|  |  |
| --- | --- |
| |  | | --- | | **INGENIERÍA EN SISTEMAS COMPUTACIONALES**  Ejercicio Sensor de 5 Módulos |   **TECNOLÓGICO DE ESTUDIOS SUPERIORES DE CHALCO** |



|  |  |
| --- | --- |
| |  | | --- | | **INTEGRANTES**  Arfaxad Zadot Aguilar Jiménez |   **SISTEMAS PROGRAMABLES** |

**DESCRIPCIÓN**

**INTRODUCIÓN**

Hace unos años apareció un proyecto libre, llamado **Arduino, el** cual facilitaba el acceso a esta clase de dispositivos a los estudiantes, dado que es una placa basada en open hardware (su diseño es libre y es posible reproducirlo por cualquier persona).

Inicialmente, la placa se conectaba a través de un puerto USB para programarlo (normalmente se hace en base a su IDE) La programación de Arduino no se realizaba a bajo nivel con ensamblador como muchos microcontroladores (a partir de ahora los llamaremos PICs), sino que se realiza con un lenguaje más comprensible por la mayoría de nosotros, C/C++, cestos elementos, un programador que no sabe de PICs, sería capaz de programar el Arduino en poco tiempo.

**OBJETIVO**

Desarrollar un monitorio de 5 módulos en señales analógicas en la dirección ip 192.168.1.75 para ser visualizado gráficamente

**MATERIALES**

* 1 computadora con IDE Arduino y Proteus 8.
* Protoboard.
* 10 resistencias de 330K.
* Foto-resistencias
* Transistores
* Modulo Ethernet
* Módulo de Agua
* Arduino Uno.

**DESARROLLO DE CODIGO**

#include <SPI.h>

#include <Ethernet.h>

byte mac[] = {

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

};

IPAddress ip(192, 168, 1, 100); //dirección ip

EthernetServer server(80);

void setup() {

Serial.begin(9600);

// inicia la conexion y el servidor:

Ethernet.begin(mac, ip);

server.begin();

Serial.print("servidor en lista");

Serial.println(Ethernet.localIP());

}

void loop() {

// Detectar los clientes entrantes

EthernetClient client = server.available();

if (client) {

Serial.println("Nuevo cliente");

boolean currentLineIsBlank = true;

while (client.connected()) {

if (client.available()) {

char c = client.read();

Serial.write(c);

// Si se ha llegado al final de la línea (recibirá una nueva línea

// con un Caracter en blanco, la petición http ha terminado,

// Para que pueda enviar una respuesta

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

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

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

client.println("Conexion Cerrada: cerrada");

client.println();

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

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

client.print("<span style='background: #ccc;'>VALORES DE ENTRADAS ANALOGICAS</span> ");

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

// Añadir una etiqueta para conseguir que el navegador se actualice cada 5 segundos:

client.println("<meta http-equiv=\"refresh\" content=\"5\">");

// salida del valor de cada entrada analogica

for (int analogChannel = 0; analogChannel < 6; analogChannel++) {

int sensorReading = analogRead(analogChannel);

client.print("Entrada analogica: ");

client.print(analogChannel);

client.print(" es <b>");

client.print(sensorReading);

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

}

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

break;

}

if (c == '\n') {

// se inicia una nueva linea

currentLineIsBlank = true;

}

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

// ha llegado a un caracter en la línea actual

currentLineIsBlank = false;

}

}

}

// facilita al navegador web un tiempo para recibir los datos

delay(1);

// cierra la conexión:

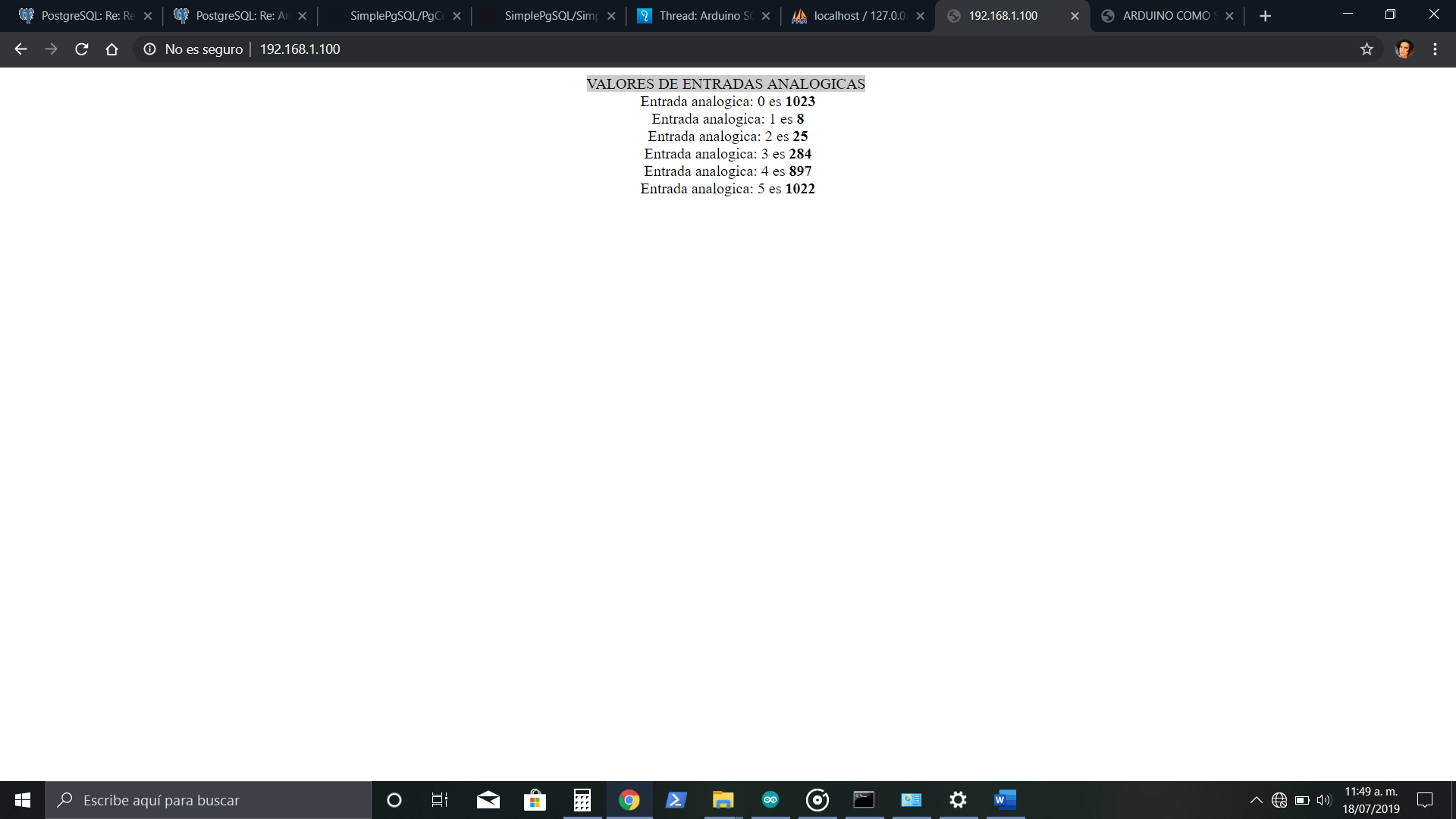
client.stop();

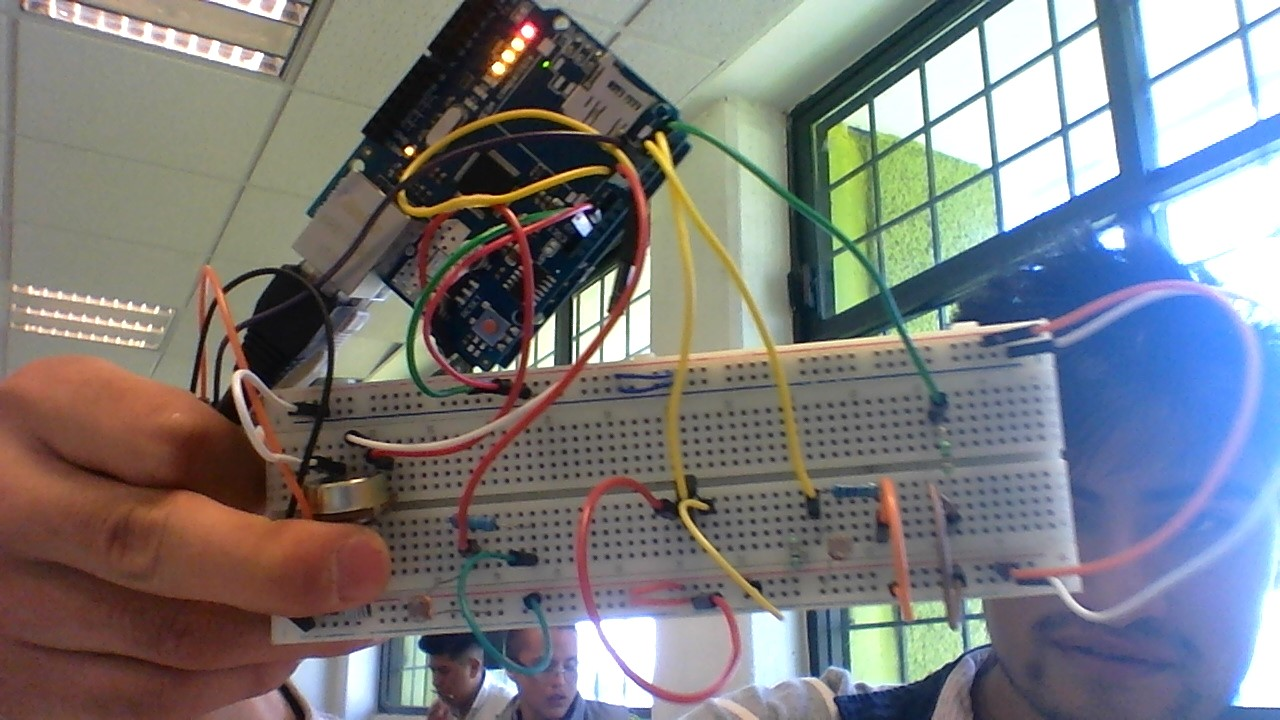
Serial.println("Cliente desconectado");

}

}

Resultado de la simulación.





**CONCLUSIONES**

Cuando se desarrollo en circuito con maquetación los 5 Volts de la batería no iban a ser eficientes para que el Arduino encendiera se optó por una pila de mayor capacidad como la de 9 Volts para que estos encendieran correctamente.