Aplicaţii pe baza modulului WiFi ESP-12 . Partea I

1. Introducere. Descrierea plăcii cu WiFi ESP-12

O rețea de dispozitive fizice, vehicole, aparate de uz casnic și alte elemente cu electronică încorporată, senzori, elemente de acţionare și conectivitate, dar și software-ul necesar funcționării acestora, toţi aceşti termeni definesc aşa numitul domeniu „Internet al obiectelor" (IoT). IoT permite acestor obiecte să se conecteze și să facă schimb de date, totodată facilitând şi detecţia sau controlul lor de la distanță, prin infrastructura de rețea existentă. O posibilitate de a realiza un dispozitiv tipic IoT este prin utilizarea cipului ESP8266, un microcip Wi-Fi de costuri reduse, cu capabilități complete de TCP / IP și cu microcontroler (produs de Espressif Systems, companie chinezească din Shanghai). [Wikipedia]

Cipul ESP8266 a intrat pe piață în august 2014 cu modulul ESP-01 (realizat de un producător terț, Ai-Thinker). Acest modul mic permite ca microcontrolerele să se conecteze la o rețea Wi-Fi și să realizeze conexiuni simple TCP / IP. Până în prezent, acestea au primit denumirea generică de module ESP-xy, cu xy = 01,02, ... 12, 13,14.

Ce reprezintă ESP 8266?

- este un SoC wireless 802.11 b/ g/ n (Wi-Fi) având costuri reduse, oferit într-un modul cu 16 pini;
- dispune de pini GPIO (General Purpose I/O), o intrare ADC și câte o interfață UART, SPI şi I2C;
- lucrează la frecvenţa de ceas CPU 80MHz, max. 160MHz, fără cuarț extern;
- deţine 64Kbytes de RAM de instrucţiuni, 96KBytes de RAM de date şi 64KBytes memorie boot ROM;
- are un Winbond SPI flash extern de 4MiB;
- folosește o arhitectură de tip RISC;
- nucleul procesor este L106 Xtensa Diamond Standard core (LX3) proiectat de Tensilica;
- modulele care utilizează acest cip sunt fabricate de mai mulți producători.

Multe dintre modulele ESP-xy includ un mic LED albastru care poate fi programat să clipească, indicând activitate pe interfața UART. Modulele ESP-xy oferă o antenă „imprimată" direct pe suportul ceramic al modulului și un conector pentru antenă externă, în banda de 2,4 GHz. Modelele recente de ESP-xx includ o punte USB-to-UART onboard și un conector Micro-USB împreună cu un convertor c.c. de 5V/ 3.3 volți (necesare pentru a furniza conectivitate cu calculatorul gazdă şi pentru alimentarea modulului prin cablu USB. La modulele ESP-xy anterioare, aceste două elemente (adaptorul USB-Serial și un regulator de 3.3 volți) trebuiau achiziționate separat.

O placă de dezvoltare realizată la noi în țară și bazată pe modul ESP8266-12 este descrisă la adresa [1], numită în această lucrare de laborator „Placa de test cu WiFi ESP-12". Modulul ESP este programabil prin IDE-ul Arduino, permiţând folosirea şi manipularea lui asemenea unui microcontroler.

În continuare, prezentăm pe scurt modulele plăcii de dezvoltare, principalele componente fiind dispozitivele ***ESP-12*** şi expanderul de porturi pe I2C al firmei Microchip, MCP 2301 ***MCP2301***.

Placa de test cu WiFi ESP-12 facilitează studiul comunicaţiei prin SPI, I2C, I2S, deoarece prin aceste interfeţe, placa interacţionează cu **7 dispozitive performante. Mai multe detalii se găsesc în ANEXA 1.**

2. Mediul Arduino şi Placa de test cu WiFi ESP-12

2.A. Instalarea elementelor necesare funcţionării plăcii de test cu mediul Arduino

Manualul convertorului USB-UART pentru ESP-01 (aflat la adresa [2]) arată detaliat modul de instalare al mediului de dezvoltare Arduino IDE şi al librăriilor ESP8266 pentru implementarea aplicaţiilor pe **Placa de test cu WiFi ESP-12**. Acest pas a fost deja parcurs pentru lucrarea de laborator curentă: Arduino IDE a fost instalat pe PC-urile din laborator, împreună cu librăriile ESP, "uneltele" şi exemplele oferite de fabricantul plăcii, mai multe detalii putându-se consulta în ANEXA 2.

Pentru lucrarea curentă, se vor parcurge ***exemplele de iniţiere*** aflate în directorul ***...\Arduino\placa_test_esp-12\exemple initiere*** Aplicaţiile plăcii conţin 55 exemple nou realizate pentru mediul Arduino, demonstrând faptul că placa e utilizabilă şi pentru dezvoltarea aplicaţiilor complexe.

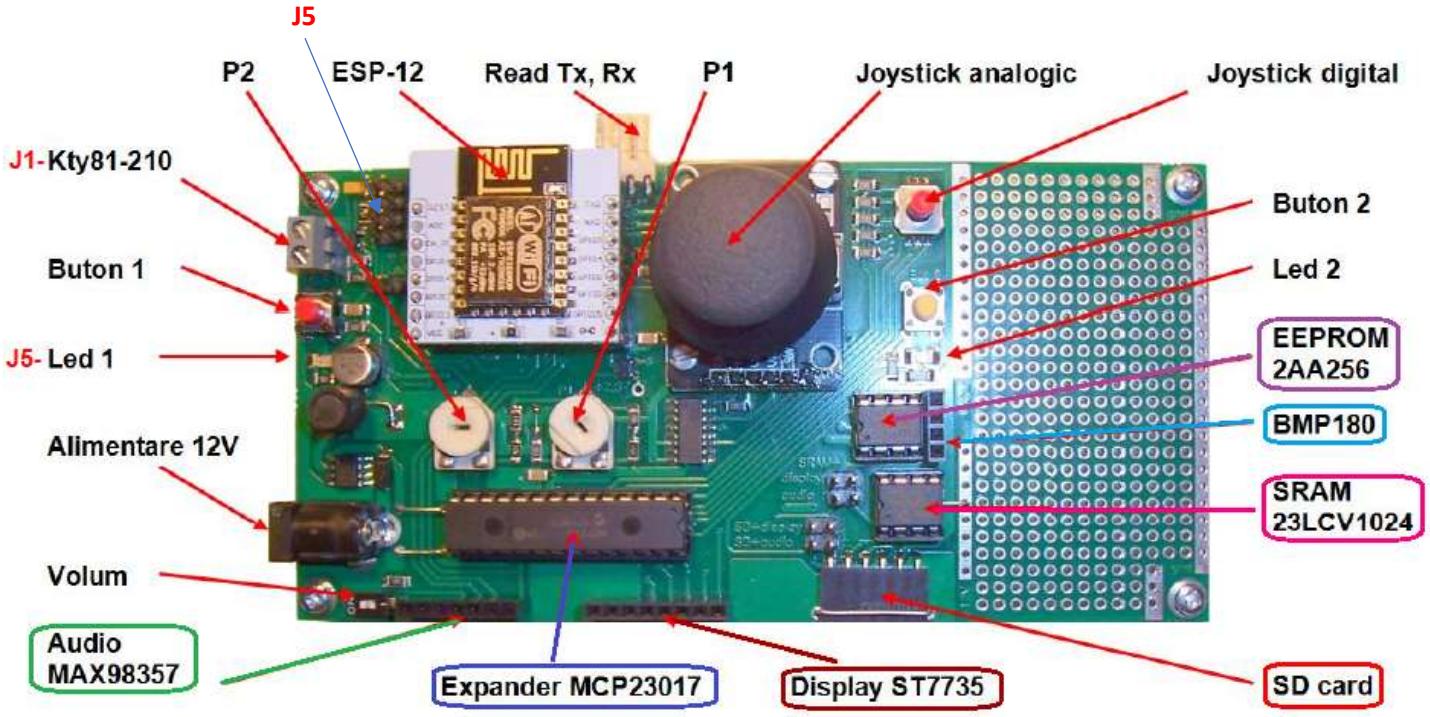


Fig. 1 Amplasarea principalelor componente pe placa de test cu WiFi ESP-12

2. B. Mediul Arduino IDE (Integrated Development Environment)

Denumirea „Arduino" se poate referi atât la partea de hardware (plăcuțe de dezvoltare bazate pe microcontrolere), cât și la partea de software destinată funcționării și programării acestora. Arduino desemnează de asemenea o companie open-source, lansată în 2005 (Ivrea, Italia), având ca scop asigurarea unei soluții ieftine și simple pentru începători, persoane pasionate de dezvoltarea de proiecte *embedded*, dar și profesioniști. Platformele hardware compatibile Arduino reprezintă dispozitive capabile să interacționeze cu mediul Arduino IDE, folosind senzori și sisteme de acționare, fiind deci dedicate zonei de sisteme încorporate (embedded). Arduino, ca hardware, este o platformă *open-source* cu microcontroller, simplu de utilizat, cu conectare prin USB. Acest aspect a propulsat-o în special în zona programelor educaţionale de pretutindeni: deşi este simplu de utilizat, platforma păstrează complexitatea sistemelor proiectate pentru dispozitivele *embedded*.

Atât hardware-ul cât şi software-ul utilizat au numele de "Arduino". Mediul Arduino IDE este gratuit, *open-source* şi *cross-platform*. Hardware-ul şi software-ul sunt *open source*, model prin care dezvoltatorii acestei platforme au publicat schemele electrice şî codul sursă, oferind utilizatorilor posibilitatea de a acţiona liber asupra procesului de producţie sau dezvoltare. Dintre avantajele plăcilor de tip Arduino, față de alte plăci de dezvltare, cele mai importante sunt reprezentate de faptul că pot fi folosite pe platforme/ S.O. diverse (Linux, Windows sau Mac) şi faptul că pot fi programate folosind portul USB 2.0 şi nu portul serial.

Arduino IDE (*Integrated Development Environment*) reprezintă un mediu de dezvoltare flexibil pentru programele ce trebuie scrise în memoria-program a microcontrollerului de pe placa de tip Arduino. Scris în limbajul de programare Java, IDE-ul este o versiune derivată a altui mediu de programare, cel pentru limbajul Process. Programele pentru Arduino sunt scrise în limbajul C sau C++. Avantajul major al mediului este reprezentat de diversitatea de procesoare și platforme pentru care poate fi utilizat, oferind suport bogat (aplicaţii şi biblioteci de funcții) pentru procesoarele suportate de mediu, precum și dinamismul cu care se dezvoltă noi aplicaţii şi biblioteci de către comunitatea Open Source.

Crearea unui nou proiect în Arduino IDE

Pornind aplicaţia, se va deschide o „schiță" nouă (Figura 2). Aici, utilizatorul poate să scrie un program în C/C++, să compileze codul şi apoi să încarce și să ruleze pe placă codul-mașină obținut; majoritatea comenzilor din interfaţa IDE sunt simple și intuitive.

Opțiuni specifice acestei interfețe:

Accesând meniul ***File->Examples*** vom găsi toate programele care vin cu mediul Arduino, dar şi cele destinate plăcii ESP dacă s-a realizat instalarea acestora aşa cum s-a precizat în ANEXA 2 (exemple de iniţiere). Acestea sunt împărţite în mai multe categorii, fiecare cu particularităţile sale (Figura 4).

Pentru a verifica şi a compila o schiţă (programul nostru) se va alege din meniul ***Sketch*** opțiunea ***Verify / Compile*** (Figura 3). După ce programul este compilat (deci tradus în codul mașină), acesta va fi încărcat în memoria flash a microcontrollerului de pe placă, prin cablul USB, folosind comanda ***File->Upload***.

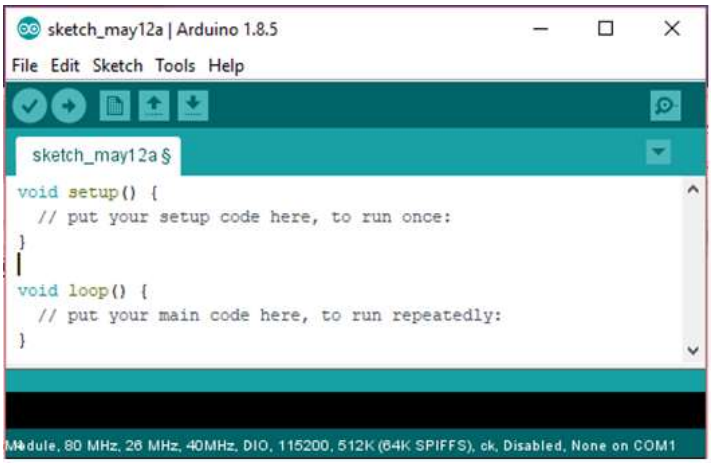


Figura 2 *Arduino IDE*

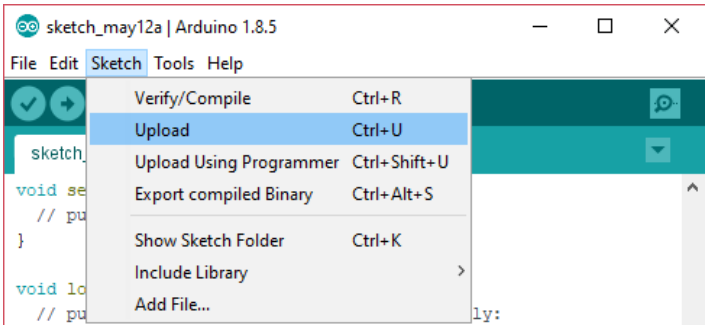


Figura 3 *Meniul schiţă (Sketch)*

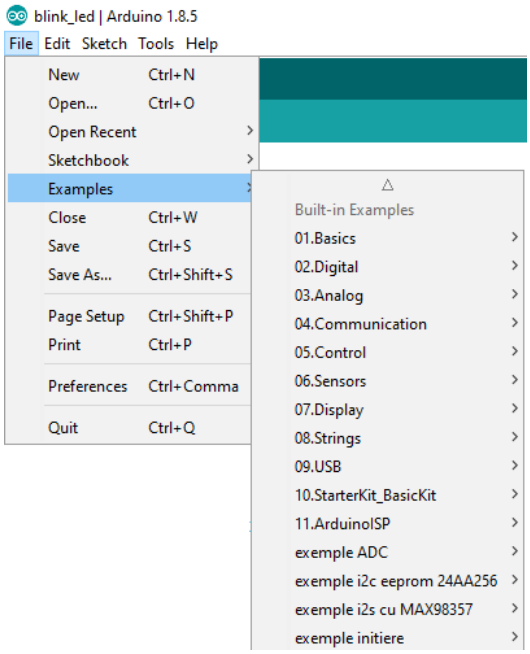


Figura 4 *Mediul Arduino IDE setat pentru lucrul cu placa de test ESP*

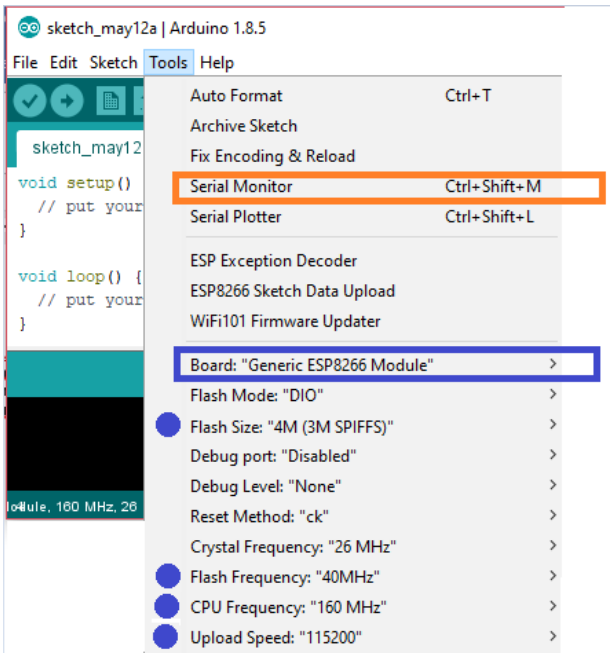


Figura 5 *Meniul de instrumente (Tools) pt monitorizarea serială şi pentru lucrul cu placa de test*

Prin intermediul meniului ***Tools-> Serial Monitor*** utilizatorul poate comunica în mod text cu procesorul rulând pe ESP prin protocolul virtual UART peste USB. Accesând acest meniu, pe ecran se va afişa fereastra-terminal '***Serial Monitor***' (Figura 5) în care utilizatorul poate afişa datele şi valorile variabilelor din program în timp real (în timp ce programul rulează pe microcontrolerul de pe placa ESP). Mesajele din program sunt transmise înapoi către PC prin interfaţa USB (pe care se emulează de fapt protocolul serial RS-232) prin intermediul funcţiei ***Serial.print()***.

<u>Realizarea unei aplicații cu IDE-ul Arduino presupune următorii paşi:</u>

1. Conectarea plăcii de dezvoltare la calculator prin cablu USB (e necesar să se verifice din meniul ***Tools->Serial Port*** apariția unui nou port serial virtual prin care se face conectarea: de obicei un port cu numele COM3, COM4, COM5, etc.); Sistemul de operare poate schimba acest nume după fiecare reintroducere a cablului USB;
2. Scrierea programului pe PC, verificarea şi compilarea acestuia utilizând mediul Arduino IDE (meniul ***Sketch->Verify / Compile***);
3. Încărcarea programului în microcontrolerul plăcii de dezvoltare prin interfaţa USB (meniul ***Sketch ->Upload***);
4. Microcontrollerul de pe placă execută apoi programul încărcat în memoria lui.

Tabelul 1 *Câteva dintre cele mai utilizate funcţii într-o aplicaţie Arduino* ([3],[6]) :

Funcţia	***Descriere***
setup()	Funcţie standard, apelată o singură dată, la început, la alimentarea plăcii sau la resetarea microcontrollerului; De obicei, în această funcţie se scrie codul de iniţializare (de ex. iniţializarea variabilelor, se stabileşte funcţia actuală a pinilor etc.).
loop()	Funcţie standard, apelată în mod repetat: este apelată automat imediat după funcţia ***setup()*** în general şi se execută în buclă infinită, atât timp cât platforma de dezvoltare este alimentată (aici se află acţiunile programului principal, logica programului).
pinMode(*pin, mode*)	Funcţia primeşte doi parametri: numărul pinului şi tipul acestuia (intrare/ ieşire). Această funcţie setează sensul (intrare sau ieşire) pt portul din care face parte pinul primit ca parametru şi îl configurează ca pin de intrare, resp. de ieşire.
analogRead(*pin*)	Returnează un număr întreg reprezentând rezultatul furnizat de convertorul analog / numeric. Parametrul de apel este numele pinului de intrare analogică, de ex. A0, A1, etc. Returnează valoarea specifică unui anumit **pin analogic** primit ca parametru. Pinii analogici sunt conectaţi la intrările convertorului analog numeric ADC al microcontrollerului de pe placă. Dacă convertorul A/N are o rezoluţie de 10 biţi => valoarea returnată de funcţia ***analogRead()*** este un număr întreg între 0-1023.
analogWrite(*pin, value*)	Setează valoare **pin analogic**. Generează pe pinul respectiv un semnal PWM (*Pulse Width Modulation*) cu factorul de umplere proporţional cu valoarea dată ca şi parametru. Valoarea de la intrare se dă în intervalul 0-255. Pentru 255, factorul de umplere al semnalului PWM va fi de 100%. Frecvența PWM este de ordinul kHz.
digitalRead(*pin*)	Returnează valoare pin digital (HIGH sau LOW). Valoarea HIGH sau LOW returnată reprezintă starea (valoarea digitală '0' sau '1') **pinului digital** primit ca parametru.
digitalWrite(*pin*)	Setează valoare **pin digital** (HIGH sau LOW). Funcţia scrie o valoare HIGH sau LOW pe un **pin digital**. Dacă pinul e configurat ca OUTPUT (cu funcţia ***pinMode()***), pe pinul respectiv se va genera o tensiune V_{DD} (de obicei 3.3 sau 5V) pentru starea HIGH, resp. 0V pentru starea LOW.
delay(*ms*)	Inserează întârzieri în program. Opreşte programul pentru un interval de timp, specificat în milisecunde. În acest timp nu se execută nicio operaţie (echivalent cu nop).
Alte funcții (pentru serial monitor):	Serial.begin() - Iniţializează serial monitor Serial.print() – mesaje de pe serial monitor de la placa de dezvoltare către PC Serial.println() - mesaje de tip log pe serial monitor cu linie nouă (nl=new line) Serail.read() – permite controlul procesului embedded de la tastatură
Tipuri de variabile oferite de compilator și constante simbolice specifice:	int – pentru valori întregi , de ex. 123 float – pentru valori zecimale, de ex. 1.15 char[] – pentru valori de tip string, de ex. "Arduino" HIGH – definire pin Digital cu nivel "1" logic LOW - definire pin Digital cu nivel "0" logic INPUT – Pinul este de sens intrare, poate fi citit prin program OUTPUT – Pinul poate fi setat prin program // comentariu pe o singură linie /* */ comentariu pe mai multe linii #define – pentru definirea unei constante #include – pentru includerea bibliotecilor externe

Programele Arduino scrise în limbaj C/ C++ se numesc schițe (***sketches) (Figura 2)***. Acestea sunt salvate de IDE ca fișiere-text cu extensia .ino. Structură de bază a unei schițe are nevoie de cel puțin două funcții în corpul programului: ***setup ()*** și ***loop ()*** (trebuie să aibă întotdeauna același nume, deoarece compilatorul C din IDE le caută și dacă aceste două nume de funcții nu sunt prezente în schița C, atunci IDE-ul Arduino nu va compila schița). Figura 7 prezintă o fereastră standard Arduino IDE și o schiță C, pe un PC bazat pe sistemul de operare Windows 10.

Funcția de configurare **setup() e**ste executată o singură dată (invocată automat de placa de tip Arduino, de fiecare dată când placa Arduino e alimentată; de aceea, este locul ideal pentru a scrie cod necesar anumitor configurări de pini de I/O, a defini variabile, etc.

Codul scris în funcția **loop()** se va executa la infinit, atâta timp cât alimentarea plăcii Arduino este prezentă. Această funcție conține logica principală a programului încorporat și poate invoca, la rândul său, alte funcții definite de utilizator.

Acomodarea cu prima schiță Arduino

Prima schiță pe care o vom studia este cel mai cunoscut program („blink") din domeniul embedded, echivalentul programului „Hello world" din programare. Programul Arduino IDE vine pre-încărcat cu o mulțime de schițe-exemplu. Toate aceste schițe exemplu sunt disponibile pentru utilizare din meniul ***File> Examples*** (Figura 4), unde se văd şi cele instalate suplimentar pentru placa ESP-12.

Figura 6. Încărcarea primei schiţe Arduino în IDE

Observație: Exemplele originale din IDE sunt pentru placa Arduino, acestea nu se vor încărca în placa ESP !

Pentru început, vom studia programul blink_led. Pentru aceasta, deschideţi Arduino IDE pe propriul PC şi mergeţi la ***File > Examples > placa_test_esp-12> exemple de initiere > Blink_led > blink_led.ino*** . Dacă totul s-a desfășurat în ordine, trebuie să apară codul din Figura 7.

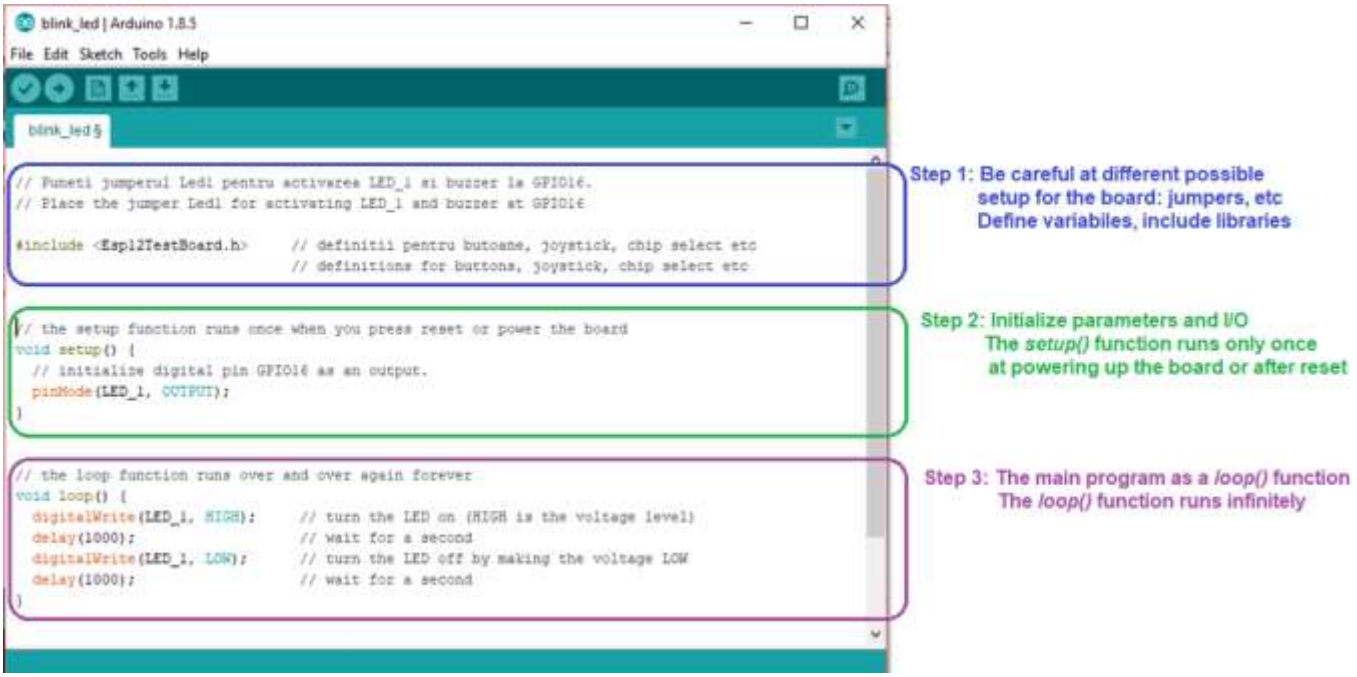


Figure 7: *Structura de bază a unei schiţe Arduino pe exemplul blink_led*

Explicaţii program Blink_led

În cadrul funcţiei ***setup(),*** următoarele linii vor fi explicate:

pinMode(LED_1, OUTPUT); spune plăcii să configureze pinul digital LED_1 ca ieşire;

Atenţie: Rețineți că atunci când un pin digital este configurat în modul de ieșire, microcontrolerul din ESP impune o tensiune de 0 V sau 3.3V prin pinul digital, corespunzătoare bitului de 0 sau 1 înscris în portul de ieșire. Această tensiune digitală este recepționată de către pinul unei componente periferice care este atașată la pinul digital care transmite semnalul digital. În cazul programului blink, anodul LED-ului de pe placă este conectat intern la pinul GPIO16 prin jumperul denumit „Led1”, iar catodul prin R10=560Ω la masă. Urmărind schema electrică, putem identifica ușor că există jumperul care conectează ansamblul LED-buzzer-piezo la pinul GPIO16. Prin urmare, tot ce trebuie să faceți este să compilați și să încărcați schița. Toţi pinii Arduino conectaţi la dispozitive trebuie definiţi ca find de sens intrare sau ieşire.

Cea mai importantă parte a schiței este funcția ***loop()***, care conține instrucțiunile ce se vor executa în buclă infinită.

digitalWrite(LED_1, HIGH); trimite „1” logic, adică valoarea tensiunii de alimentare a ESP-ului pe pinul cu nr specificat. În acest caz, apare un nivel llogic HIGH prin pinul digital LED_1 și LED-ul de pe placă începe să lumineze.

Cea de-a doua funcţie folosită în funcţia **loop()** este:

delay(1000); opreşte curgerea programului pentru un număr de milisecunde atât cât e specificat în paranteză. În acest caz, pentru 1 secundă. Urmează apoi comanda:

digitalWrite(LED_1, LOW); se înscrie un semnal digital LOW pe pinul LED_1, deci 0 V și astfel LED-ul de pe placă se stinge. După aceea, urmează o nouă întârziere de 1 secundă:

delay(1000); După trecerea unei secunde, bucla se reia, deoarece funcţia loop este invocată din nou.

Astfel, secvenţa de cod prezentată nu face altceva decât să comute periodic între stările *LED aprins* şi *LED stins*.

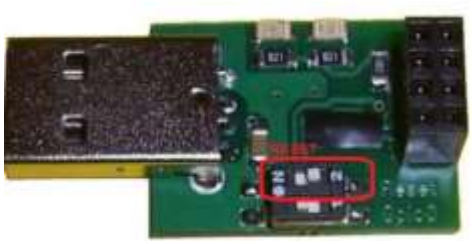
Compilarea, încărcarea şi rularea unei schiţe

Primul pas îl constituie conectarea plăcii la PC folosind un cablu *USB-A (Female) to USB-A cable (Male)* **(conectarea plăcii ESP-12 e diferită ca la Arduino Uno, de exemplu).**

Lansaţi apoi aplicaţia Arduino IDE şi placa va fi detectată în mod automat de IDE. Mergeţi în meniu la Tools | Port, unde ar trebui să vi se ofere posibilitatea de a selecta portul COM pe care este văzută placa de PC, de exemplu COM6. Dacă placa nu e detectată, închideţi IDE-ul şi porniţi din nou. După ce placa de test a fost detectată şi selectată cu succes, următorul pas este compilarea schiţei. Pentru aceasta, apăsaţi butonul marcat cu bifă şi veţi observa că IDE-ul începe compilarea schiţei. În caz că sunt erori, acestea sunt afişate în

consolă, în partea de jos a ferestrei ecranului IDE. La compilarea cu succes, IDE-ul va afişa un mesaj prin care se indică faptul că s-a realizat cu succes compilarea (va afişa şi cantitatea de memorie ocupată de schiţă). Arduino UNO are o memorie de 32KB, dar placa de test ESP are o memorie de 1MB, deci nu sunt probleme de optimizare a codului. ! Partajarea Flash Size recomandă ca 1 Mbyte să fie alocat programului și 3 Mbyte să fie alocați fișierelor de date. Numele fișierelor de date sunt în format 8.3. Numele fișierului are maxim 8 caractere, de ex. pisica_1, urmată de '.' și extensia 3 caractere, precum bmp, wav, mp3, txt etc.

După compilarea cu succes, codul compilat trebuie încărcat (upload-at) pe placă
Pentru aceasta, se va acţiona butonul cu săgeată, iar IDE-ul va începe să încarce codul compilat (cod hex) pe placă. **Cele 2 comutatoare de Reset şi GPIO0 sunt des acţionate ! înainte de fiecare *upload* trebuie acţionat RESET !**



Pentru încărcarea programului pe placă, sunt necesari 3 pasi:
1. Aduceți comutatorul 1 (**GPIO0**) în stare **ON** (la masă). ESP- este configurat astfel pentru încărcarea programului.
2. Resetați circuitul ESP: aduceți comutatorul **Reset** în stare **ON** (la masă) după care reveniț cu el în starea **OFF**.
3. Selectați opțiunea ***Upload*** din Arduino/Sketch/Upload.

Figura 7. Convertorul USB-UART. Ambele comutatoare sunt OFF

3. Mersul lucrării : *Aplicatii de iniţiere pe placa de test ESP-12*

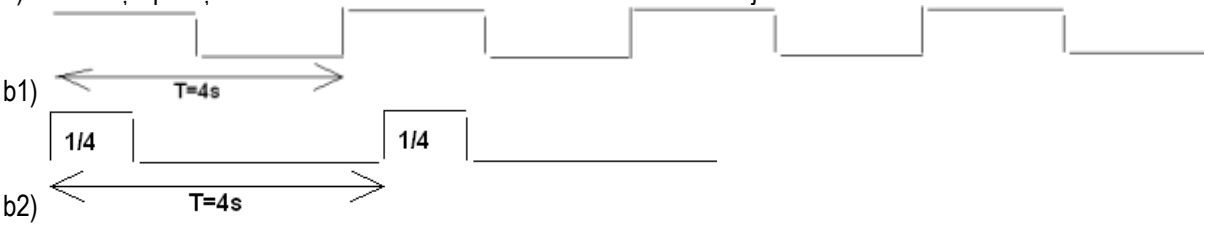
Înainte de a trece efectiv la rularea exemplelor, asiguraţi-vă că v-aţi însuşit următoarele:

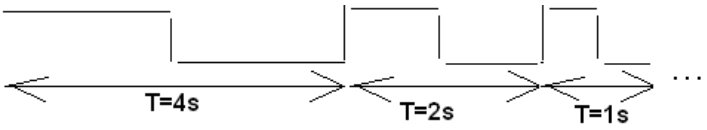
A. Pregătirea placii de test ESP-12 pentru lucru: **Placa de test** funcționează împreună cu **convertorul USB-UART** (figura 7) pentru ESP-01. Acesta asigură încărcarea, depanarea programelor şi alimentarea plăcii de test prin USB. Comutatoarele pinilor Reset şi GPIO0 sunt marcate în clar pe circuitul imprimat. Placa de test are conectorul pereche cu 8 pini montat pe partea inferioară. Atenţie la conectarea celor 2 module, să nu decalați din greșeală pinii ! La prima conectare, așteptați ca îndrumătorul de LAB să verifice conexiunile, după aceea intorduceți mufa USB în PC.

B. Configurarea mediului de dezvoltare Arduino:
1. Conectați convertorul USB-UART la placa test ESP-12
2. Deschideți mediul de dezvoltare Arduino.
3. Conectați un cablu prelungitor USB între PC și convertorul USB-UART.
4. Verificați numărul *COM port* identificat de PC în Device Manager/Ports și selectați același port în *Arduino/Tools/ Port.* Selectați Upload Speed 115200.
5. In Arduino/Tools verificați:
- **Flash Size: 4M(3M SPIFFS)**
- **Board: Generic ESP8266 Module**
- **FlashFrequency: 80MHz**
- **CPU Frequency: 160MHz (aşa cum arată Figura 5)**

C. Rularea exemplelor
Se vor consulta exemplele de iniţiere.
Fiecare exemplu descrie jumperele folosite: niciunul, unul singur, maxim 2. Singurele jumpere "nevinovate" sunt cele analogice, pinul ADC nefiind partajat.
!: Schema electrică trebuie sa fie întotdeauna în faţa dumneavoastră !
!: Cititi cu atenție notele informative de la începutul exemplelor. Aflați astfel dispozitivele auxiliare pe care le folosiți și cum să poziționați jumperele.
! La fiecare exemplu nou creat/ modificat, realizați un nou sketch. Pentru încărcarea programului pe placă, este obligatoriu să dați un RESET plăcii prin comutarea în ON și apoi în OFF a "SW2" de pe placuța cu convertorul USB-UART

3 A. Blink_led
a) Analizaţi programul şi specificați forma semnalului care controlează LED-ul în programul blink_led.ino.
b) Modificați aplicația astfel încât să realizeze formele de undă de mai jos:



b3) 

pana la T=0.5 sec inclusiv, după care se reia de la început, în buclă infinită

c) explicați ce formă de undă se va genera cu următoarea secvență de cod:

```
void loop()
{
  int i, val, n;
  for (i=0;i<5;i++)
 {
   val=5-i;
   n=200*val;
   digitalWrite(LED_1, HIGH);
   delay(n);
   digitalWrite(LED_1, LOW);
   delay(n);
  }
 }
```

d) analizând schema electrică din anexă, modificați programul astfel încât să scoată un ton de aproximativ 500Hz pe durata cât LED-ul luminează, iar pe durata cat LED-ul e stins, să "tacă" (să nu genereze sunet). Folosiţi funcţia digitalWrite.

3 B. blink led expander: repetaţi toate cerinţele de la a) … d). Care este efectul obţinut ?

3. C. Blink_led _pwm_sos: analizaţi codul şi explicaţi diferenţa între funcţiile DigitalWrite şi AnalogWrite.

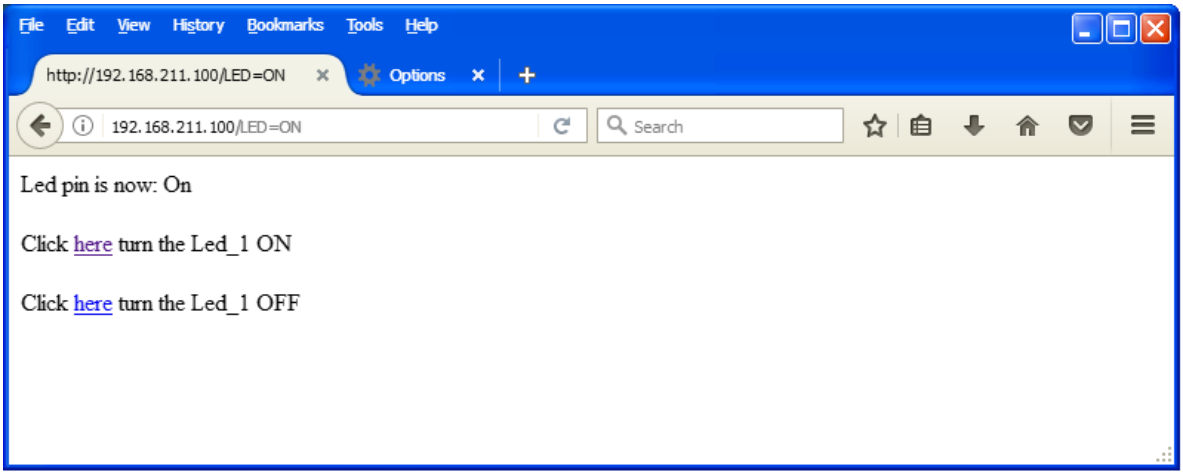
3. D. Analizaţi şi explicaţi **PWM_fade_led.**

3. E. Analizaţi şi explicaţi **simple_button.**

3. F. Analizaţi **send_serial_command** şi introduceţi voi o nouă comandă, de exemplu "stop" care să stingă LED_1.

3. G. Analizaţi aplicaţia **web_command_led** şi comunicaţi cu placa prin smartphone-ul personal.

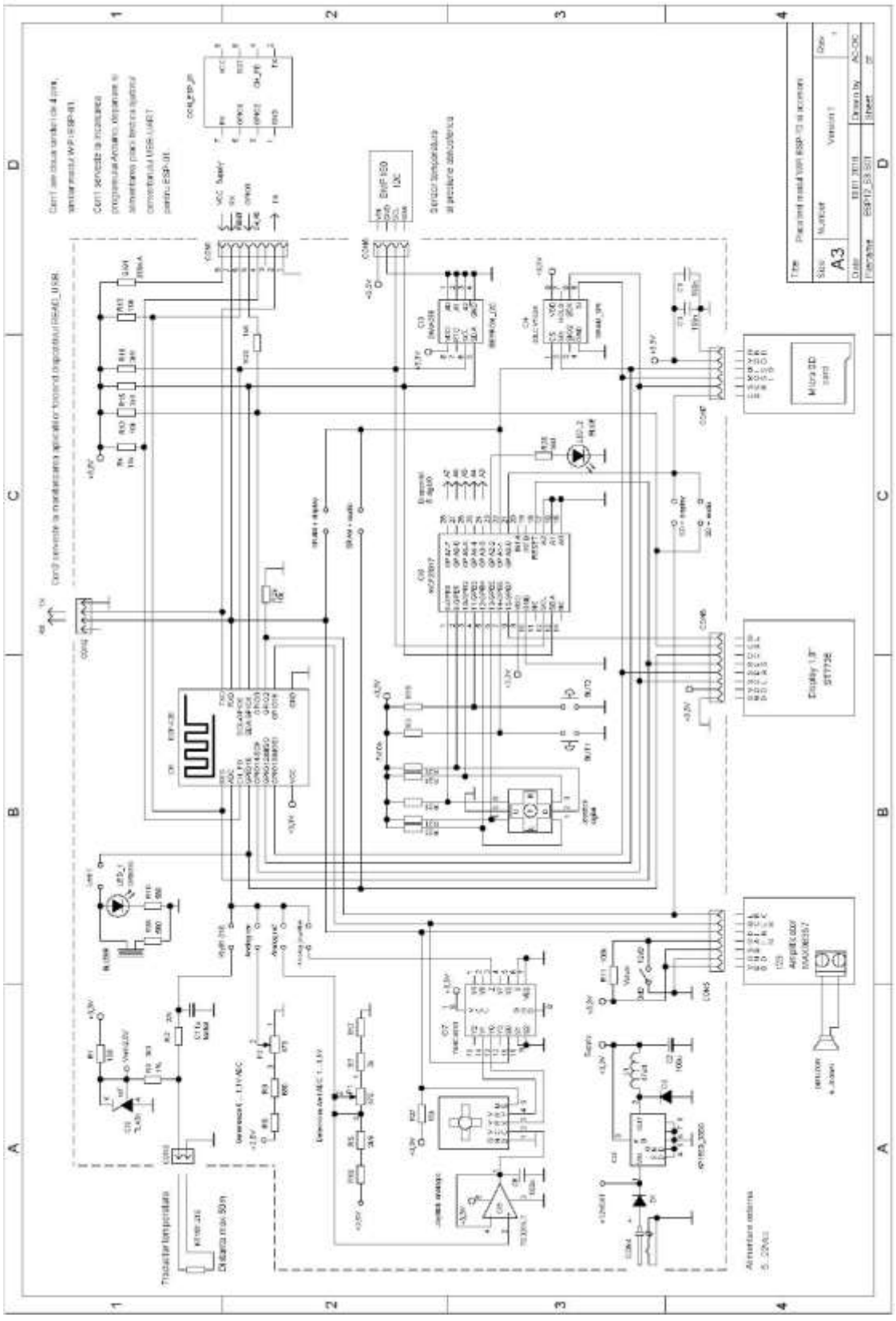
Pentru aceasta, ESP8266 e server web (placa va fi conectată la routerul 211 din sala pe baza de *ssid* și *password*) și deține pagina HTML. Comanda de aprindere / stingere a LEDului e lansată din browser de la un PC sau smart phone conectat la router în LAN sau WLAN. Adresa WLAN - ESP8266 o aflati în modul de lucru *run*, imediat dupa *reset*, pe monitorul serial Arduino, aşa cum se prezintă în exemplul de mai jos:



Anexe:

ANEXA 1: Informaţii despre placa de test

Schema electrică a plăcii



Placa de test cu WiFi ESP-12 facilitează studiul comunicaţiei prin SPI, I2C, I2S, deoarece prin aceste interfeţe, placa interacţionează cu **7 dispozitive performante**:

Prin **SPI:**	Prin **I2C:**	Prin **I2S:**
(Dev1) micro SD card	**(Dev4) traductor BMP180 pentru temperatura, presiune si altitudine**	**(Dev7) audio DAC MAX98357**
(Dev2) display ST7735	**(Dev5) expander MCP23017**	
(Dev3) memorie sram 23LCV1024	**(Dev6) memorie flash 24AA256**	

Funcţionarea simultană a dispozitivelor **(Dev5) expander MCP23017**, **(Dev1) micro SD card** , **(Dev2) display ST7735** şi **(Dev7) audio DAC MAX98357** necesită 11 pini GPIO, însă ***ESP-12*** oferă doar 10 ! ***Soluţia propusă*** este partajarea pinilor GPIO: **Audio MAX98357** elibereaza **3 pini** atunci cand lipseste, iar **display ST7735** elibereaza **2 pini**.
Procedura i2s cu **MAX98357** si procedura grafica cu **ST7735** sunt foarte rapide. Atunci când aceste dispozitive folosesc fişiere în SD card, cer ca şi SD card să fie rapid. Funcţionarea e posibilă doar atunci când pinii CS (chip select) sunt pini rapizi GPIO.

E important să folosiţi corect jumperele: (**J1**) Kty81-210, (**J2**) Analog var, (**J3**) Analog ref, (**J4**) Analog joystick, (**J5**) Led 1, (**J6**) SD + display, (**J7**) SD + audio, (**J8**) SRAM + display, (**J9**) SRAM + audio. Aceste jumpere sunt marcate în clar pe circuitul imprimat, de exemplu:

Pinii GPIO ai modulului ESP-12: ESP-12 are 16 pini. Pinii Gnd, Vcc, Ch_Pd, Reset, Txd si ADC au intrebuintare unica. Raman doar 10 pini la dispozitia utilizatorului. Pinii comunicatiilor spi, i2c, i2s si uart sunt neconfigurabili.

Pinii **SPI** :	Pinii **I2C**:	Pinii **I2S**:	Pinii **UART**:
• SCK pin GPIO14	• SCL pin GPIO5	• LRC pin GPIO2	• RXD pin GPIO3
• MISO pin GPIO12	• SDA pin GPIO4	• BCLK pin GPIO15	• TXD pin GPIO1
• MOSI pin GPIO13		• DIN pin GPIO3, partajat cu RXD	

Pinii expandati de i2c MCP23017: Circuitul expander i2c MCP23017 asigura 16 pini suplimentari dig I/O pe porturile A si B. Numerotarea pinilor se face antiorar, incepand de la GPA0 (pinul 0) pana la GPB7 (pinul 15). ! Interfata i2c e emulata software (bit bitbanging) si lucreaza la 400kHz. Timpul de raspuns e 200µs. Pentru comparatie, pinii rapizi GPIO raspund in 5µs.

Sunt **7 intrari expandate:**	Sunt **4 iesiri expandate:**	Raman **5 pini expandati**, disponibili pentru utilizari viitoare.
• BUTON_1 13, • BUTON_2 14, • DIG_JOY_1 12,	• CS_SD_EXP 0, • CS_RAM_EXP 1	
• DIG_JOY_2 9, • DIG_JOY_3 10, • DIG_JOY_4 13	• LED_2 2, • LCD_BACKLIGHT 5	
• DIG_JOY_6 8		

Cele trei dispozitive **(Dev1) micro SD card** , **(Dev2) display ST7735** şi **(Dev7) audio DAC MAX98357** pot functiona simultan atunci când **display ST7735** si **audio MAX98357** nu folosesc fişiere salvate în **SD card**. Pot folosi fişiere salvate în **flash**, în **memoria program** şi **server web http**. In acest fel, **SD card** foloseşte CS expandat şi îndeplineşte alte activităţi, de ex. achiziţie date, scrierea/ citirea valorilor în fisiere .txt, etc. Atunci când unul din dispozitivele **display ST7735** sau **audio MAX98357** folosesc fişiere salvate în **SD card**, programul atribuie automat un pin rapid GPIO pentru CS al SD card:
• Chip select SD+audio atunci când folosiţi SD card şi audio MAX98357.
• Chip select SD+display atunci când folosiţi SD card şi display ST7735.

! Jumperele SD+audio şi SD+display trebuie poziţionate corespunzător acestor situaţii !

ANEXA 2: *Instalarea elementelor necesare funcţionării plăcii de test*

Manualul convertorului USB-UART pentru ESP-01 (aflat la adresa **http://www.acdcelectronics.ro/convertor_usb_uart_pentru_esp_01.pdf**) arata detaliat modul de instalare al mediului de dezvoltare Arduino IDE si al librariilor ESP8266 pentru funcţionarea aplicaţiilor propuse pe **Placa de test cu WiFi ESP-12**. Acest pas a fost deja parcurs pentru lucrarea de laborator curentă: Arduino IDE a fost instalat împreună cu librăriile, uneltele şi programele de test necesare. **Pentru aceasta, s-a descărcat arhiva placa_test_esp12 de pe site, s-a copiat in radacina Arduino directorul placa_test_esp12,** şi apoi s-au instalat pe rând librăriile, uneltele, programele aşa cum se prezintă mai jos.

a) Instalarea librariilor auxiliare

Directorul placa_test_esp12\anexe\librarii are 11 librarii auxiliare:
• Adafruit-GFX-Library-master, • Adafruit-MCP23017-Arduino-Library-master,
• Adafruit-ST7735-Library-master, • BMP180-Breakout_Arduino-Library-master,
• ESP-12_TestBoard, • ESP8266_Spiram-master, • ESP8266Audio-master, • Morse,
• SD_expander, • SpiRAM_expander, • Time-master
S-au copiat aceste librarii in directorul *Arduino\libraries*.
ESP12TestBoard.h e libraria proprie a placii test si cuprinde procedura expander MCP23017, definitii pentru butoane, leduri, chip select etc.
Procedura expander MCP23017 este apelata atat in programul sursa cat si in librariile modificate SDexp.h si SpiRAMexp.h.
Modificarile librariilor SDexp.h si SpiRAMexp.h sunt minore si se refera doar la atribuirea chip select expandat pentru SD card si spi SRAM atunci cand este nevoie.
Atunci cand aceste dispozitive folosesc chip select pini rapizi GPIO, librariile modificate functioneaza in forma originala a librariilor SD.h si SpiRAM.h.

b) Instalarea uneltelor

Directorul placa_test_esp12\anexe \tools cuprinde 2 unelte necesare. S-au copiat ESP8266FS si EspException Decoder in *Arduino\tools*.

c) Instalarea/ copierea exemplelor de test

S-au copiat toate exemplele în *Arduino\examples*. Pentru lucrarea curentă, se vor parcurge *exemplele de iniţiere*

Bibliografie

1. **http://www.acdcelectronics.ro/manual_utilizare_placa_test_esp-12.pdf**
2. **http://www.acdcelectronics.ro/convertor_usb_uart_pentru_esp_01.pdf**
3. Arduino Home Page, www.arduino.cc
4. Arduino Language Reference, http://www.arduino.cc/en/Reference/HomePage
5. Adeel Javed, Building Arduino projects for the Internet of Things, Apres 2016, ISBN-13 (pbk): 978-1-4842-1939-3
6. G.Toderean, O.Buza, A.Balogh – “Aplicaţii cu microcontrolere”, Risoprint 2016
7. ESP 8266
8. https://tttapa.github.io/ESP8266/Chap01%20-%20ESP8266.html
9. https://blog.robofun.ro/2018/03/06/cum-sa-realizam-un-repetor-wifi-utilizand-esp8266/
10. https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/using-arduino-ide
11. https://www.esp8266.com/
12. https://github.com/esp8266/Arduino
13. https://www.espressif.com/en/support/download/other-tools?keys=&field_type_tid%5B%5D=14