

Aplicatii pe baza modului WiFi ESP-12. Partea II

1. Digitizarea semnalelor

În aplicațiile practice, majoritatea semnalelor întâlnite sunt analogice, deci prezente la fiecare moment de timp și având amplitudini continue variabile (de exemplu: temperatura, presiunea, viteza, semnalul vocal, etc.). **Prelucrarea digitală** a semnalelor presupune reprezentarea lor digitală (sub forma unei secvențe de numere), cu scopul de a extrage informația dorită sau de a realiza transformări.

În ultimele decenii prelucrarea digitală constituie un domeniu în continuă evoluție, atât din punct de vedere tehnologic, cât și economic. În cursul anilor a apărut o multitudine de circuite digitale complexe, care sunt din ce în ce mai „inteligente”, rapide, miniaturizate și cu posibilități de interconectare sporite: circuite integrate dedicate ASIC (Application Specific Integrated Circuit), arii de porți logice programabile FPGA (Field Programmable Gate Array), microcontrolere MCU, microprocesoare de uz general GPP (General Purpose Processors).

Semnalele analogice sunt transformate la intrare din domeniul analogic în domeniul numeric prin intermediul unui convertor analog-numeric ADC (Analog to Digital Converter, fig. 1) iar la ieșire din domeniul numeric în domeniul analogic prin intermediul unui convertor numeric-analog (DAC).

Etapile principale în digitizarea unui semnal sunt **eșantionarea** și **cuantizarea**.

- 1) Procesul de **eșantionare** convertește variabila independentă (timpul) dintr-o mărime care evoluează continuu într-o mărime care poate lua doar valori discrete (se discretizează).
- 2) **Cuantizarea** convertește variabila dependentă (tensiunea) dintr-o mărime continuă într-una discretizată. Orice eșantion al semnalului digitizat poate avea o eroare de maxim $\pm 1/2$ LSB (Least Significant Bit), unde LSB desemnează distanța dintre două niveluri de cuantizare adiacente.

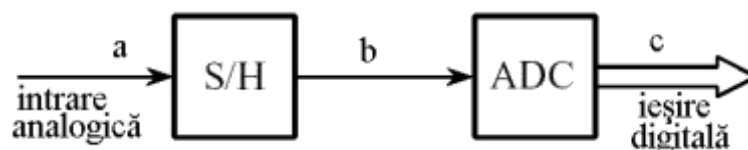


Fig. 1. Digitizare semnal, etape principale: esantionare (S/H - Sample and Hold) + cuantizare propriuzisa (ADC)

- În fig.2 avem
- a semnalul analogic de intrare, continuu în domeniul timp și în domeniul amplitudine
- b semnalul analogic după esantionare/memorare (S/H)
- c semnal digitizat, cuantizat și convertit de ADC în secvența de numere
- d eroarea de cuantizare, diferența dintre semnalul analogic original și semnal digitizat convertit înapoi în semnal analogic cu un DAC

Semnalul digital rezultat este o funcție, în care atât timpul cât și amplitudinea iau valori discrete și finite ca număr.

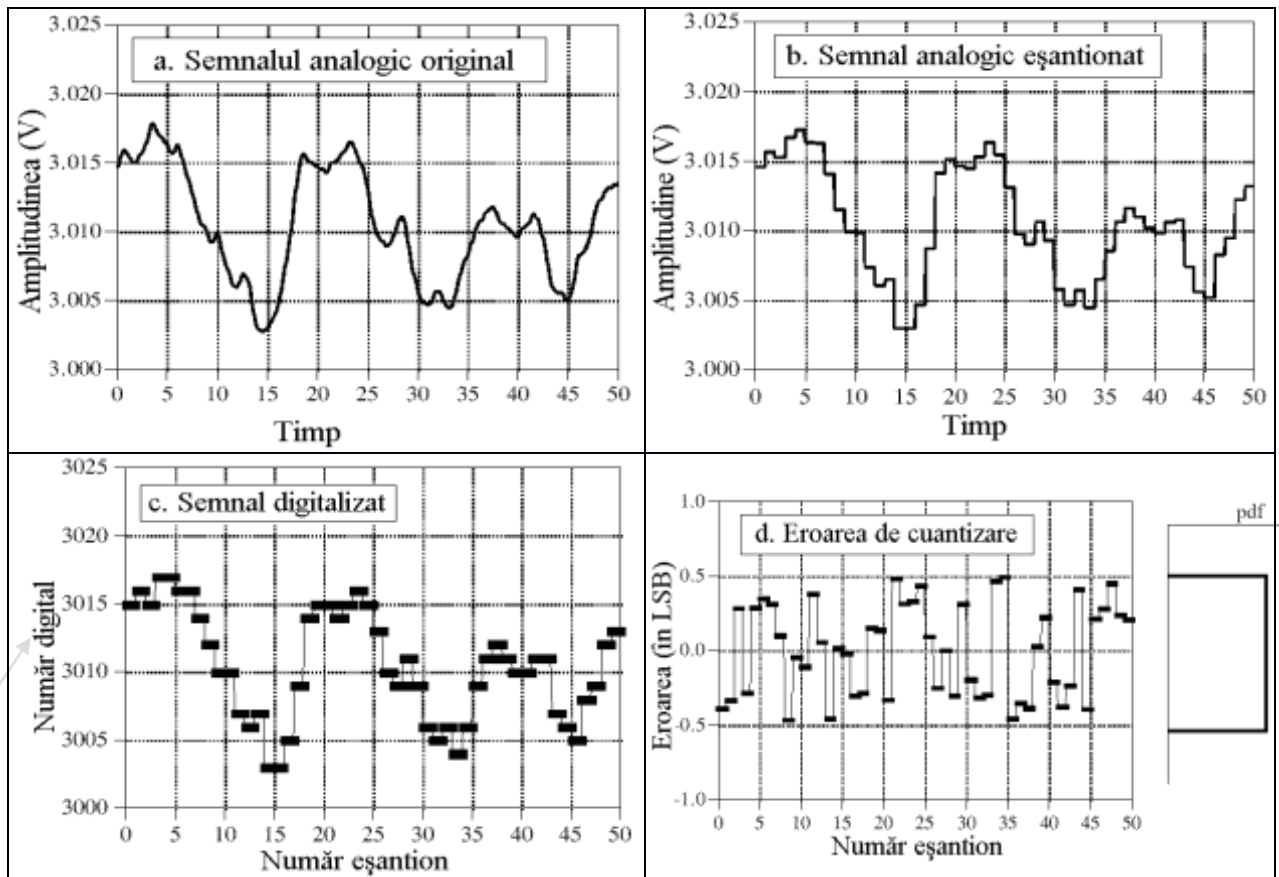


Fig. 2. Exemplu digitizare semnal - esantionare si cuantizare

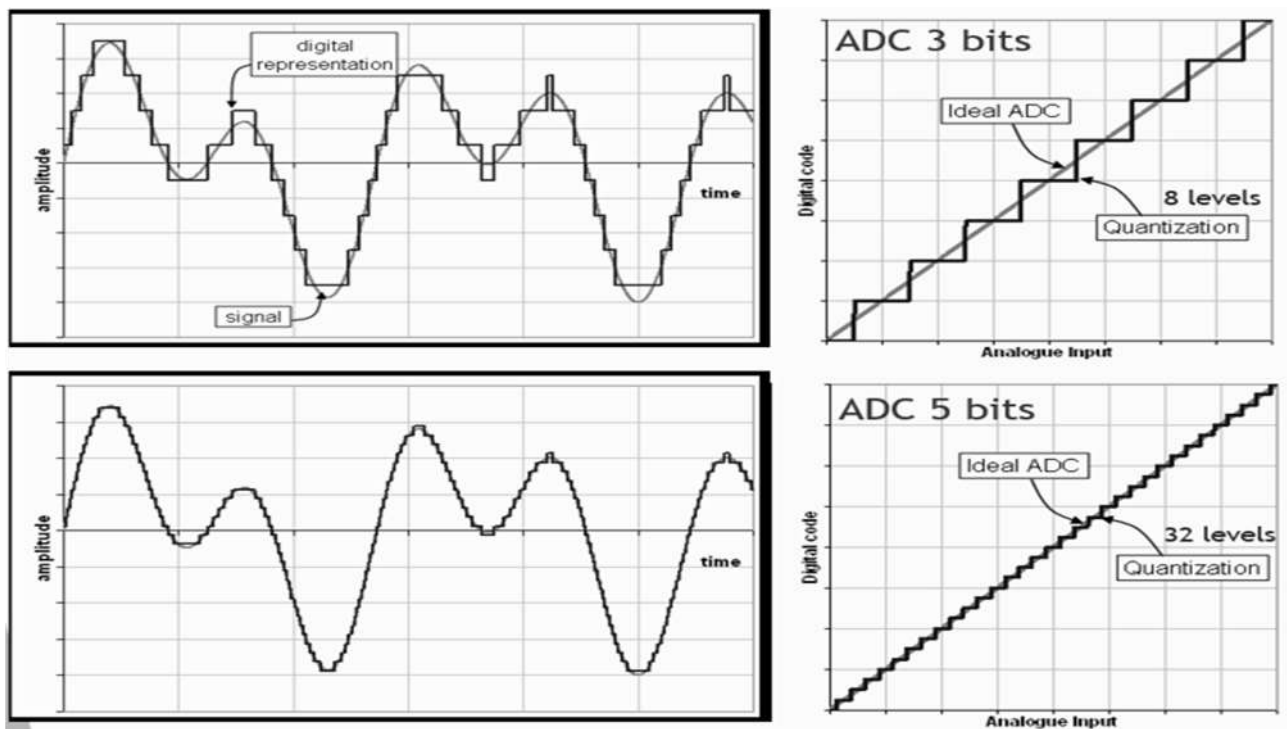


Fig. 3. Reprezentare digitala folosind diferite nivele de cuantizare

2. Determinarea referinței convertorului ADC la ESP-12E

ESP8266-v12, include un convertor analog digital pe 10 biti, având 1024 nivele de cuantizare. Modulul poate prelucra semnale de tensiune provenite de la senzori analogici. Documentatia oferita de producator, pentru ADC-ul din ESP8266 este destul de redusă si imprecisa. Au fost observate aceste deficiente facand primele experimente cu modulul WiFi ESP-12.

Foia de catalog mentioneaza ca valoarea referintei analogice V_{REFADC} este aproximativ 1,0V. Rezulta că atunci când aplicam la intrarea ADC o tensiune $\leq V_{REFADC} - \frac{1}{2}LSB$, numarul pe 10 biti furnizat de ADC trebuie sa fie 1023. Valoarea 1024 nu se mai poate reprezenta pe 10biți, ea însemnând depășirea domeniului admis.

Experimentele noastre arata ca in realitate $Aref \neq 1,0V$ si variaza de la modul la modul. Numarul furnizat de ADC pt. aceeasi tensiune de intrare diferă de la un exemplar ESP la altul. Aceste erori nu pot fi acceptate.

Solutia este **determinarea experimentală a valorii** V_{REFADC} a fiecarui modul ESP in parte.

Obs:

Jumper-ul sa fie pozitionat pe Analog Ref! (vezi și schema electrică). Rotim potentiometrul P1 pana când Nadc raportat periodic de aplicația `check_analog_ref` alternează între 1023 și 1024. Atunci tensiunea pe cursorul lui P1 este chiar valoarea referinței analogice interne, V_{REFADC} . Cand tensiuni pe pinul ADC mai mari decât V_{REFADC} , se obtine $Nadc = 1024$.

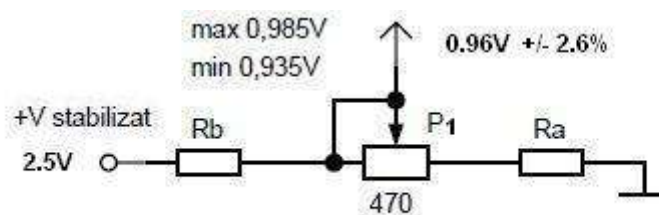


Fig. 4. Dimensionarea divizorului rezistiv P1

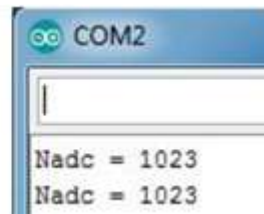
Tema 1:

- 1) Pornind de la valoarea +V stabilizat de care dispuneti, dimensionati divizorul astfel incat la rotirea potentiometrului P1, tensiunea in cursor sa fie cuprinsa intre 0,935~0,985V, iar $V_{stabilizat} = 2,5V$.
- 2) Incarcati programul `check_analog_ref.ino` si conectati cursorul la intrarea ADC / ESP-12 cu jumperul pe pozitia AnalogRef.
- 3) Rotiti potentiometrul pana cand numarul pasilor de conversie Nadc afisat pe monitorul serial Arduino e la limita 1023/1024. Tensiunea masurata in cursor este valoarea referintei analogice Aref. Notati aceasta valoare, veti avea nevoie de ea pentru aplicatiile ulterioare!
- 4) **Șurubul trebuie strâns, altfel nu face contactul la GND**

```

void loop() {
  int Nadc = analogRead(A0);
  Serial.print("Nadc = ");
  Serial.println(Nadc);
  delay(1000);
}

```



scala 2000mV
Fig.5. Determinare experimentală a valorii V_{REF_ADC}

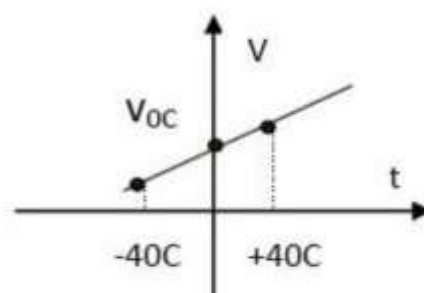
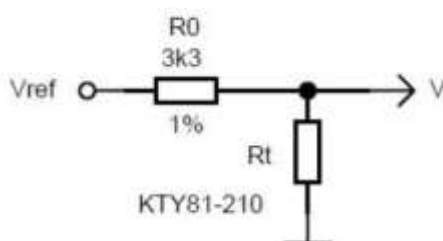
3. Măsurarea temperaturii ambiante cu traductorul KTY81-210

În continuare prezentăm metoda pentru măsurarea temperaturii ambiante între $-40 - +40\text{ }^{\circ}\text{C}$, folosind o metoda bazată pe senzorul de temperatura KTY81-210.

- 1) Traductorul KTY81-210 este o termorezistență. Liniaritatea funcției de transfer $V = f(\theta)$ se garantează de producător dacă rezistența serie R_0 are valoarea exact 3300Ω
- 2) Traductorul poate fi legat la max 50m distanță cu un cablu bifilar $2 \times 0,75\text{mm}^2$.

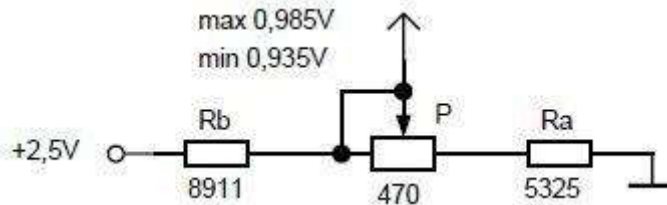
Ecuatiile traductorului KTY81-210

Valoarea $R_0 = 3300\Omega$ asigură o liniaritate foarte bună a tensiunii de ieșire V în funcție de temperatura t , indiferent de tensiunea V_{ref} .



$$V = V_{ref} \frac{R_t}{R_t + R_0} \quad (1) \quad \text{alegem } V_{ref} = 2,5V$$

Divizorul pentru determinarea valorii referinței analogice A_{ref} a modului ESP-12 este:



Valorile R_a și R_b rezulta din calcul atunci când $P = 470\Omega$, însă potențiometrii au toleranță 20%. Valorile exacte R_a și R_b rezulta din valoarea măsurată P :

- $R_a = 11,33 \times P$
- $R_b = 18,96 \times P$

Valorile R_a și R_b se realizează din inserierea unor rezistențe standardizate.

Foaia de catalog KTY81-210 arată valorile termorezistenței R_t la diverse temperaturi (anexa 2):

- 1630Ω la $0^\circ C$
- 1135Ω la $-40^\circ C$
- 2245Ω la $+40^\circ C$
- 2000Ω la $+25^\circ C$

Tensiunea V furnizată de traductor poate fi scrisă ca o dreaptă în funcție de temperatura t :

$$V = \alpha \cdot t + V_{oc} \quad (2) \quad \text{în care } \alpha \text{ e panta și } V_{oc} \text{ e ordonată în origine}$$

Valorile numerice V_{oc} și α se determină din ecuația (1).

$$V_{0C} V_{oc} = 2,495 \cdot \frac{1630}{1630 + 3300} = \frac{1630}{1630 + 3300} = 0,824919V$$

$$V_{-40C} V_{-40C} = 2,495 \cdot \frac{1135}{1135 + 3300} = \frac{1135}{1135 + 3300} = 0,638517V$$

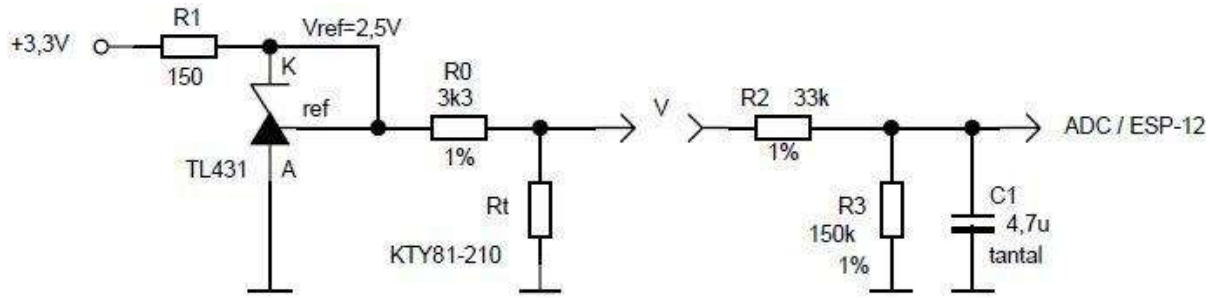
$$V_{+40C} V_{+40C} = 2,495 \cdot \frac{2245}{2245 + 3300} = \frac{2245}{2245 + 3300} = 1,010149V$$

$$\alpha = \frac{V_{+40C} - V_{-40C}}{40C - (-40C)} = \frac{1,010149 - 0,638517}{80} =$$

$$\frac{V_{+40C} - V_{-40C}}{40C - (-40C)} = \frac{1,010149 - 0,638517}{80} = 4,6454 \cdot 10^{-3} \text{ } 10^{-3} V/C$$

Referința TL431 are valoare centrată $V_{ref} = 2,495V$.

Montajul electronic arata astfel:



Divizorul R2, R3 coboara tensiunea V furnizata de traductor la plaja de intrare a ADC / ESP-12.

Intrarea ADC converteste tensiunea de la iesirea divizorului in numar de pasi N_{adc} in functie de valoarea A_{ref} determinata experimental:

$$V * \frac{R3}{R2 + R3} = N_{adc} * \frac{A_{ref}}{1024} \quad (3) \quad V * \frac{R3}{R2 + R3} = N_{adc} * \frac{A_{ref}}{1024} \quad (3)$$

inlocuim numeric $R2$, $R3$ si le trecem in dreapta:

$$V = N_{adc} * \frac{A_{ref}}{1024} * \frac{183}{150} \quad (4)$$

Eliminam variabila V in ecuatiile (4) si (2):

$$\alpha * t + V_{oc} = N_{adc} * \frac{A_{ref}}{1024} * \frac{183}{150}$$

Extragem temperatura t si inlocuim α si V_{oc} cu valorile numerice calculate mai sus:

$$t = N_{adc} * \frac{A_{ref}}{1024} * \frac{183}{150} * \frac{1000}{4,6454} - \frac{824,919}{4,6454} \quad (5)$$

Amplificam ecuatia cu 10.000 si o impartim in final doar cu 1.000.

$$N_{adc} * \frac{(A_{ref} * 10000)}{1024} * \frac{183}{150} * \frac{1000}{4,6454} - \frac{8249190}{4,6454}$$

$$t * 10 = \frac{N_{adc} * \frac{(A_{ref} * 10000)}{1024} * \frac{183}{150} * \frac{1000}{4,6454} - \frac{8249190}{4,6454}}{1000} \quad (6)$$

notam $A_{ref1} = A_{ref} * 10000$, facem calcule partiale si rezulta

$$t * 10 = \frac{N_{adc} * A_{ref1} * 183}{713533} - 1776 \quad (7)$$

Exemplarul de test ESP-12 are valoarea **Aref = 0,9715V**, determinata experimental, deci **Aref1=9715**.

Obs:

Experimentele au fost facute pe 5 exemplare ESP-12. Valorile Aref determinate experimental au fost 0,9756V, 0,9672V, 0,9853V, 0,9715V, 0,9859V si 0,9791V.

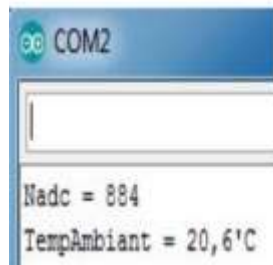
E util un voltmetru cu 4 ½ digiti.

Obs: Am evitat operatiile aritmetice in virgula mobila. Rezolutia de masura este ~ 0,3 °C.

Tema 2:

In programul [adc_kty81_210.ino](#) inlocuiti **Aref1** cu valoarea obtinuta si rulati programul. Acesta prindeza partea intreaga cu semn, virgula si prima zecimala a temperaturii masurate:

```
Serial.print("TempAmbiant = ");  
Serial.print(TempAmbiant/10);           //temperatura in valoare intreaga si semn  
Serial.print(",");                       //virgula zecimala  
Serial.print(abs(TempAmbiant)%10);      //valoarea zecimala cu ajutorul functiei modulo 10  
Serial.println("'C");
```



Tema 3:

- a. Studiati de asemenea programele [adc_kty81_210_isr.ino](#) si [web_kty81_210.ino](#).
- b. Precizati diferentele dintre [adc_kty81_210.ino](#) si [adc_kty81_210_isr.ino](#).
- c. Scrieti o aplicatie care sa genereze un semnal dreptunghiular al carui factorul de umplere sa varieze proportional cu valoarea tensiunii aplicata pinului ADC. Pentru vizualizare activati Led 1.
Sugestii: Puneti jumperul Led 1 pentru activarea LED_1 si buzer la GPIO16.
Puneti jumperul Analog var si rotiti potentiometrul P2. Ce observati?

OBS.

Dacă in Arduino IDE apare jos „error: espcomm_upload_mem failed”, ati uitat sa dati Reset la ESP

Bibliografie

http://cnic.ro/telecom/adc_dac.htm
https://www.nxp.com/docs/en/data-sheet/KTY81_SER.pdf
http://www.acdcelectronics.ro/index_files/wifi_esp8266_utilizari_aplicatii_tutoriale.html
<https://www.arduino.cc/en/Reference/WiFi>

Anexa 1

1. Sursa program: `check_analog_ref.ino`



```
meas_V_REF_ADC | Arduino 1.8.5
File Edit Sketch Tools Help

meas_V_REF_ADC

// Programul permite sa masuraram valoarea referintei interne al ADC-ului din ESP

// Puneti jumperul pe poz. „Analog ref” si rotiti pot.metrul P1 pana cand Nadc = 1023.
// Tensiunea existanta acum pe pinul ADC este valoarea referintei analogice V_REF.
// Atunci cand tensiunea in pinul ADC >= V_REF, se obtine Nadc = 1024.  */

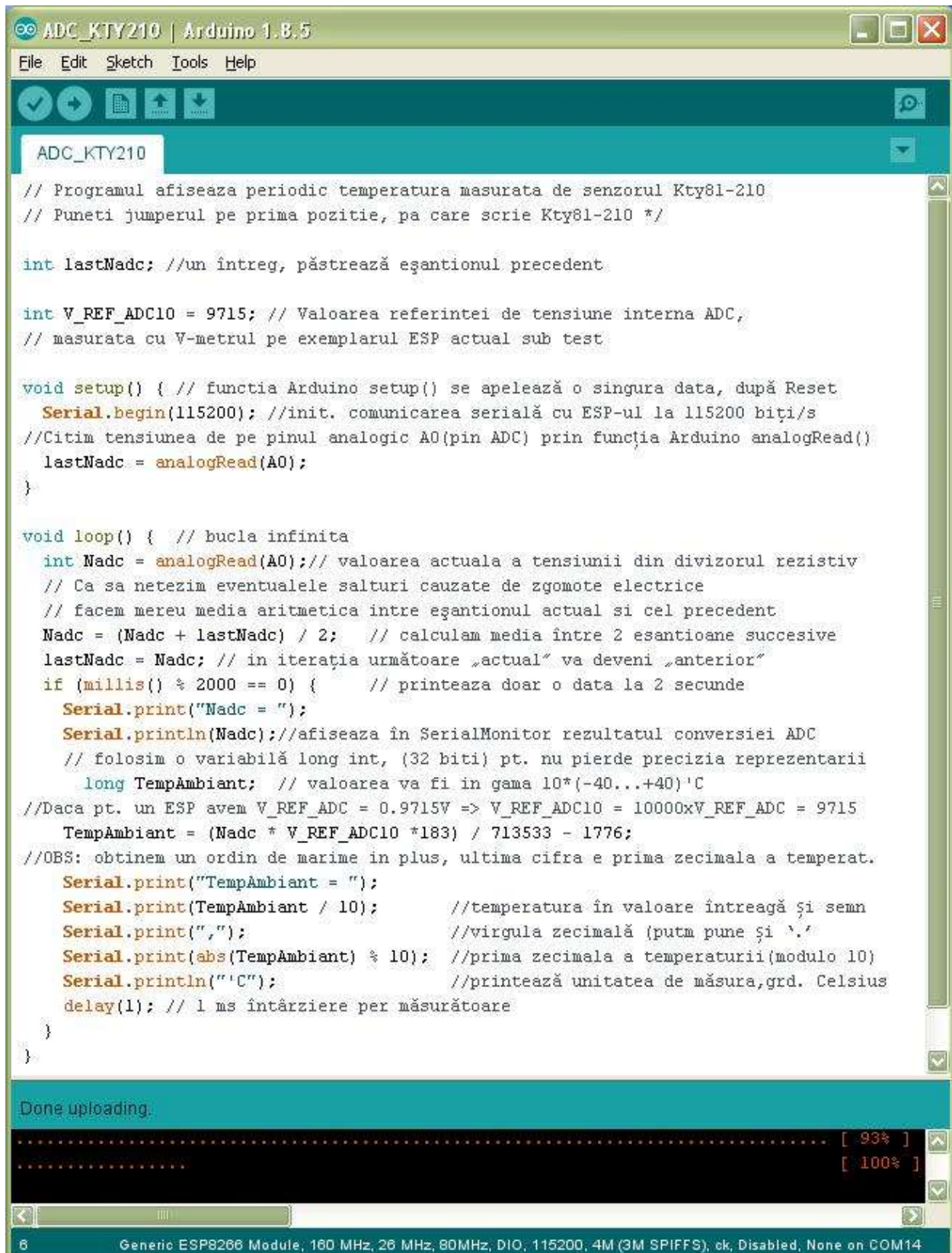
// functia setup(), rulează dupa Reset.
// Cand se alimentează placa, 8266 se auto-reseteaza datorită unui circuit intern
void setup() {
  Serial.begin(115200); //initializează comunicarea serială la 115200 biti pe secundă
}

void loop() { // bucla infinita
  int Nadc = analogRead(A0); // functie Arduino.
  // returneaza un intreg Nadc = 1024 * v_iA / V_REF , unde
  // v_iA = tensiunea pe pinul de intrare analogica al ESP, A0(pinul „ADC” al ESP)
  // V_REF = tensiunea interna de referinta „bandgap” al ADC,
  // aprox. 1V, compensat termic
  Serial.print("Nadc = "); // un text trimis catre afisare in SerialMonitor
  Serial.println(Nadc); // afisează in continuare rezultatul conversiei ADC
  delay(1000); // 1 secundă asteptare dupa fiecare măsurătoare.
}

Done compiling.
Sketch uses 258659 bytes (24%) of program storage space. Maximum is 1044464 bytes.
Global variables use 33272 bytes (40%) of dynamic memory, leaving 48648 bytes for local variables.

1 Generic ESP8266 Module, 160 MHz, 28 MHz, 80MHz, DIO, 115200, 4M (3M SPIFFS), ck, Disabled, None on COM14
```


2. Sursa program: `adc_kty_210.ino`



```
ADC_KTY210 | Arduino 1.8.5
File Edit Sketch Tools Help

ADC_KTY210

// Programul afiseaza periodic temperatura masurata de senzorul Kty81-210
// Puneti jumperul pe prima pozitie, pa care scrie Kty81-210 */

int lastNadc; //un întreg, păstrează eșantionul precedent

int V_REF_ADC10 = 9715; // Valoarea referintei de tensiune internă ADC,
// masurata cu V-metrul pe exemplarul ESP actual sub test

void setup() { // functia Arduino setup() se apelează o singura data, după Reset
  Serial.begin(115200); //init. comunicarea serială cu ESP-ul la 115200 biți/s
  //Citim tensiunea de pe pinul analogic A0(pin ADC) prin funcția Arduino analogRead()
  lastNadc = analogRead(A0);
}

void loop() { // bucla infinita
  int Nadc = analogRead(A0); // valoarea actuala a tensiunii din divizorul rezistiv
  // Ca sa netezim eventualele salturi cauzate de zgomote electrice
  // facem mereu media aritmetica între eșantionul actual si cel precedent
  Nadc = (Nadc + lastNadc) / 2; // calculam media între 2 esantioane succesive
  lastNadc = Nadc; // in iterația următoare „actual” va deveni „anterior”
  if (millis() % 2000 == 0) { // printeaza doar o data la 2 secunde
    Serial.print("Nadc = ");
    Serial.println(Nadc); //afiseaza în SerialMonitor rezultatul conversiei ADC
    // folosim o variabilă long int, (32 biti) pt. nu pierde precizia reprezentarii
    long TempAmbiant; // valoarea va fi in gama 10*(-40...+40)°C
    //Daca pt. un ESP avem V_REF_ADC = 0.9715V => V_REF_ADC10 = 10000xV_REF_ADC = 9715
    TempAmbiant = (Nadc * V_REF_ADC10 *183) / 713533 - 1776;
    //OBS: obtinem un ordin de marime in plus, ultima cifra e prima zecimala a temperat.
    Serial.print("TempAmbiant = ");
    Serial.print(TempAmbiant / 10); //temperatura în valoare întreagă și semn
    Serial.print(","); //virgula zecimală (puta pune și '.')
    Serial.print(abs(TempAmbiant) % 10); //prima zecimala a temperaturii(modulo 10)
    Serial.println("'C"); //printează unitatea de măsură,grd. Celsius
    delay(1); // 1 ms întârziere per măsurătoare
  }
}

Done uploading.
..... [ 93% ]
..... [ 100% ]

6 Generic ESP8266 Module, 160 MHz, 26 MHz, 80MHz, DIO, 115200, 4M (3M SPIFFS), ck, Disabled, None on COM14
```

3. Sursa program: **web_kty81_210.ino**

/* Programul transmite periodic prin WLAN valoarea temperaturii.
Contine doua noutăți fata de exemplul precedent, adc_kty_210.ino
1. include biblioteca ticker.h care utilizeaza intreruperea periodica de Timer din uController
2. si biblioteca de clase Arduino implementata pe ESP pt. comunicare prin interfata-aer, ESP8266WiFi.h

Nadc = Valoarea conversiei ADC e determinat in Timer ISR, care apare asincron fata de executia instructiunilor din bucla principala loop().

Puneti jumperul de selecție a intrarii ADC pe Kty81-210 */

/*Biblioteca ESP8266WiFi.h oferă o colecție de clase C++
pentru a configura și opera modulul ESP8266 ca stație sau punct de acces(soft acces point)
Exemple de clase ce pot fi utilizate:

Station:

pentru a conecta modulul ESP la o rețea WI-Fi stabilită de un punct de acces

Soft Acces Point:

permite accesul altor dispozitive la rețeaua Wi-Fi și le conectează în continuare la o rețea cu fir

Scan

pentru a scana și afișa rețelele WLAN disponibile

Client:

crează clienți care pot accesa serviciile furnizate de servere pentru a trimite, primi și procesa date

Client Secure

extensie a clasei Client în care conexiunea și schimbul de date cu serverele se efectuează utilizând criptare.

Suportă protocolul TLS 1.1, (Transport Layer Security) dar nu și TLS 1.2

Server

crează servere care oferă funcționalitate altor programe sau dispozitive,numite clienți

UDP

permite transmiterea și recepționarea mesajelor prin protocolul User Datagram Protocol.

Deși UDP ofera sume de control pentru integritatea datelor și a antetului, acest nivel nu memoreaza starea transmisiei.

Folosește numere de port pentru adresarea diferitelor funcții la sursa și destinația datagramelor.

Generic

utilizată pentru gestionarea evenimentelor Wi-Fi cum ar fi:

conectarea, deconectarea, obținerea unei adrese IP, modificări ale modulului Wi-Fi */

```
#include <ESP8266WiFi.h>
// biblioteca originala Arduino, pt. „shield”-ul sau WLAN se numea <WiFi.h>

const char* ssid = "211B";           // Service Set ID, numele rețelei
const char* password = "sala211b";  // password (programul trimite parola)

WiFiServer server(80); // creeaza un server web care așteaptă clienți
// asculta pe portul nr. 80, asociat protocolului HTTP, transmite și rec. pagini HTML
```

```

//bibliotecă pentru apelarea funcțiilor în mod repetat cu o anumită perioadă
#include <Ticker.h> // Ticker Library -> procedura ISR
Ticker get_Nadc; // Pot fi create mai multe instante Ticker ruland simultan

int Nadc, lastNadc;//variabile de tip întreg pentru măsurarea actuală și precedentă
int V_REF_ADC10=9715; // ESP-ul nostru are V_REF_ADC = 0,9715V masurat experimental.
// pt. a lucra cu întregi(fara a utiliza biblioteca float), o înmultim cu 10000

void average_Nadc() { // Functia care se apeleaza din ISR,numita si rutina "callback"
// Temperatura e o marime analogica si nu poate varia prin salturi
// Facem media între măsurarea actuala si măsurarea precedentă
Nadc = analogRead(A0);//citim valoarea actuala pe pinul analogic A0(pinul ADC)
Nadc = (Nadc + lastNadc) / 2; // calculam media între cele 2 măsurători succesive
lastNadc = Nadc;
}

void setup() { // funcția este apelata o singura data dupa reset
Serial.begin(115200);//inițializează comunicarea serială la 115200 biți/s
delay(100);//0.1 secunde întârziere per măsurătoare
lastNadc = analogRead(A0);//initializam valoarea precedentă
//init. ticker: average_Nadc() se va apela la fiecare 9ms
get_Nadc.attach_ms(9, average_Nadc);

Serial.print("\n\nConnecting to "); // Connecting to WiFi network
Serial.println(ssid);//printează numele rețelei

WiFi.begin(ssid, password);//se apelează funcția pentru conectarea la WI-Fi
// conectarea poate dura câteva secunde, asteptam acest lucru in următoarea buclă
while (WiFi.status() != WL_CONNECTED) { //bucla va rula în continuare pana cand
//WiFi.status este altul decât "WL Connected"
delay(500);// așteptare de 500ms (0.5s)
Serial.print("."); // progress indicator
}
Serial.println(""); // linie goala
Serial.println("WiFi connected");// afișează confirmarea că s-a conectat

server.begin(); // Porneste serverul
Serial.println("Server started");//afișează confirmarea că serverul este funcțional

// Print the IP address
Serial.print("Use this URL to connect: http://");
Serial.print(WiFi.localIP());//se printează adresa IP atribuită modulului ESP de către DHCP
Serial.println("/");
}

void loop() { // loop() se apeleaza in bucla infinita din main()-ul "ascuns"
WiFiClient client = server.available();//serverul așteaptă clientul să se conecteze
if (!client)
return; // "early return" din functia loop(), care este imediat reapelata din nou
Serial.println("new client");
while (!client.available()) { // serverul așteaptă ca clientul să trimită o cerere
delay(1); // întârziere 1ms
}

String request = client.readStringUntil('\r'); // citește prima linie a cererii
Serial.println(request); // printează prima linie
client.flush(); //așteaptă ca toate caracterele de ieșire din buffer au fost trimise
client.println("HTTP/1.1 200 OK");// returnează răspunsul, printează în html
client.println("Content-Type: text/html");
client.println(""); // acesta linie goala e mandatorie
client.println("<!DOCTYPE HTML>");
client.println("<html>");

//calculeaza valoarea temperaturii
long TempAmbiant; //in gama -40...+40'C //variabilă de tip întreg pe 32 de biți
TempAmbiant = (Nadc * V_REF_ADC10 * 183) / 713533 - 1776;

```

```

//are un ordin de marime in plus -> prima zecimala a temperaturii

client.println("<br>"); // scrie continutul pe pagina web
client.println("Click <a href=\"/\">Get new temperature</a>");
client.println("</html>");

client.println("<br><br>");
client.print("Temperatura ambianta = ");
client.print(TempAmbiant / 10); //temperatura in valoare intreaga si semn
client.print(","); //virgula zecimala
client.print(abs(TempAmbiant) % 10); //prima zecimala a temperaturii
client.println("'C"); // Grade Celsius
client.println("</html>");

delay(1); // asteptare 1ms
Serial.println("Client disconnected");//clientul este deconectat
Serial.println("");

//intrerupe conexiunea in curs si asteapta una noua de la acelasi client sau altul
client.stop();
} // loop()
/*La rulareapare in Tools> SerialMonitor
=====
.....
WiFi connected
Server started
Use this URL to connect: http://192.168.1.201/
new client
GET / HTTP/1.1
Client disconnected

new client
GET /favicon.ico HTTP/1.1
Client disconnected
*/

```

Pagina de web vizibila la client:



Client(PC-ul Alpis) si server (ESP) conetati la acelasi LAN prin router-ul wireless al laboratorului 211B

The screenshot shows a 'DHCP Clients Table' window in Mozilla Firefox. It displays a table of active DHCP clients on the network. The DHCP Server IP Address is 192.168.1.1. The table lists two clients: 'alpis' and 'ESP_347EE5'.

Client Host Name	IP Address	MAC Address	Expires	Delete
alpis	192.168.1.200	00:13:8F:29:0E:9E	23:35:16	<input type="checkbox"/>
ESP_347EE5	192.168.1.201	60:01:94:34:7E:E5	23:56:58	<input type="checkbox"/>

Table 10. Ambient temperature, corresponding resistance, temperature coefficient and maximum expected temperature error for KTY81/210 and KTY81/220

$I_{sen(cont)} = 1 \text{ mA}$.

Ambient temperature		Temperature coefficient (%/K)	KTY81/210				KTY81/220			
(°C)	(°F)		Resistance (Ω)			Temperature error (K)	Resistance (Ω)			Temperature error (K)
			Min	Typ	Max		Min	Typ	Max	
-55	-67	0.99	951	980	1009	±3.02	941	980	1019	±4.02
-50	-58	0.98	1000	1030	1059	±2.92	990	1030	1070	±3.94
-40	-40	0.96	1105	1135	1165	±2.74	1094	1135	1176	±3.78
-30	-22	0.93	1218	1247	1277	±2.55	1205	1247	1289	±3.62
-20	-4	0.91	1338	1367	1396	±2.35	1325	1367	1410	±3.45
-10	14	0.88	1467	1495	1523	±2.14	1452	1495	1538	±3.27
0	32	0.85	1603	1630	1656	±1.91	1587	1630	1673	±3.08
10	50	0.83	1748	1772	1797	±1.67	1730	1772	1814	±2.88
20	68	0.80	1901	1922	1944	±1.41	1881	1922	1963	±2.66
25	77	0.79	1980	2000	2020	±1.27	1960	2000	2040	±2.54
30	86	0.78	2057	2080	2102	±1.39	2036	2080	2123	±2.68
40	104	0.75	2217	2245	2272	±1.64	2194	2245	2295	±2.97
50	122	0.73	2383	2417	2451	±1.91	2359	2417	2475	±3.28
60	140	0.71	2557	2597	2637	±2.19	2531	2597	2663	±3.61
70	158	0.69	2737	2785	2832	±2.49	2709	2785	2860	±3.94
80	176	0.67	2924	2980	3035	±2.8	2894	2980	3065	±4.3
90	194	0.65	3118	3182	3246	±3.12	3086	3182	3278	±4.66
100	212	0.63	3318	3392	3466	±3.46	3284	3392	3500	±5.05
110	230	0.59	3523	3607	3691	±3.93	3487	3607	3728	±5.61
120	248	0.53	3722	3817	3912	±4.7	3683	3817	3950	±6.59
125	257	0.49	3815	3915	4016	±5.26	3775	3915	4055	±7.31
130	266	0.44	3901	4008	4114	±6	3861	4008	4154	±8.27
140	284	0.33	4049	4166	4283	±8.45	4008	4166	4325	±11.46
150	302	0.20	4153	4280	4407	±14.63	4110	4280	4450	±19.56