Cristian-Florin DONE - □ WatchHouse □

Autorul poate fi contactat la adresa: **Login pentru adresa**

Introducere

Pe măsură ce înaintăm în timp tehnologia avansează și odată cu acesta și dispozitivele pe care noi le folosim. Tendința este ca toate dispozitivele de lângă noi să devină unele smart, dotate cu cât mai multe funcționalități care să ne ușureze viața de zi cu zi. În ultima perioadă s-au dezvoltat din ce în ce mai multe dispozitive de tip Smart House.

Scopul acestui proiect este dezvoltarea unui modul pentru Smart House care să permită monitorizarea anumitor parametrii precum temperatura, luminozitatea și umiditatea. De asemenea modulul ne va permite controlarea instantanee sau după un anumit timp prin intermediul unor timere a unui corp de iluminat din casă . Toate aceste funcționalități vor fi disponibile de pe un smartphone cu sistem de operare Android, conectat la internet sau la rețeaua locală.

Descriere generală

Schema bloc

Hardware Design

Listă de piese

Număr piese	Nume piesă	Observații
1	Placă de bază cu ATMega324A-PU	Alimentare la 3.3V
1	Modul LCD SPI de 1.44'' (128×128)	Compatibil cu Nokia 5110
1	Senzor de lumină ambientală TEMT6000	Senzor analogic
1	Senzor de temperatură și umiditate DHT22	Senzor digital
1	Modul Ethernet ENC28J60	
1	Releu 5V	

1	Translator de nivel	Pentru releu
1	Rezistor 4.7kΩ	Rezistor de pull-up pentru senzorul DHT22

Scheme electrice

Microcontroler-ul ATmega324A-PU se va conecta cu componentele menționate anterior după următoarele scheme electrice.

LCD

LCD-ul folosit pentru acest proiect este unul compatibil cu cel de pe telefonul Nokia 5110, similar cu cel de la laborator. Comunicația cu acest LCD este realizată prin SPI, pinii folosiți fiind CS, SDA(MOSI din SPI), SCK. LCD-ul oferă încă doi pini LED și RESET pentru controlul backlight-ului respectiv resetarea dispozitivului.

Senzorul de lumină ambientală TEMT6000

Senzorul de lumină folosit(TEMT6000) este unul analogic motiv pentru care acesta este conectat la canalul 0 al convertorului analog-digital(ADC0) de pe pinul PA0 al microcontroler-ului nostru.

Senzorul de temperatură și umiditate DHT22

Senzorul de temperatură și umiditate(DHT22) folosit în acest proiect este unul digital, pinul pentru date putând fi conectat la oricare port. În mod arbitrar am ales ca acesta să fie conectat la pinul PC0 al microcontroler-ului. Menționez că pentru a funcționa, senzorul are nevoie de un rezistor de pull-up pentru linia de date.

Modulul ethernet ENC28J60

Modulul folosit pentru comunicația prin internet este unul bazat pe microcontroler-ul ENC28J60, comunicația cu acesta realizându-se prin SPI. Cei 4 pini folosiți în comunicație sunt: CS, SI(MOSI din SPI), SCK și SO(MISO din SPI).

Releu

Ultima componentă ce rămâne a fi discutată este releul. Întrucât nu am reușit să găsesc un releu comandat la 3.3V am decis să folosesc unul de 5V împreună cu un translator de nivel pentru a face trecerea de la tensiunea de alimentare a microcontroler-ului de 3.3V la 5V necesari funcționării releului. Releul va fi controlat de microcontroler prin intermediul pinului PC1 și va fi folosit ca switch pentru un corp de iluminat.

Software Design

ATmega324A

Firmware-ul aplicației se împarte pe mai multe componente

LCD

Pentru afișarea textului dorit pe LCD am folosit codul din laboratorul 5, display-ul folosit fiind similar celui folosit la laborator. Există totuși câteva modificări aduse codului însă sunt doar pentru schimbarea pinilor folosiți.

Codul pentru display se găsește în arhivă, în folder-ul **avr/lcd**.

Senzorul de lumină ambientală TEMT6000

Senzorul de lumină ambientală este unul analog iar pentru citirea acestuia am folosit convertorul analog-digital(ADC) împreună cu sistemul de întreruperi din interiorul microcontroler-ului nostru. Pentru obținerea valorilor de la senzor am realizat o mică bibliotecă ce se poate găsi în directorul **avr/light**.

Prima metodă realizată din bibliotecă este **light_init()**. Acesta este folosită pentru a activa ADC-ul(**ADEN** = 1), pentru a selecta tensiunea de referința(**REFS[1:0]** = 1 deci se va folosi AVCC = 3.3V), pentru a activa întreruperea la terminarea conversiei(**ADIE** = 1), pentru a selecta rata de

eșantionare(**ADPS[2:0]** = 5 deci se va folosi 32), pentru a selecta pinul pentru citire(**DDRA0** = 0), pentru a selecta canalul folosit(**MUS[4:0]** = LIGHT_CHANNEL) respectiv pentru a iniția prima citire a senzorului.

```
void light_init()
{
    DDRA &= ~(1 << PA0);
    ADMUX = (1 << REFS0) | (LIGHT_CHANNEL << MUX0);
    ADCSRA = (1 << ADEN) | (1 << ADIE) | (1 << ADSC) | (5 << ADPS0);

}
```

Pentru a inițializa citirea senzorului de lumină se folosește functia **light_init_read()**. Acesta selectează canalul și inițializează citirea senzorului prin ADC.

```
// Initializare citire senzor
void light_init_read()
{
    // Setare citire senzor LIGHT_CHANNEL
    ADMUX = (ADMUX & ~(0x1f << MUX0)) | (LIGHT_CHANNEL << MUX0);

    // Start conversion ADC
    ADCSRA |= (1 << ADSC);
}
```

În momentul în care ADC-ul a terminat de citit valoarea, se generează întreruperea **ADC**, iar în rutina de tratare a întreruperii se stochează valoarea într-o variabilă.

```
static int light_val;

ISR(ADC_vect)
{
    light_val = ADC;
}
```

Pentru interogarea ultimei valori citire se folosește funcția **light_get()**.

```
int light_get() {
    return light_val;
}
```

Senzorul de umiditate și temperatură DHT22

Pentru obținerea temperaturii și a umidității am creat un wrapper peste biblioteca avr-dhtxx.

Întrucât valorile întoarse de bibliotecă sunt reprezentate ca unsigned int și reprezintă valoarea umidității/temperaturii înmulțită cu DHTXX_MUL(10) am preferat să realizez un wrapper care să

întoarcă chiar valorile reale ca float. Menționez că temperatura întoarsă este reprezentată în grade Celsius iar umiditatea în procente. Codul din wrapper este următorul:

```
static int temp, humid;

// Initializare citire senzor
void dht22_init_read()
{
    dhtxxread( DHTXX_DHT22, &DHT22_PORT, &DHT22_DDR,
        &DHT22_PIN, DHT22_MASK, &temp, &humid );
}

// Obtinerea temperaurii in grade Celsius
float dht22_get_temperature()
{
    return temp/DHTXX_MUL;
}

// Obtinerea umidiatii in procente
float dht22_get_humidity()
{
    return humid/DHTXX_MUL;
}
```

Codul complet se găsește în folder-ul **avr/dht22** din arhiva cu codul sursă.

Releu

Pentru comandarea releului tot ce trebuie să facem este să scriem un bit într-un registru. Pentru a abstractiza operația am scris o mică bibliotecă. Ea se găsește în directorul **avr/relay** și oferă următoarele funcționalități: inițializare, pornire/oprire releu și verificare dacă releul este pornit. Codul folosit este următorul:

```
// Initializare
void relay_init()
{
    DDR_RELAY |= (1 << PIN_RELAY);
    PORT_RELAY |= (1 << PIN_RELAY);
}

// Intoarce 0 daca releul nu este activat sau o valoare diferita de 0
// in cazul in care releu este dezactivat
int relay_is_on()
{
    return (PORT_RELAY & (1 << PIN_RELAY)) == 0;
}
```

```
// Activare releu
void relay_turn_on()
{
    PORT_RELAY &= ~(1 << PIN_RELAY);
}

// Dezactivare releu
void relay_turn_off()
{
    PORT_RELAY |= (1 << PIN_RELAY);
}
```

Modulul ethernet ENC28J60

Transmiterea de pachete prin internet se realizează prin intermediul modulului ethernet ENC28J60, cu ajutorul bibliotecii avr-enc28j60. Aceasta se găsește în directorul **avr/enc28j60**. Pentru realizarea comunicației cu smartphone-ul am ales să realizez un server web, pornind de la exemplul din bibliotecă.

Inițializarea modului presupune setarea adresei MAC respectiv a IP-ului static și se realizează în funcția **ethernet_init()** din fișierul **avr/main.c**.

```
// Initializare ethernet
void ethernet_init()
{
    CLKPR = (1<<CLKPCE);
    CLKPR = 0;
    _delay_loop_1(50);
    ENC28J60_Init(mymac);
    ENC28J60_ClkOut(2);
    _delay_loop_1(50);
    ENC28J60_PhyWrite(PHLCON,0x0476);
    _delay_loop_1(50);
    init_network(mymac,myip,mywwwport);
}
```

O dată la 100ms se face polling pentru a vedea dacă s-au primit date pentru serverul web, prin intermediul metodei **ethernet_pool()**, iar în cazul în care s-au primit date, se procesează cererea http prin intermediul funcției **webserver(char *url)** ce primește ca unic parametru resursa cerută, asociată cererii. Ambele funcții se găsesc în fișierul **avr/main.c**.

Întrucât nu ne dorim ca oricine să aibă acces la dispozitivul nostru, folosirea oricărei funcționalități impune cunoașterea parolei asociate dispozitivului. Orice cerere HTTP trebuie să fie de forma

```
GET /parola/comanda HTTP/1.1
```

pentru a fi acceptată de server. Altfel spus accesarea server-ului se face prin-un url de forma

http://ip/parola/comanda, unde **comanda** poate lua următoarele forme:

- **info** - pentru informații despre senzori și timere
- **relay_on** - pentru activarea releului
- **relay_off** - pentru dezactivarea releului
- **add_command/cmd/timp** - pentru adăugarea unui timer care pornește releul(în cazul în care **cmd** = 1) respectiv îl oprește(în cazul în care **cmd** = 0) peste **timp** secunde
- **remove_command/id** - pentru ștergerea timer-ului identificat prin **id**

Fiecare din comenzile menționate anterior au ca rezultat întoarcerea unui pagini web al cărui conținut este un JSON cu următoarea formă:

```
{"light": 97, "temp": 25.0, "humid": 63.0, "light_status": false, "timers": {}}
```

Câmpurile din JSON sunt următoarele:

- **light** - valoarea luminozității(un întreg din intervalul [0,1023])
- **temp** - temperatură în grade Celsius reprezentată ca float
- **humid** - umiditatea relativă măsurată în procente ca float în intervalul [0,100]
- **light_status** - starea luminii(**true** daca lumina este pornită și **false** în caz contrar)
- **timers** - informații despre timere(valoarea este un alt JSON ce are drept câmpuri numere naturale reprezentând identificatorul **id** al timer-ului iar valorile sunt alte structuri de tip JSON)

Câmpurile dintr-un JSON asociat unui timer sunt:

- **turn_on** - **true** dacă timer-ul are ca scop pornirea luminii și **false** dacă timer-ul are ca scop oprirea luminii
- **time_left** - numărul de secunde până la expirarea timer-ului

Descriere generală

Dispozitivul procesează o dată la 100ms valorile obținute de la senzori și le afișează afișează pe LCD. În cazul în care apar pachete în acel interval acestea sunt procesate.

Valorile sunt preluate de la senzori o dată la 100ms pentru senzorul de lumină ambientală, respectiv o dată la 1s în cazul senzorului de umiditate și temperatură. O dată pe secundă sunt verificate și timerele(comenzile). Un timer este considerat activ dacă timpul până la expirare este mai mare ca 0. În cazul în care există timere(comenzi) active, se decrementează timpul asociat iar dacă acestea ajung la 0 se execută comanda asociată(pornire/oprire lumină). Toate aceste evenimente se realizează cu ajutorul timer-ului 1 asociat microcontroler-ului, ce generează o întreruperea la fiecare 100ms. Citirea senzorilor și prelucrarea timerelor(comenzilor) se realizează în întreruperea asociată timer-ului 1.

Timer-ul folosește prescaler-ul de 64(**CS1[2:0]** = 3), modul CTC cu top OCR1A(**WGM1[3:0]** = 4), generează întrerupere(**OCIE1A** = 1), fiind executat o dată la 100ms(**OCR1A** = 0x493e). Codul pentru inițializarea acestui timer este următorul:

```
// Initializare timer - 100ms
void timer_init()
```

```
{
    TCCR1A = 0;
    TCCR1B = (3 << CS10) | (1 << WGM12);
    TIMSK1 = (1 << OCIE1A);
    OCR1A = 0x493E; // 100ms
}
```

Android

În ceea ce privește aplicația pentru smartphone, aceasta poate fi instalată pe orice dispozitiv mobil cu sistem de operare Android, versiunea 4.0 sau mai mare. Interfața este separată în trei părți(fragmente) diferite: General, Timers și Settings.

Utilizatorul are opțiunea să seteze ip-ul, port-ul respcetiv parola dispozitivului și intervalul de timp la care să se trimită cereri de informații către device prin intermediul interfeței Settings. Aceste setări sunt persistente, fiind memorate în sistemul de fișiere, într-o locație specifică aplicației.

În interfața general, utilizatorul poate vizualiza valorile senzorilor, poate aprinde/stinge lumina respectiv să vizualizeze timerele.

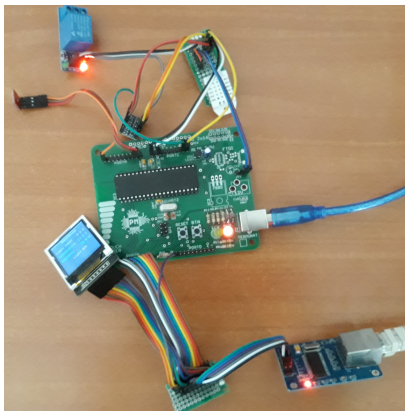
În interfața timers există posibilitatea de a vizualiza/șterge/adăuga timere. În cazul in care utilizatorul optează să adauge timere, este pus să aleagă momentul de timp la care să se activeze comanda și ce efect va avea(pornire/oprire lumină). Întrucât conexiunea cu dispozitivul(ATmega324A) poate să eșueze, aplicația reține exect momentul în care se dorește activarea comenzii și calculează precis peste câte secunde se va activa comanda, la fiecare încercare de a comunica cu microcontroler-ul nostru. În cazul în care utilizatorul a decis adăugarea unei comenzi care să se activeze peste un interval de timp dar nu s-a putut realiza conexiunea în timp util(a trecut intervalul de timp până la activare), în momentul în care dispozitivul mobil reușește să se conecteze la server, comanda va fi direct de a opri/porni lumina, fără să mai adauge un timer.

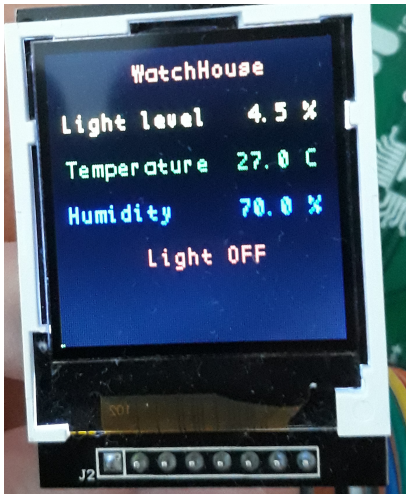
Interfața cu utilizatorul este realizată printr-un Activity ce schimbă fragmentul afișat în funcție de alegerile utilizatorului. La fiecare secundă verifică dacă există o comandă ce trebuie să fie rulată și nu există o altă conexiune cu server-ul. În caz afirmativ se realizează o conexiune la server pentru efectuarea comenzii dorite(poate să fie doar actualizarea informațiilor) iar în momentul când se primește răspunsul se actualizează conținutul fragmentului curent dacă este cazul(General/Timers).

Codul pentru aplicația android se găsește în directorul **android**.

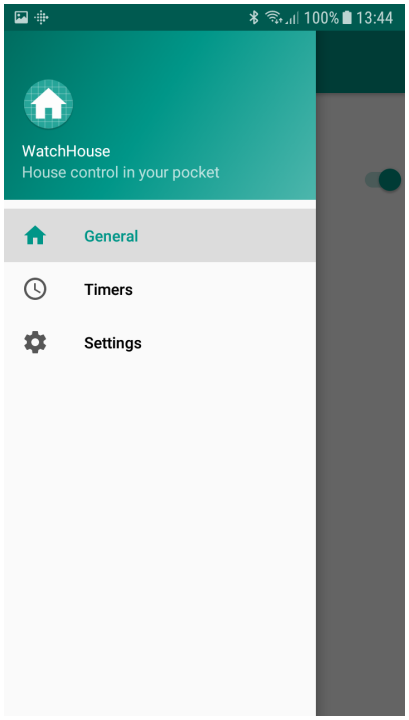
Rezultate Obţinute

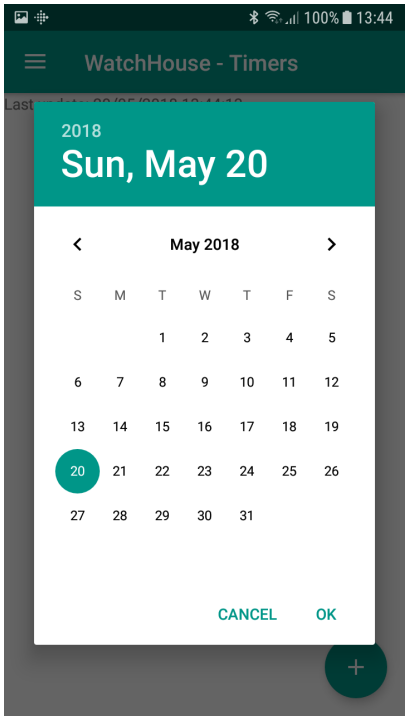
ATmega324A




WatchHouse
Light level 4.5 %
Temperature 27.0 C
Humidity 70.0 %
Light OFF
J2

Android


100% 13:44
WatchHouse
House control in your pocket
General
Timers
Settings



Concluzii

Acest proiect reprezintă un modul pentru Smart House ce ne permite să obținem temperatura, luminozitatea respectiv umiditatea din casă și totodată să controlăm un corp de iluminat la orice moment de timp. Funcția de temporizare poate fi folosită atât pentru a ne asigura că nu adormim și uităm lumina aprinsă cât și pentru a ne trezi de dimineața.

Proiectul poate fi îmbunătățit în viitor prin adăugarea de notificări pe telefon în cazul în care uităm lumina aprinsă, controlul mai multor corpuri de iluminat, sau chiar pornirea automată a corpurilor de iluminat dacă luminozitatea este sub o anumită limită și există o persoană în casă.

Download

Cod sursă

Jurnal

21 Mai 2018 - Finalizare wiki

20 Mai 2018 - Finalizare proiect

22 Aprilie 2018 - Adăugare descriere generală și listă de componente

6 Mai 2018 - Realizare schemă electrică

Bibliografie/Resurse

- Datasheet ATmega324A
- Biblioteca LCD din laborator
- Datasheet DHT22
- Biblioteca avr-dhtxx
- Biblioteca avr-enc28j60
- Android Developer
- Documentația în format PDF