



UNIVERSITATEA DIN CRAIOVA  
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI  
ELECTRONICĂ  
DEPARTAMENTUL DE AUTOMATICĂ, ELECTRONICĂ ȘI  
MECATRONICĂ



## PROIECT DE DIPLOMĂ

Ionuț-Cristian Minică

COORDONATOR ȘTIINȚIFIC

Șef lucrări dr. ing. Ionuț Cristian Reșceanu

Septembrie 2014

CRAIOVA



UNIVERSITATEA DIN CRAIOVA  
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI  
ELECTRONICĂ  
DEPARTAMENTUL DE AUTOMATICĂ, ELECTRONICĂ ȘI  
MECATRONICĂ



Sistem de monitorizare inteligent pentru mediul industrial folosind Arduino  
Uno V3

Ionuț-Cristian Minică

COORDONATOR ȘTIINȚIFIC

Șef lucrări dr. ing. Ionuț Cristian Reșceanu

Septembrie 2014

CRAIOVA

*„Învățătura este o comoară care își urmează stăpânul pretutindeni.”*

Proverb popular

## DECLARAȚIE DE ORIGINALITATE

Subsemnatul Ionuț-Cristian Minică, student la specializarea Automatică și Informatică Aplicată din cadrul Facultății de Automatică, Calculatoare și Electronică a Universității din Craiova, certific prin prezenta că am luat la cunoștință de cele prezentate mai jos și că îmi asum, în acest context, originalitatea proiectului meu de licență:

- cu titlul “Sistem de monitorizare inteligent pentru mediul industrial folosind Arduino Uno V3”
- coordonată de Sl. Dr. Ing. Ionuț Cristian Reșceanu
- prezentată în sesiunea Septembrie 2014.

La elaborarea proiectului de licență, se consideră plagiat una dintre următoarele acțiuni:

- reproducerea exactă a cuvintelor unui alt autor, dintr-o altă lucrare, în limba română sau prin traducere dintr-o altă limbă, dacă se omit ghilimele și referința precisă,
- redarea cu alte cuvinte, reformularea prin cuvinte proprii sau rezumarea ideilor din alte lucrări, dacă nu se indică sursa bibliografică,
- prezentarea unor date experimentale obținute sau a unor aplicații realizate de alți autori fără menționarea corectă a acestor surse,
- însușirea totală sau parțială a unei lucrări în care regulile de mai sus sunt respectate, dar care are alt autor.

Pentru evitarea acestor situații neplăcute se recomandă:

- plasarea între ghilimele a citatelor directe și indicarea referinței într-o listă corespunzătoare la sfârșitul lucrării,
- indicarea în text a reformulării unei idei, opinii sau teorii și corespunzător în lista de referințe a sursei originale de la care s-a făcut preluarea,
- precizarea sursei de la care s-au preluat date experimentale, descrieri tehnice, figuri, imagini, statistici, tabele et caetera,
- precizarea referințelor poate fi omisă dacă se folosesc informații sau teorii arhicunoscute, a căror paternitate este unanim cunoscută și acceptată.

Data,

Semnătura candidatului,



UNIVERSITATEA DIN CRAIOVA  
Facultatea de Automatică, Calculatoare și Electronică  
Departamentul de Automatică, Electronică și Mecatronică

Aprobat la data de .....  
Șef de departament,  
Prof. dr. ing.  
Emil PETRE

## PROIECTUL DE DIPLOMĂ

Numele și prenumele studentului/-ei:	Minică Ionuț-Cristian
Enunțul temei:	„Sistem de monitorizare inteligent pentru mediul industrial folosind Arduino Uno V3”
Datele de pornire:	Arduino, Aplicații industriale, Interfață internet
Conținutul proiectului:	<p>În capitolul “Introducere” sunt prezentate scopul, motivația și avantajele acestui proiect.</p> <p>În capitolul “Descrierea tehnologiilor folosite” sunt prezentate din punct de vedere teoretic tehnologiile folosite în acest proiect.</p> <p>În capitolul “Descrierea aplicației practice” este prezentată arhitectura generală a proiectului din punct de vedere practic și modul de funcționare a modulelor acestuia.</p> <p>În capitolul “Concluzii” sunt prezentate concluziile autorului.</p>
Material grafic obligatoriu:	
Consultații:	săptămânale
Conducătorul științific (titlul, nume și prenume, semnătura):	Șef lucrări dr. ing. Ionuț Cristian Reșceanu
Data eliberării temei:	03.02.2014

Termenul estimat de predare a proiectului:	01.09.2014
Data predării proiectului de către student și semnătura acestuia:	



UNIVERSITATEA DIN CRAIOVA  
Facultatea de Automatică, Calculatoare și Electronică

Departamentul de Automatică, Electronică și Mecatronică

## REFERATUL CONDUCĂTORULUI ȘTIINȚIFIC

Numele și prenumele candidatului: Minică Ionuț-Cristian  
Automatică și informatică aplicată

Specializarea:

Titlul proiectului: Sistem de monitorizare inteligent pentru mediul industrial folosind Arduino Uno V3

Locația în care s-a realizat practica de documentare (se bifează una sau mai multe din opțiunile din dreapta): În facultate ☐

În producție ☐

În cercetare ☐

Altă locație:

În urma analizei lucrării candidatului au fost constatate următoarele:

Nivelul documentării		Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
Tipul proiectului		Cercetare <input type="checkbox"/>	Proiectare <input type="checkbox"/>	Realizare practică <input type="checkbox"/>	Altul [se detaliază]
Aparatul matematic utilizat		Simplu <input type="checkbox"/>	Mediu <input type="checkbox"/>	Complex <input type="checkbox"/>	Absent <input type="checkbox"/>
Utilitate		Contract de cercetare <input type="checkbox"/>	Cercetare internă <input type="checkbox"/>	Utilare <input type="checkbox"/>	Altul [se detaliază]
Redactarea lucrării		Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
Partea grafică, desene		Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
Realizarea practică	Contribuția autorului	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Mare <input type="checkbox"/>	Foarte mare <input type="checkbox"/>
	Complexitatea temei	Simplă <input type="checkbox"/>	Medie <input type="checkbox"/>	Mare <input type="checkbox"/>	Complexă <input type="checkbox"/>
	Analiza cerințelor	Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>

	Arhitectura	Simplă <input type="checkbox"/>	Medie <input type="checkbox"/>	Mare <input type="checkbox"/>	Complexă <input type="checkbox"/>
	Întocmirea specificațiilor funcționale	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
	Implementarea	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
	Testarea	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
	Funcționarea	Da <input type="checkbox"/>	Parțială <input type="checkbox"/>	Nu <input type="checkbox"/>	
Rezultate experimentale		Experiment propriu <input type="checkbox"/>		Preluare din bibliografie <input type="checkbox"/>	
Bibliografie		Cărți	Reviste	Articole	Referințe web
Comentarii și observații					

În concluzie, se propune:

ADMITEREA PROIECTULUI <input type="checkbox"/>	RESPINGEREA PROIECTULUI <input type="checkbox"/>
---	---

Data,

Semnătura conducătorului științific,



## REZUMATUL PROIECTULUI

Proiectul constă în aplicația practică de monitorizare și reglare a unor caracteristici unui mediu industrial. Se monitorizează nivelurile parametrilor fizici de importanță semnificativă precum nivelul de gaze inflamabile, temperatura. Datele prelevate de placa de dezvoltare Arduino sunt procesate de un calculator personal ce le interpretează utilizându-le spre popularea unei pagini web. Pagina web poate fi accesată de pe un dispozitiv conectat la internet, fie ele mobil sau nu. În cazul unor valori ce depășesc un prag setat apriori, considerat periculos, aplicația semnalează starea de avarie, avertizând personalul ce se află în vecinătate de situația periculoasă și acționează spre îndepărtarea pericolului.

**Termenii cheie:** domotică, Arduino, automatizare, aplicații industriale, C++, php, internet, server, interfață, Win32Api.

## **MULȚUMIRI**

Îi mulțumesc Monicăi Stefania pentru răbdarea cu care m-a verificat și m-a ajutat la prezenta lucrare.

Mi-aș dori să îmi exprim mulțumirea față de Dr. Ing. Ionuț Cristian Reșceanu pentru implicarea, contribuția și răbdarea de-a lungul procesului de planificare, construcție și redactare a acestui proiect.

## PROLOG

Încă din cele mai vechi timpuri omenirea a căutat să își simplifice viața, să o automatizeze. Astfel eforturile de-a lungul timpului ale oamenilor învățați s-au îndreptat spre înțelegerea, mai mult sau mai puțin în detaliu, a proceselor fizice și chimice. Automatizarea necesită cunoștințe și din domenii variate precum mecanica, hidraulica, pneumatica, electrica, electronica și calculatoare, de obicei în diverse combinații.

Scopul automatizării este de a reduce sau înlocui munca efectuată de factorul uman, de a sporii calitatea, cantitatea și/sau timpul de producere a unui bun. Din aceasta cauză apar conflicte de ordin etic precum reducerea locurilor de muncă pentru oameni. Totuși există domenii în care acest conflict dispare, acela al securizării locurilor periculoase, al sarcinilor imposibil de efectuat de către om, pe scurt orice sarcină care pune în pericol semnificativ omul este o pretendentă pentru automatizare.

# CUPRINSUL

<b>1</b>	<b>INTRODUCERE .....</b>	<b>1</b>
1.1	SCOPUL.....	1
1.2	MOTIVAȚIA.....	1
1.3	AVANTAJE PROIECT.....	1
<b>2</b>	<b>DESCRIEREA TEHNOLOGIILOR FOLOSITE .....</b>	<b>2</b>
2.1	INTERFAȚA UTILIZATOR .....	2
2.1.1	HTML.....	2
2.1.2	PHP.....	4
2.1.3	Internet Server .....	6
2.2	NIVELUL HARDWARE .....	8
2.2.1	Arduino Uno V3.....	8
2.2.2	Senzorii folosiți.....	12
2.2.3	Actuatorii folosiți .....	15
2.2.4	Mediul de programare Arduino .....	19
2.3	LEGĂTURA HARDWARE – INTERFAȚA .....	22
2.3.1	Windows .....	22
2.3.2	Limbajul de programare C++ .....	23
2.3.3	Interfața de programare Windows - C++.....	25
2.3.4	Comunicații seriale prin portul USB .....	25
2.3.5	Serverul Apache .....	28
2.4	ALTE APLICAȚII SEMNIFICATIVE .....	29
2.4.1	Git și GitHub.....	29
2.4.2	Notepad++ .....	30
2.4.3	XAMPP .....	30
<b>3</b>	<b>DESCRIEREA APLICAȚIEI PRACTICE.....</b>	<b>32</b>
3.1	ARHITECTURĂ GENERALĂ .....	32
3.2	APLICAȚIA ARDUINO .....	33
3.3	APLICAȚIA WINDOWS .....	35
3.4	APLICAȚIA PHP/HTML .....	38
<b>4</b>	<b>CONCLUZII .....</b>	<b>43</b>
<b>5</b>	<b>BIBLIOGRAFIE .....</b>	<b>44</b>
<b>6</b>	<b>REFERINȚE WEB.....</b>	<b>45</b>

A.	CODUL SURSĂ.....	47
B.	SITE-UL WEB AL PROIECTULUI .....	60
C.	CD / DVD .....	63
	INDEX .....	64

# LISTA FIGURILOR

FIGURĂ 1 .....	9
FIGURĂ 2 .....	11
FIGURĂ 3 .....	13
FIGURĂ 4 .....	15
FIGURĂ 5 .....	16
FIGURĂ 6 .....	18
FIGURĂ 7 .....	19
FIGURĂ 8 .....	21
FIGURĂ 9 .....	24
FIGURĂ 10 .....	29
FIGURĂ 11 .....	31
FIGURĂ 12 .....	33
FIGURĂ 13 .....	34
FIGURĂ 14 .....	35
FIGURĂ 15 .....	40
FIGURĂ 16 .....	41
FIGURĂ 17 .....	41
FIGURĂ 18 .....	42

# **1 INTRODUCERE**

## **1.1 Scopul**

Scopul elaborării acestui proiect este de a dobândii experiență practică în folosirea microcontrolerelor, senzorilor, actuatorilor și a folosirii tehnologiilor răspândite din domeniul calculatoarelor precum sistemul de operare Windows, serverul Apache, limbajul de programare C++ și PHP.

## **1.2 Motivația**

Tema aleasă însumează o mulțime a deprinderilor și cunoștințelor teoretice acumulate în perioada studiilor, fiind un prilej de exersare a acestora. Un alt motiv important este acela de a observa cum pot fi folosite aceste cunoștințe spre a rezolva o problemă din viața reală.

## **1.3 Avantaje proiect**

Placa de dezvoltare Arduino este folosită de un număr foarte mare de oameni fapt ce a condus la crearea de forum-uri specializate și cluburi de pasionați ce oferă suport și continuă să dezvolte platforma în colaborare cu utilizatorii. Astfel ea devine o alegere foarte bună, automată chiar, pentru începători.

Un alt avantaj este timpul de răspuns, acesta fiind aproape instantaneu în cazul unei situații considerate periculoase, procesarea și acționarea fiind făcută la nivelul microcontrolerului.

Accesul la interfața utilizator se poate face de pe orice dispozitiv conectat la internet și are un browser compatibil HTML.

Proiectul poate fi ușor scalat, putându-se executa mult mai multe instrucțiuni decât în prezent la nivelul Arduino-ului. Se mai pot adăuga senzori/actuatori, iar dacă se dorește scalarea cu mai multe plăci s-ar putea folosi un shield(modul ușor de conectat cu Arduino) Bluetooth pentru comunicația serială astfel nefiind nevoie decât de un port la nivelul calculatorului.

## **2 DESCRIEREA TEHNOLOGIILOR FOLOSITE**

### **2.1 Interfața Utilizator**

#### **2.1.1 HTML**

##### *Introducere în HTML*

În anul 1980 Tim Berners-Lee a construit un sistem de împărtășirea documentelor între oamenii de știință din sistemul CERN. Influențat de experiența proiectului a propus crearea unui sistem hypertext bazat pe internet. Mai târziu, în 1990, a scris browser-ul, server-ul și prima specificație a HTML.

Hyper Text Markup Language este un limbaj de marcare sau o mulțime de tag-uri de marcare folosit pentru descrierea modului în care o pagina web trebuie afișată de către o aplicație precum un browser web. Este limbajul standard folosit în ziua de azi pentru a crea pagini web. Deși este un standard pentru paginile web, rar se mai găsesc pagini web ce conțin numai HTML și activele folosite de pagină(poze, imagini, animații, sunete etc.), majoritate conțin și fragmente de JavaScript(se execută la nivelul browserului), PHP(se execută la nivelul serverului), baze de date(pentru stocarea informațiilor, de exemplu utilizatorul și parola), CSS etc.

Un browser web poate citi și interpreta acest document, creând și afișând pagina cerută.

##### *Elemente și taguri*

Un element poate fi definit ca: „o componentă individuală dintr-un document HTML”. Elementele sunt formate din cel puțin un tag de început, dar în funcție de tipul elementului poate avea și un tag de sfârșit și/sau alte atribute.

Un tag poate fi definit ca : „ un cod folosit în HTML pentru a defini un format de schimb sau link de tip hypertext”.



## Structura unui document HTML

Un document HTML este format din mai multe elemente, de obicei structurate arborescent.

În următorul exemplu de document HTML este prezentat programul clasic “Hello World”, folosit pentru evaluarea/compararea unui limbaj, fie el de programare sau de marcare(cazul de față).

```
<!DOCTYPE html>
<html>

  <head>
    <title>Title</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

Se observă cu ochiul liber structura arborescentă a organizării limbajului, chiar dacă acesta are doar 9 linii de cod.

Secvența `<!DOCTYPE html>` are rolul de a declara tipul documentului. În primele versiuni de HTML scopul secvenței era de a activa validarea și parsarea documentului de către browser. În browser-ele moderne definiția ajută la stabilirea modului de randare.

Textul dintre tagurile `<html>` `</html>`, numit și element HTML, conține descrierea paginii web. Elementul notifică browser-ul că urmează un document HTML. Este și rădăcina documentului, el conținând toate elementele ulterioare cu excepția `<!DOCTYPE html>`.

Textul dintre tagurile `<head>` `</head>`, conține elemente precum titlul paginii, stiluri și meta informație.

Textul dintre tagurile `<body>` `</body>`, conține elementele ce trebuiesc afișate în pagina web. Elementul definește practic corpul documentului.

### 2.1.2 PHP

PHP a fost dezvoltat inițial de Rasmus Lerdorf în 1994, un programator ce avea nevoie de o metodă de a întreține pagina sa personală. Ulterior acesta a fost extins adăugându-se capacități de lucru cu formulare HTML precum și comunicarea cu baze de date.

Urmează exemplul tipic de comparare a unui limbaj, programul „Hello World” :

```
<!DOCTYPE html>
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <?php echo '<p>Hello World</p>'; ?>
  </body>
</html>
```

Codul PHP este inserat în interiorul documentului HTML. Ceea ce se află între tagurile <?php ?>, este codul PHP efectiv ce se execută de interpretorul PHP.

PHP:Hyper Text Processor este un limbaj de scripting (deci interpretat), un limbaj cu tipuri de date dinamice(nu trebuie definite apriori).

Ca și tipuri de date, PHP stochează numerele întregi precum și cele flotante într-o gamă specifică platformei sub care a fost instalat interpretorul. Numerele flotante pot fi specificate folosind notația în virgulă mobilă sau cele două forme de notație științifică. PHP conține și un tip de date boolean, ce poate avea cele două valori adevărat sau fals, precum și tipul de date null, el reprezentând o valoare ce nu există.

O variabilă reprezintă un alias pentru o zonă de memorie utilizată pentru stocarea de informație. Definiția unei variabile în PHP începe cu semnul dolar \$. O variabilă nu poate începe decât cu o literă sau caracterul underscore \_. O variabilă nu poate începe cu o cifră, trebuie să folosească numai caractere alfa-numerice și/sau underscore. Numele variabilelor este case-sensitive.

```
<?php
```

```
$intreg = 10;
```

```
$boolean = true;
```

```
$flotant = 10.365;
```

```
$flotant_notatie stiintifica1 = 2.4e3;
$flotant_notatie_stiintifica2 = 8E-5;

$vector=array("Volvo","BMW","Toyota");
?>
```

PHP are sute de funcții oferite de funcționalitatea limbă de bază și multe altele disponibile prin diferite extensii. Aceste funcții sunt bine documentate în documentația on-line PHP.

Pe lângă funcțiile built-în PHP, ne putem crea propriile noastre funcții.

O funcție este un bloc de declarații care pot fi utilizate în mod repetat, într-un program. Ea nu va executa imediat când o pagină se încarcă ci printr-un apel al acesteia.

```
<?php
function scrieMsg() {
    echo "Hello world!";
} //definitia functiei

scrieMsg (); // apelul functiei
?>
```

O clasă este o structură care definește șablonul unui obiect și conține declarația proprietăților și metodelor. Un obiect este un tip de date care stochează date și metode ce conțin modul de a procesa aceste date.

În PHP, un obiect trebuie să fie declarat în mod explicit. În primul rând trebuie să ne declarăm o clasă de obiecte. Pentru aceasta, vom folosi cuvântul cheie class.

```
<?php

class Masina
{
    var $culoare;
    function Masina ($culoare ="green") {
        $this-> culoare = $ culoare;
    }
    function what_color() {
        return $this-> culoare;
    }
}
}??>
```

### 2.1.3 Internet Server

#### *Introducere*

Un server este un exemplu de funcționare a unei aplicații ( software ) capabile de a accepta cereri de la client și de a da răspuns în consecință. Serverele pot rula pe orice calculator sau calculator dedicat, care este, de asemenea, adesea menționată ca "server", în multe cazuri, un calculator poate furniza mai multe servicii și pe acesta rulează mai multe servere. Avantajul de a rula servere pe un calculator dedicat este acela de securitate sporită.

Serverele funcționează în cadrul unei arhitecturi client-server. Ele sunt programe de calculator care rulează pentru a servi cererile de alte programe, clienții. Astfel, serverul efectuează unele activități în numele clienților. Clienții sunt conectați de obicei la server prin intermediul rețelei, dar pot rula și de pe același computer. În contextul de Internet Protocol ( IP ) de rețea, un server este un program care funcționează ca un ascultător.

Serverele oferă adesea servicii esențiale într-o rețea, fie pentru utilizatorii privați, în interiorul unei organizații mari sau a utilizărilor publice prin intermediul internetului. Serverele de calcul tipice sunt:

- server de baze de date
- server de fișiere
- server de mail
- server de imprimare
- server web
- server de jocuri
- server de aplicații

Numeroase sisteme folosesc acest model de rețea client-server, inclusiv site-uri web și servicii de e-mail. Un model alternativ, de tip peer- to-peer permite tuturor calculatoarelor să acționeze ca un server sau client după cum este necesar.

### *Modelul Client-Server*

Modelul client-server de calcul este o structură aplicație distribuită care împarte sarcini de lucru între furnizorii de o resursă sau serviciu, numiți servere, și solicitanții de servicii, numiți clienți.

De multe ori clienții și serverele comunică printr-o rețea, dar ele pot exista simultan pe același sistem. O gazdă server rulează unul sau mai multe programe de server care împărtășesc resursele lor cu clienți

Exemple de aplicații informatice care folosesc modelul client – server sunt e-mail și World Wide Web.

Caracteristica client-server descrie relația de programe care au cooperat într-o aplicație. Componenta de server oferă o funcție sau serviciu la unul sau mai mulți clienți, care inițiază cereri pentru astfel de servicii.

Serverele sunt clasificate pe serviciile pe care le furnizează. De exemplu, un server de web servește pagini web și un server de fișiere servește fișiere de calculator. O resursă comună poate fi orice legat de software, cât și unele componente ale server-ului, de la programe ce procesează date până la dispozitive de stocare. Împărțirea resurselor unui server se numește serviciu.

Dacă un calculator este un client, un server, sau ambele, este determinat de natura cererii care necesită funcțiile de serviciu. De exemplu, un singur calculator poate rula server web și software de server de fișiere, în același timp, pentru a servi date pentru clienții care fac diferite tipuri de solicitări. Software-ul client poate comunica, de asemenea, cu software de tip server în același computer. Comunicarea între servere, cum ar fi sincronizarea datelor, este uneori numit inter-server sau comunicare de server la server.

## 2.2 Nivelul hardware

### 2.2.1 Arduino Uno V3

#### *Introducere*

Arduino este o placă de dezvoltare cu un singur microcontroler (ATMEGA 328 cazul de față), destinată dezvoltării de aplicații practice accesibile din punct de vedere tehnic pentru studenți sau pasionați.

Introdusă în 2005, platforma Arduino a fost proiectată pentru a oferi un mod ieftin și ușor pentru pasionați, studenți și profesioniști, să poată crea dispozitive care interacționează cu mediul lor, folosind senzori și actuatori. Exemple comune pentru amatori sau începători includ roboți simpli, termostate și detectoare de mișcare. Acesta este dotat cu un mediu simplu integrat de dezvoltare (IDE), care rulează pe computere personale obișnuite și permite utilizatorilor să scrie programe pentru Arduino folosind C sau C++.

Plăcile Arduino pot fi achiziționate pre-asamblat sau ca și kituri. Informațiile de proiectare a plăcii de dezvoltare sunt disponibile pentru cei care doresc să construiască un Arduino. A fost estimat la jumătatea anului 2011, că peste 300.000 plăci Arduino construite de dezvoltatorii proiectului (oficiale) au fost produse comercial și în 2013, 700.000 de plăci oficiale au fost în mâinile utilizatorilor.

#### *Specificatii tehnice*

Placa de dezvoltare are 14 pini digitali de intrare / ieșire (din care 6 pot fi folosiți ca ieșiri PWM), 6 intrări analogice, un rezonator ceramic de 16 MHz, o conexiune USB, un jack de alimentare și un buton de resetare.

Uno diferă de toate modelele precedente, în care se folosesc chip driver FTDI USB-to-serial pentru comunicație. În schimb, este dotat cu Atmega16U2, programat ca un convertor USB - to- serial.

- Microcontroler: ATmega328
- Tensiune de alimentare: 5V
- Tensiune de intrare (recomandat): 7-12V
- Tensiune de intrare (limite): 6-20V
- Pini digitali I / O: 14 (din care 6 oferă ieșire PWM)
- Pinii de intrare analogică: 6

- 

ARDUINO is a registered trademark.

Use of the ARDUINO name must be compliant with <http://www.arduino.cc/en/Main/Policy>

9

Arduino Uno poate fi alimentat prin conexiunea USB sau cu o sursă de alimentare externă. Sursa de energie este selectată automat.

Alimentarea externă poate veni fie de la un adaptor AC -DC sau de la o baterie. Adaptorul poate fi conectat prin mufa de 2.1mm centru - pozitiv în fișa de alimentare a plăcii de dezvoltare. Bornele unei baterii pot fi introduse în pinii GND și Vin.

Placa poate funcționa cu o sursă externă de 6-20 volți. În cazul în care este furnizat mai puțin de 7V, pinul de 5V va furniza mai puțin de 5V și placa poate fi instabilă. Dacă utilizați mai mult de 12V, regulatorul de tensiune se poate supraîncălzi și deteriora placa.

### *Comunicarea*

Arduino Uno are un număr de facilități pentru a comunica cu un calculator, un alt Arduino, sau alte microcontrolere. ATmega328 oferă UART TTL ( 5V ) comunicare serială, care este disponibilă pe pinii digitali 0 ( RX ) și 1 ( TX ). Un ATmega16U2 conectat la circuitul de apărare ca un port COM virtual pentru software-ul de pe computer. Firmware 16U2 folosește driverele USB standard COM, și nu este nevoie de nici un driver extern. Cu toate acestea, pe Windows, este necesar un fișier.inf. Software-ul Arduino include și o aplicație ce monitorizează porturile seriale, permițând transferul de date textuale simple către și de la Arduino. Pinii RX și TX sunt conectați la 2 LED-uri notate, RX respectiv TX, pe placa de dezvoltare, acestea aprinzându-se când se transmit date prin conexiunea USB-Serial la computer.



## Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash program memory (ATmega48PA/88PA/168PA/328P)
  - 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
  - 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C<sup>(1)</sup>
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I<sup>2</sup>C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 20 MHz @ 1.8 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 µA
  - Power-save Mode: 0.75 µA (Including 32 kHz RTC)



**8-bit AVR<sup>®</sup>**  
**Microcontroller**  
**with 4/8/16/32K**  
**Bytes In-System**  
**Programmable**  
**Flash**

**ATmega48PA**  
**ATmega88PA**  
**ATmega168PA**  
**ATmega328P**

Rev. 8161D-AVR-10/09



Figură 2

### 2.2.2 Senzorii folosiți

Un senzor este un dispozitiv care detectează schimbări în proprietățile și oferă o ieșire corespunzătoare, în general, un semnal electric sau optic; de exemplu, un termocuplu convertește temperatura la o tensiune de ieșire. Dar un termometru cu mercur, din sticlă este de asemenea un senzor; se convertește temperatura măsurătorii în expansiune și contracție a unui lichid, care poate fi citit pe un tub de sticlă calibrat.

Senzorii sunt folosiți în obiecte de zi cu zi, cum ar fi butoane de lift sensibile la atingere (senzor tactil) și lămpi care se întindecă sau luminează mai puternic prin atingerea bazei, pe lângă alte nenumărate aplicații de care majoritatea oamenilor nu sunt conștienți.

Sensibilitatea unui senzor indică cât de mult mărimea de ieșire a senzorului se schimbă atunci când cantitatea de intrare se modifică. De exemplu, în cazul în care mercurul în termometre se mută de 1 cm atunci când se schimbă temperatura cu  $1^{\circ}\text{C}$ , sensibilitatea este de  $1\text{ cm} / ^{\circ}\text{C}$  (aceasta este de fapt panta  $Dy / Dx$  presupunând o caracteristică liniară). Uni senzori pot avea, de asemenea, un impact asupra a ceea ce se măsoară; de exemplu, un termometru ce se afla la temperatura camerei inserat într-o ceașcă fierbinte de lichid răcește lichidul din ceașca în timp ce se încălzește termometrul. Senzorii trebuie să fie proiectați pentru a avea un efect mic asupra a ceea ce se măsoară. Progresul tehnologic permite tot mai mulți senzori să fie fabricați la scară microscopică folosind tehnologia MEMS.

#### *Senzor fum și gaze inflamabile MQ-2*

Materialul senzitiv din senzorul de gaz MQ-2 este  $\text{SnO}_2$ . Materialul are o proprietate ce îl face interesant pentru aplicația curentă, aceea de a avea o conductivitate scăzută în aerul curat. Când combustibil este însă prezent în vecinătatea lui, conductivitate crește și corespunde semnalului de ieșire. Senzorul este sensibil la GPL, propan și hidrogen.

Avantajele principale ale acestuia sunt:

- Sensitivitate bună la gaze combustibile din o gamă variată.
- Sensitivitate sporită la GPL, propan și hidrogen
- Circuit simplu de folosit
- Costuri mici

Specificații:

- Alimentare : 5V
- Temperatura de funcționare : -20 +50
- Curent: 150mA
- Dimensiuni : 16.8 mm diametru 9.3 mm înălțime fără pini



Figură 3

### *Senzor temperatură LM-50*

Senzorul LM50 este un circuit integrat de precizie, ce poate sesiza o diferență de temperatură între valorile de  $-40^{\circ}\text{C}$  și  $+125^{\circ}\text{C}$ ; intervalul de temperatură, folosind o singură sursă de tensiune continuă pozitivă. Tensiunea de ieșire LM50 este proporțional liniară cu temperatura ambientală ( $+10\text{ mV}/^{\circ}\text{C}$ ) și are un deplasament de curent continuu de  $+500\text{ mV}$ . Deplasamentul permite citirea temperaturii negative fără a fi nevoie de o sursă de tensiune negativă.

Principalele avantaje:

- Potrivit pentru aplicații la distanță
- Funcționează de la  $4.5\text{V}$  la  $10\text{V}$
- Calibrat direct în grade Celsius
- Mai puțin de  $130\text{ mA}$  curent de scurgere
- Factor de scalare liniar cu  $+10,0\text{ mV}/^{\circ}\text{C}$
- Precizia specificată  $\pm 2^{\circ}\text{C}$  la  $+25^{\circ}\text{C}$
- Funcționează în intervalul  $-40^{\circ}\text{C}$  la  $+125^{\circ}\text{C}$  cu o precizie de  $\pm 4^{\circ}\text{C}$

Principalele domenii de utilizare sunt :

- Unitățile de disc
- Managementul baterie
- Automotiv
- Aparat fax
- Imprimante
- Instrumente medicale portabile

### 2.2.3 Actuatorii folosiți

#### *LED*

Pentru a semnala luminos situațiile de avarie am folosit LED-uri.

LED-ul este o diodă ce emite lumină când este activată. Dioda este formată dintr-un material semiconductor dopat cu impurități pentru a crea o joncțiune de tip PN. LED-ul exploatează principiul electro-luminiscent, pentru a transforma energia electrică în energie luminoasă.

#### *Buzzer-ul piezoelectric*

Pentru a semnala sonor situațiile de avarie am folosit buzzer-ul piezoelectric.

Acesta este un dispozitiv ce este capabil sa creeze unde sonore atunci când primește un semnal. Ca și aplicații tipice un buzzer se găsește în dispozitive de alarmă, timere etc.

Un element piezoelectric este format dintr-un un circuit electronic oscilant(variază repetitiv), un amplificator și un difuzor piezoelectric.



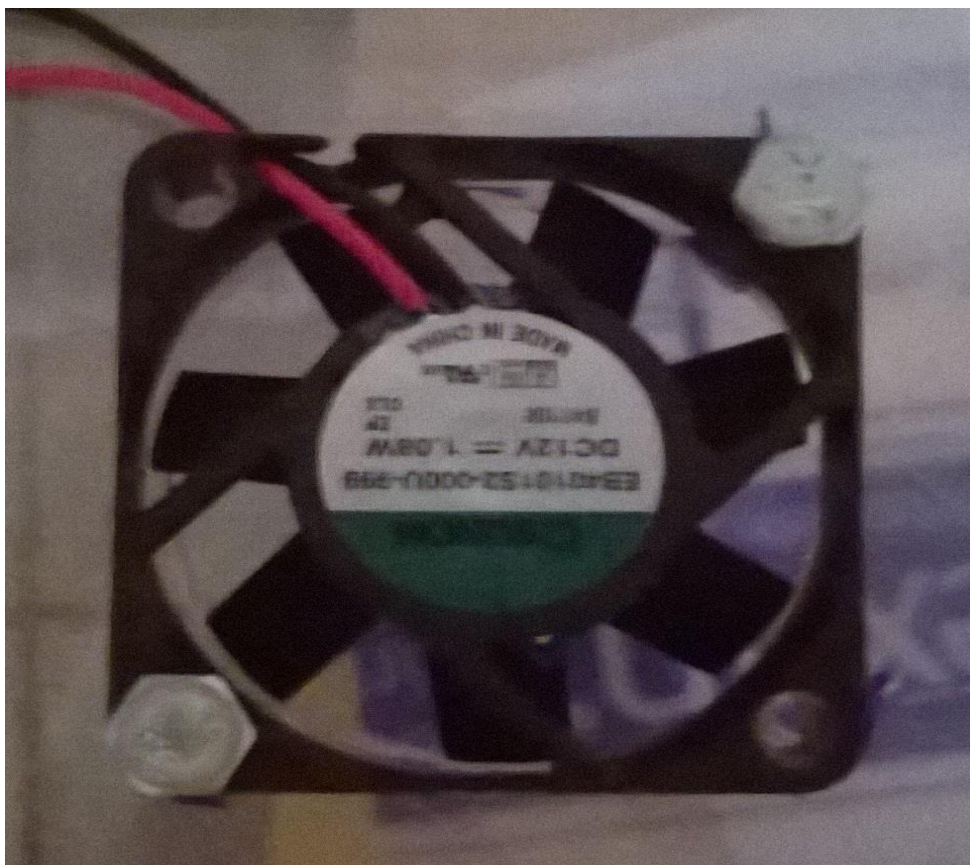
**Figură 4**

#### *Ventilatoarele*

Microventilatorul Sunon EB40101S2:

- Tensiune alimentare: 12V DC
- Dimensiune ventilator: 40x40x10mm
- Randament ventilatoare: 11.9m3/h
- Nivel zgomot: 27dBA
- Consum curent: 1.08W
- Tensiune de lucru: 6-13.8V

- Viteza de rotație: 5800 ( $\pm 15\%$ )rot/min
- Masă: 17g
- Curent nominal: 0.09A
- Material elice: termoplast
- Material carcasă: termoplast
- Clasă izolație: E
- Temperatura de lucru: -10...70°C
- Clasă inflamabilitate: UL94V-0
- Acționare ventilator: motor DC fără perii
- Lungime cablu: 300mm
- Dimensiune conductor: 26AWG



**Figură 5**

## Ventilatorul Sunon

- Tensiune alimentare: 12V DC
- Dimensiune ventilator: 80x80x25mm
- Randament ventilatoare: 68,4 m<sup>3</sup>/h
- Nivel zgomot: <15dBA
- Consum curent: 2,6W
- Tensiune de lucru: 5...13.8V
- Viteza de rotație: 3000(±15%)rot/min
- Masă: 123g
- Curent nominal: 0.165A
- Material: elice termoplast
- Material carcasă: termoplast
- Clasă izolație: E
- Temperatura de lucru: -10...70°C
- Clasă inflamabilitate: UL94V-0
- Acționare ventilator: motor DC fără perii
- Lungime cablu: 300mm
- Dimensiune conductor: 24AWG



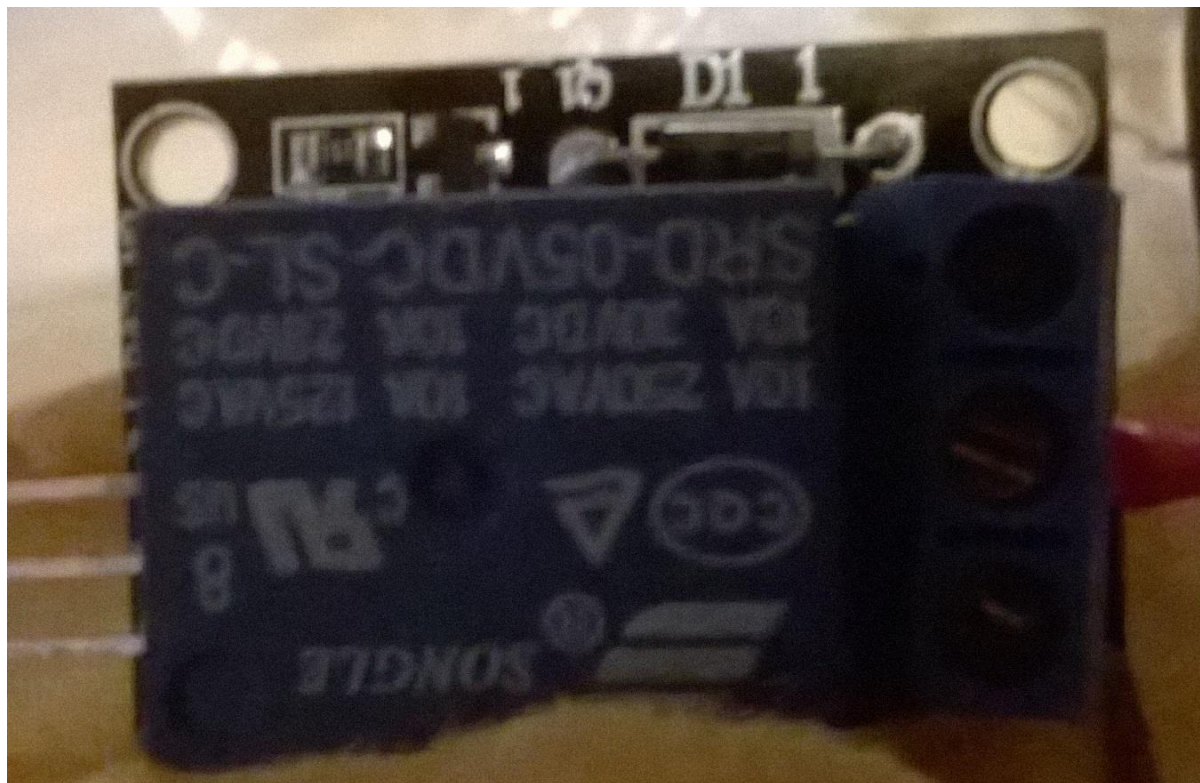


Figură 6



## Releul

Releul electromagnetic este un întrerupător acționat electric. Ca principiu de funcționare, un releu, folosește electromagnetul pentru a acționa mecanic întrerupătorul. Un releu își găsește utilizare în aplicații în care elementele trebuiesc controlate cu un semnal de mică putere, dar și alte aplicații precum controlarea mai multor elemente prin o singură comandă.



Figură 7

### 2.2.4 Mediul de programare Arduino

Mediul Arduino de dezvoltare integrat ( IDE ) este o aplicație cross-platform scrisă în Java, și este derivată de la IDE pentru limbajul Processing. Acesta include un editor de cod cu caracteristici, cum ar fi sintaxa, bretele de potrivire, și indentare automată, și este, de asemenea, capabil de întocmirea și încărcarea de programe pe placa de dezvoltare cu un singur clic. Un program sau cod scris pentru Arduino este numit o "schită ".

Programele Arduino sunt scrise în C sau C++. Arduino IDE vine cu o bibliotecă de software numită "Cablarea", ceea ce face ca multe operații comune de intrare/ieșire să fie efectuate mult mai ușor. Utilizatorii trebuie doar să definească două funcții pentru a face un program să fie executate ciclic :

- `setup ( )` : o funcție ce rulează o singură dată, la începutul unui program. Se poate folosi pentru setările inițiale.
- `loop ( )` : o funcție apelată în mod repetat.

Un prim program simplu pentru Arduino ce aprinde pur și simplu un LED. În mediul Arduino, utilizatorul ar putea să scrie un program similar cu următorul:

Integrat pin 13 LED

```
# define LED_PIN 13

void setup ( ) {

    pinMode ( LED_PIN, OUTPUT ) ; // Activați pin 13 pentru ieșire digitală

}

void loop ( ) {

    digitalWrite ( LED_PIN, HIGH ) ; // Porniți pe LED

    delay ( 1000 ) ; // Așteaptă o secundă ( 1,000 milisecunde )

    digitalWrite ( LED_PIN, LOW ) ; // Opriți LED

    delay ( 1000 ) ; // Așteaptă o secundă

}
```



Figură 8

Cele mai multe plăci Arduino au o rezistență de sarcină și un LED conectate între pinul 13 și masă; o caracteristică convenabilă pentru mai multe teste simple sau pentru debugarea programelor. Codul anterior nu ar fi compilat cu succes de un compilator standard C++, astfel atunci când utilizatorul face clic pe butonul "Upload to I / O " în IDE Arduino, o copie a codului este scrisă într-un fișier temporar, la care se adaugă un antetul funcției main în partea de jos implementarea pentru a face un program C++ valid.

Arduino IDE folosește uneltele de GNU și AVR pentru a compila programe, și programul avrdude pentru a încărca programele pe placa de dezvoltare.

Ca platforma Arduino utilizează microcontrolere Atmel, mediu de dezvoltare Atmel, AVR Studio sau mai nou Atmel Studio, pot fi, de asemenea, folosite pentru a dezvolta software-ul pentru Arduino.

## **2.3 Legătura hardware – interfața**

### **2.3.1 Windows**

Microsoft Windows este o serie de sisteme de operare cu interfață grafică dezvoltate, comercializate, și vândute de către Microsoft.

Microsoft a introdus un mediu de operare numit Windows pe 20 noiembrie 1985 ca un supliment grafic al sistemului de operare pentru MS-DOS, un răspuns la interesul tot mai mare în interfețe grafice ( GUI ). Microsoft Windows a ajuns să domine piața mondială de sisteme de operare pentru calculatorul personal cu peste 90 % cotă de piață, depășind Mac OS, care a fost introdus în 1984, sau distribuțiile Linux.

Printre principalele avantaje se numără răspândirea, suportul existent atât pentru versiunile curente cât și pentru cele mai vechi, varietatea de aplicații deja existente dar mai ales faptul că este compatibil cu procesoarele care au setul de instrucțiuni X86/X64 și modelul de PC IBM.

## 2.3.2 Limbajul de programare C++

### *Istorie*

C++ (pronunțat si plus plus ) este un limbaj de programare general. Are caracteristici imperative orientate-obiect și de programare generică, oferind în același timp, facilitățile de manipulare a memoriei la nivel scăzut.

Acesta este conceput cu precădere pentru programarea de sisteme ( de exemplu, sisteme integrate, kernel de sistem de operare ), cu performanță, eficiență și flexibilitate de utilizare fiind cerințele sale de design. C++ și-a găsit utilitatea în multe alte contexte, inclusiv aplicații desktop, servere ( de exemplu, e-commerce, căutare web, SQL ), aplicații unde sunt restricții de performanță(de exemplu, centrale telefonice, sonde spațiale ) și software-ul de divertisment, cum ar fi jocurile video.

Este un limbaj compilat, cu implementări disponibile pe majoritatea platformelor. Diferite organizații furnizează compilatorul de C++, inclusiv FSF, LLVM, Microsoft și Intel.

C++ este standardizat de către Organizația Internațională de Standardizare ( ISO ), care cel mai recent care a fost ratificat și publicat de ISO în septembrie 2014 ca ISO/IEC 14882:2014, informal cunoscut sub numele de C++ 14. Limbajul de programare a fost inițial standardizat în 1998 ca ISO / IEC 14882: 1998, care a fost modificată apoi prin C++ 03, ISO / IEC 14882 : 2003, și C++11, ISO/IEC 14882:2014. Standardul actual ( C++ 14 ) înlocuiește acestea, cu noi caracteristici și o bibliotecă standard extinse.

Înainte de standardizare ( începând cu 1989 ), C++ a fost dezvoltat de Bjarne Stroustrup de la Bell Labs, începând încă din 1979, care a dorit un limbaj flexibil și eficient ( cum ar fi C ), dar care să furnizeze și caracteristici de nivel înalt pentru organizare de program.

Multe alte limbaje de programare au fost influențate de C++, inclusiv C #, Java, și versiuni mai noi ale C ( după 1998 ).

### *Filosofie*

De-a lungul vieții C++-lui, dezvoltarea și evoluția sa a fost guvernată neoficial de un set de reguli care ce trebuiau urmate :

- Evoluția limbajului trebuie să fie condusă de problemele reale și caracteristicile sale ar trebui să fie utile imediat în programele din lumea reală.
- Fiecare caracteristică specificată în standard trebuie să poată fi implementată ( cu un mod rezonabil evident de a face acest lucru ).

- Programatorii ar trebui să aibă libertatea de a alege stilul lor de programare propriu, și acesta ar trebui să fie pe deplin sprijinit de C++.
- Existența unei caracteristici utilă este mai important decât a preveni orice abuz posibil al limbajului C++.
- Acesta ar trebui să ofere facilități pentru organizarea de programe în părți separate bine definite, și oferă facilități pentru a combina piesele dezvoltate separat (modularizare).
- Orice schimbări implicite la nivelul tipului de date nu sunt permise, ele trebuiesc făcute explicite de către programator.
- Orice caracteristici ale limbajului ce nu sunt folosite, nu trebuie plătite.
- Ar trebui să existe niciun o limbaj sub C++ ( cu excepția limbaj de asamblare ).
- C++ ar trebui să lucreze alături de alte limbaje de programare pre - existente, mai degrabă decât să fie o parte din propriul mediu de programare separată și incompatibil.
- Dacă ceea ce programatorul vrea să facă este necunoscut, să permită programatorului să specifice ce vrea să facă ( asigură un control manual ).

```

1  #include "Conex.h"
2
3  bool Conex::Connect()
4  {
5      SP = new SerialPort("\\\\.\\COM4"); // adjust as needed
6      if (SP->IsConnected())
7      {
8          printf("We're connected\n");
9          return true;
10     }
11     return false;
12 }
13 int Conex::ReadInfo(char* buffer, int dataLength, int *queueSize)
14 {
15     int readResult = 0;
16     if (SP->IsConnected())
17     {
18         //Request
19         readResult = SP->ReadData(buffer, dataLength, queueSize);
20     }
21     return readResult;
22 }
23
24 bool Conex::WriteInfo(char* writeChar, int size)
25 {
26     printf(writeChar);
27     if (SP->IsConnected())
28     {
29         bool result = SP->WriteData(writeChar, size);
30         if (result)
31             return true;
32     }
33     return false;

```

Figură 9

### 2.3.3 Interfața de programare Windows - C++

Windows API, neoficial WinAPI, este set de bază de interfețe de programare a aplicațiilor (API) disponibile în sistemele de operare Microsoft Windows. Numele Windows API se referă în mod colectiv la o serie de implementări ce există pe diferite platforme. Toate programele Windows interacționează, într-un fel sau altul cu API-ul Windows-ului.

Suportul pentru dezvoltatori este disponibil sub forma de Kit Dezvoltare Software pentru Windows ( SDK ), acesta furnizează documente și instrumente necesare pentru a construi software-ul bazat pe API-ul Windows și interfețele de Windows asociate.

Windows API ( Win32 ) se concentrează în primul rând pe limbajul de programare C, în care funcțiile sale expuse și structurile de date sunt descrise în acest. Cu toate acestea, API-ul poate fi folosit de orice compilator (de limbaj de programare sau de asamblare) capabil de manipularea ( bine definite ) structurilor de date de nivel scăzut, împreună cu convențiile de așteptare prescrise pentru apeluri de funcții. În mod similar, la implementarea funcționalității API-ului Windows au fost folosite mai multe limbaje. În ciuda faptului că limbajul C este lipsit de orice noțiune de programare orientată pe obiecte pentru Windows API, precum și pentru Windows în sine, acesta a fost uneori descris ca orientat pe obiect. Au fost, de asemenea, mai multe clase ce “înveleau” API-ul și extensii ( de la Microsoft sau alte surse ), acestea putând fi folosite în limbaje orientate pe obiecte sau care au făcut această structură mai orientată pe obiecte ( MSFC, VCL, GDI +, etc ). De exemplu, Windows 8, oferind în același timp Windows API, oferă, de asemenea, API WinRT care este pus în aplicare în C++ și este orientat pe obiect prin design.

### 2.3.4 Comunicații seriale prin portul USB

O resursă de comunicare este un dispozitiv fizic sau logic care oferă un singur flux bidirecțional, asincron de date. Porturi seriale, porturi paralele, mașini de fax și modem-uri sunt exemple de resurse de comunicații. Pentru fiecare resursă de comunicare, există un furnizor de servicii, constând dintr-o bibliotecă sau „driver”, care permite aplicațiilor să acceseze resursa.

Funcțiile de manipularea fișierelor ( CreateFile, CloseHandle, readfile, ReadFileEx, WriteFile, și WriteFileEx ) asigură interfața de bază pentru deschiderea și închiderea unui cârlig(handle) de resurse de comunicare și pentru efectuarea operațiunilor de citire și scriere.

Windows acceptă atât comunicații sincrone și asincrone( suprapuse ) pentru operațiunile asupra resurselor de comunicații seriale. Operațiuni suprapuse permite firului de așteptare(waiting thread) pentru a efectua alte sarcini în timp ce operațiunea execută în fundal. Un fir(thread) utilizează funcția ReadFile sau ReadFileEx pentru a citi dintr-o resursă de comunicare, și WriteFile sau funcția WriteFileEx pentru a scrie o resursă de comunicare. ReadFile și WriteFile pot fi efectuate sincron sau asincron. ReadFileEx și WriteFileEx pot fi efectuate numai asincron.

Un fir (thread) poate, de asemenea, scrie o resursă de comunicare prin utilizarea funcției TransmitCommChar, care inserează un caracter(specificat în funcție) înainte de orice date în așteptare din memoria tampon ( bufferul ) de ieșire. Această funcție este utilă pentru transmiterea unui semnal de mare prioritate la sistemul de recepție.

Un fir poate folosi funcția PurgeComm a se debarasa(discard) de toate caracterele din memoria tampon(buffer) de ieșire sau de intrare a dispozitivului. PurgeComm poate rezilia, de asemenea cererile de citire sau scriere, chiar dacă operațiunile nu au fost încă finalizate. În cazul în care un fir utilizează PurgeComm pentru a debarasa(discard) un tampon de ieșire, caracterele șterse nu sunt transmise. Pentru a goli bufferul de ieșire asigurând în același timp că datele în așteptare sunt transmise, un fir se poate apela funcția FlushFileBuffers ( o operație sincronă ).

Pentru a deschide un cârlig(handle) la o resursă de comunicație se folosește funcția CreateFile pentru. De exemplu, specificând COM1 deschide un cârlig la un port serial, și LPT1 deschide un cârlig la un port paralel. În cazul în care resursa specificat este în prezent utilizată de un alt proces, CreateFile nu reușește. Orice fir al procesului poate utiliza cârligul returnat de CreateFile pentru a identifica resursă în oricare dintre funcțiile ce vor opera pe acel cârlig.

Când procesul de apeluri CreateFile pentru a deschide o resursă de comunicare, se specifică următoarele atribute :

- Ce tip de acces de citire/scriere se folosește pentru resursa specificată.
- Dacă cârligul poate fi moștenit de procese copil.
- Dacă cârligul poate fi utilizat în operații sincrone sau asincrone(suprapuse).

Când procesul folosește CreateFile pentru a deschide o resursă de comunicare, acesta trebuie să specifice anumite valori pentru următorii parametri :

- Parametrul fdwShareMode trebuie să fie zero, semnifică deschiderea resursei pentru acces exclusiv.



- Parametrul `fdwCreate` trebuie să specifice steagul(flag-ul) `OPEN_EXISTING`.
- Parametrul `hTemplateFile` trebuie să fie `NULL`.

Atunci când se utilizează `CreateFile` pentru a deschide un cârlig direct la un aparat, o aplicație trebuie să utilizeze caracterele speciale "`\\.\`", pentru a identifica dispozitivul. De exemplu, pentru a deschide un cârlig unitatea A, specificați "`\\.\o :`". Pentru parametrul `lpzName` de `CreateFile`. Procesul de așteptare poate folosi cârligul în funcția `DeviceIoControl` de a trimite coduri de control la aparat.

Când funcția `CreateFile` deschide un cârlig pentru o resursă de comunicații serială, sistemul inițializează și configurează resursa conform valorilor stabilite ultima dată când resursa a fost deschisă. Conservarea setărilor anterioare permite utilizatorului să-și păstreze setările specificate printr-un modul de comandă atunci când aparatul este redeschis. Valorile moștenite de la operația deschisă anterior includ setările de configurare ale blocului de control al dispozitivului (o structura DCB), precum și valorile de timp-out folosite în operațiunile de I / O. Dacă dispozitivul nu a fost deschis, acesta este configurat cu valorile implicite de sistem.

Pentru a determina configurația inițială a unei resurse de comunicații seriale, un proces apelează funcția `GetCommState`, care umple într-o structură DCB setările curente. Pentru a modifica această configurație, un proces specifică o structură DCB într-un apel la funcția `SetCommState`.

Membrii structurii DCB specifică setările de configurare, cum ar fi rata de transfer, numărul de biți de date pe octet, și numărul de biți de stop pe octet. Alți membri DCB specifică caractere speciale și pentru a permite verificarea parității și de control a fluxului. Când un proces are nevoie pentru a modifica doar câteva dintre aceste setări de configurare, va trebui să solicite mai întâi `GetCommState` pentru a umple într-o structură DCB cu configurația curentă.

Funcția `SetCommState` reconfigurează resursa de comunicare, dar aceasta nu afectează tamponale interne intrare sau ieșire ale driverului specificat.

Funcția `ClearCommError` extrage informații despre o eventuală eroare de comunicare și raportează starea curentă a unui dispozitiv de comunicare. Funcția se apelează atunci când apare o eroare de comunicare, ea ștergând steag-ul de eroare a dispozitivului pentru a permite operații suplimentare de intrare și ieșire ( I / O ).

Funcția `ReadFile` citește date din fișierul specificat sau de la dispozitivul intrare / ieșire ( I / O ). Parametrii funcției sunt următorii

- `HANDLE hFile`: cârligul către dispozitivul de comunicație

- LPVOID lpBuffer: un pointer la memoria tampon care primește datele citite de la dispozitiv.
- DWORD nNumberOfBytesToRead: numărul maxim de baiți ce urmează a fi citați (pot corespunde cu mărimea memoriei tampon).
- LPDWORD lpNumberOfBytesRead: un pointer la o variabilă care primește numărul de baiți citați atunci când se utilizează un parametru hFile sincron. ReadFile stabilește această valoare la zero înainte de a face orice citire, inițializare sau verificare de eroare.
- LPOVERLAPPED lpOverlapped: este un pointer la o structură necesar dacă parametrul hFile a fost deschis cu FILE\_FLAG\_OVERLAPPED, în caz contrar acesta poate fi NULL.

### 2.3.5 Serverul Apache

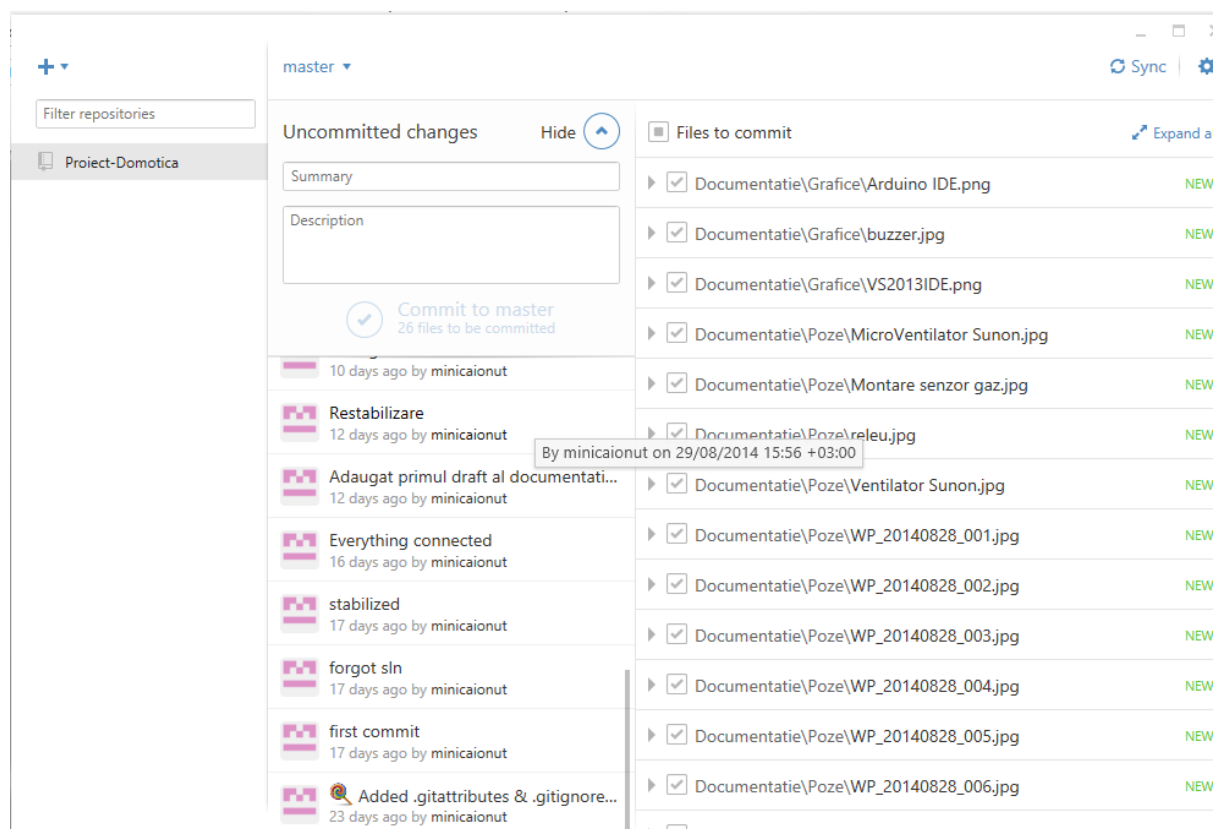
Apache HTTP Server, numit colocvial Apache, este o aplicație server Web notabilă pentru rolul cheie ce l-a jucat în creșterea inițială a World Wide Web. Inițial bazată pe serverul NCSA HTTPd, dezvoltarea Apache a început în anul 1995, după dezvoltarea codului NCSA a stagnat. Apache a preluat rapid NCSA HTTPd ca server HTTP dominant, și a rămas cel mai popular server HTTP în uz din aprilie 1996. În 2009, el a devenit primul software de tip server Web pentru a servi mai mult de 100 de milioane de site-uri web.

## 2.4 Alte aplicații semnificative

### 2.4.1 Git și GitHub

Git este un sistem de control, revizuire și management al codului sursă distribuit ( CSM ), cu un accent pe viteză, integritatea datelor, și suport pentru fluxurile de lucru non-liniare distribuite. A fost inițial proiectat și dezvoltat de către Linus Torvalds pentru dezvoltarea kernel-ului Linux, în 2005 a devenit sistemul de control al versiunii cel mai larg adoptat pentru dezvoltarea de software.

GitHub este un serviciu de găzduire Git bazat pe web care oferă toată funcționalitatea Git de gestionare a codului sursă (CSM), precum și adăugarea mai multor caracteristici proprii. Spre deosebire de Git, ceea ce este strict un instrument de linie de comandă, GitHub oferă o interfață grafică bazată pe web și desktop, precum și integrarea pe mobil. Acesta oferă, de asemenea, controlul accesului și mai multe caracteristici de colaborare, cum ar fi wiki, management de activitate, precum și de urmărire a erorilor și cereri de facilități pentru fiecare proiect.



Figură 10

### **2.4.2 Notepad++**

Notepad ++ este un editor de text pentru Windows. Aceasta diferă de la Notepad-ul preinstalat pentru Windows deoarece Notepad ++ permite editarea cu tab-uri, care permite lucrul cu mai multe fișiere deschise într-o singură fereastră plus multe alte beneficii. Notepad++ deschide fișiere mari semnificativ mai rapid decât Windows Notepad.

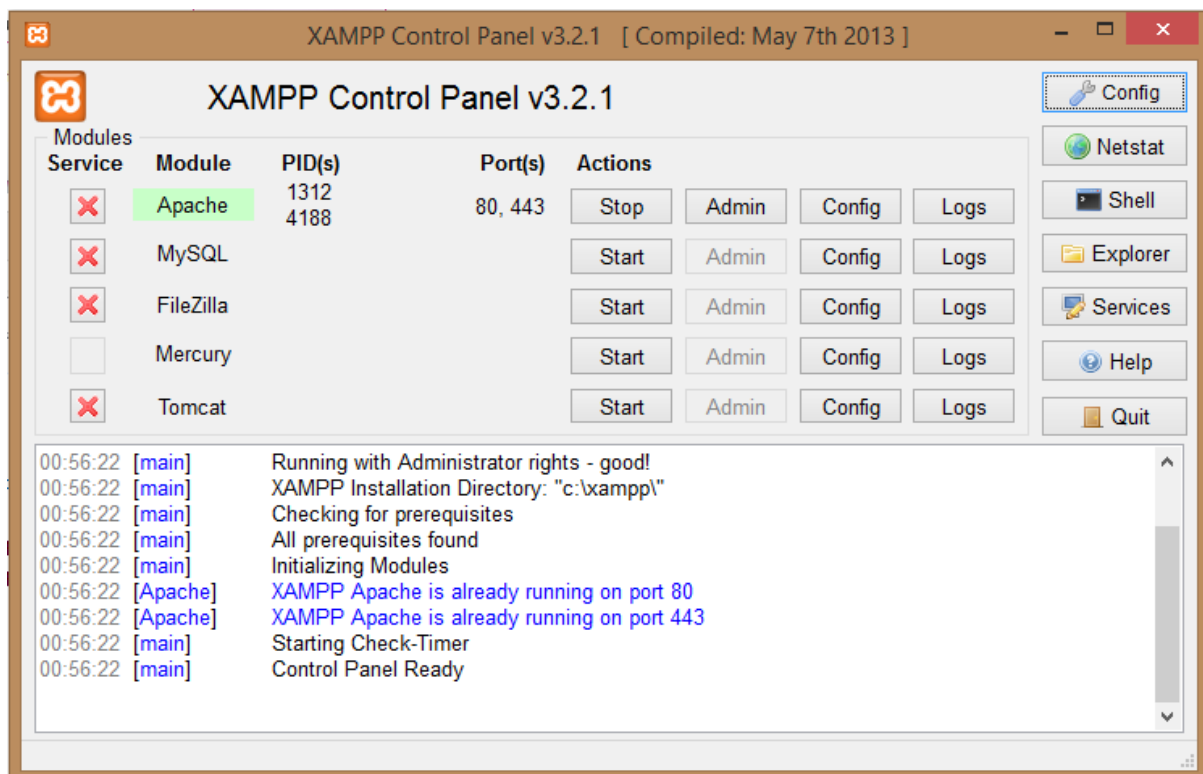
Notepad ++ este distribuit ca software liber. Proiectul a fost găzduit pe SourceForge.net, de unde a fost descărcat de peste 28 de milioane de ori și a câștigat de două ori Choice Award SourceForge comunitare pentru cel mai bun Developer Tool. Proiectul a fost găzduit pe Tuxfamily din iunie 2010 Notepad ++ folosește componenta editor Scintilla.

### **2.4.3 XAMPP**

XAMPP este un software deschis( “open source”) , valabil pe mai multe platforme( „cross-platform” ) ce formează o soluție de server web, constând în principal din baza de date(MySQL), server Apache HTTP Server, și interpretori(interpreter) pentru scripturi scrise în PHP și Perl( limbaje de programare folosite în mediul Web).

Oficial, designerii XAMPP au considerat că pachetul va fi folosit doar ca un instrument de dezvoltare, pentru a permite designerilor site-ul și programatorilor pentru a testa munca pe computerele lor, fără nici un acces la internet. Pentru a face acest lucru cât mai ușor posibil, mai multe caracteristici importante de securitate sunt dezactivate în mod implicit. În practică, însă, XAMPP este folosit uneori pentru a servi efectiv pagini web pe World Wide Web. Un instrument special este oferit pentru a proteja prin parolă cele mai importante părți ale pachetului.

XAMPP oferă de asemenea suport pentru crearea și manipularea bazelor de date în MySQL și SQLite, printre altele.



Figură 11

### 3 DESCRIEREA APLICAȚIEI PRACTICE

#### 3.1 Arhitectură generală

*Privire ansamblu*



Aplicația are rolul de a măsura cantitatea de gaz și temperatura din cele două incinte, de a acționa spre corectarea acestora, cât și semnalizarea defecțiunii.

Dacă valorile măsurate depășesc anumite valorile maxime admise setate apriori, sistemul intră în starea de avarie.

Atunci când sistemul intră în starea de avarie se activează actuatorii corespunzători incintei/incintelor periculoase, corectând parametrii măsurați și avertizând persoanele aflate în zonă.

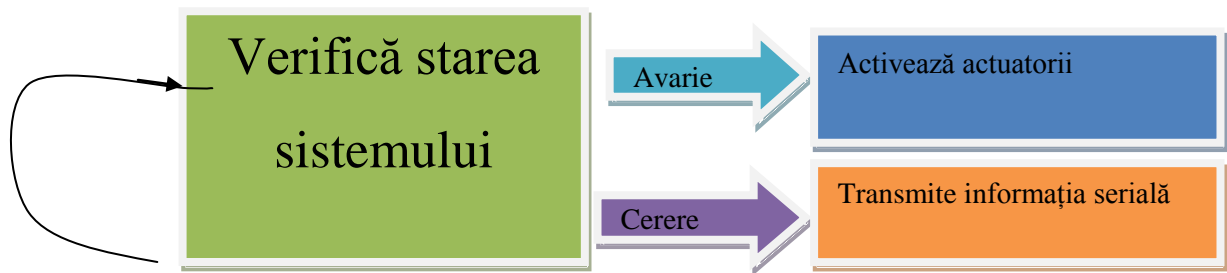
Comunicare dintre calculatorul personal și Arduino Uno V3 se face prin portul serial. Programul instalat pe placa de dezvoltare eșantionează ciclic valorile de la nivelul senzorilor și stabilește dacă sunt periculoase sau nu. După stabilirea stării sistemului și eventuala acționare spre corectare, dacă programul Windows a cerut informații despre valorile senzorilor, acesta le transmite prin portul serial.

Programul Windows prelucrează informațiile primite și apoi le scrie într-un fișier ce va fi folosit pentru a transmite informațiile mai departe spre interfața web.

La nivelul interfeței HTML se afișează valorile extrase de la senzori și se prezintă eventuala stare de avarie prin "aprinderea" LED-ului corespunzător și simularea funcționării ventilatorului respectiv.

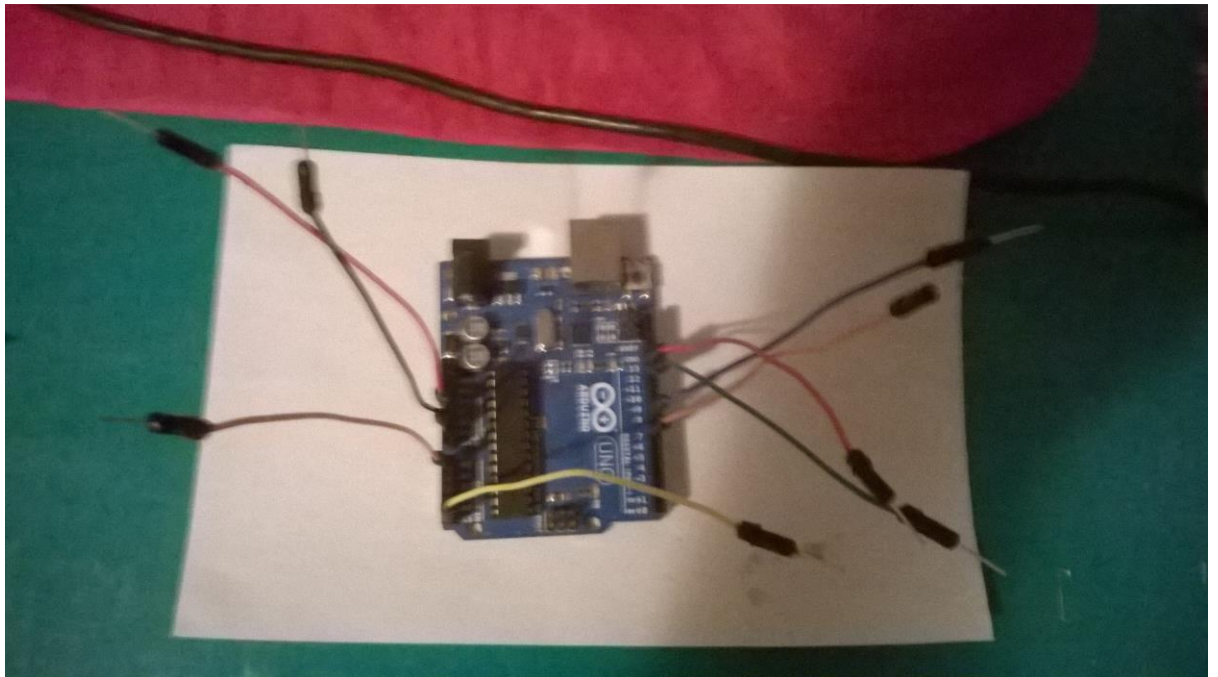
Am împărțit proiectul după cum urmează:

### 3.2 Aplicația Arduino



La nivelul aplicației Arduino se efectueaza trei roluri cheie:

1. Evaluarea stării sistemului
2. Acționarea în caz de avarie
3. Trimiterea informațiilor prin portul serial.



**Figură 12**

În cazul în care sunt depistate anumite valori ce nu corespund, sistemul intră în starea de avarie.

Există două stări posibile de avarii, și anume starea de avarie datorată nivelului de gaz periculos din încăperea și cea datorată temperaturii excesive, cele două stări au comportament similar, diferența fiind ventilatorul pornit.

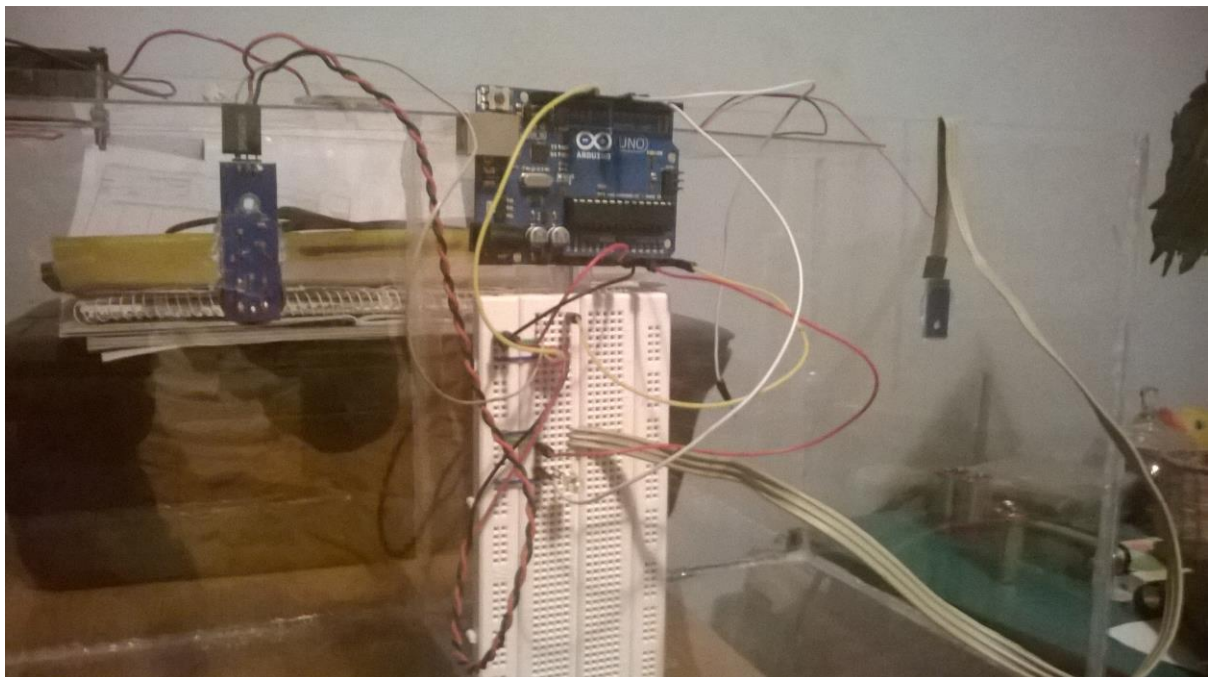
În cazul în care în incinta unu se măsoară un nivel periculos de gaz, microcontrolerul setează pinul corespunzător pe valoare logică „adevărat” cuplând releul ce pornește ventilatorul evacuând amestecul de gaze.

În cazul în care în incinta doi se măsoară un nivel al temperaturii excesive, microcontrolerul setează pinul corespunzător pe valoare logică „adevărat” cuplând microventilatorul ce pornește evacuând aerul încălzit din camera doi.

Proiectul presupune că există posibilitatea ca "muncitorii" să existe în zona de testare. Aceștia la rândul lor trebuie să fie avertizați de starea de avarie în care se află sistemul, putând fi în pericol, astfel dacă sistemele de siguranță nu ar putea, spre exemplu, evacua gazul suficient de repede din încăperea unu, să se poată deplasa într-un loc sigur. Folosind cele două metode și "muncitorii" cu dizabilități vor putea fi avertizați.

Ca și metode de avertizare se vor folosi:

- un LED ce avertizează printr-un semnal luminos
- un buzzer piezo electric ce semnalizează starea de pericol printr-un semnal sonor.



**Figură 13**

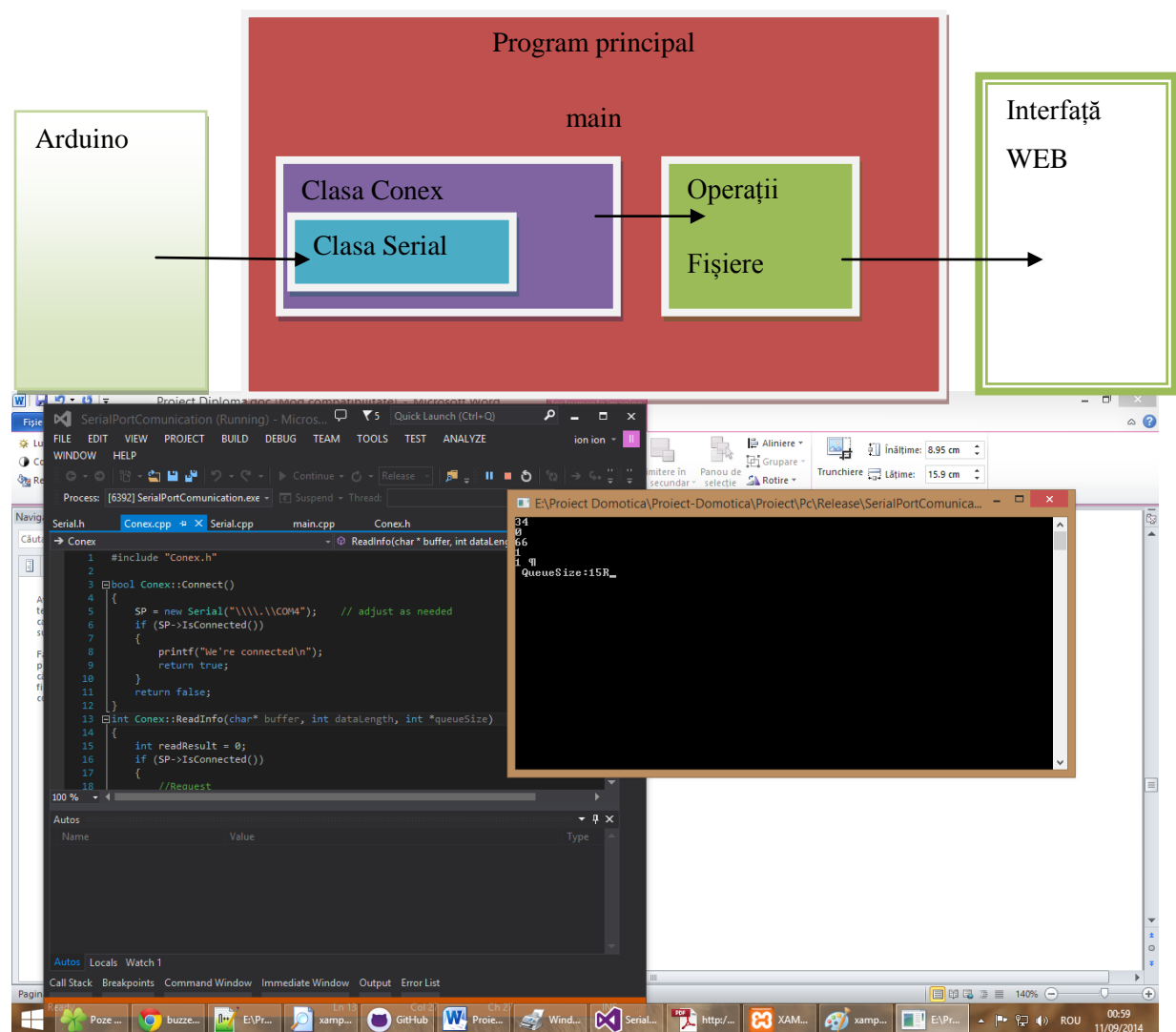


### 3.3 Aplicația Windows

La nivelul aplicației Windows se face citirea datelor de la Arduino, și scrierea acestora în fișierul ce va fi folosit de către aplicația PHP/HTML.

La intervale regulate aplicația Windows face cereri către Arduino pentru informații. O dată primite, informațiile nu sunt prelucrate, ci transmise direct către aplicația PHP/HTML.

Aplicația Windows este structurată astfel:



Figură 14

Clasa “Serial” declarată în fișierul Serial.h și definită în fișierul Serial.cpp abstractizează comunicația cu portul serial și conține apelurile la funcțiile conținute în Windows API.

- CreateFileA
- ReadFile
- WriteFile
- SetCommState
- GetCommState
- ClearCommError
- GetLastError
- ClearCommError
- CloseHandle

Declarația clasei arată în felul următor:

```
class Serial
{
private:
    //Serial comm handler
    HANDLE hSerial;

    //Connection status
    bool connected;

    //Get various information about the connection
    COMSTAT status;

    //Keep track of last error
    DWORD errors;
```

```

        int queueSize;

public:

    Serial(char *portName);

    ~Serial();

    int ReadData(char *buffer, unsigned int nbChar, int*
queueSize);

    bool WriteData(char *buffer, unsigned int nbChar);

    bool IsConnected();

    int GetQueueSize(){ return queueSize; }

};

```

Pentru simplificarea mai mult a operațiilor cu clasa Serial, clasa Conex este următorul nivel de abstractizare. Operațiile de citire si conectare sunt făcute prin clasa Serial.

```

class Conex

{
    Serial* SP;

    bool SendInfoRequest();

public:

    bool Connect();

    int ReadInfo(char* buffer, int dataLength, int *queueSize);

    bool WriteInfo(char* writeChar, int size);

};

```

În programul principal:

Declararea și demararea conexiunii se face astfel:

```
Conex c;  
  
c.Connect();
```

Pentru a reduce consumul de resurse, Arduino nu scrie nimic la portul serial decât dacă este cerut.

```
c.WriteInfo(READ_COMMAND, sizeof(READ_COMMAND - 1));
```

Citirea datelor de la portul serial se face astfel:

```
int readBytes = c.ReadInfo(incomingData, dataLength, &queueSize);
```

Declareare și deschiderea fișierului :

```
FILE *fInput;  
  
errno_t erInput = fopen_s(&fInput, INPUT_FILENAME, "w");
```

Scrierea în fișier:

```
fwrite(incomingData, sizeof(char), readBytes, fInput);
```

### **3.4 Aplicația PHP/HTML**

La nivelul aplicației PHP/HTML se citesc datele din fișierul generat de aplicația Windows informațiile sunt prelucrate și pe baza lor se generează o pagină HTML ce va fi afișată pe dispozitivul ce face cererea via internet.

Administratorul sistemului, de exemplu, poate oricând să observe nivelul parametrilor considerați periculoși și starea, de avarie sau nu, a sistemului în general de pe orice dispozitiv conectat la internet, fie el mobil sau nu, ce are instalat un browser web compatibil HTML.

Practic orice dispozitiv ce este conectat la internet și poate afișa pagina Google, poate fi folosit pentru accesul interfeței.

Deschiderea și citirea fișierului se face folosind funcțiile fopen și fgets astfel :

```

//Aici se citeste din fisier

while($file_handle == FALSE )

{

    @$file_handle = fopen("E:\Proiect Domotica\Proiect-
Domotica\Proiect\Pc\SerialPortCommunication\Input", "r");

}

while (!feof($file_handle)) {

    $valoriFisier[1] = fgets($file_handle);

    .

    .

    .

    $valoriFisier[5] = fgets($file_handle);

}

```

Funcția CreateChambers creează tabelul HTML și îl populează cu valorile din fișier.

```

function CreateChambers(

    $tempValue, $tempCooler, $gasValue,







    $gasCooler, $gasLed

)

```

Interfața arată în felul următor:

# Status

<b>Gaz</b>  Valoare Gaz: 23 	<b>Alerta Gaz</b>   
<b>Temperatura</b>  Valoare Temperatura:30 	<b>Alerta Temperatura</b>   

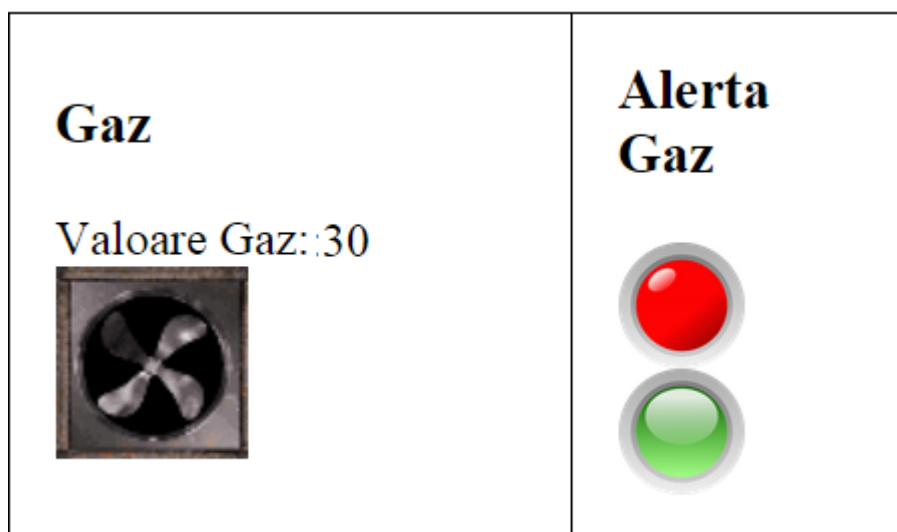
Figură 15

Să presupunem că temperatura ar crește peste o valoare admisă, atunci se va semnaliza starea de avarie, și va porni animația ventilatorului.



Figură 16







Un alt caz ipotetic ar fi atunci când valoarea gazului ar trece peste o valoare admisă, atunci pagina corespunzătoare încăperi 1 va arăta astfel:



Figură 17

În cazul în care ambele situații de avarii sunt semnalate interfața va arăta astfel:

# Status

<b>Gaz</b>  Valoare Gaz: 23 	<b>Alerta Gaz</b>   
<b>Temperatura</b>  Valoare Temperatura:30 	<b>Alerta Temperatura</b>   

Figură 18



## **4 CONCLUZII**

Suportul existent (forumuri, documentație etc.) pentru programarea sau implementarea diverselor proiecte face dezvoltarea unui proiect nou mult mai simplă, mai rapidă și mai ieftină.

Aplicația poate fi adaptată unui mediu industrial simplu. Schimbările necesare pentru adaptarea proceselor cu o complexitate medie sau mare nu au fost studiate pe deplin.

În concluzie, există posibilitatea folosirii unui Arduino/Atmega328, pentru aplicații industriale simple.

## 5 BIBLIOGRAFIE

Bibliografia va fi ordonată alfabetic după eticheta fiecărei element (de ex. DOOM05 în lista de mai jos este o etichetă). Etichetele materialelor consultate vor fi formate folosind:

- primele litere ale primului autor urmate de cele două cifre semnificative ale anului apariției materialului, sau
- dintr-un acronim popular al lucrării respective, urmat din nou de cele două cifre semnificative ale anului apariției.

[DOOM05] – *Dicționarul ortografic, ortoepic și morfologic al limbii române*, Editura Univers Enciclopedic, București, 2005

[BAN09] –Massimo Banzi, *Getting Started with Arduino*, Editura O'Reilly, 2009

[PET98] –Charles Petzold, *Programming Windows, 5<sup>th</sup> Edition* , Editura Microsoft Press, 1998

[STR13] –Bjarne Stroustrup, *The C++ Programming Language (4th Edition)*, Editura Addison-Wesley, 2013

[TAT02] – Kevin Tatroe, Peter MacIntyre, Rasmus Lerdorf, *Programming PHP*, Editura O'Reilly, 2002

## 6 REFERINȚE WEB

- [ARD14] - <http://en.wikipedia.org/wiki/Arduino>
- [APC14] - <http://en.wikipedia.org/wiki/Apache>
- [ARR14] - <http://Arduino.cc/en/Reference/HomePage>
- [AUT14] - <http://en.wikipedia.org/wiki/Automation>
- [COM14] - [http://msdn.microsoft.com/en-us/library/windows/desktop/aa363196\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/aa363196(v=vs.85).aspx)
- [CPP14] - <http://en.wikipedia.org/wiki/C%2B%2B>
- [GIT14] - <http://en.wikipedia.org/wiki/GitHub>
- [HTML14] - <http://en.wikipedia.org/wiki/HTML>
- [HTML14] - [http://en.wikipedia.org/wiki/HTML\\_element](http://en.wikipedia.org/wiki/HTML_element)
- [LM50] - <http://www.ti.com/lit/ds/symlink/lm50.pdf>
- [MQ2] - [http://www.pololu.com/file/download/MQ2.pdf?file\\_id=0J309](http://www.pololu.com/file/download/MQ2.pdf?file_id=0J309)
- [NOT14] - <http://en.wikipedia.org/wiki/Notepad++>
- [PHP14] - <http://en.wikipedia.org/wiki/PHP>
- [REL14] - <http://en.wikipedia.org/wiki/Relay>
- [SEN14] - <http://en.wikipedia.org/wiki/Sensor>
- [SER14] - [http://en.wikipedia.org/wiki/Server\\_\(computing\)](http://en.wikipedia.org/wiki/Server_(computing))
- [SERM14] - [http://en.wikipedia.org/wiki/Client%E2%80%93server\\_model](http://en.wikipedia.org/wiki/Client%E2%80%93server_model)
- [WIN14] - [http://en.wikipedia.org/wiki/Microsoft\\_Windows](http://en.wikipedia.org/wiki/Microsoft_Windows)
- [W3S14] - <http://www.w3schools.com/>
- [XAM14] - <http://en.wikipedia.org/wiki/XAMPP>

[Alm08] – Pedro de Almeida, Patrik Fuhrer, Documentation Guidelines for Diploma and Master Thesis, Universitatea din Fribourg, Elveția, 2008, disponibil on-line la adresa <http://diuf.unifr.ch/drupal/softeng/teaching/guidelines>

[Olt07] – Th. Olteanu, C. Albu, *Ghid pentru redactarea lucrării de diplomă sau a disertației de masterat*, Universitatea Română de Arte și Științe „Gheorghe Cristea”, 2007, disponibil via web la adresa [http://www.ugc.ro/tpl/GHID\\_REDACTARE\\_DIPLOMA\\_LICENTA.pdf](http://www.ugc.ro/tpl/GHID_REDACTARE_DIPLOMA_LICENTA.pdf)

## A. CODUL SURSĂ

### A.1 Aplicație Arduino:

```
#define READ_COMMAND 'R'

#define LIGHT_SENSOR 1

#define TEMP_SENSOR 0
#define TEMP_MAX 35
#define TEMP_COOLER 7

#define GAS_SENSOR 5
#define GAS_BUZZER 5
#define GAS_LED 13
#define GAS_COOLER 8
#define GAS_MAX 60

// #define DEBUG

// the setup routine runs once when you press reset:
void setup() {
    // init serial comm
    pinMode(GAS_LED, OUTPUT);

    pinMode(GAS_COOLER, OUTPUT);
    pinMode(TEMP_COOLER, OUTPUT);
    digitalWrite(GAS_LED, LOW);
    Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {

    static int i=0;
    static bool wasGasDangerousState = false;
    static bool wasTempDangerousState = false;
    String lBuffer;

    // check for dangerous gas situations
    int gas = GasPercentage();
    if (gas >= GAS_MAX)
    {
        StartLed(GAS_LED);
        StartCooler(GAS_COOLER);
        StartBuzzer(GAS_BUZZER);
        wasGasDangerousState = true;
    }
}
```

```

}
else if (wasGasDangerousState)
{
    StopLed(GAS_LED);
    StopCooler(GAS_COOLER);
    StopBuzzer(GAS_BUZZER);
    wasGasDangerousState = false;
}
//check for dangerous temperature situations
int temp = TempInCelsius(5);
if(temp >= TEMP_MAX)
{
    StartCooler(TEMP_COOLER);
    wasTempDangerousState = true;
}
else if (wasTempDangerousState)
{
    StopCooler(TEMP_COOLER);
    wasTempDangerousState = false;
}
//
int lum = LuminosityPercentage();

if (Serial.available() && Serial.read() == 'R')
{
    Serial.flush();
    //Temperature chamber
    lBuffer += temp;
    lBuffer += "\n";
    if (wasTempDangerousState)
    {
        lBuffer += "1 \n";
    }
    else
    {
        lBuffer += "0 \n";
    }
    //Gas Chamber
    lBuffer += gas;
    lBuffer += "\n";
    if (wasGasDangerousState)
    {
        lBuffer += "1 \n";
        lBuffer += "1 \n";
    }
    else
    {
        lBuffer += "0 \n";
        lBuffer += "0 \n";
    }
}

```

```

    }

    int len = lBuffer.length();
    char* charBuff = new char[len + 1];
    charBuff[len] = 0;
    lBuffer.toCharArray(charBuff, len);
    Serial.write((uint8_t*)charBuff, len);
    Serial.flush();
    delete charBuff;
}
delay(1200);
}

int LuminosityPercentage()
{
    int analog_reading = analogRead(LIGHT_SENSOR);
    return map(analog_reading, 0, 1023, 0, 100);
}
int GasPercentage()
{
    int analog_reading = analogRead(GAS_SENSOR);
    return map(analog_reading, 0, 1023, 0, 100);
}
int TempInCelsius(int count)
{
    float temperaturaMediata = 0;
    float sumaTemperatura;
    for (int i = 0; i < count; i++) {
        int reading = analogRead(TEMP_SENSOR);
        float voltage = reading * 5.0;
        voltage /= 1024.0;
        float temperatureCelsius = (voltage - 0.5) * 100 ;
        sumaTemperatura = sumaTemperatura +
temperatureCelsius;
    }
    return sumaTemperatura / (float)count;
}

void StartCooler(int coolerNumber)
{
    digitalWrite(coolerNumber, HIGH);
}
void StopCooler(int coolerNumber)
{
    digitalWrite(coolerNumber, LOW);
}
void StartLed(int led)
{

```

```
    digitalWrite(led, HIGH);  
}  
void StopLed(int led)  
{  
    digitalWrite(led, LOW);  
}  
void StartBuzzer(int buzzer)  
{  
    digitalWrite(buzzer, HIGH);  
}  
void StopBuzzer(int buzzer)  
{  
    digitalWrite(buzzer, LOW);  
}
```



## A.2 Aplicația Windows

### A.2.1 Fișierul *Serial.h*

```
#ifndef SERIALCLASS_H_INCLUDED
#define SERIALCLASS_H_INCLUDED

#define ARDUINO_WAIT_TIME 2000

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

class Serial
{
private:
    //Serial comm handler
    HANDLE hSerial;
    //Connection status
    bool connected;
    //Get various information about the connection
    COMSTAT status;
    //Keep track of last error
    DWORD errors;
    int queuesize;
public:
    //Initialize Serial communication with the given COM
    port
    Serial(char *portName);
    //Close the connection
    //NOTA: for some reason you can't connect again
    before exiting
    //the program and running it again
    ~Serial();
    int ReadData(char *buffer, unsigned int nbChar, int*
    queueSize);
    //Writes data from a buffer through the Serial
    connection
    //return true on success.
    bool WriteData(char *buffer, unsigned int nbChar);
    //Check if we are actually connected
    bool IsConnected();

    int GetQueueSize(){ return queuesize; }

};

#endif // SERIALCLASS_H_INCLUDED
```

### A.2.2 Fişierul *Serial.cpp*

```
#include "Serial.h"

Serial::Serial(char *portName)
{
    //We're not yet connected
    this->connected = false;

    //Try to connect to the given port through CreateFile
    this->hSerial = CreateFileA(portName,
        GENERIC_READ | GENERIC_WRITE,
        0,
        NULL,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL,
        NULL);

    //Check if the connection was successful
    if (this->hSerial == INVALID_HANDLE_VALUE)
    {
        //If not successful display an Error
        if (GetLastError() == ERROR_FILE_NOT_FOUND){

            //Print Error if necessary
            printf("ERROR: Handle was not attached.
Reason: %s not available.\n", portName);

        }
        else
        {
            printf("ERROR!!! %d", GetLastError());
        }
    }
    else
    {
        //If connected we try to set the comm
parameters
        DCB dcbSerialParams = { 0 };

        //Try to get the current
        if (!GetCommState(this->hSerial,
&dcbSerialParams))
        {
            //If impossible, show an error
            printf("failed to get current serial
parameters!");
        }
        else
```

```

        {
            //Define serial connection parameters for
the arduino board
            dcbSerialParams.BaudRate = CBR_9600;
            dcbSerialParams.ByteSize = 8;
            dcbSerialParams.StopBits = ONESTOPBIT;
            dcbSerialParams.Parity = NOPARITY;

            //Set the parameters and check for their
proper application
            if (!SetCommState(hSerial, &dcbSerialParams))
            {
                printf("ALERT: Could not set Serial Port
parameters");
            }
            else
            {
                //If everything went fine we're connected
                this->connected = true;
                queuesize = this->queuesize;
                //We wait 2s as the arduino board will be
reseting
                Sleep(ARDUINO_WAIT_TIME);
            }
        }
    }

Serial::~Serial()
{
    //Check if we are connected before trying to disconnect
    if (this->connected)
    {
        //We're no longer connected
        this->connected = false;
        //Close the serial handler
        CloseHandle(this->hSerial);
    }
}

void Replace(char* buffer, int size)
{
    for (int i = 0; i < size; ++i)
        if (buffer[i] == 0)
            buffer[i] = 20;
}

int Serial::ReadData(char *buffer, unsigned int nbChar, int*
queueSize)
{
    //Number of bytes we'll have read
    DWORD bytesRead;
    //Number of bytes we'll really ask to read
    unsigned int toRead;

```

```

        //Use the ClearCommError function to get status info on
the Serial port
        ClearCommError(this->hSerial, &this->errors, &this-
>status);
        *queueSize = this->status.cbInQue;
        //Check if there is something to read
        if (this->status.cbInQue > 0)
        {
            //If there is we check if there is enough data to
read the required number
            //of characters, if not we'll read only the
available characters to prevent
            //locking of the application.
            toRead = this->status.cbInQue;
            if (toRead > nbChar)
            {
                printf("Warning BUFFER OVERFLOW!!!\n");
                toRead = nbChar;
            }
            //Try to read the require number of chars, and
return the number of read bytes on success
            if (ReadFile(this->hSerial, buffer, toRead,
&bytesRead, NULL) && bytesRead != 0)
            {
                char* link = buffer + (char)bytesRead + toRead
+ 1;

                *link = 0;
                Replace(buffer, bytesRead);
                return bytesRead;
            }
            else
            {
                printf("Error! Could not read from COM port!
To read size:%d", toRead);
            }
        }

        //If nothing has been read, or that an error was detected
return -1
        return 0;
    }

bool Serial::WriteData(char *buffer, unsigned int nbChar)
{
    DWORD bytesSend;

    //Try to write the buffer on the Serial port
    if (!WriteFile(this->hSerial, (void *)buffer, nbChar,
&bytesSend, 0))
    {

```

```

        //In case it don't work get comm error and return
false      ClearCommError(this->hSerial, &this->errors, &this-
>status);
        return false;
    }
    else
        return true;
}

bool Serial::IsConnected()
{
    //Simply return the connection status
    return this->connected;}

```

### A.2.3 Fişierul Conex.h

```
#ifndef __CONEX__
#define __CONEX__

#include "Serial.h"

#define SLEEP_TIME 2000
class Conex
{
    Serial* SP;
    bool SendInfoRequest();
public:
    bool Connect();

    int ReadInfo(char* buffer, int dataLength, int
*queueSize);

    bool WriteInfo(char* writeChar, int size);
};
#endif
```

#### A.2.4 Fişierul Conex.cpp

```
#include "Conex.h"

bool Conex::Connect()
{
    SP = new Serial("\\\\.\\COM4");    // adjust as
needed
    if (SP->IsConnected())
    {
        printf("We're connected\n");
        return true;
    }
    return false;
}

int Conex::ReadInfo(char* buffer, int dataLength, int
*queueSize)
{
    int readResult = 0;
    if (SP->IsConnected())
    {
        //Request
        readResult = SP->ReadData(buffer, dataLength,
queueSize);
    }
    return readResult;
}

bool Conex::WriteInfo(char* writeChar, int size)
{
    printf(writeChar);
    if (SP->IsConnected())
    {
        bool result = SP->WriteData(writeChar, size);
        if (result)
            return true;
    }
    return false;
}
```

### A.2.5 Fişierul main.cpp

```
#include <stdio.h>
#include <tchar.h>
#include <string>
#include <time.h>
#include "Conex.h"
#define LENGTH 500
#define READ_COMMAND "R"
#define WRITE_COMMAND "W"
#define INPUT_FILENAME "Input"
#define SANITY_CHECK 14

int _tmain(int argc, _TCHAR* argv[])
{
    Conex c;
    c.Connect();

    while (true)
    {
        char incomingData[LENGTH];
        int dataLength = LENGTH - 1;
        incomingData[LENGTH-1] = 0;

        int queueSize;
        REQUEST:
        c.WriteInfo(READ_COMMAND, sizeof(READ_COMMAND - 1));
        Sleep(500);
        READBYTES:
        int readBytes = c.ReadInfo(incomingData, dataLength,
        &queueSize);
        if (readBytes)
        {
            if (readBytes > SANITY_CHECK)
            {
                FILE *fInput;
                errno_t erInput = fopen_s(&fInput,
                INPUT_FILENAME, "w");
                if (erInput)
                {
                    printf("\nFailed to open file!Error:
                %d!", erInput);

                    exit(1);
                }
                fwrite(incomingData, sizeof(char),
                readBytes, fInput);
                system("cls");
                printf(incomingData);
                printf("\n QueueSize:%d", queueSize);
            }
        }
    }
}
```



```

        }
    else
    {
        printf("\n QueueSize:%d", queueSize);
        printf("\n Size:%d", readBytes);
        goto REQUEST;
    }
}
else
{
    Sleep(1000);
    goto READBYTES;
}
_fcloseall();
}
return 0;
}

```

## B. SITE-UL WEB AL PROIECTULUI

Fisierul index.php:

```
<html>
<head>
<title>Interfata Casa</title>
<meta http-equiv="refresh" content="2"><!--...-->
<style>
table,th,td
{
    border:1px solid black;
    border-collapse:collapse;
}
th, td
{
    padding:15px;
}

<?php
function CreateChambers(
$tempValue, $tempCooler, $gasValue,
$gasCooler, $gasLed
){
    $stableInit = " <table border=\"1\"
style=\"width:300px\"> \n <tr> ";
    $stableEnd = " </tr>\n </table>";
    $fanActivePath = "<img src=\"Fan.gif\" \">>";
    $fanStopPath = "<img src=\"StoppedFan.png\" \">>";
    $redLedOn = "<img src=\"RedLedOn.png\" height=\"42\"
width=\"42\" \">>";
    $redLedOff = "<img src=\"RedLedOff.png\"
height=\"42\" width=\"42\" \">>";
    $greenLedOn = "<img src=\"GreenLedOn.png\"
height=\"42\" width=\"42\" \">>";
    $greenLedOff = "<img src=\"GreenLedOff.png\"
height=\"42\" width=\"42\" \">>";
    //GasChamber creation
    $gasChamber = "<td> " .
    "<h3>Gaz</h3> " .
    " Valoare Gaz: " . $gasValue;
    if ($gasCooler)
    {
        $gasChamber = $gasChamber . $fanActivePath .
"</td>";
    }
    else
```

```

        {
            $gasChamber = $gasChamber . $fanStopPath .
"</td>";
        }

        $gasAlert = "<td>" . "<h3>Alerta Gaz</h3> " ;
        if ($gasLed)
        {
            $gasAlert = $gasAlert . $redLedOn .
$greenLedOff;
        }
        else{
            $gasAlert = $gasAlert . $redLedOff .
$greenLedOn;
        }

        //Temp chamber creation
        $tempChamber = "<td>" . "<h3>Temperatura</h3> " .
"Valoare Temperatura:" . $tempValue;
        if($tempCooler)
        {
            $tempChamber = $tempChamber . $fanActivePath .
"</td>";
        }
        else
        {
            $tempChamber = $tempChamber . $fanStopPath .
"</td>";
        }
        $tempAlert = "<td>" . "<h3>Alerta Temperatura</h3>
" ;
        if ($tempCooler)
        {
            $tempAlert = $tempAlert . $redLedOn . "<br>"
. $greenLedOff;
        }
        else{
            $tempAlert = $tempAlert . $redLedOff . "<br>"
. $greenLedOn;
        }

        $gasTable = $tableInit . $gasChamber . $gasAlert .
$tableEnd;
        $tempTable = $tableInit . $tempChamber . $tempAlert
. $tableEnd;

        return $gasTable . $tempTable;

```

```

    }
    //Scrierea titlului
    echo '</style> <H1>Status</H1>';

    $valoriFisier = array(5);
    @$file_handle = FALSE ;
    //Aici se citeste din fisier
    while($file_handle == FALSE )
    {
        @$file_handle = fopen("E:\Proiect
Domotica\Proiect-
Domotica\Proiect\Pc\SerialPortCommunication\Input", "r");
    }
    while (!feof($file_handle)) {
        $valoriFisier[1] = fgets($file_handle);
        $valoriFisier[2] = fgets($file_handle);
        $valoriFisier[3] = fgets($file_handle);
        $valoriFisier[4] = fgets($file_handle);
        $valoriFisier[5] = fgets($file_handle);
    }
    fclose($file_handle);
    //Apelarea functiei ce generează pagina html
    echo CreateChambers(
        intval($valoriFisier[1]) ,intval($valoriFisier[2])
        ,intval($valoriFisier[3]),
        intval($valoriFisier[4]) ,intval($valoriFisier[5]));

    ?>

</body>
</html>

```

## **C. CD / DVD**

Autorul atașează în această anexă obligatorie, versiunea electronică a aplicației, a acestei lucrări, precum și prezentarea finală a tezei.



# INDEX

## *B*

Bibliografie..... 44

## *C*

CUPRINSUL .....xii

## *L*

LISTA FIGURILOR.....xiv

## *R*

Referințe web..... 45