# Final Software Implementation

I will provide the code here and in a digital format when submitting the deliverables.

## CI301.cpp (Main board – arduino mega)

#include <EasyScheduler.h>

#include <dht.h>

#include <SPI.h>

#include <Ethernet.h>

#include "sha1.h"

#include "mysql.h"

//Pin references

int pin\_dht[4] = {54, 55, 56, 57}; //{A0,A1,A2,A3}

int temps[4] = {0, 0, 0, 0};

boolean dimmable[] = {1, 1, 1, 1};

int desired\_temp[] = {20, 20, 20, 20};

int desired\_tolerance[] = {4, 4, 4, 4} ;

boolean relayHeat[] = {0, 0, 0, 0};

//int minimum\_dim[] = {0, 0, 0, 0}; //0 - 255

//Sensor nicknames

char\* dhtNN[4] = {

"Zone1 - A", // DHT0

"Zone1 - B", // DHT1

"Zone2 - C", // DHT2

"Zone2 - D" // DHT3

};

dht DHT;

Schedular setWeb;

Schedular setMysql;

Schedular setSend;

// Enter a MAC address and IP address for your controller below.

// The IP address will be dependent on your local network:

byte mac[] = {

0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED

};

IPAddress addr\_me(192, 168, 0, 66);

IPAddress addr\_mysql(198, 23, 57, 27);

char user[] = "iansmi9\_ard";

char password[] = "arduino";

char SQL\_POWERON\_TEST[] = "INSERT INTO `iansmi9\_ard`.`poweron` VALUES (NULL, CURRENT\_TIMESTAMP, 'ok')";

/\* Setup the MySQL connector \*/

Connector my\_conn; // The Connector reference

// Initialize the Ethernet server library

// with the IP address and port you want to use

// (port 80 is default for HTTP):

EthernetServer server(80);

void setup() {

// Open serial communications and wait for port to open:

Serial.begin(115200);

while (!Serial) {

; // wait for serial port to connect. Needed for Leonardo only

}

Serial.println();

Serial.println("Hello, I'm about to start! One minute :)");

Serial.println();

if (Ethernet.begin(mac) == 0) {

Serial.println("Failed to configure Ethernet using DHCP");

Serial.println("Retrying!!");

setup();

}

Ethernet.begin(mac, addr\_me);

server.begin();

// print your local IP address:

Serial.print("My IP address is ");

for (byte thisByte = 0; thisByte < 4; thisByte++) {

// print the value of each byte of the IP address:

Serial.print(Ethernet.localIP()[thisByte], DEC);

Serial.print(".");

}

Serial.println();

//Test mysql connection by adding a poweron record

if (my\_conn.mysql\_connect(addr\_mysql, 3306, user, password))

{

//delay(15);

// run the SQL query

my\_conn.cmd\_query(SQL\_POWERON\_TEST);

Serial.println("Power up query success!");

}

else {

Serial.println("Connection failed. Retrying...");

setup();

}

Serial.println();

for (int j = 0; j < 4; j++) {

pinMode(pin\_dht[j], INPUT);

}

setWeb.start();

setMysql.start();

setSend.start();

}

void loop() {

setWeb.check(WebServer, 1);

setSend.check(relaySwitch, 1000);

setMysql.check(sendReadings, 1800000); //half hour

}

void WebServer() {

// listen for incoming clients

EthernetClient client = server.available();

if (server.available() ) {

Serial.println("Server AVAILABLE");

} else {

Serial.print(".");

}

if (client) {

Serial.println("new client");

// an http request ends with a blank line

boolean currentLineIsBlank = true;

while (client.connected()) {

if (client.available()) {

char c = client.read();

Serial.print(c);

// if you've gotten to the end of the line (received a newline

// character) and the line is blank, the http request has ended,

// so you can send a reply

if (c == '?' && currentLineIsBlank) {

Serial.println("QUESTION MARK!??!?!?!?!");

}

else if (c == '\n' && currentLineIsBlank) {

/\*

// if (my\_conn.mysql\_connect(addr\_mysql, 3306, user, password)) {

// delay(10);

my\_conn.cmd\_query("SELECT \* FROM iansmi9\_ard.log");

my\_conn.show\_results();

// We're done with the buffers so Ok to clear them (and save precious memory).

my\_conn.free\_columns\_buffer();

my\_conn.free\_row\_buffer();

// }

// else {

// Serial.println("Connection failed.");

// }

\*/

// send a standard http response header

client.println("HTTP/1.1 200 OK");

client.println("Content-Type: text/html");

client.println("Connection: close"); // the connection will be closed after completion of the response

client.println("Refresh: 60"); // refresh the page automatically every 5 sec

client.println();

//and metadata...

client.println("<!DOCTYPE HTML>");

client.println("<html>");

client.println("<head>");

client.println("<title>Environment Live Times</title>");

client.println("<link rel=\"shortcut icon\" type=\"image/x-icon\" href=\"http://arduino.cc/en/favicon.png\" />");

client.println(" <script type=\"text/javascript\" src=\"https://www.google.com/jsapi?autoload={'modules':[{ 'name':'visualization', 'version':'1', 'packages':['corechart'] }]}\"></script>");

client.println(" <script type=\"text/javascript\">");

client.println(" google.setOnLoadCallback(drawChart);");

client.println(" ");

client.println(" function drawChart() {");

client.println(" var data = google.visualization.arrayToDataTable([");

client.println(" ['Year', 'DHT1', 'DHT2'],");

client.println(" ['2004', 1000, 400],");

client.println(" ['2005', 1170, 460],");

client.println(" ['2006', 660, 1120],");

client.println(" ['2007', 1030, 540]");

client.println(" ]);");

client.println(" ");

client.println(" var options = {");

client.println(" title: 'Company Performance',");

client.println(" curveType: 'function',");

client.println(" legend: { position: 'bottom' }");

client.println(" };");

client.println(" ");

client.println(" var chart = new google.visualization.LineChart(document.getElementById('curve\_chart'));");

client.println(" ");

client.println(" chart.draw(data, options);");

client.println(" }");

client.println(" </script>");

//and CSS style data...

client.println("<style type=\"text/css\">");

client.println(" container {");

client.println(" position: relative;");

client.println(" width: 100%;");

client.println(" float: left;");

client.println(" margin-bottom: 25px;");

client.println(" margin-left:auto;");

client.println(" margin-right:auto;");

client.println(" }");

client.println(" reading {");

client.println(" float: left;");

client.println(" width: 100px;");

client.println(" font-family: \"MS Serif\", \"New York\", serif;");

client.println(" font-size: 12px;");

client.println(" text-align: center;");

client.println(" padding: 20px, 20px. 20px, 20px;");

client.println(" }");

client.println(" reading H1 {");

client.println(" font-family: Georgia, \"Times New Roman\", Times, serif;");

client.println(" font-size: 14px;");

client.println(" font-style: oblique;");

client.println(" }");

client.println(" relay {");

client.println(" position: relative;");

client.println(" float: left;");

client.println(" width: 100px;");

client.println(" font-family: \"MS Serif\", \"New York\", serif;");

client.println(" font-size: 12px;");

client.println(" text-align: center;");

client.println(" padding: 20px, 20px. 20px, 20px;");

client.println(" }");

client.println(" relay H1 {");

client.println(" font-family: Georgia, \"Times New Roman\", Times, serif;");

client.println(" font-size: 14px;");

client.println(" font-style: oblique;");

client.println(" }");

client.println("");

client.println("</style>");

client.println("</head>");

//and body content.

client.println("<body>");

client.println(" <container>");

client.println(" <container>");

// output the value of each sensor

for (int i = 0; i < 4; i++) {

DHT.read11(pin\_dht[i]);

client.print(" <reading><h1>");

client.print(dhtNN[i]);

client.print("</h1><br />Humidity<br /> ");

client.print(DHT.humidity);

client.print("%.<br /><br />Temperature<br /> ");

client.print(DHT.temperature);

client.println("C</reading>");

};

client.println(" </container> <br /> ");

client.println("<form name = \"form\" method = \"post\" action = \"?test:test1;key:value;\">");

client.println(" <container>");

for (int i = 0; i < 4; i++) {

client.print("<relay><H1>Relay ");

client.print(i);

client.print(" :");

client.println(" </H1> <p>");

client.println(" Temp Desired <br />");

client.print(" <input type=\"text\" name=\"relay");

client.print(i);

client.print("\_desired\" id=\"relay");

client.print(i);

client.println("\_desired\" style=\"width: 90px; \" >");

client.println(" <br />");

client.println(" <br />");

client.println(" Temp Tolerance <br />");

client.print(" <input type=\"text\" name=\"relay");

client.print(i);

client.print("\_tolerance\" id=\"relay");

client.print(i);

client.println("\_desired\" style=\"width: 90px; \" >");

client.println(" <br />");

client.println(" <br />");

client.println(" DHT Monitored");

client.print(" <select name=\"relay");

client.print(i);

client.print("\_dht\" id=\"relay");

client.print(i);

client.println("\_dht\" style=\"width: 90px; \" >");

client.println(" <option value=\"1\">DHT1</option>");

client.println(" <option value=\"2\">DHT2</option>");

client.println(" <option value=\"3\">DHT3</option>");

client.println(" <option value=\"4\">DHT4</option>");

client.println(" </select>");

client.println(" <br />");

client.println(" <br /> Dimmable <br /> ");

client.print(" <select name=\"relay");

client.print(i);

client.print("\_dimmable\" id=\"relay");

client.print(i);

client.println("\_dimmable\" style=\"width: 90px; \" >");

client.println(" <option value=\"0\">NO!</option>");

client.println(" <option value=\"1\">YES!</option>");

client.println(" </select>");

client.println("<br />");

client.println(" <br /> Cooling? <br /> ");

client.print(" <select name=\"relay");

client.print(i);

client.print("\_cooling\" id=\"relay");

client.print(i);

client.println("\_cooling\" style=\"width: 90px; \" >");

client.println(" <option value=\"0\">Heating</option>");

client.println(" <option value=\"1\">Cooling</option>");

client.println(" </select>");

client.println("<br />");

client.println(" </p>");

client.println("</relay>");

client.println("");

};

client.println(" </container>");

client.println(" <input name=\"btnSubmit\" type=\"submit\" style=\"width: 400px;\">");

client.println(" </form>");

client.println(" <container>");

client.println(" <!--Div that will hold the pie chart-->");

client.println(" <div id=\"curve\_chart\" style=\"width: 100%; height: 100%;\"></div>");

client.println(" </container>");

client.println(" </container>");

client.println("</body>");

client.println("</html>");

break;

}

if (c == '\n') {

// you're starting a new line

currentLineIsBlank = true;

}

else if (c != '\r') {

// you've gotten a character on the current line

currentLineIsBlank = false;

}

}

}

// give the web browser time to receive the data

delay(1);

// close the connection:

client.stop();

}

}

void sendReadings() {

int sensorvalue[8];

//fill the sensorvalue array

for (int i = 0; i < 4; i++) {

DHT.read11(pin\_dht[i]);

sensorvalue[i \* 2] = DHT.humidity;

sensorvalue[(i \* 2) + 1] = DHT.temperature;

}

//build the query, correcting any variable usage/data type issues

char SQL\_SEND\_READINGS[128];

sprintf(SQL\_SEND\_READINGS, "INSERT INTO iansmi9\_ard.log VALUES (NULL, CURRENT\_TIMESTAMP, '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d');",

sensorvalue[0], sensorvalue[1], sensorvalue[2], sensorvalue[3],

sensorvalue[4], sensorvalue[5], sensorvalue[6], sensorvalue[7]);

/\* run the SQL query \*/

my\_conn.cmd\_query(SQL\_SEND\_READINGS);

Serial.println(SQL\_SEND\_READINGS);

}

void relaySwitch() {

Serial.println("");

for (int i = 0; i < 4; i++) {

int DHTnum = i;

Serial.print("Checking DHT: ");

Serial.println(DHTnum);

DHT.read11(pin\_dht[DHTnum]);

int temp = DHT.temperature;

if ((temp > 0) && (temp < 60) && (temp != temps[DHTnum])) {

if (dimmable[DHTnum]) {

int minimum\_dim = 0;

//Tell MySQL

char SQL\_SEND\_READINGS[100];

sprintf(SQL\_SEND\_READINGS, "INSERT INTO iansmi9\_ard.log\_relays VALUES (NULL, CURRENT\_TIMESTAMP, '%d', '2');", DHTnum);

my\_conn.cmd\_query(SQL\_SEND\_READINGS);

//Serial.println();

//Find the PWM value

double pwmNumerator = (double) (temp - (desired\_temp[DHTnum] - (desired\_tolerance[DHTnum] / 2) ) );

double pwmDecimal = pwmNumerator / desired\_tolerance[DHTnum];

if (pwmDecimal < 0) {

pwmDecimal = 0.00;

};

if (pwmDecimal > 1) {

pwmDecimal = 1.00;

};

//Serial.println(pwmDecimal);

//Serial.println(minimum\_dim);

int pwmMaximum = (255 - minimum\_dim);

//Serial.println(pwmMaximum);

double dim\_component = pwmMaximum \* pwmDecimal;

//Serial.println(dim\_component);

int pwmValue = (int) minimum\_dim + dim\_component;

//Serial.println(pwmValue);

//Write this to arduino 2

char toSend[12];

sprintf(toSend, "#%d %03d$", DHTnum, pwmValue);

Serial.write(toSend);

//Tell Serial monitor

char toTell[36];

sprintf(toTell, " Sensor %d has temperature %3d! PWM: %d!", DHTnum, temp, pwmValue);

Serial.println(toTell);

temps[i] = temp;

}

else if ((!dimmable) && (temp > (desired\_temp[DHTnum] + ((int) (desired\_tolerance[DHTnum] / 2)) )) ) {

//Tell MySQL

//build the query, correcting any variable usage/data type issues

char SQL\_SEND\_READINGS[88];

sprintf(SQL\_SEND\_READINGS,

"INSERT INTO iansmi9\_ard.log\_relays VALUES (NULL, CURRENT\_TIMESTAMP, '%d', '1');", DHTnum);

/\* run the SQL query \*/

my\_conn.cmd\_query(SQL\_SEND\_READINGS);

Serial.println(SQL\_SEND\_READINGS);

//Write this to arduino 2

char toSend[7];

sprintf(toSend, "#%d 255$", DHTnum);

Serial.write(toSend);

//Tell Serial Monitor

Serial.print("Too high on sensor: ");

Serial.print(DHTnum);

Serial.println(".");

Serial.print(temp);

Serial.print(" - Turning Cooling ON - ");

Serial.println(DHTnum);

temps[i] = temp;

}

else if ((!dimmable) && (temp < (desired\_temp[DHTnum] - ((int) (desired\_tolerance[DHTnum] / 2)) )) ) {

//Tell MySQL

char SQL\_SEND\_READINGS[88];

sprintf(SQL\_SEND\_READINGS,

"INSERT INTO iansmi9\_ard.log\_relays VALUES (NULL, CURRENT\_TIMESTAMP, '%d', '0');", DHTnum);

/\* run the SQL query \*/

my\_conn.cmd\_query(SQL\_SEND\_READINGS);

Serial.println(SQL\_SEND\_READINGS);

//Write this to arduino 2

char toSend[7];

sprintf(toSend, "#%d 000$", DHTnum);

Serial.write(toSend);

//Tell Serial Monitor

Serial.print("Too low on sensor: ");

Serial.print(i);

Serial.println(".");

Serial.print(temp);

Serial.print(" - Turning Cooling OFF - ");

Serial.println(DHTnum);

temps[i] = temp;

}

else {

Serial.print(" -- OUTLYING TEMPERATURE - ");

Serial.println(temp);

}

}

}

}

## DimmingBoard.cpp – (secondary board/ arduino Uno)

//main board = 0;

//relays are at 6, 7, 8, 9;

int relayPins[4] = {6, 9, 10, 11};

void setup() {

Serial.begin(115200);

Serial.println();

//pinMode(input, INPUT);

for (int i = 0; i < 4 ; i++ ) {

pinMode(relayPins[i], OUTPUT);

}

Serial.println("STARTED");

}

void loop() {

//If there is an input

char data = Serial.read();

String dataString;

if (data == '#') {

//Serial.println("Start");

data = Serial.read();

//print until stop character found

for (int i = 0; (data != '$'); data = Serial.read()) {

if (data > -1) {

dataString += data;

//Serial.print(data);

}

}

//Serial.println(dataString);

//Serial.println("STOP");

// get the value from input

int relay = dataString.charAt(0) - '0';

char inputchars[3];

sprintf(inputchars, "%d%d%d" , (dataString.charAt(2) - '0'), (dataString.charAt(3) - '0'), (dataString.charAt(4) - '0'));

String valuestring = inputchars;

int value = valuestring.toInt();

//output what we're changing

String printme = "Dimming Relay: ";

printme += relay;

printme += " to Value: ";

printme += value;

Serial.println(printme);

if (value > 255) {

value = 255;

}

else if (value < 0) {

value = 0;

}

//Set the value of that relay.

analogWrite( relayPins[relay], value );

}

}