



Proiectare cu microprocesoare:
Proiectarea unei matrice LED si controlarea acesteia
wireless cu ajutorul modulului ESP8266
Proiect de laborator

Cristian Serbu, Grupa 30235
Email: cristi.serbu17@gmail.com

Profesor indrumator: Razvan Itu



Contents

1	Introducere & Componente	3
2	Implementare	4
2.1	Crearea matricii	4
2.2	Controlarea cu Arduino	7
2.3	adaugarea ESP8266	11
3	Rezultat final si concluzii	12

Chapter 1

Introducere & Componente

In acest proiect am proiectat o matrice de leduri adresabile individual, de la 0, pe care o voi programa apoi cu diferite moduri de iluminat, folosind o placuta Arduino. Am folosit apoi un modul Wi-Fi ESP8266, pentru a adauga control wireless asupra matricei. Pentru demonstrarea principiilor pe care urmeaza sa le folosesc, am ales dimensiunea matricii de 5 linii si 8 coloane, ceea ce ne duce la un numar total de 40 de leduri.

Componentele necesare pentru realizarea acestui proiect sunt:

- 40 de leduri (am ales leduri albastre)
- bucata PCB pentru lipirea ledurilor
- un circuit integrat 74HC595 (8 bit shift register)
- 5 tranzistori npn 2N2222
- rezistente de diferite valori (6x 220 ohmi, 1x 470 ohmi)
- un condensator ceramic de 1 nF
- un Arduino Uno
- un modul ESP8266
- breadboard si fire jumper
- fir de cupru neizolat
- fier de lipit, fludor si sacaz

Chapter 2

Implementare

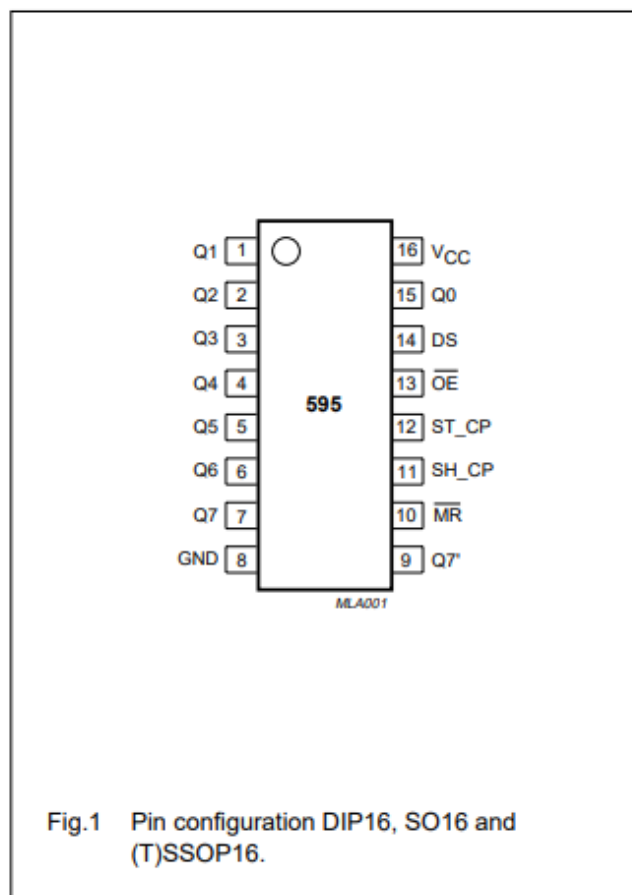
2.1 Crearea matricii

Am stabilit deja ca matricea va fi constituita de 40 de leduri lipite pe un pcb, iar ledurile vor fi dispuse sub forma a 5 randuri si 8 coloane. Ledurile vor avea anod comun pe coloana si catod comun pe rand, adica anozii de pe fiecare coloana vor fi lipite impreuna, iar catozii de pe fiecare din cele 5 randuri vor fi lipiti impreuna. Astfel, vom obtine o matrice led adresabila individual. Modul de functionare: Aplicand 5v la capatul unei coloane si ground la capatul unui rand, ledul aflat la intersectia celor 2 se va aprinde. De exemplu, daca vom aplica 5v pe prima coloana si ground pe primul rand, ledul din coltul stanga sus se va aprinde. Daca aplicam 5v pe toate coloanele matricei, si legam un singur rand la ground, tot randul respectiv se va aprinde.

Acum ca avem o matrice functionala, vrem sa afisam o animatie, sau litere pentru a afisa un mesaj. Pentru asta, vom lega periodic, pe rand, cate un rand al matricei la ground, si vom aplica 5v pe coloanele pe care ne dorim sa le aprindem. Vom pacali ochiul folosindu-ne de persistenta viziunii: ochiul si creierul uman pot sa proceseze undeva intre 10 si 12 imagini pe secunda, insemnand ca daca vom repeta ciclul de legare a randurilor la ground de mai mult de 12 ori pe secunda, pentru ochiul uman va parea ca ledurile sunt aprinse tot timpul, desi ele sunt aprinse pentru o fractiune de secunda. Aceasta modalitate de iluminat este de asemenea eficienta din punctul de vedere al economisirii electricitatii.

Coloanele vor fi adresate cu ajutorul unui circuit integrat 74hc595, un shift register de 8 biti. Fiecare din cele 8 outputuri va fi legat la cate o coloana a matricei. Ca sa stim care sunt pinii de output ai circuitului integrat, ne vom uita la pinoutul din datasheetul sau.

Din imaginea de mai jos reiese ca pinii 1-7 sunt pinii de output la care trebuie conectate coloanele matricei. De asemenea pinul 8 este groundul, si pinul 16 este VCC, fiind pinii de alimentare. Alti pini importanti pentru noi sunt cel de date, pinul 14, si cel de clock pentru bistabile de registru, pinul 12. Circuitul 74HC595 are si un registru intermediar de stocare, iar pinul pentru ceasul acestuia este pinul 11. Intre vcc si ground voi pune un condensator ceramic pentru a filtra zgomotul venind de la sursa de alimentare.




2003 Jun 25

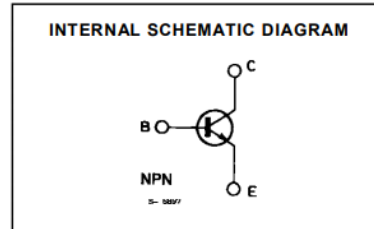
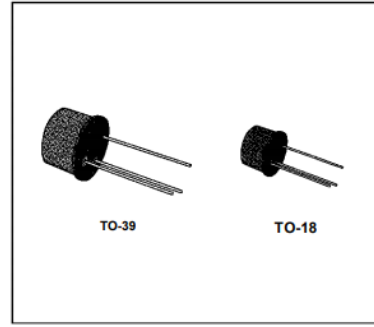
4

Fiecare rand al matricei va fi conectat colectorul unui tranzistor npn, in timp ce emitorul tranzistorilor va fi legat la ground. legarea pe rand a randurilor la ground, precum si scrierea de date pe shift register vor fi realizate de catre Arduino. Am ales modelul 2N2222 din 2 motive: Trebuie sa poata sa-si schimbe starea foarte repede, si trebuie sa poata face fata la curentul ledurilor aprinse. Aruncand o privire la datasheetul acestuia, observam ca 2N2222 are o frecventa foarte mare de switching, si poate sa faca fata la un curent pe colector de pana la 800 mA ($8 \text{ leduri} * 10\text{mA} = 80\text{mA}$, de 10 ori mai putin decat curentul maxim indicat). Aceste lucruri il fac alegerea ideala in caz ca dorim sa adaugam mai multe coloane sau/si randuri de leduri in viitor.

DESCRIPTION

The 2N2218, 2N2219, 2N2221 and 2N2222 are silicon planar epitaxial NPN transistors in Jedec TO-39 (for 2N2218 and 2N2219) and in Jedec TO-18 (for 2N2221 and 2N2222) metal cases. They are designed for high-speed switching applications at collector currents up to 500 mA, and feature useful current gain over a wide range of collector current, low leakage currents and low saturation voltages.

 2N2218/2N2219 approved to CECC 50002-100, 2N2221/2N2222 approved to CECC 50002-101 available on request.



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CB0}	Collector-base Voltage ($I_E = 0$)	60	V
V_{CE0}	Collector-emitter Voltage ($I_B = 0$)	30	V
V_{EB0}	Emitter-base Voltage ($I_C = 0$)	5	V
I_C	Collector Current	0.8	A
P_{tot}	Total Power Dissipation at $T_{amb} \leq 25^\circ\text{C}$ for 2N2218 and 2N2219	0.8	W
	for 2N2221 and 2N2222	0.5	W
	at $T_{case} \leq 25^\circ\text{C}$ for 2N2218 and 2N2219	3	W
	for 2N2221 and 2N2222	1.8	W
T_{stg}	Storage Temperature	- 65 to 200	$^\circ\text{C}$
T_J	Junction Temperature	175	$^\circ\text{C}$

ELECTRICAL CHARACTERISTICS ($T_{amb} = 25^\circ\text{C}$ unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
I_{CBO}	Collector Cutoff Current ($I_E = 0$)	$V_{CB} = 50\text{ V}$ $V_{CB} = 50\text{ V}$ $T_{amb} = 150^\circ\text{C}$			10 10	nA μA
I_{EBO}	Emitter Cutoff Current ($I_C = 0$)	$V_{EB} = 3\text{ V}$			10	nA
$V_{(BR)CBO}$	Collector-base Breakdown Voltage ($I_E = 0$)	$I_C = 10\text{ }\mu\text{A}$	60			V
$V_{(BR)CEO}^*$	Collector-emitter Breakdown Voltage ($I_B = 0$)	$I_C = 10\text{ mA}$	30			V
$V_{(BR)EBO}$	Emitter-base Breakdown Voltage ($I_C = 0$)	$I_E = 10\text{ }\mu\text{A}$	5			V
$V_{CE(sat)}^*$	Collector-emitter Saturation Voltage	$I_C = 150\text{ mA}$ $I_B = 15\text{ mA}$ $I_C = 500\text{ mA}$ $I_B = 50\text{ mA}$			0.4 1.6	V V
$V_{BE(sat)}^*$	Base-emitter Saturation Voltage	$I_C = 150\text{ mA}$ $I_B = 15\text{ mA}$ $I_C = 500\text{ mA}$ $I_B = 50\text{ mA}$			1.3 2.6	V V
h_{FE}^*	DC Current Gain	for 2N2218 and 2N2221 $I_C = 0.1\text{ mA}$ $V_{CE} = 10\text{ V}$ $I_C = 1\text{ mA}$ $V_{CE} = 10\text{ V}$ $I_C = 10\text{ mA}$ $V_{CE} = 10\text{ V}$ $I_C = 150\text{ mA}$ $V_{CE} = 10\text{ V}$ $I_C = 500\text{ mA}$ $V_{CE} = 10\text{ V}$ $I_C = 150\text{ mA}$ $V_{CE} = 1\text{ V}$ for 2N2219 and 2N2222 $I_C = 0.1\text{ mA}$ $V_{CE} = 10\text{ V}$ $I_C = 1\text{ mA}$ $V_{CE} = 10\text{ V}$ $I_C = 10\text{ mA}$ $V_{CE} = 10\text{ V}$ $I_C = 150\text{ mA}$ $V_{CE} = 10\text{ V}$ $I_C = 500\text{ mA}$ $V_{CE} = 10\text{ V}$ $I_C = 150\text{ mA}$ $V_{CE} = 1\text{ V}$	20 25 35 40 20 20		120	
f_T	Transition Frequency	$I_C = 20\text{ mA}$ $V_{CE} = 20\text{ V}$ $f = 100\text{ MHz}$	250			MHz
C_{CBO}	Collector-base Capacitance	$I_E = 0$ $f = 100\text{ kHz}$ $V_{CB} = 10\text{ V}$			8	pF
$R_{e(hie)}$	Real Part of Input Impedance	$I_C = 20\text{ mA}$ $f = 300\text{ MHz}$ $V_{CE} = 20\text{ V}$			60	Ω

* Pulsed : pulse duration = 300 μs , duty cycle = 1 %.

2.2 Controlarea cu Arduino

Dupa ce am creat matricea de leduri si i-am atasat si tranzistorii si circuitul integrat, este timpul sa facem legaturile la Arduino. Datele le vom trimite catre 74HC595 in mod serial, prin interfata SPI. Interfata SPI pentru Arduino este descrisa pe <https://www.arduino.cc/en/reference/SPI>. Ce ne intereseaza pe noi este sa trimitem date de la Master (placuta Arduino) la slave (74HC595). Deci ne vor trebui pinii MOSI (master out slave in) si SCKL (serial clock), care sunt pinii 11, respectiv 13 pe Arduino.

Connections

The following table display on which pins the SPI lines are broken out on the different Arduino boards:

Arduino / Genuino Board	MOSI	MISO	SCK	SS (slave)	SS (master)	Level
Uno or Duemilanove	11 or ICSP- 4	12 or ICSP- 1	13 or ICSP- 3	10	-	5V
Mega1280 or Mega2560	51 or ICSP- 4	50 or ICSP- 1	52 or ICSP- 3	53	-	5V
Leonardo	ICSP- 4	ICSP- 1	ICSP- 3	-	-	5V
Due	SPI-4	SPI-1	SPI-3	-	4, 10, 52	3,3V
Zero	ICSP- 4	ICSP- 1	ICSP- 3	-	-	3,3V
101	11 or ICSP- 4	12 or ICSP- 1	13 or ICSP- 3	10	10	3,3V
MKR1000	8	10	9	-	-	3,3V

Daca aruncam o privire la datasheetul 74hc595 observam ca pinul de date este pinul 14, iar pinul de clock este pinul 12. Dupa ce facem conexiunile necesare, trecem la conectarea bazelor tranzistorilor la Arduino. Ne uitam in datasheetul 2N2222 la Base-emitter Saturation Voltage si facem conexiunea intre placuta si baza tranzistorului print-o rezistenta de 220ohmi pentru a atinge tensiunea dorita.

Am folosit pinii 3, 4, 5, 6, si 7 de pe Arduino pentru controlul randurilor. Acum ca avem Arduino conectat la matricea noastra, este timpul sa scriem functia care scrie date pe matrice, si functii pentru animatii ale matricei.

```
1 void scrie(){
2   for(int i=3;i<8;i++){
3     digitalWrite(slaveSelect, LOW);
4     SPI.transfer(row[i-3]);
5     digitalWrite(slaveSelect, HIGH);
6     digitalWrite(i,HIGH);
7     //delay(1);
8     digitalWrite(i,LOW);
9   }
10 }
```

Functia scrie() trimite datele din vectorul row in mod serial cu ajutorul functiei SPI.transfer, apoi face pinul respectiv randului pe care vrem sa il afisam HIGH, pentru a face conexiune cu

groundul prin tranzistor, urmand sa traga acel pin LOW imediat. Acel delay() a fost comentat deoarece dupa ce am conectat modulul ESP8266, cauza un efect de flickering, care a fost inlaturat doar de inlaturarea delayului. Vectorul row are 5 elemente, fiecare corespunzand cate unui rand: row[0] = primul rand, row[1] = al doilea rand. etc. Cei 8 biti ce vor fi trimisi sunt cei mai putin semnificativi 8 biti, iar de-a lungul programului am lucrat cu datele in hexa, ca sa fie mai usor de scris si inteles (0x83 e mult mai usor de inteles decat 0b10000011)

```
1 void set( unsigned long a[5]){
2     for(int ij =0;ij<5;ij++)
3         row[ij]=a[ij];
4 }
5
6 void checkit(int count){
7
8     if(row[count]==0x80){
9         if(count==0)
10             row[4]=0x80;
11         else
12             row[count-1]=0x80;
13         row[count]=0x00;
14     }
15 }
16
17
18 void sterge(){
19     for(int i=0;i<5;i++)
20         row[i]=0x00;
21 }
22
23 void aprinde(){
24     //row[4]=0x40;
25     for(int i=0;i<5;i++)
26         row[i]=0xff;
27 }
28
29 void start(){
30     if(millis()-StartTime>200UL){
31         StartTime=millis();
32         if(on==1)
33         {
34             aprinde();
35             on=0;
36         }
37         else if(on==0)
38         {
39             sterge();
40             on=1;
41         }
42     }
43 }
44
45 void fillRandom(){
46     int i = random(0,5);
47     int j=random(0,8);
48     int ok=1;
49     int rcount=0;
50     while(ok==1){
51         if(row[i]==0xff)
52         {
```



```

53     i++;
54     if(i==5) i=0;
55     rcount++;
56 }
57 if(rcount==5)
58 ok=0;
59 if((row[i] & (0x01<<j)) ==0){
60     ok=0;
61 }
62 else
63 {
64     j++;
65     if(j==8)
66     j=0;
67 }
68 }
69 row[i]=row[i]|0x01<<j;
70 }
71
72
73 void emptyRandom(){
74     int i = random(0,5);
75     int j=random(0,8);
76     int ok=1;
77     int rcount=0;
78     while(ok==1){
79         if(row[i]==0x00)
80         {
81             i++;
82             if(i==5) i=0;
83             rcount++;
84         }
85         for(int k=0;k<5;k++)
86
87             if(rcount==5)
88             ok=0;
89             if((row[i] & (0x01<<j))) {
90                 ok=0;
91             }
92             else
93             {
94                 j++;
95                 if(j==8)
96                 j=0;
97             }
98         }
99         if(row[0] || row[1] || row[2] || row[3] || row[4])
100 row[i]=row[i]^(0x01<<j);
101     }
102
103 void plimbare(){
104     if(semnal)
105     {
106         row[count]=0x01;
107         semnal=0;
108     }
109     else{
110         row[count-1]=0x00;
111         if(count==0)
112         row[4]=0x00;

```

```

113 row[count]=row[count]<<1;
114 if(semnals)
115 {
116     count++;
117     if(count==5)
118         count=0;
119     row[count]=0x01;
120     semnals=0;
121 }
122 if(row[count]==0x80){
123     if(count==0)
124         row[4]=0x80;
125     else
126         row[count-1]=0x80;
127     row[count]=0x00;
128     semnals=1;
129 }
130 }
131 }

```

Funcția set() seteaza vectorul row la o valoare predefinita. Am creat variabile pentru fiecare litera din alfabet, si prin repetate apelari ale functiei set putem sa scriem mesaje. **Funcția sterge()** seteaza fiecare rand la 0x00. **Funcția aprinde()** seteaza fiecare rand la 0xff, aprinzand efectiv fiecare led de pe fiecare coloana. **Funcția start()** apeleaza functiile sterge() si aprinde() intermitent, la interval de 200 de milisecunde (de 5 ori pe secunda).

Funcția fillRandom() vrea sa gaseasca si sa aprinda un singur led, iar prin apelari repetate putem umple toata matricea in mod random intr-un interval variabil. Mai intai alege un rand si o coloana random, si apoi daca randul respectiv e plin, incrementam variabila de rand, pana gasim unul care e gol. Daca ledul de la pozitia j de la randul gasit este stins, il aprindem, altfel incrementam variabila coloana pana gasim un led stins. Aprinderea ledului se face cu un sau pe biti intre randul ales si 0x01 shiftat de un numar de ori egal cu variabila coloana gasita.

Funcția emptyRandom() face exact opusul, si stinge un led care era aprins, facand un xor pe biti intre randul ales si 0x01 shiftat la dreapta cu un nr de pozitii egal cu coloana aleasa.

Funcția plimbare() initializeaza un rand cu 0x01, adica un singur led aprins in toata matricea, apoi shifteaza aceasta valoare la dreapta pana la valoarea 0x08. Cand am ajuns la aceasta valoare, in loc de shiftare, ii dam valoarea 0x00 acestui rand si valoarea 0x01 randului urmator. Astfel prin apelari repetate, plimbam un singur led de la dreapta la stanga, prin toata matricea.

```

1
2 void setup()
3 {
4     pinMode(slaveSelect, OUTPUT);
5     SPI.begin();
6     StartTime = millis();
7     SPI.beginTransaction(SPISettings(10000000, LSBFIRST, SPI_MODE0));
8
9     pinMode(3, OUTPUT);
10    pinMode(4, OUTPUT);
11    pinMode(5, OUTPUT);
12    pinMode(6, OUTPUT);
13    pinMode(7, OUTPUT);
14    pinMode(9, OUTPUT);
15
16    randomSeed(analogRead(0));
17 }

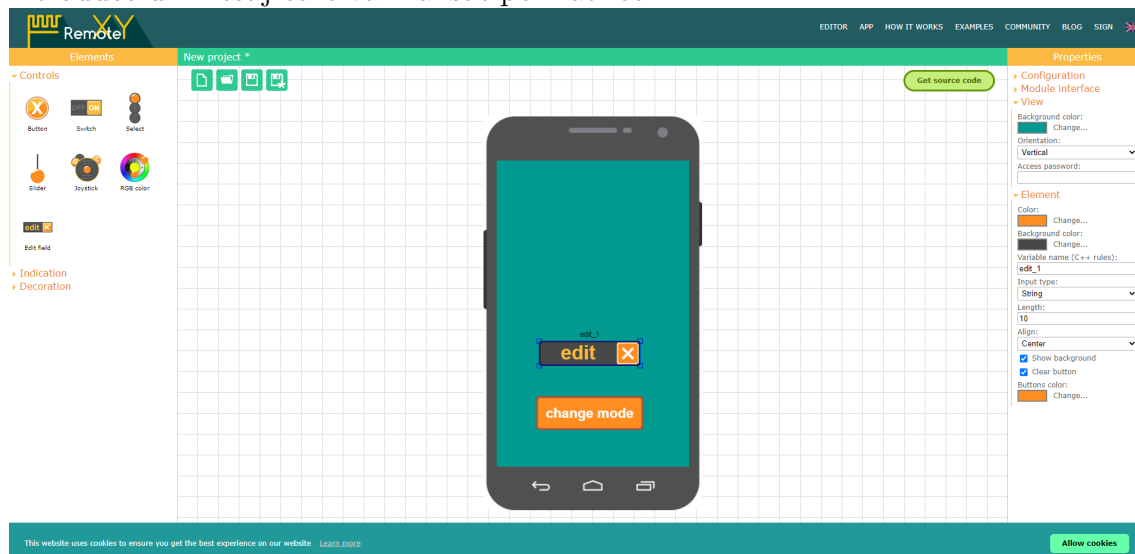
```

Pana acum, functia setup arata asa, protocolul SPI este initializat, pinii 3-7 sunt setati ca si

output, dar vreau sa evidentiez cum am reusit sa obtin valori random adevarate: daca la functia randomSeed se da un numar oarecare, de exemplu 42, se va genera un sir infinit de numere random pe baza acestui numar, doar ca va fi generat acelasi sir de numere la fiecare rulare a programului. Cum nu am dorit acest efect pseudo-random, cu functia analogRead pe un pin analog voi obtine mereu un numar diferit, si in consecinta, siruri de numere random diferite.

2.3 adaugarea ESP8266

Modulul ESP este sensibil la 5v, tensiunea sa de alimnetare este de 3.3v, asa ca va fi alimentat din iesirea de 3.3v de la Arduino. Pentru partea de cod, voi folosi libraria RemoteXY pentru generarea codului si a unei aplicatii pe telefon cu o interfata grafica user friendly. Butonul de change v-a schimba intre animatiile matricei, iar in casuta de deasupra utilizatorul poate introduce un mesaj care va fi afisat pe matrice.



Pentru comunicare intre Arduino si ESP vom lega pinul TX de la ESP la RX de la Arduino si RX de la ESP la TX de la Arduino. Pinul TX de la Arduino este trecut printr-un divizor de tensiune format din 2 rezistente de 220 ohm, respectiv 470 ohm pentru a reduce tensiunea de la 5v la 3v. De asemenea, legam pinul de enable de la ESP la 3.3v.

Enter any three known values and press "Calculate" to solve for the other.

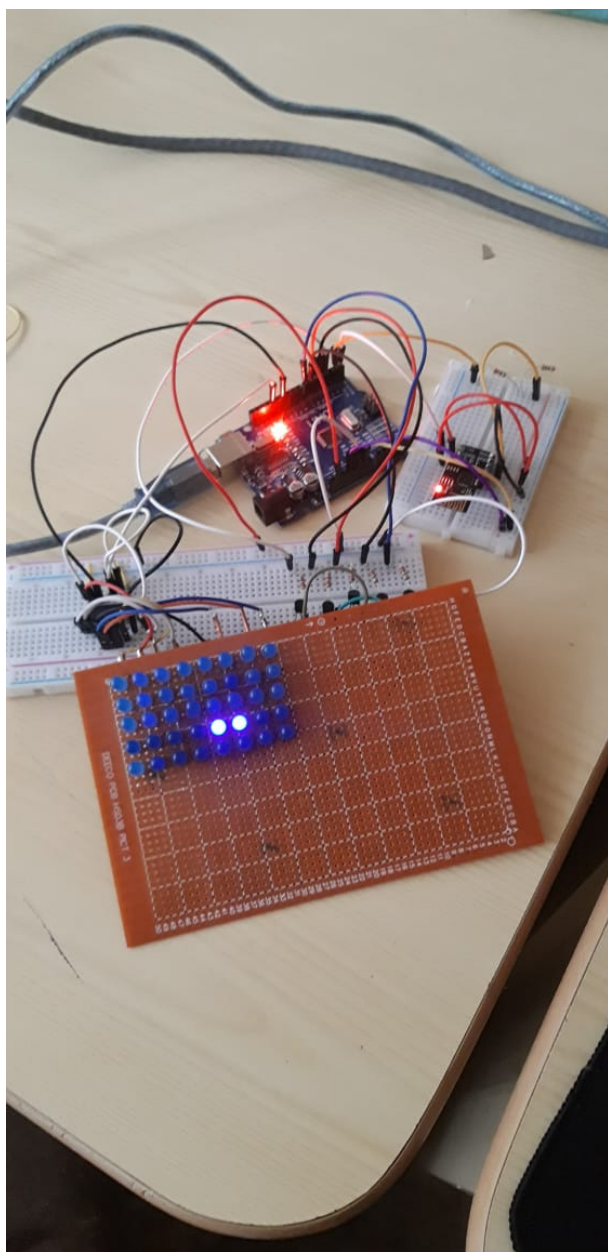
Voltage Source (V_s)	<input type="text" value="5"/>	Volts (V)
Resistance 1 (R_1)	<input type="text" value="220"/>	ohms (Ω)
Resistance 2 (R_2)	<input type="text" value="470"/>	ohms (Ω)
Output Voltage (V_{out})	<input type="text" value="3.486"/>	Volts (V)

Click "Calculate" to update the field with orange border.

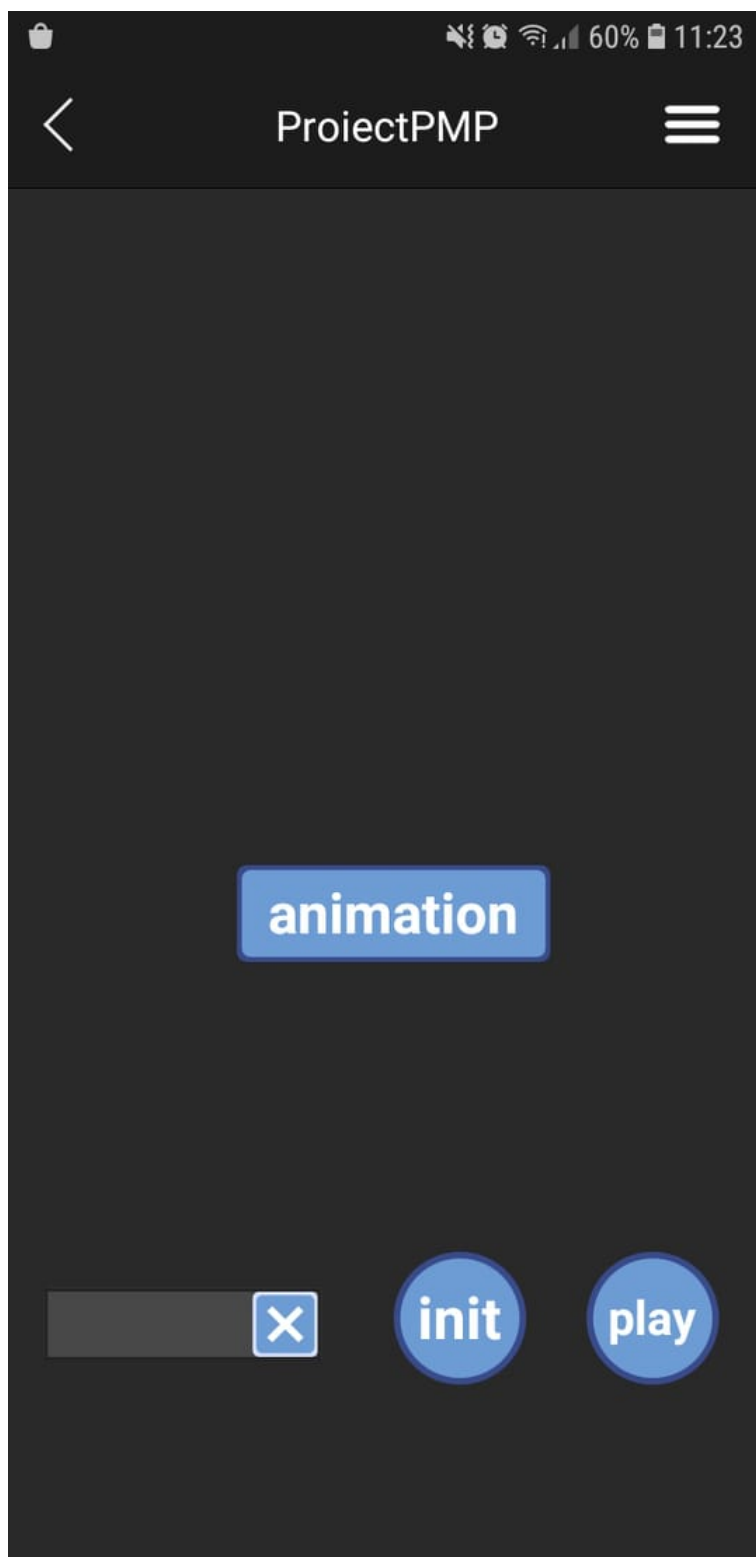
Chapter 3

Rezultat final si concluzii

Produsul final este o matrice led de 8x5 adresabila individual, programata cu cateva animatii si controlabila Wireless.



Aplicatia pe telefon este generata de codul RemoteXY si arata asa:



In concluzie, a fost un proiect interesant, distractiv, a presupus multe provocari si multe concepte noi si pot spune ca sunt complet multumit de rezultatul final. Matricea se poate extinde in continuare prin adaugarea mai multor linii si coloane de leduri, a mai multor tranzistori si a mai multor circuite 74HC595. De asemenea se pot programa in continuare mai multe animatii, si se pot aduce imbunatatiri animatiilor prezente.

