time of last update

Create a new variable for Date()

Set up the text for the respective month.

{

If dt.getmonth()+1 equals 1, then its January.

If dt.getmonth()+1 equals 2, then its February

…

}

If condition to determine whether is AM or PM

If hours is greater than 12

Hours = hours -12

Day/night indicator is PM

Otherwise, day/night indicator is AM

setTimeout:

allows for webpage to refresh so that time can update when data is updated

Foot:

Group member names and class number/semester

setInterval function

if the webpage is ready to be updated, then send a request to get temperature/humidity

Processor:

Replaces placeholder values in html document with actual values.

Setup:

Start the serial port at a baud rate of 115200

Start the dht

Connect to the wifi. Print out the local ip address.

Check if clock is on. If the clock battery lost power, set the time to the current time.

Setup task 1 and task 2. Start task scheduler.

Server setup

If the server gets a request, determine if the request was for temperature or humidity data. Send the respective data value to the webpage.

Start the server

Full Code:

// Import required libraries

#include <stdio.h>

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "WiFi.h"

#include "ESPAsyncWebServer.h"

#include <Adafruit\_Sensor.h>

#include <DHT.h>

#include "RTClib.h"

// Replace with your network credentials

const char\* ssid = "House Wifi";

const char\* password = "\*\*\*\*\*\*\*\*\*";

#define TimeSetDelaySeconds 10 //specify how long of a delay you would like. Keep above 2\*\*

#define STACK\_SIZE 10000

DHT dht(27, DHT22); //

RTC\_DS3231 rtc; //Create DS3231 object

AsyncWebServer server(80);// Instantiate an AsyncWebServer object with port 80

String DHTTemperature; // The RTOS will ensure these readings are kept up to date.

String DHTHumidity; // The rate at which the server updates itself is its own business.

String readDHTTemperature(void);

String readDHTHumidity(void);

void task1(void \*pvParameter)

{

const portTickType xDelay = 1000 / portTICK\_RATE\_MS;

while(1)//for(int i = TimeSetDelaySeconds; i > 0; i--)

{

vTaskDelay( xDelay );

//Serial.println("LOOP");

DHTTemperature = readDHTTemperature();

}

// Serial.println("exit loop");

}

void task2(void \*pvParameter)

{

const portTickType xDelay = 1000 / portTICK\_RATE\_MS;

while(1)//for(int i = TimeSetDelaySeconds; i > 0; i--)

{

vTaskDelay( xDelay );

//Serial.println("LOOP");

DHTHumidity = readDHTHumidity();

}

// Serial.println("exit loop");

}

//Read from sensor. Sensor reading may take up to 2 seconds. //Important to not set an time for sampling of less than 2 seconds.

String readDHTTemperature() {

float t = dht.readTemperature(true); // true condition reads temp as Fahrenheit. //For Celsius leave condition blank.

// Check if any reads failed

if (isnan(t)) {

Serial.println("Failed to read from DHT sensor!");

return "--.--"; //return a blank time for display

}

else {

Serial.println(t);

return String(t);

}

}

String readDHTHumidity() {

// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

float h = dht.readHumidity();

if (isnan(h)) {

Serial.println("Failed to read from DHT sensor!");

return "--.--";

}

else {

Serial.println(h);

return String(h);

}

}

//HTML JavaScript stuff

// <!--Comments are written like this for html-->

const char index\_html[] PROGMEM = R"rawliteral(

<!DOCTYPE HTML><html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr" crossorigin="anonymous">

<style>

html {

font-family: Arial;

display: inline-block;

margin: 0px auto;

text-align: center;

}

h2 { font-size: 3.0rem; }

p { font-size: 2.5rem; }

.units { font-size: 1.2rem; }

.dht-labels{

font-size: 1.5rem;

vertical-align:middle;

padding-bottom: 20px;

}

</style>

</head>

<body>

<h2>Atmospheric Condition Monitor</h2>

<p>

<i class="fas fa-thermometer-half" style="color:Tomato;"></i>

<span class="dht-labels">Temperature</span>

<span id="temperature">%TEMPERATURE%</span>

<sup class="units">&deg;F</sup>

</p>

<p>

<i class="fas fa-tint" style="color:#00add6;"></i>

<span class="dht-labels">Humidity</span>

<span id="humidity">%HUMIDITY%</span>

<sup class="units">%</sup>

</p>

<p>Last updated: <span id="datetime"></span></p>

<script>

var dt = new Date();

if((dt.getMonth()+1) == 11)

{

var mon = "Nov.";

}

else if((dt.getMonth()+1) == 12)

{

var mon = "Dec.";

}

else if((dt.getMonth()+1) == 1)

{

var mon = "Jan.";

}

else if((dt.getMonth()+1) == 2)

{

var mon = "Feb.";

}

else if((dt.getMonth()+1) == 3)

{

var mon = "Mar.";

}

else if((dt.getMonth()+1) == 4)

{

var mon = "Apr.";

}

else if((dt.getMonth()+1) == 5)

{

var mon = "May.";

}

else if((dt.getMonth()+1) == 6)

{

var mon = "Jun.";

}

else if((dt.getMonth()+1) == 7)

{

var mon = "Jul.";

}

else if((dt.getMonth()+1) == 8)

{

var mon = "Aug.";

}

else if((dt.getMonth()+1) == 9)

{

var mon = "Sep.";

}

else if((dt.getMonth()+1) == 10)

{

var mon = "Oct.";

}

var ampm = "AM"

var h = dt.getHours()

if(h > 12)

{

h = (h - 12);

ampm = "PM"

}

if (h == 0)

{

h = 12;

}

//<!- - Network Time - ->

document.getElementById("datetime").innerHTML = mon +" "+ (("0"+dt.getDate()).slice(-2)) +" at "+ h +":"+ (("0"+dt.getMinutes()).slice(-2)) + " " + ampm;

setTimeout;

</script>

<!-- Footer -->

<div>

<footer class="w3-center w3-light-grey w3-padding-8">

<br><br><br><br><br>

<p style="margin:0;"><font size="2">Created by: Mubashar Khan, Samantha Chan, Lanny Dao<br>ECE 5436 Fall 2019</font></p>

<p style="margin:2;"><font size="2"></font></p>

</footer>

</div>

</body>

<script>

setInterval(function ( ) {

var xhttp = new XMLHttpRequest();//Allows part of the webpage to be updated without changing the whole page.

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) { //When readyState is 4 and status is 200, the response is ready: The webpage can be updated.

document.getElementById("temperature").innerHTML = this.responseText;

}

};

xhttp.open("GET", "/temperature", true);

xhttp.send();

}, 10000 ) ;

setTimeout(function() {

location.reload();

}, 10000);

setInterval(function ( ) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("humidity").innerHTML = this.responseText;

}

};

xhttp.open("GET", "/humidity", true);

xhttp.send();

}, 10000 ) ;

</script>

</html>)rawliteral";

// Replaces placeholder with DHT values

String processor(const String& var){

//Serial.println(var);

if(var == "TEMPERATURE"){

return DHTTemperature; // readDHTTemperature();

}

else if(var == "HUMIDITY"){

return DHTHumidity; // readDHTHumidity();

}

return String();

}

void setup(){

// Serial port for debugging purposes

Serial.begin(115200);

dht.begin();

// Connect to Wi-Fi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connecting to WiFi..");

delay(3000); //Time for the clock to set itself up; if it is not set up...

if (! rtc.begin())

{

Serial.println("Couldn't find RTC");

while (1);

}

if (rtc.lostPower())

{

Serial.println("RTC lost power, lets set the time!");

// following line sets the RTC to the date &amp; time this sketch was compiled

rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

// This line sets the RTC with an explicit date &amp; time, for example to set

// January 21, 2014 at 3am you would call:

// rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0)); //Year, month, Day, hour, minute, second

}

}

// Create both tasks

xTaskCreate(

task1, /\* Task function. \*/

"TaskOne", /\* String with name of task. \*/

10000, /\* Stack size in bytes. \*/

NULL, /\* Parameter passed as input of the task \*/

1, /\* Priority of the task. \*/

NULL); /\* Task handle. \*/

xTaskCreate(

task2, /\* Task function. \*/

"TaskTwo", /\* String with name of task. \*/

10000, /\* Stack size in bytes. \*/

NULL, /\* Parameter passed as input of the task \*/

1, /\* Priority of the task. \*/

NULL); /\* Task handle. \*/

// Print ESP32 Local IP Address

Serial.println(WiFi.localIP());

// Route for root / web page

server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/html", index\_html, processor);

});

server.on("/temperature", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", DHTTemperature.c\_str()/\*readDHTTemperature().c\_str()\*/);

});

server.on("/humidity", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", DHTHumidity.c\_str()/\*readDHTHumidity().c\_str()\*/);

});

// Start server

server.begin();

vTaskStartScheduler();

}

void loop(){

DateTime now = rtc.now();

}