# Sistem pentru notificarea pacinetului.

# Tipul de medicamente, numărul lor, setare de alarme

# Descriere proiect

Proiectul are ca scop notificarea pacientului cu privire la medicamentele pe care acesta trebuie sa le administreze. Utilizator are posibilitatea de a seta in interfata grafica o serie de alarme (ziua, lunia, ora, minut, medicament, cantitate) care vor fi inregistrate in baza de date. Cu ajutorul modulului wifi ne conectam la un router care are si rol de server de DHCP si ne este astfel asignata o adresa ip. Protocolul prin care se realizeaza comunicarea este HTTP. La nivelul protocolului se executa metoda de GET. Se construieste header-ul de GET, iar server-ul va raspunde cu continulul dorit, si anume inregistrarile din baza de date pe care le stocam intr-o variabila globala, de fiecare data cand se face acest request.

Pe ecran se vor afisa data calendaristica si ora, iar in momentul in care este timpul ca o alarma sa sune, buzzer-ul ca suna, iar pe ecran se afiseaza medicamentul si cantitatea de administrat pana in momentul in care alarma este oprita prin apasarea unui buton.

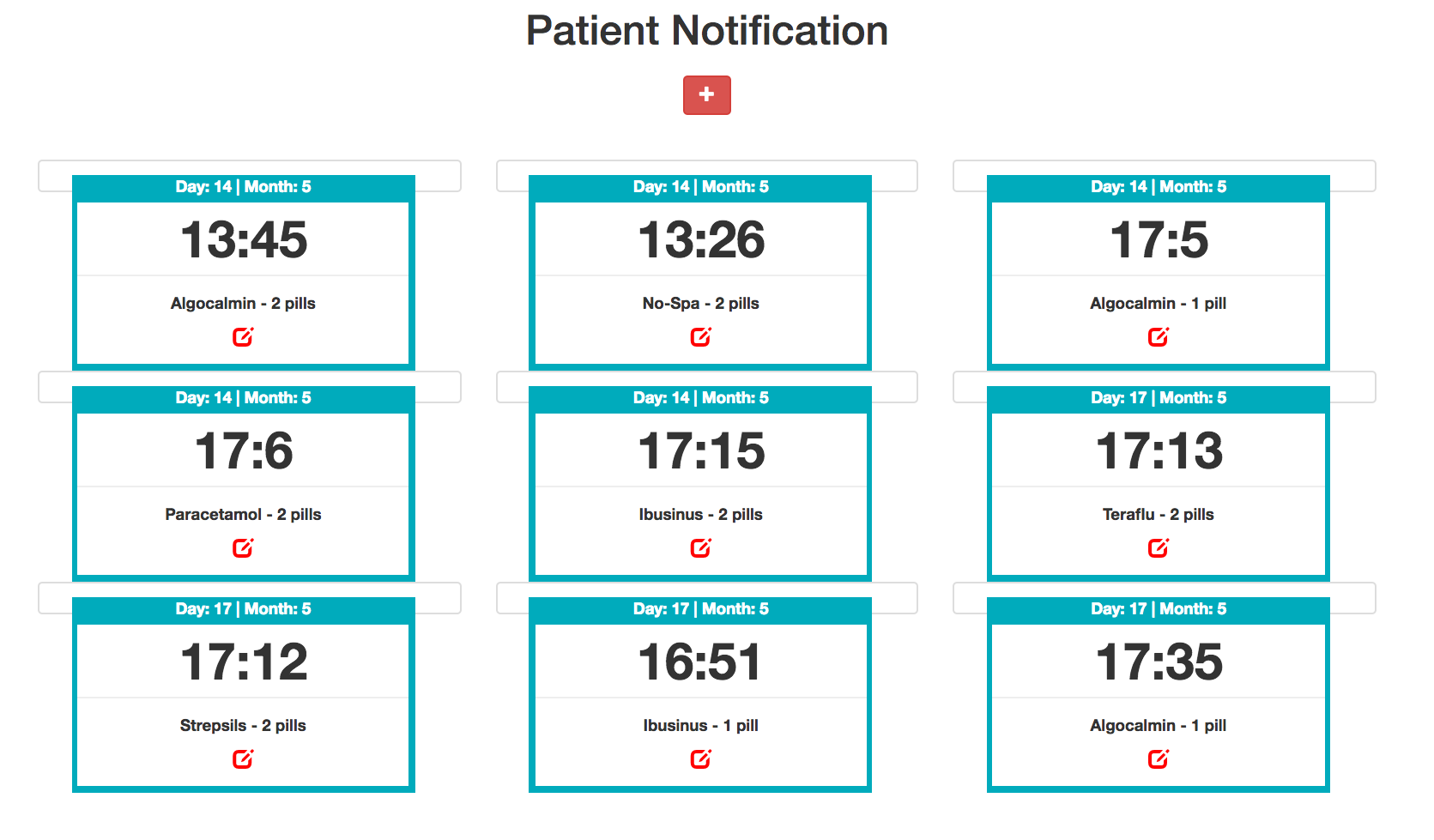
# Componente folosite

* Arduino Mega
* Modul wifi Adafruit cc3000
* Modul ceas in timp real DS1307
* LCD 16x2
* Potentiometru Stereo 10k
* Buton cu retinere
* Buzzer
* Led-uri

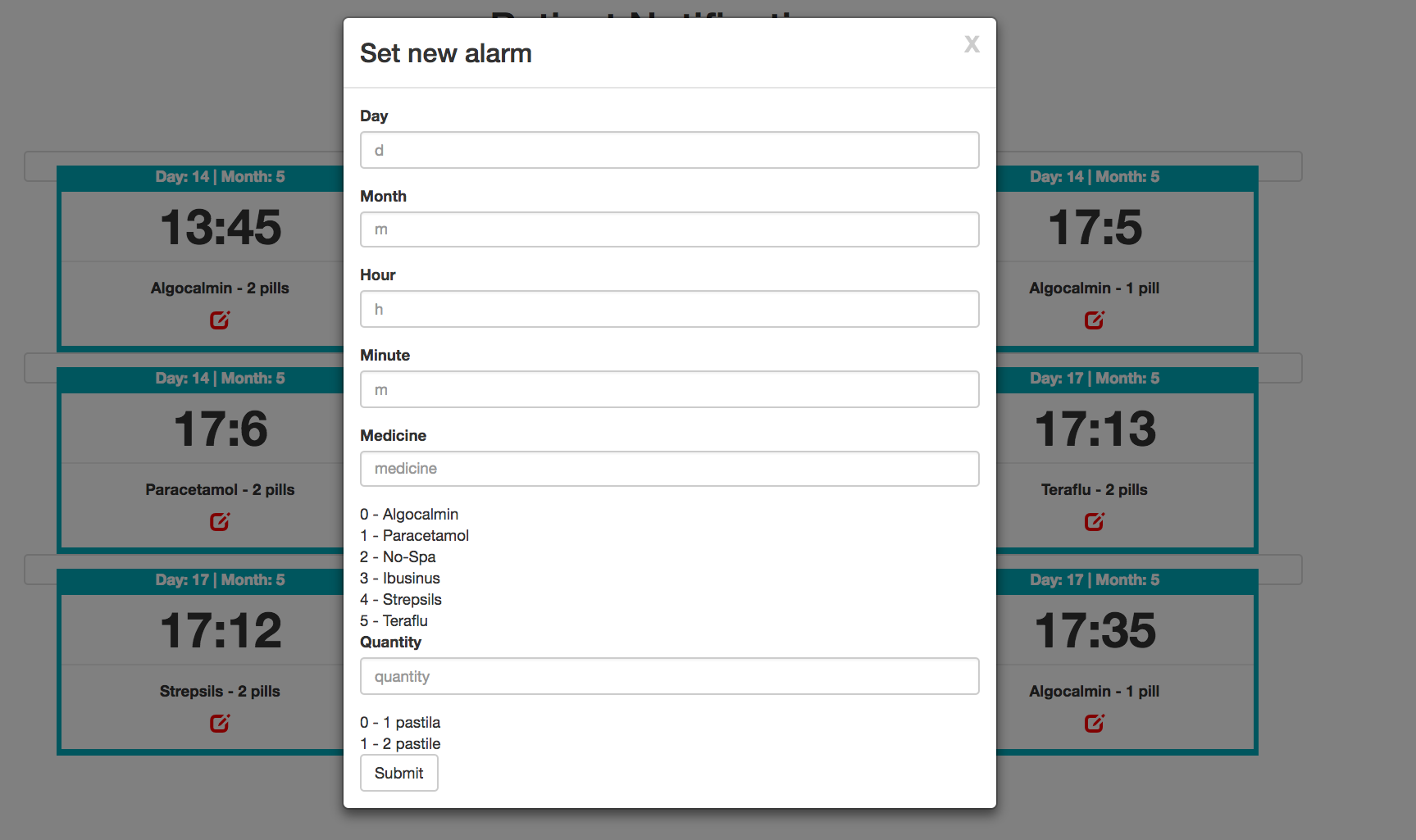




# Interfata cu utilizatorul



Pe apasarea butonului plus se poate adauga o alarma noua:



# Cod Arduino

#include <Wire.h>

#include "RTClib.h"

#include <LiquidCrystal.h>

#include <Adafruit\_CC3000.h>

#include <ccspi.h>

#include <SPI.h>

#include <string.h>

#include "utility/debug.h"

// These are the interrupt and control pins

#define ADAFRUIT\_CC3000\_IRQ 3 // MUST be an interrupt pin!

// These can be any two pins

#define ADAFRUIT\_CC3000\_VBAT 5

#define ADAFRUIT\_CC3000\_CS 10

// Use hardware SPI for the remaining pins

// On an UNO, SCK = 13, MISO = 12, and MOSI = 11

Adafruit\_CC3000 cc3000 = Adafruit\_CC3000(ADAFRUIT\_CC3000\_CS, ADAFRUIT\_CC3000\_IRQ, ADAFRUIT\_CC3000\_VBAT,

SPI\_CLOCK\_DIVIDER); // you can change this clock speed but DI

//#define WLAN\_SSID "Vlad" // cannot be longer than 32 characters!

//#define WLAN\_PASS "vladmanole"

#define WLAN\_SSID "Vlad" // cannot be longer than 32 characters!

#define WLAN\_PASS "brah1234"

// Security can be WLAN\_SEC\_UNSEC, WLAN\_SEC\_WEP, WLAN\_SEC\_WPA or WLAN\_SEC\_WPA2

#define WLAN\_SECURITY WLAN\_SEC\_WPA2

Adafruit\_CC3000\_Client www; //defines how we will refer to the cc3000 connection object

// What page to grab!

#define WEBSITE "192.168.43.56" //NOTE: CC3000 doesn't seem to like the default localhost address of 127.0.0.1 - need to enter actual IP 192.168.0.109

#define WEBPAGE "/SMP\_test/json\_v4.php"

#define MAXLINES 10

LiquidCrystal lcd(12, NULL, 11, 9,8,7,6);

//LiquidCrystal\_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

//RTC\_Millis rtc;

RTC\_DS1307 rtc;

const int buzzerPin = 4;

const int ledPin = 1; //in loc de 6

const int buttonPin = 2; //in loc de 8

//shift register

const int latchPin = 17;

const int clockPin = 18;

const int dataPin = 16;

//this is the counter that will be used for uploading data

long pollCounter = 1;

//interval at which to poll base station

//takes a value in seconds; add more multipliers if you want a more convenient unit

long pollInterval = 100L \* 1000L; //converts from millis; initial default value is to poll every 10 sec

long lastPoll; //stores the last time we polled

boolean pollFlag = false; //tells the sketch whether to poll or not

//declare a variable to hold a numeric IP address

//can be overridden below if you use lookup

//uint32\_t ip = (192L << 24) | (168L<<16) | (43<<8) | 56;

uint32\_t ip = (192L << 24) | (168L<<16) | (43<<8) | 56;

// Setting Buzzer mode to False

boolean buzzer\_mode = true;

// For LED

int ledState = LOW;

long previousMillis = 0;

long interval = 100; // Interval at which LED blinks

int alarms[MAXLINES][4];

int button\_state;

int counter[MAXLINES];

String medicine[6] = {"Algocalmin","Paracetamol", "No-Spa", "Ibusinus", "Strepsils", "Teraflu"};

String quantity[2] = {"1 pastila" , "2 pastile"};

int line = 0;

int openBracketPositions[MAXLINES];

int closedBracketPositions[MAXLINES];

int index1 = 0;

int index2 = 0;

void displayMedicine(int i){

//Serial.print("DisplayMedicine");

lcd.clear();

lcd.setCursor(3,0);

Serial.print(medicine[alarms[i][2]]);

lcd.print(medicine[alarms[i][2]]);

lcd.setCursor(4,1);

lcd.print(quantity[alarms[i][3]]);

delay(3000);

}

void alarm\_ON(int button\_state, int i){

DateTime now = rtc.now();

Serial.print(" \n\nSTARE BUTON alarmON: ");

Serial.print(button\_state);

if (button\_state) {buzzer\_mode = false; counter[i] = counter[i] + 1;}

Serial.print("\nCounter: ");

Serial.print(counter[i]);

Serial.print("\nIndex:");

Serial.print(i);

if (buzzer\_mode){

digitalWrite(ledPin, HIGH);

tone(buzzerPin,50000);

displayMedicine(i);

}

// If alarm is off

if (buzzer\_mode == false) {

// No tone & LED off

Serial.print("--- NU SE MAI CANTA ---\n");

noTone(buzzerPin);

shiftOut(dataPin, clockPin, MSBFIRST, 0);

digitalWrite(ledPin, LOW);

}

}

void isTime2(){

DateTime now = rtc.now();

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

if((counter[i] == 0) && (alarms[i][0] == now.hour()) && (alarms[i][1] == now.minute())){

button\_state = digitalRead(buttonPin);

alarm\_ON(button\_state,i);

}

else {buzzer\_mode = true;}

}

}

void setup() {

Serial.begin(9600);

Wire.begin();

rtc.begin();

lcd.begin (16,2);

pinMode(latchPin, OUTPUT);

pinMode(dataPin, OUTPUT);

pinMode(clockPin, OUTPUT);

digitalWrite(latchPin, LOW);

shiftOut(dataPin, clockPin, MSBFIRST, 0);

digitalWrite(latchPin, HIGH);

if (! rtc.isrunning()) {

Serial.println("RTC is NOT running!");

// following line sets the RTC to the date & time this sketch was compiled

rtc.adjust(DateTime(\_\_DATE\_\_, \_\_TIME\_\_));

}

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

counter[i] = 0;

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

alarms[i][j] = 60;

}

}

/\* Initialise the module \*/

//Serial.println(F("\nInitialising the CC3000 ..."));

if (!cc3000.begin())

{

Serial.println(F("Unable to initialise the CC3000! Check your wiring?"));

while(1);

}

if (!cc3000.connectToAP(WLAN\_SSID, WLAN\_PASS, WLAN\_SECURITY)) {

Serial.println(F("Failed!"));

while(1);

}

Serial.println(F("Connected!"));

/\* Wait for DHCP to complete \*/

Serial.println(F("Request DHCP"));

while (!cc3000.checkDHCP())

{

delay(100); // ToDo: Insert a DHCP timeout!

}

//The Following are our output

pinMode(ledPin,OUTPUT);

pinMode(buzzerPin,OUTPUT);

//Button is our Input

pinMode(buttonPin, INPUT);

}

void displayTime(){

//Serial.print("\n---- SE AFISEAZA TIMPU ---\n");

byte second, minute, hour, day, month, year, dayOfWeek;

DateTime now = rtc.now();

year = now.year();

month = now.month();

day = now.day();

hour = now.hour();

minute = now.minute();

second = now.second();

dayOfWeek = now.dayOfTheWeek();

//LCD

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(day);

lcd.print("/");

lcd.print(month);

lcd.print("/");

lcd.print(now.year());

lcd.print(" ");

switch(dayOfWeek){

case 1:

lcd.print("Mon");

break;

case 2:

lcd.print("Tue");

break;

case 3:

lcd.print("Wed");

break;

case 4:

lcd.print("Thu");

break;

case 5:

lcd.print("Fri");

break;

case 6:

lcd.print("Sat");

break;

case 7:

lcd.print("Sun");

break;

}

lcd.setCursor(4, 1);

lcd.print(hour);

lcd.print(":");

if(now.minute() < 10){

lcd.print("0");

}

lcd.print(minute);

lcd.print(":");

if(now.second() < 10){

lcd.print("0");

}

lcd.print(now.second());

}

void loop () {

displayTime();

//Serial.print("Inainte de leduri");

digitalWrite(latchPin, LOW);

shiftOut(dataPin, clockPin, MSBFIRST, 255);

digitalWrite(latchPin, HIGH);

delay(1000);

//check if it is time to poll

if(pollFlag == true){

String json = "";

index1 = 0;

index2 = 0;

Serial.println();

Serial.print("Poll counter: "); Serial.println(pollCounter);

String targetURI = WEBPAGE;

/\* Try connecting to the website \*/

Serial.print("\r\nAttempting connection with data ");

Serial.println(targetURI);

Adafruit\_CC3000\_Client www = cc3000.connectTCP(ip, 80);

if (www.connected()) {

Serial.println("Connected; requesting commands");

www.fastrprint(F("GET "));

www.print(targetURI); //can't use fastrprint because it won't accept a variable

www.fastrprint(F(" HTTP/1.0\r\n"));

www.fastrprint(F("Host: ")); www.fastrprint(WEBSITE); www.fastrprint(F("\n"));

www.fastrprint(F("\r\n"));

www.fastrprint(F("\r\n"));

www.println();

} else {

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

return;

}

boolean readflag = false;

while (www.connected()) {

while (www.available()) {

char c = www.read();

//Serial.print(c);

if (c=='[') {readflag = true;}

if (c=='!') {readflag = false;}

if ((readflag == true)) {

json = json + c;

}

}

}

Serial.print("\nJson: ");

Serial.print(json);

for(int i = 0; i < json.length(); i++){

if (json[i]=='[') {openBracketPositions[index1++] = i;}

if (json[i]==']') {closedBracketPositions[index2++] = i;}

}

for(int k = 0; k < MAXLINES; k++){

String commands = "";

for(int l = openBracketPositions[k]+1; l < closedBracketPositions[k]; l++){

commands = commands + json[l];

int comma1 = commands.indexOf(',');

int comma2 = commands.indexOf(',',comma1 + 1);

int comma3 = commands.lastIndexOf(',');

alarms[k][0] = (commands.substring(0,comma1)).toInt();

alarms[k][1] = (commands.substring(comma1+1,comma2)).toInt();

alarms[k][2] = (commands.substring(comma2+1, comma3)).toInt();

alarms[k][3] = (commands.substring(comma3+1)).toInt();

}

}

www.close();

Serial.print("\n\nAlarms: \n");

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

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

Serial.print(alarms[i][j]);

Serial.print(" ");

}

Serial.print("\n-------\n");

}

//now that we've polled, get ready for next poll

pollCounter = pollCounter + 1; //increment the counter

pollFlag = false;

lastPoll = millis();

Serial.print("\r\n Next poll in ");

Serial.println(pollInterval/1000);

} //end of what we do if pollFlag == true

else //what we do if pollFlag == false

{

//check to see if it's time to poll

if (millis() >= (lastPoll + pollInterval))

{ //it's time to poll!

pollFlag = true;

}

}

isTime2();

}

# COD SCRIPT PHP:

<?php

//No HTTP header or tags.

//To minimize the amount of data the Arduino has to parse, we send ONLY the JSON string.

//First we collect the info passed from the Arduino in the form of GET arguments

//collect passed value for record ID from the GET arg

$id=$\_GET['id'];

//echo "ID = ". $id;

include('sql-connect2.php');

$result = mysqli\_query($con,"SELECT \* FROM " . $table . " WHERE recordID =" . $id);

while($row = mysqli\_fetch\_array($result)){

$json = array(

+($row["hour"]),

+($row["minute"]),

+($row["second"])

);

$description = $row["description"];

}

echo json\_encode($json);

echo "!" . $description . "!";

//var\_dump($json);

mysqli\_close($con);

?>