Implementación de Hardware libre en la medición de parámetros micro meteorológicos.

Franco Martin Muñoz Oses. - naqu_oficialok@hotmail.com
Diego Fernandez de Sa. - diegofernandezdesa@gmail.com
Tomás Navarro Febre. - tnavarrofebre@gmail.com

INTRODUCCIÓN.

La toma de datos y el censo de parámetros es fundamental para la investigación científica, permitiendo estudiar, monitorear, analizar, describir y comprender un proceso o evento en particular. Las mediciones manuales pueden llevar mucho tiempo y mano de obra intensiva, lo que resulta en la recopilación de datos a baja frecuencia, con largos intervalos de tiempo entre las mediciones. La automatización del proceso de recopilación de datos puede reducir los requisitos de mano de obra y aumentar considerablemente la frecuencia y regularidad de las mediciones. Las tecnologías que permiten aplicar estrategias de automatización son costosas y muy a menudo los equipos de monitoreo bajo patente están diseñados para operar con aquellos sensores y extensiones producidos por el propio fabricante. Por estos motivos suelen resultar prohibitivas para gran parte de la comunidad científica.

Sin embargo el surgimiento del hardware libre y de licencia abierta se ha colado en el segmento de consumidores que requieren hardware altamente personalizado y de bajos volumenes de produccion. Grupo en el que se enmarca la comunidad científica y de desarrollo técnico. Junto a estos implementos se ofrecen softwares también bajo código libre, el cual puede emplearse, estudiarse, copiarse, modificarse y redistribuirse sin restricción, o con limitaciones que aseguran que otros usuarios tienen los mismos derechos que aquellos bajo los que se obtuvieron en primer lugar.

Estas tecnologías suelen ser asequibles por la mayoría de instituciones científicas y junto a su carácter de creación comunitaria se ha producido un crecimiento exponencial de nuevas herramientas, planos y código. Estos han beneficiado ampliamente a campos tan diversos de la ciencia como son la medicina, o la meteorología, y sectores de la producción como es el control de procesos de cultivo en la agricultura. Lo que ha dado como resultado la aparición de cientos de nuevos implementos accesibles para todo aquel que lo requiera llegando a los usuarios como producto final no un artefacto, si no una idea. Esta idea puede ingresar

nuevamente en el mismo circuito que la produjo, pudiendo ser reproducida, estudiada, analizada y modificada.

Uno de los proyectos de hardware abierto más difundidos es la plataforma de desarrollo basada en microcontroladores llamada Arduino. Este hardware consiste en un microprocesador programable montado sobre una placa que proporciona acceso a los pines de entrada y salida del procesador y conectividad a una computadora que permite su programación y la interacción con el usuario. Los investigadores han comenzado a desarrollar e implementar dispositivos basados en Arduino por su gran variedad de componentes compatibles y el lenguaje de programación estandarizado.

Cabe mencionar que Arduino no es la única plataforma de hardware libre. Existen otras tales como: Open Compute Project enfocado al data center, RapRap pensada para el control de impresoras 3D, VIA OneBook para el desarrollo de equipamiento original -OEM- (Original Equipment Manufacturer). Otras como Cubieboard o Raspberry se han diseñado como pequeñas computadoras en las cuales se puede instalar un sistema operativo, generalmente GNU-Linux con el cual están estrechamente ligados por su carácter libre.

El objetivo de este artículo es presentar el desarrollo de un circuito y su código para la obtención de medidas de parámetros micro meteorológicos. Tanto el circuito y el código estarán basados en la plataforma de hardware libre Arduino.

Se ha optado por el testeo de parámetros micro meteorológicos para el desarrollo de este trabajo dado que las técnicas de la micrometeorología implican mediciones in-situ que no perturban el medioambiente. Estas técnicas permiten las mediciones continuas y del promedio en el tiempo de estas mediciones en un área delimitada, se pueden obtener conclusiones generales sobre la meteorología de la zona en estudio. La micrometeorología implica la medición de variables meteorológicas en intervalos cortos de tiempo y espacio. Generalmente en un radio no mayor a un kilómetro desde los sensores.

En este proyecto mediremos presión, temperatura, y humedad relativa. Los tres parámetros están relacionados con la comodidad del ser humano respecto al clima pudiendo obtener de la humedad y la temperatura absoluta la sensación térmica. La acumulación de estas medidas a largo plazo permiten predecir el clima de forma local y tener un entendimiento de cómo se relaciona el suelo con estratos más altos de la atmósfera.

COMPONENTES.

Plataforma de desarrollo Arduino.

La actual plataforma de desarrollo estándar de Arduino está basada en un microprocesador programable ATmega328 de 8 bits. Una placa de circuito impreso coloca el microcontrolador en un circuito de modo que los pines de entrada y salida (IO) son fácilmente accesibles. El microcontrolador contiene 32 kilobytes de memoria flash para almacenamiento de programas y ocho kilobyte de memoria ROM para almacenamiento de datos. Las líneas IO constan de 14 pines digitales y 6 pines analógicos, que proporcionan 6 canales de 10 bits. Los dispositivos funcionan a un nivel de 5 V y una velocidad del oscilador de 16 MHz o un nivel de 3.3-V y 8 MHz.

El microcontrolador contiene muchas características integradas, incluyendo temporizadores y contadores, interrupciones internas y externas, serie y otras capacidades de protocolo de comunicación, temporizador de vigilancia programable y modos de bajo consumo de energía.

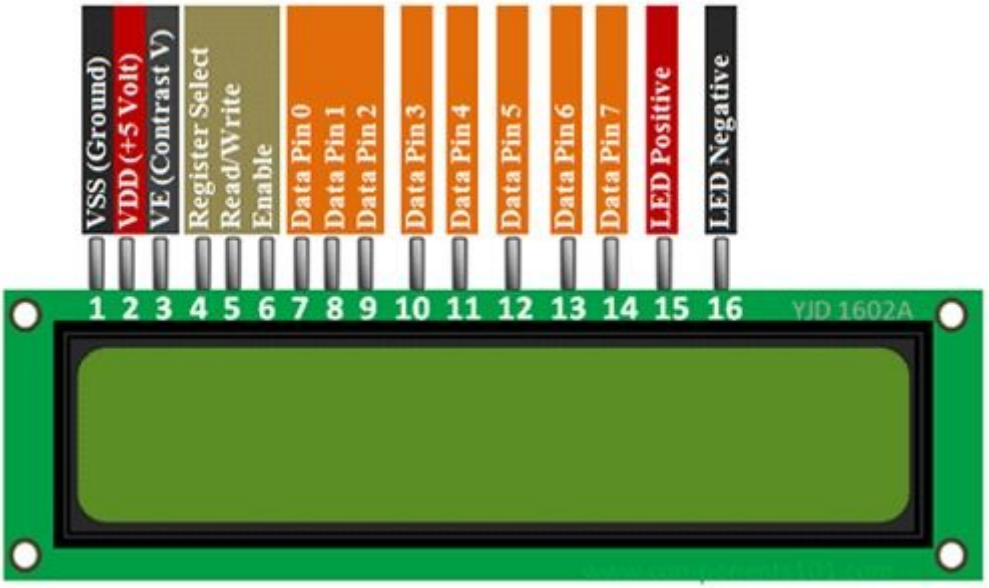
La placa Arduino está diseñada para permitir la expansión a través de la conexión de placas auxiliares. Se conectan a través de clavijas de acoplamiento dispuestas en la misma configuración física que la placa Arduino. Estas placas son controladas por el programa y el microcontrolador Arduino, por lo que no se requieren rutinas de inyección de código adicionales.

Hardware.

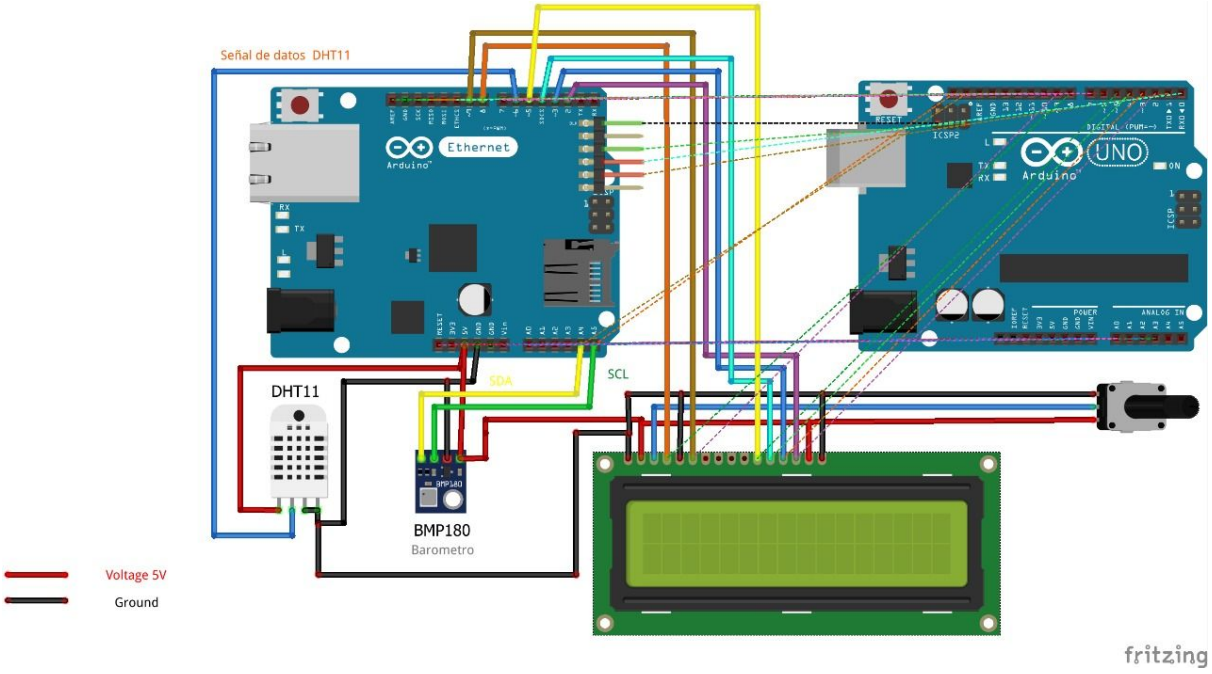
El sensor DHT11 se encuentra conectado al pin digital 6 de la placa Arduino y alimentado por la misma placa con una tensión de 5 V CC. En los puertos analogico 4 y analogicos 5 se conecta el sensor BMP 180 por el protocolo de comunicación I2C.

El ethernet shield montado sobre la arduino provee una interfaz de red tipo ethernet (IEEE 802.3x) que nos brinda comunicación con sistemas externos.

Un LCD 16x02 conectando;



pin 1 (vss) a ground (masa).
pin 2 (vdd) a alimentación 5V+.
pin 3 (ve contraste) al pin central del potenciómetro.
pin 4 (register select) del LCD al pin d8 de la placa Arduino.
pin 5 (r/w) a ground (masa).
pin 6 (enable) del LCD al 9 de la placa.
pines digitales 11, 12, 13, y 14 a los pines digitales 5 4 3 2 de la placa Arduino transfiere los datos.
pines 15 y 16 de la luz de fondo a 5V positivos y masa respectivamente.



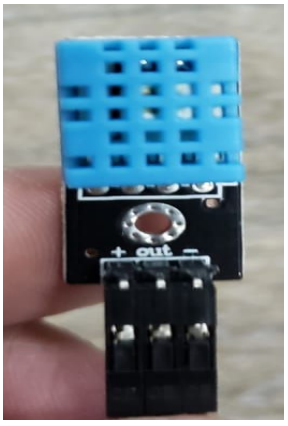
Se anexan fotografías al final del informe.

Presión y temperatura.

Para la medición de la presión y la temperatura se utilizó un sensor BMP 180 el cual utiliza una alimentación típica de 2,5 V y 5 µA registrando mínimos y máximos operativos a 25ºC de 1,62 V - 3 µA y 3,6 V - 7 µA respectivamente.

En condiciones útiles de 0ºC a 65ºC se puede obtener un rango de lecturas de presión con mínimos en los 300 hPA y máximos en 1100 hPa (error típico 0,12 hPa) con una resolución de 0,01 hPa.

Dentro del mismo rango operativo las lecturas de la temperatura tiene el mínimo en 0ºC y el máximo en 65ºC (error típico 1.0ºC) con una resolución de 0.1ºC.



Humedad.

Para la medición de la humedad se utilizó un sensor DHT11 , el cual utiliza una alimentacion tipica de 5 V , registrando límites mínimos operativos de 3V y 0.5mA , asi como tambien máximos operativos de 5,5 V y 2,5 mA.

En condiciones útiles de 0ºC a 25ºC se puede obtener un rango de lecturas de humedad con mínimos en los 20% y máximos en 90 % , con una error típico de 5% , con una resolución de 1% de HR .

Software.

Usando el IDLE de Arduino se desarrolló un script para el microcontrolador el cual permite integrar los sensores a las placas y tomar medidas de presión y temperatura desde el sensor BMP 180 y humedad desde el sensor DHT11.

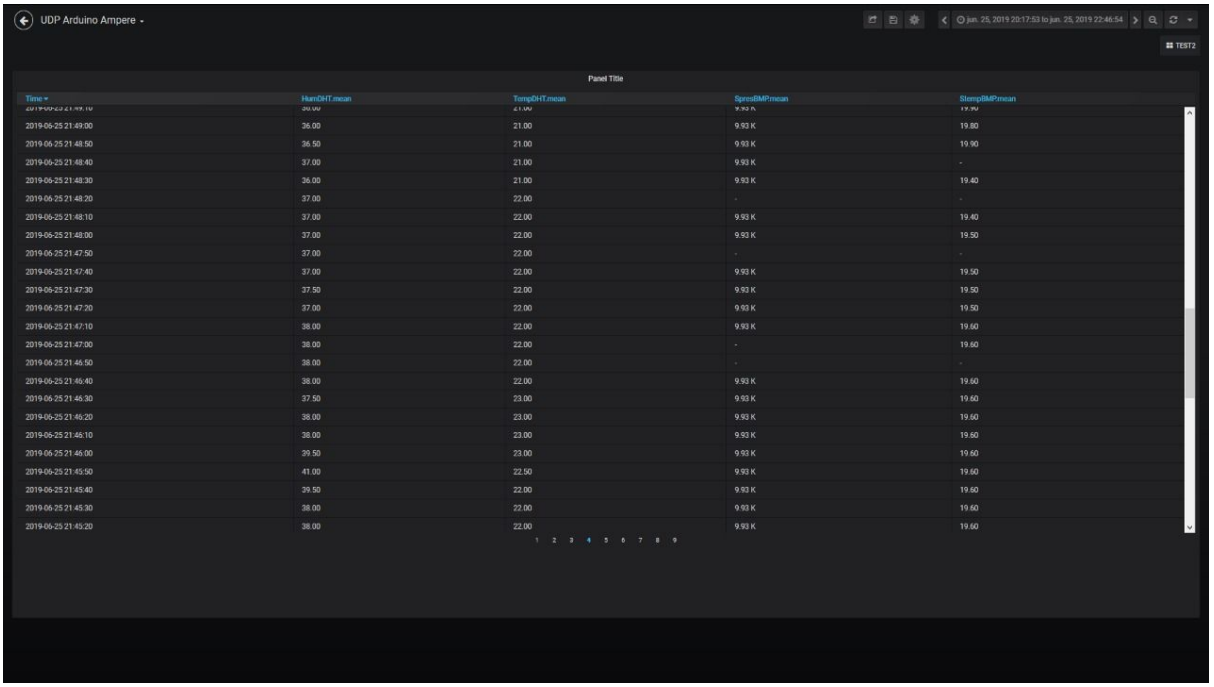
Luego el script envía los datos adecuadamente formateados a un LCD y a un web server local accesible desde la placa ethernet anexada al microprocesador. Los datos son enviados con una tasa de refresco de quince segundos. El código se encuentra anexo al final del informe.

Por otro lado con la librería ethernetudp.h a través del protocolo udp enviamos los datos recolectados a través de la red a una base de datos de secuencia de tiempo (influxDB). Posteriormente se realizó la gráfica en tiempo real utilizando la plataforma Grafana. Estos dos servicios alojados en un servidor linux independiente del sistema antes descripto.

Mediciones y conclusiones.

Se realizaron mediciones testigo durante la puesta en funcionamiento del sistema las cuales eran coincidentes con los datos obtenidos del servicio meteorológico nacional e instrumentos de medición con los que se contaba en el momento, tales son un reloj inteligente con el cual se obtenía presión y temperatura y un termómetro ambiental de alcohol.

Luego los registros se guardaron en una base de datos desde la cual se podían presentar en tiempo real en una interfaz construida con el paquete Grafana. tal como muestran las siguientes figuras.



Se puede concluir que el muestreo de variables micro meteorológicas es fácilmente resoluble con una plataforma de hardware libre el cual constituye una opción robusta que cumple con los objetivos planteados.

Sería interesante evaluar la capacidad del implemento para censar datos a la intemperie.

Anexo.

```
#include #include #include #include #define I2C_ADDRESS 0x77
#include #include #include #include #include

const int rs = 8, en =9 , d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// declaracionde del sensor DHT 11
int pinDHT11 = 6;
#define DHTPIN 6
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);


//create an BMP180 object using the I2C interface
BMP180I2C bmp180(I2C_ADDRESS);

// Parametros de RED
```

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 3, 177);
IPAddress gateway(192, 168, 3, 1);
IPAddress subnet(255, 255, 255, 0);
byte host[] = {192, 168, 3, 20}; // Donde esta la base de datos
int port = 8089;
EthernetServer server(80); // Puerto dedl server Web
EthernetClient client;
EthernetUDP udp;
//MDNS mdns(udp);

void setup() {
// Inicio de Servicios

Serial.begin(9600);

while (!Serial) {
;
}

Wire.begin(); // Inicia Comunicacion I2C pada modulo BMP180

if (!bmp180.begin())
{
Serial.println("Falla en la comunicacion del Modulo BMP180 por Interface I2C.");
while (1);
}
bmp180.resetToDefaults(); // resetea Modulo
bmp180.setSamplingMode(BMP180MI::MODE_UHR); // Habilitar modo de lectura de alta resolucio

Ethernet.begin(mac, ip,gateway,subnet); // Inicia el hard ethernet

// chequea la comunicacion con el Hardware Ethernet
if (Ethernet.hardwareStatus() == EthernetNoHardware) {
Serial.println("Ethernet shield no esta presente o falla la comunicacion hardware.");
}
if (Ethernet.linkStatus() == LinkOFF) {
Serial.println("El cable de red no esta conectado");
}

// Inicia el web server
server.begin();
Serial.print("Direcion IP del Servidorcito ");
Serial.println(Ethernet.localIP());
Serial.print("http://climafisica.local/ \n");

// regitrto ded nombre y servicio ded la APP

//mdns.begin(Ethernet.localIP(), "fisica");

// Inicializa e inicia LCD
```

```
lcd.begin(16, 2); //Inicia LCD
lcd.setCursor(0,0);
lcd.print("Sever en");
lcd.setCursor(0,1);
lcd.print(Ethernet.localIP());

delay(1000);

dht.begin();

delay(1000);
}

void loop() {

float tempDHT = dht.readTemperature();
float tempDHTf = dht.readTemperature(true);
float humDHT = dht.readHumidity();
float sensf = dht.computeHeatIndex(tempDHTf, humDHT);
float sensc = dht.computeHeatIndex(tempDHT, humDHT, false);

if (isnan(humDHT) || isnan(tempDHT) || isnan(tempDHTf)) {
Serial.println(F("Failed to read from DHT sensor!"));
lcd.clear();
lcd.print("ERROR DHT");
return;
}

if (!bmp180.measureTemperature())
{
Serial.println("Error de lectura Temperatura BMP180");
lcd.clear();
lcd.print("ERROR BMP Temp");
return;
}

do
{
delay(100);
} while (!bmp180.hasValue());

double tempBMP = bmp180.getTemperature(); //carga el dato en varible
//Serial.println(tempBMP);

if (!bmp180.measurePressure())
{
Serial.println("Error de lectura Presion BMP180");
lcd.clear();
lcd.print("ERROR BMP Pres");
return;
}
```

```
//wait for the measurement to finish. proceed as soon as hasValue() returned true.
do
{
delay(100);
} while (!bmp180.hasValue());

float presBMP = bmp180.getPressure(); //carga el dato en varible
//Serial.println(presBMP);

/// fanal de lecturas

/// se muestran los datos por web , a la base , serial y LCD

// LCD print

//delay(2000);

lcd.clear();
//lcd.setCursor(0,0);

lcd.print(tempBMP,1);
lcd.print(" C ");
lcd.print(humDHT,0);
lcd.print("%");

lcd.setCursor(0,1);
lcd.print(presBMP,0);
lcd.print("Pa ");
lcd.print(millis()/1000);

//mdns.run();

EthernetClient client = server.available();
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.write(c);

if (c == '\n' && currentLineIsBlank) {
// 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: 15"); // refresco de la pagina automatico 15 sec
client.println();
client.println("");
```

```
client.println("");

client.println("ClimaExactas");

client.println("

                                        ClimaExactas

");
client.println("
```

```
");

client.println("

Temperatura: ");

client.println(bmp180.getTemperature());

client.println(" degC");

client.println("

");

client.println("

Presion: ");

client.println(bmp180.getPressure());

client.println(" Pa");

client.println("

");

//// DHT 11

client.println("

Temperatura:

");
client.println("
```

");

client.println(tempDHT);

client.println(" degC");

client.println("

");

client.println("

Temperatura Farenheit:

");
client.println("

");

client.println(tempDHTf);

client.println(" degC");

client.println("

");

client.println("

Humedad:

");

client.println("

");

client.println(humDHT);

client.println(" %");

client.println("

");

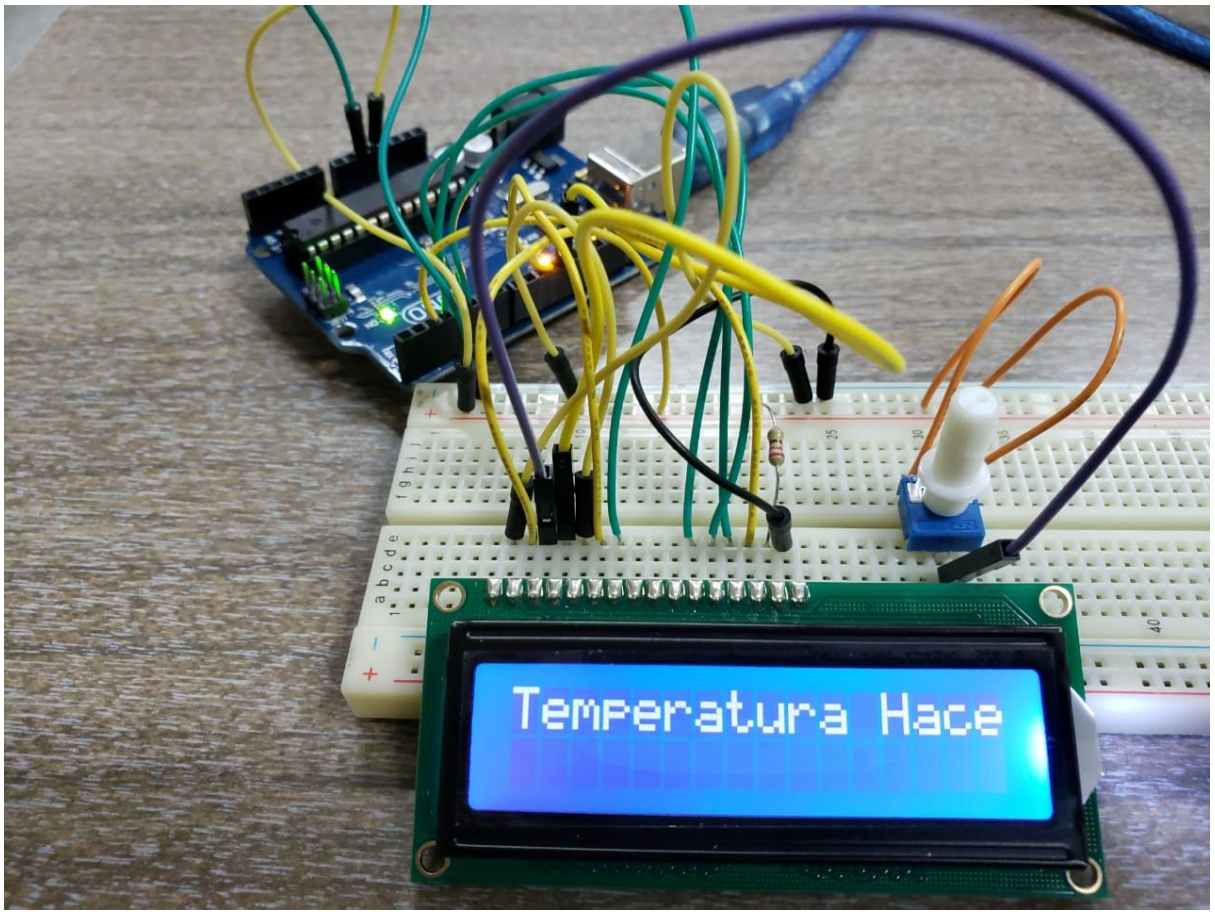
client.println("

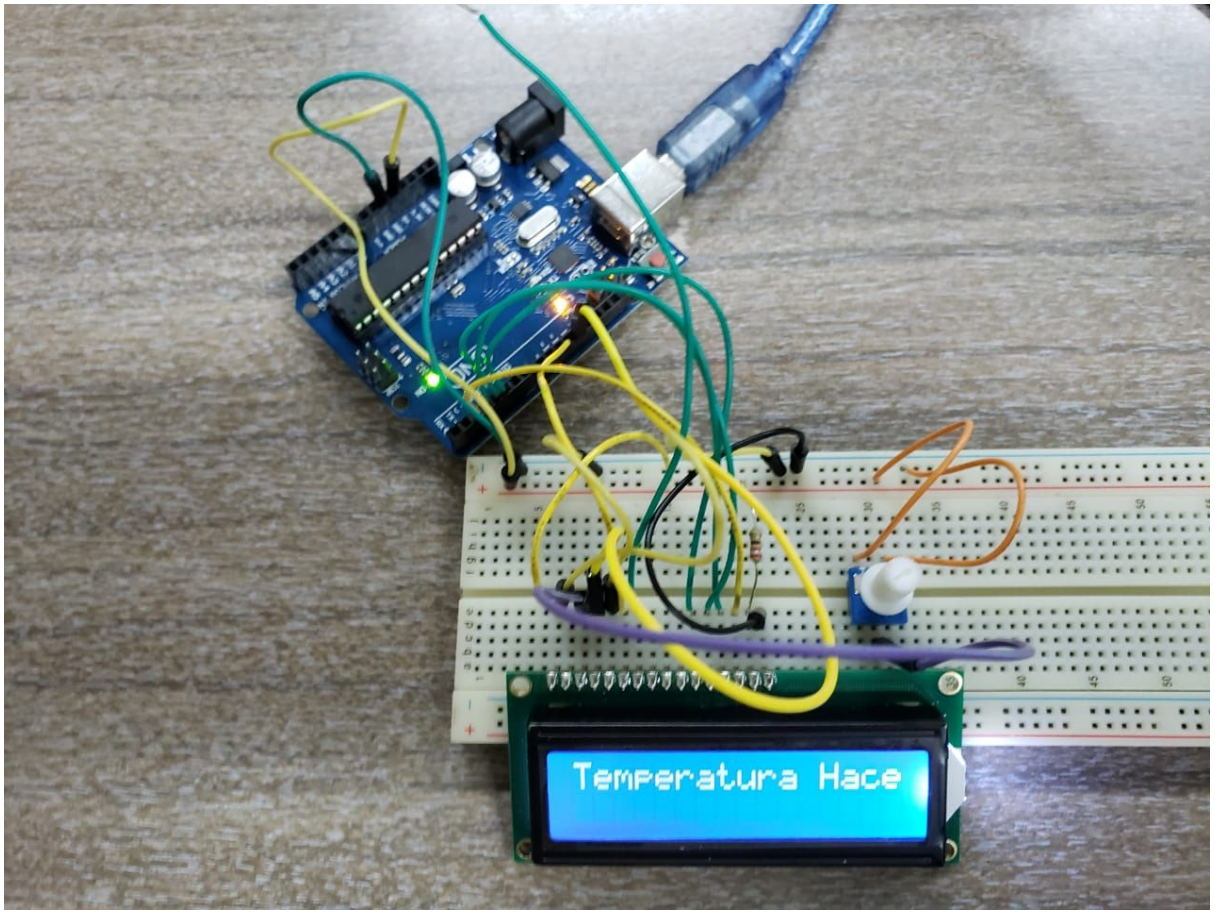
Sensaciontermica C:

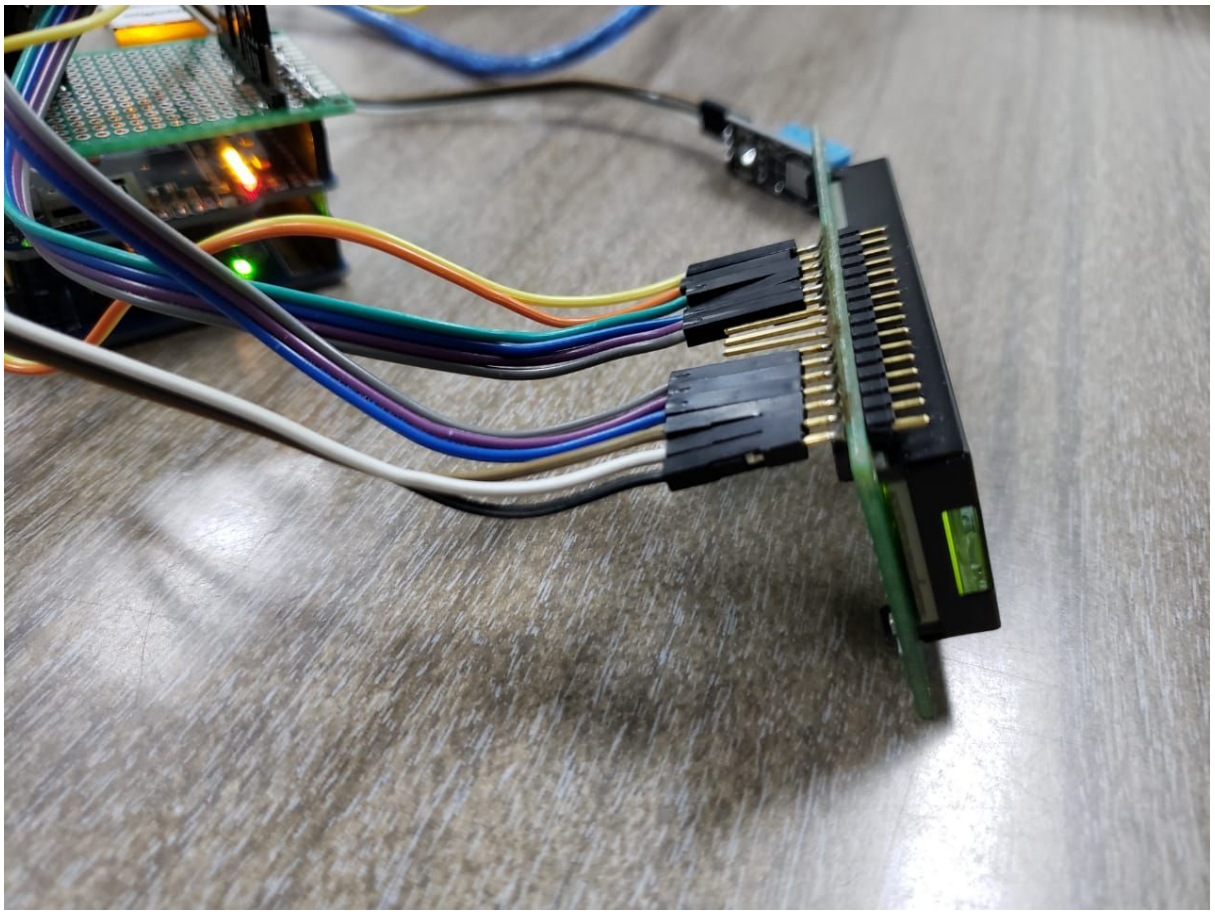
");

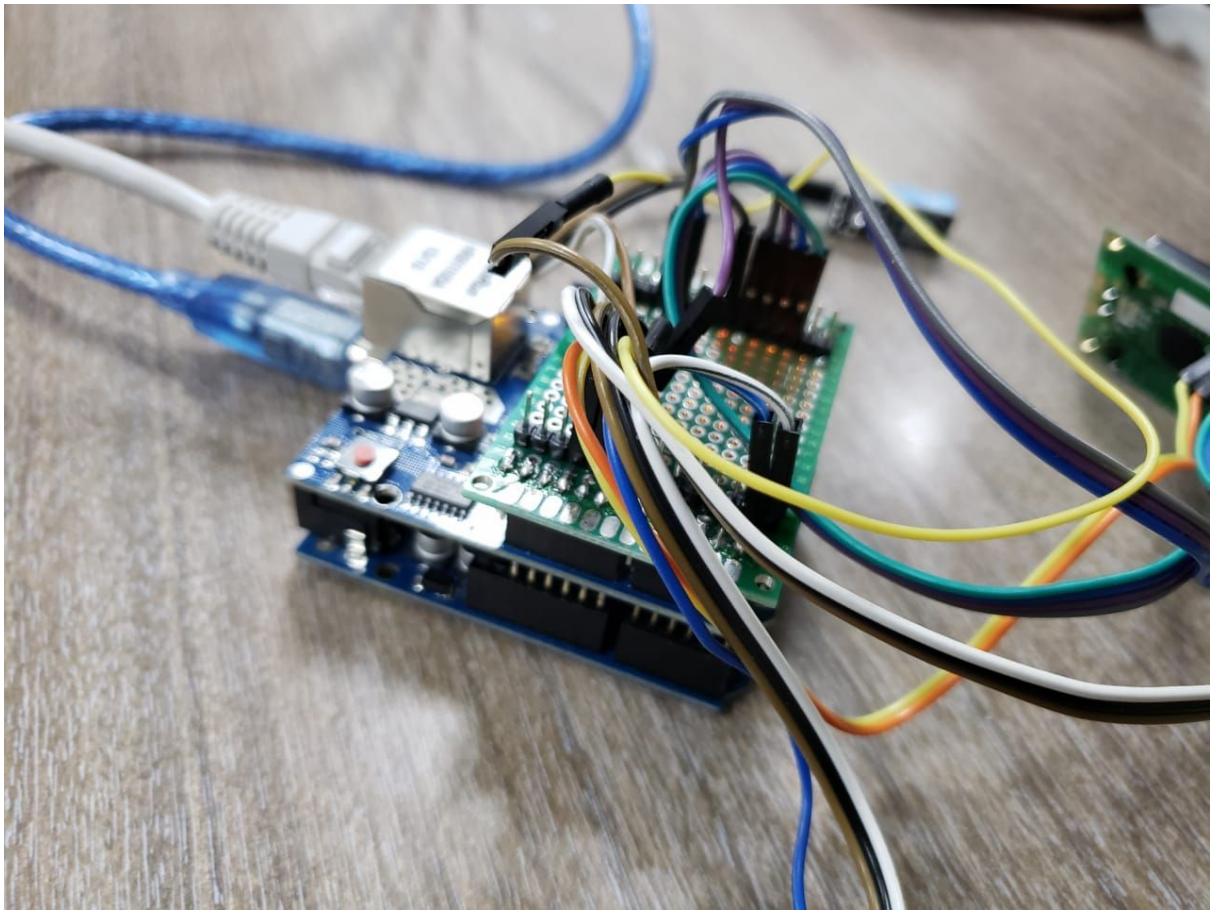
```
client.println("
");

client.println(sensc);

client.println(" C");

client.println("

");

client.println("
Sensaciontermica F:

");
client.println("
");

client.println(sensf);

client.println(" F");

client.println("


");

/// Fin DHT 11

client.println("");
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();
Serial.println("client disconnected");
```

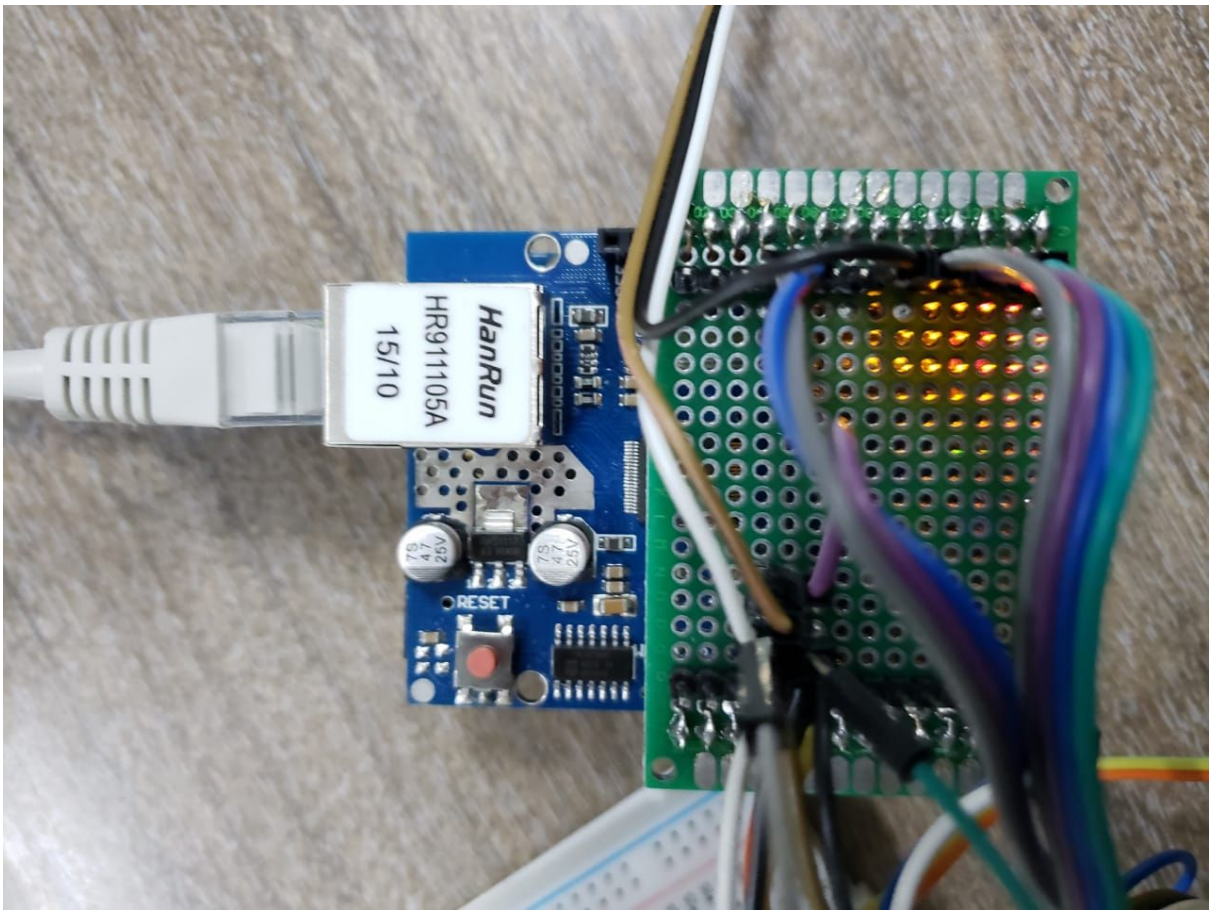
```
}

}
```

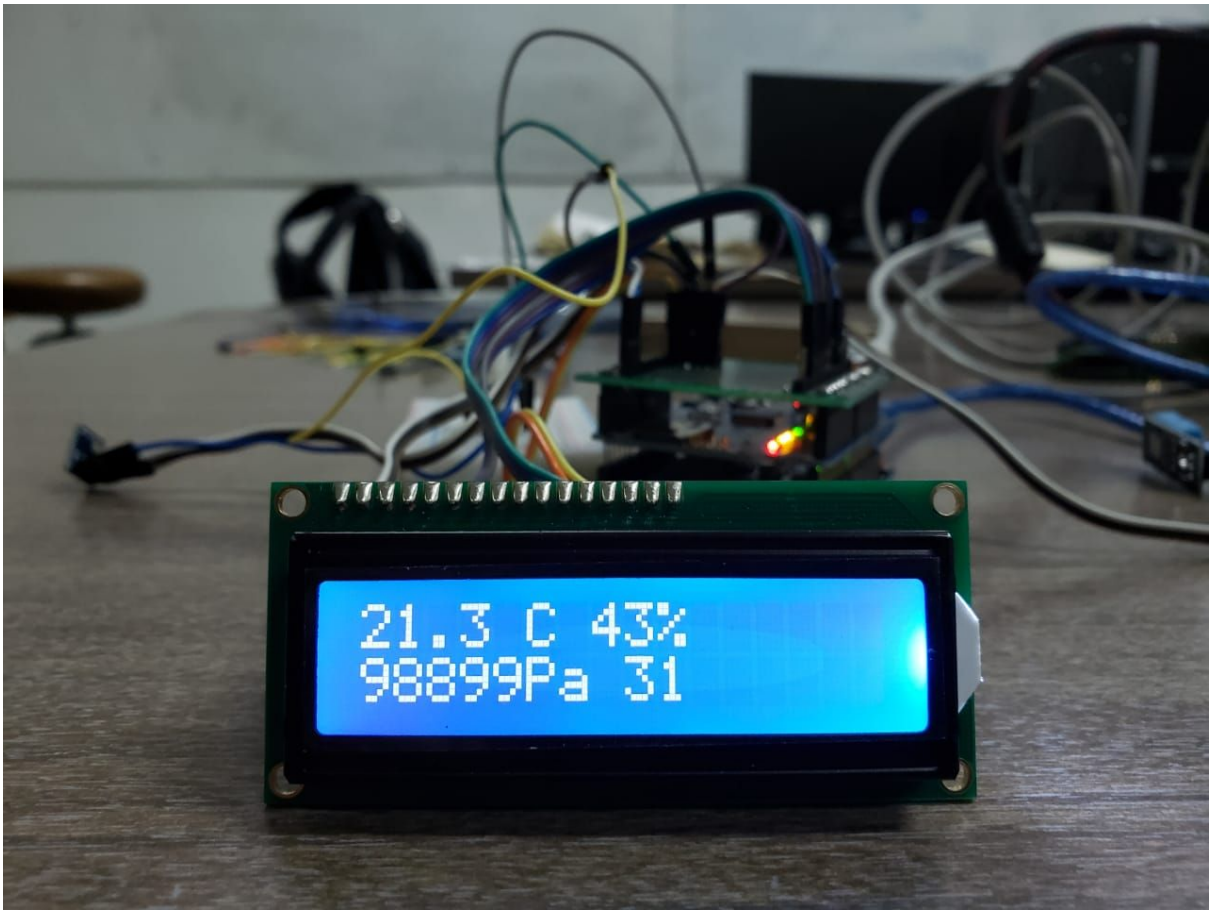

Temperatura Hace


Temperatura Hace






HanRun
HR911105A
15/10
RESET


21.3 C 43%
98899Pa 31

Bibliografía.

Ariffin Noordin, K.,et.al.,"A Low-Cost Microcontroller-based Weather Monitoring System mohd noor derahman View project Solar Cell View project A Low-Cost Microcontroller-based Weather Monitoring System"(2006),Journal(vol: 5 (1)).

Baldocchi, D., Hincks, B., Meyers, T., "Measuring Biosphere-Atmosphere Exchanges of Biologically Related Gases with Micrometeorological Methods" (1988), Ecology (vol: 69 (5) pp: 1331-1340)

Fedi, A.,et.al., "The ACRONET paradigm and open hardware project BYU ScholarsArchive Citation" (2013), Open Water Journal (vol: 2 (1) pp: 7)

Fisher, D., Gould, P., "Open-Source Hardware Is a Low-Cost Alternative for Scientific Instrumentation and Research" (2012), Modern Instrumentation(vol: 1 pp: 8-20)

Pearce, J., "Building Research Equipment with Free, Open-Source Hardware" (2012), Science (vol: 337 (6100) pp: 1303-1304)

Pearce, J., "Quantifying the Value of Open Source Hardware Development" (2015), consultado en línea [fecha de consulta:28/06/2019] en: creativecommons.org/licenses/by/4.0/.

Wittbrod, B., "Life-cycle economic analysis of distributed manufacturing with open-source 3-D printers" (2013), Mechatronics (vol: 23 (6) pp: 713-726)