

leeassam / arduino-bootcamp Public[Code](#)[Issues](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)[Insights](#)

master ▾

...

[arduino-bootcamp](#) / [Weather\\_Station](#) / [Final](#) / [Online\\_Weather\\_Station](#) /  
[Online\\_Weather\\_Station.ino](#)

leeassam Updated Online Weather Station Code



1 contributor

265 lines (223 sloc) | 8.94 KB

...

```
1  /*
2   Retrieve Weather From Weather Underground
3
4   This sketch connects to the Weather Underground API to make a request
5   using an Arduino Wifi shield. The result is displayed on an LCD display.
6
7   created 10/30/2016
8   modified 10/30/2016
9   by: Lee Assam
10
11  */
12
13  #include <WiFi.h>
14  #include <LiquidCrystal.h>
15
16
17  char ssid[] = "Your network name";    // your network SSID (name)
18  char pass[] = "Your network password"; // your network password
19  int keyIndex = 0;                     // your network key Index number (needed only for WEP)
20
21  int status = WL_IDLE_STATUS;
22
23  // Initialize the Wifi client library
24  WiFiClient client;
25
26  // server address:
27  char server[] = "api.wunderground.com";
28
29  unsigned long lastConnectionTime = 0; // last time you connected to the server, in milliseconds
30  const unsigned long postingInterval = 30L * 1000L; // delay between updates, in milliseconds
31
```

```
32
33 String responseString = "";           // Response string from server for .JSON page
34 int APItemp;                         // Integer temperature from .JSON page
35 boolean tempFound = false;           // Indicator for temperature found
36 boolean tempStringFound = false;     // Indicator for weather temperature found
37 boolean weatherFound = false;        // Indicator for weather found
38 boolean humidityFound = false;       // Indicator for humidity found
39
40 char weather[30];
41 char weatherString[30];
42 char humidity[4];
43
44 //API information
45 String apiKey = "Your api key";
46 //US
47 String state = "IL";
48 String city = "Bloomington";
49 String weatherLocation = String(state + "/" + city);
50
51 //International
52 //String country = "Name of your Country";
53 //String city = "City in your Country";
54 //String weatherLocation = String(country + "/" + city);
55
56 // initialize the library with the numbers of the interface pins
57 LiquidCrystal lcd(2, 3, 4, 5, 8, 9);
58
59 void setup() {
60     //Initialize serial and wait for port to open:
61     Serial.begin(9600);
62
63     // set up the LCD's number of columns and rows:
64     lcd.begin(16, 2);
65     lcd.clear();
66     lcd.print("Retrieving");
67     lcd.setCursor(0, 1);
68     lcd.print("Weather...");
69
70     while (!Serial) {
71         ; // wait for serial port to connect. Needed for native USB port only
72     }
73
74     // check for the presence of the shield:
75     if (WiFi.status() == WL_NO_SHIELD) {
76         Serial.println("WiFi shield not present");
77         // don't continue:
78         while (true);
79     }
80
81     String fv = WiFi.firmwareVersion();
82     if (fv != "1.1.0") {
83         Serial.println("Please upgrade the firmware");
```

```
84     }
85
86     // attempt to connect to Wifi network:
87     while (status != WL_CONNECTED) {
88         Serial.print("Attempting to connect to SSID: ");
89         Serial.println(ssid);
90         // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
91         //status = WiFi.begin(ssid, pass);
92         status = WiFi.begin(ssid, pass);
93         // wait 10 seconds for connection:
94         delay(10000);
95     }
96     // you're connected now, so print out the status:
97     printWifiStatus();
98 }
99
100 void loop() {
101     // if there's incoming data from the net connection.
102     // send it out the serial port. This is for debugging
103     // purposes only:
104
105     boolean startCapture = false;           // Boolean for indicating JSON capture
106     String response = "";                   // Initialize the response string
107     tempFound = false;                     // Indicator for temperature found
108     tempStringFound = false;
109     weatherFound = false;                   // Indicator for weather found
110     humidityFound = false;
111
112     while (client.available())
113     {
114         char inChar = client.read();
115         if (inChar == '"')
116             startCapture = true;
117         else if (startCapture)
118         {
119             responseString += inChar;
120             if (inChar == ',')
121             {
122                 startCapture = false;
123                 checkData();
124             }
125         }
126     }
127
128     // if ten seconds have passed since your last connection,
129     // then connect again and send data:
130     if (millis() - lastConnectionTime > postingInterval) {
131         httpRequest();
132     }
133
134     displayWeather();
135 }
```

```
136 }
137
138 // this method makes a HTTP connection to the server:
139 void httpRequest() {
140     // close any connection before send a new request.
141     // This will free the socket on the WiFi shield
142     client.stop();
143
144     // if there's a successful connection:
145     if (client.connect(server, 80)) {
146         Serial.println("connecting...");
147         // send the HTTP PUT request:
148         String urlRequest = String("GET /api/" + apiKey + "/conditions/q/" + weatherLocation + ".json");
149         client.println(urlRequest);
150         client.println("Host: api.wunderground.com");
151         client.println("User-Agent: ArduinoWiFi/1.1");
152         client.println("Connection: close");
153         client.println();
154
155         // note the time that the connection was made:
156         lastConnectionTime = millis();
157     } else {
158         // if you couldn't make a connection:
159         Serial.println("connection failed");
160     }
161 }
162
163
164 void printWifiStatus() {
165     // print the SSID of the network you're attached to:
166     Serial.print("SSID: ");
167     Serial.println(WiFi.SSID());
168
169     // print your WiFi shield's IP address:
170     IPAddress ip = WiFi.localIP();
171     Serial.print("IP Address: ");
172     Serial.println(ip);
173
174     // print the received signal strength:
175     long rssi = WiFi.RSSI();
176     Serial.print("signal strength (RSSI):");
177     Serial.print(rssi);
178     Serial.println(" dBm");
179 }
180
181
182
183 /*****
184 void checkData()
185 *****/
186
187 Purpose: Upates the response values from a string containing the API response .JSON file
188 Entry: int APItemp and char weather contain outdated values
```

```

188     Exit: int APItemp - updated
189     char[] weather - updated
190     *****
191 void checkData()
192 {
193     int i; // Variable for string length
194     if (responseString.startsWith(F("temp_f")) & !tempFound) // If this is the temp_f line
195     {
196         char temp2[3]; // Allocate temporary temperature array
197         tempFound = true; // We have found temperature
198         for (i = 7; i < responseString.length() - 1; ++i) // For the length of the response string
199         {
200             if (responseString[i] == '.') // If we found a '.' break
201                 break;
202             temp2[i - 7] = responseString[i]; // Otherwise add the character to the array
203         }
204         APItemp = atoi(temp2); // Convert char to int and update APItemp
205         Serial.print("Temperature: ");
206         Serial.println(APItemp);
207     }
208     else if (responseString.startsWith(F("weather")) & !weatherFound) // If this is the weather line
209     {
210         memset(weather, 0, sizeof(weather));
211         weatherFound = true; // We have found weather, set weatherFound to true
212         for (i = 8; i < responseString.length() - 1; ++i) // For the length of the response string
213             weather[i - 8] = responseString[i]; // Append the character on to the weather array
214         Serial.print("Weather: ");
215         Serial.println(weather);
216     }
217     else if (responseString.startsWith(F("relative_humidity")) & !humidityFound) // If this is the humidity line
218     {
219         memset(humidity, 0, sizeof(humidity));
220         humidityFound = true; // We have found weather, set humidityFound to true
221         for (i = 18; i < responseString.length() - 1; ++i) // For the length of the response string
222             humidity[i - 18] = responseString[i]; // Append the character on to the humidity array
223         Serial.print("Humidity: ");
224         Serial.println(humidity);
225     }
226     else if (responseString.startsWith(F("temperature_string")) & !tempStringFound) // If this is the temperature_string line
227     {
228         memset(weatherString, 0, sizeof(weatherString));
229         tempStringFound = true; // We have found weather, set tempStringFound to true
230         for (i = 19; i < responseString.length() - 1; ++i) // For the length of the response string
231             weatherString[i - 19] = responseString[i]; // Append the character on to the weatherString array
232         Serial.print("Weather String: ");
233         Serial.println(weatherString);
234     }
235     responseString = ""; // Clear the response string
236 }
237
238 void displayWeather() {
239     if (strlen(weatherString) != 0) {

```

```
240 //Display weather description
241 lcd.clear();
242 lcd.setCursor(0, 0);
243 lcd.print("Weather: ");
244 lcd.setCursor(0, 1);
245 lcd.print(weather);
246 delay(3000);
247 //Display temperature
248 lcd.clear();
249 lcd.setCursor(0, 0);
250 lcd.print("Temperature: ");
251 lcd.setCursor(0, 1);
252 lcd.print(weatherString);
253 delay(3000);
254 //Display humidity
255 lcd.clear();
256 lcd.setCursor(0, 0);
257 lcd.print("Humidity: ");
258 lcd.setCursor(0, 1);
259 lcd.print(humidity);
260 delay(3000);
261 }
262 }
263
264
265
```