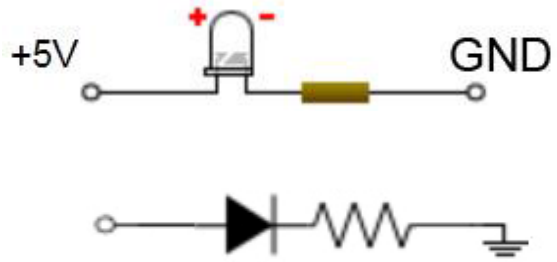
Laborator 3
Utilizarea pinilor I/O pentru aprinderea LED-urilor

Secţiuni lucrare:

Echipamente necesare lucrării:

- Placă de dezvoltare Nucleo F103RB
- Cablu de conectare USB
- Breadboard + fire de interconectare
- 8 leduri + 8 rezistori 220ohm
- 1 shift-register pe 8 biți (TPIC6B595)

Conectarea unui led la un pin I/O



Una dintre cele mai simple modalități de utilizare a unui pin I/O este comanda unui element de tip LED. Pentru a putea aprinde un led este necesară inserarea unui rezistor pentru limitarea curentului.

Calcularea valorii rezistorului se face în funcție de parametrii electrici ai ledului: curentul de aprindere și tensiunea de alimentare. Inserarea unui rezistor mai mare va scădea intensitatea de aprindere a ledului, inserarea unui rezistor mai mic va forța ledul la intensitate maximă scăzându-i durata de viață.

Citirea valorii unui rezistor se face pe baza codului de bare colorate înscrise pe rezistor. Benzile 1, 2 și 3 (numai în cazul codului cu 5 bare) reprezintă cele mai reprezentative cifre ale valorii, banda 3 multiplicatorul valorii și banda 4 toleranța. Mai jos avem un tabel cu codul culorilor pentru rezistori.

Culoare	Banda 1	Banda 2	Banda 3	Banda 4 (toleranță)
Negru	0	0	X 1	
Maro	1	1	X 10	
Roșu	2	2	X 100	
Portocaliu	3	3	X 1,000	
Galben	4	4	X 10,000	
Verde	5	5	X 100,000	
Albastru	6	6	X 10^6	
Violet	7	7	X 10^7	
Gri	8	8	X 10^8	
Alb	9	9	X 10^9	
Auriu			X 0.1	5%
Argintiu			X 0.01	10%
Fără culoare				20%

Tabel 1. Codul culorilor pentru rezistențe

Proiect joc de lumini (conectare individuală)

Conectarea LED-urilor

Presupunem următoarea schemă de interconectare a 8 leduri cu placa de dezvoltare Nucleo F103RB în mod individual – fiecare led este comandat în mod direct de câte un pin al plăcii de dezvoltare (led 1 – pin 2, led 2 – pin 3…. led 8 – pin 9):

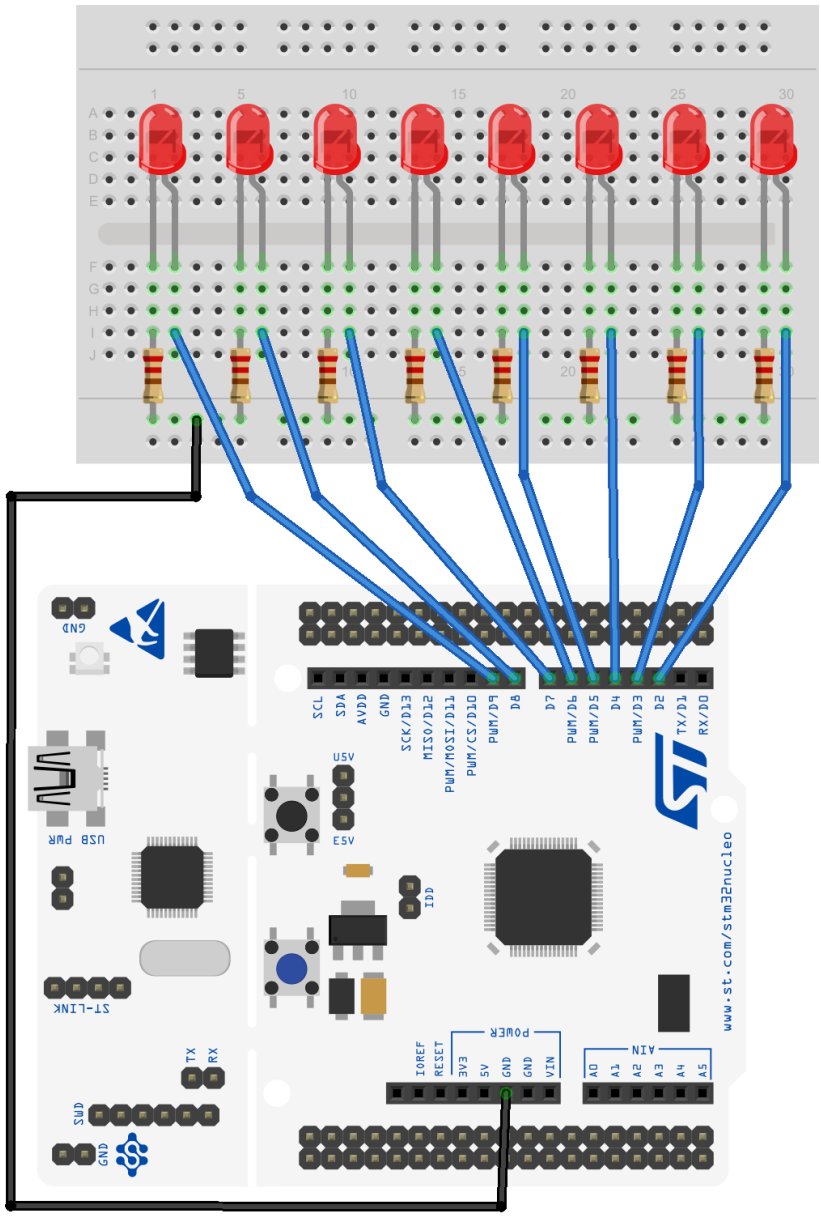


Figura 1. Schema de interconectare

Pentru a putea conecta corect dispozitivele la pinii unei plăci, trebuie să se urmarească funcţiile pinilor în foile de catalog. În figura 2 se poate observa asignarea pinilor pe placa Nucleo F103RB.

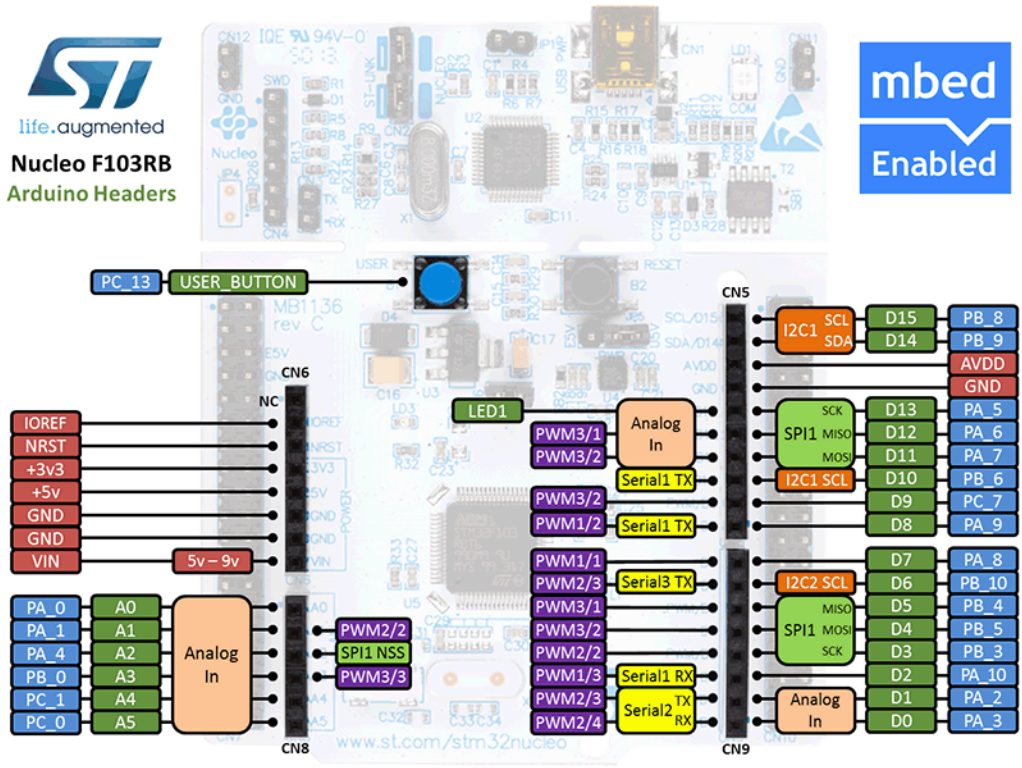


Figura 2. Nucleo F103RB Pinout

Implementarea jocurilor

Dorim să implementăm un joc de lumini cu patru tipare de aprindere. Selectarea tiparului de aprindere se va face de către utilizator prin intermediul unui meniu implementat peste comunicație serială (Figura 3). Jocul 1 va aprinde cele opt leduri în mod intermitent intercalat patru câte patru. Jocul 2 va aprinde ledurile pe rând și apoi le va stinge pe rând (ca un efect de umplere și golire). Jocul 3 va muta un led aprins de pe poziția 1 până pe poziția a 8-a iar jocul 4 va aprinde și va stinge ledurile în mod aleatoriu.

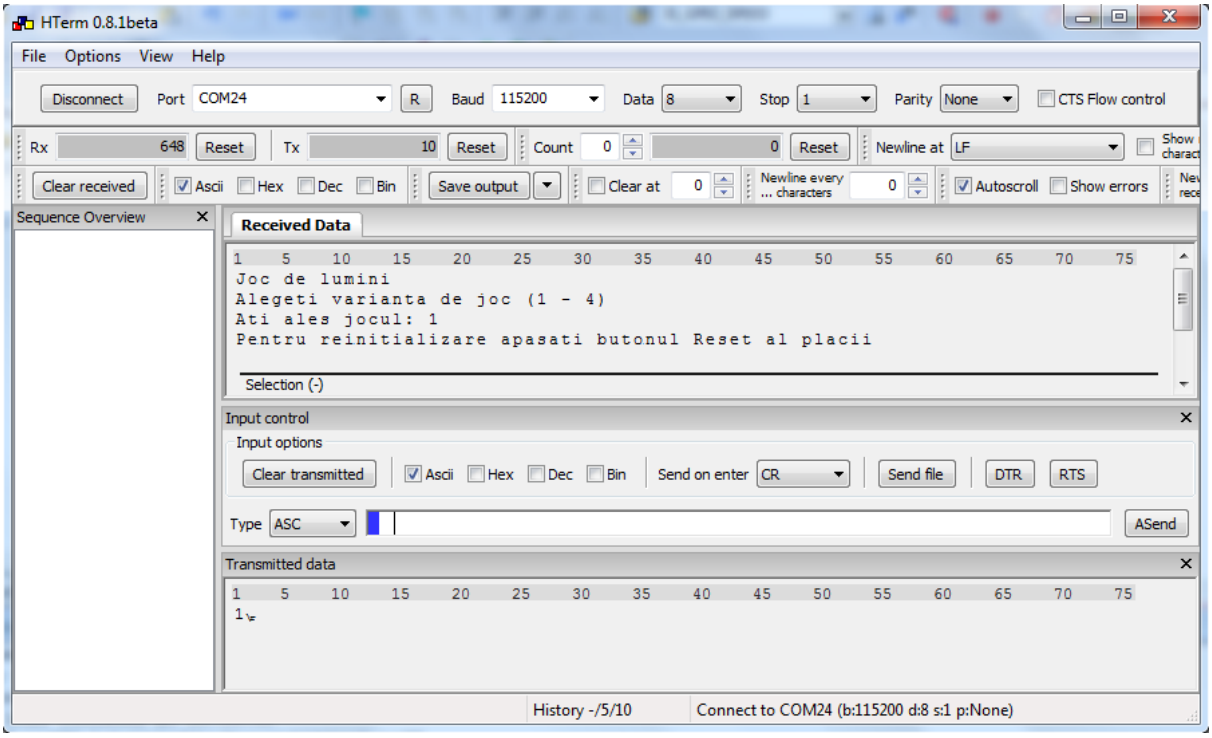


Figura 3. Meniu pentru alegere joc

Funcția principală main() va prelua prin comunicație serială selecția efectuată de utilizator (un întreg între 1 și 4) și va rula la infinit jocul de lumini ales prin apelarea funcției corespondente. Reluarea alegerii jocului de lumini se poate face doar prin resetare hardware a plăcii care va conduce la reinițializarea programului.

```
int main ()
{
	char alege = '0';
	SysTick_Config( SystemCoreClock / 1000 );
	initUsart();
	initGPIO_USART();
	initGPIO_LED();

	sendString("Joc de lumini \n");
	sendString("Alegeti varianta de joc (1 - 4) \n");

	while( ! (strchr("1234", alege)) )
	 {
		alege = readChar();
		 Delay(500);
	 }
	sendString("Ati ales jocul: ");
```

```
		sendChar(alege);
		sendString("\nPentru reinitializare apasati butonul Reset al placii ");

		switch (alege)
		{
			case '1':  joc1();
			case '2':  joc2();
			case '3':  joc3();
			case '4':  joc4();
		 }
}
```

Pentru a putea implementa cu uşurinţă jocurile de lumini (aprinderea sau stingerea LED-urilor într-o anumită ordine), s-au ales doi vectori care să aibă ca elemente porturile şi respectiv pinii la care sunt conectate LED-urile (figurile 1 şi 2).

```
uint16_t pin [] = {GPIO_Pin_10, GPIO_Pin_3, GPIO_Pin_5, GPIO_Pin_4, GPIO_Pin_10, GPIO_Pin_8, GPIO_Pin_9,
GPIO_Pin_7 } ;

GPIO_TypeDef * port[]={GPIOA, GPIOB, GPIOB, GPIOB, GPIOB, GPIOA, GPIOA, GPIOC};
```

Astfel pentru aprinderea led-ului ***i*** se va scrie o intrucţiune de tipul:

`port[i]->BSRR = pin[i];` iar pentru stingerea lui se va scrie o intructiune de tipul

`port[i]->BRR = pin[i];`

Functia porneste clock-ul pe magistralele la care sunt conectate LED-urile şi configurează pinii de ieşire:

```
void initGPIO_LED()
{
		// enable clock to GPIOA
		RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOA, ENABLE );
		// enable clock to GPIOB
		RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOB, ENABLE );
		RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOC, ENABLE );

		// enable AFIO clock
		RCC_APB2PeriphClockCmd( RCC_APB2Periph_AFIO, ENABLE );
		// disable JTAG and JTAG-DP
		GPIO_PinRemapConfig( GPIO_Remap_SWJ_NoJTRST, ENABLE );
		GPIO_PinRemapConfig( GPIO_Remap_SWJ_JTAGDisable, ENABLE );


		// save pin speed and pin mode
		GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10 | GPIO_Pin_8 | GPIO_Pin_9;
		GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
		GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
		GPIO_Init( GPIOA, &GPIO_InitStructure );
```

```
		// save pin speed and pin mode
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_10;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_Init( GPIOB, &GPIO_InitStructure );

		// save pin speed and pin mode
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_Init( GPIOC, &GPIO_InitStructure );
}
```

Jocul 1 de lumini va aprinde alternativ la un interval de 1 secundă ledurile de rang impar și ledurile de rang par (led 1, led 3, led 5, led 7 – led 2, led 4, led 6, led 8).

```
	void joc1()
	{
		uint8_t i;
		while (1)
		{
			for (i=0; i<8; i++)
			{
				if ((i%2)==0)
					port[i]->BSRR = pin[i];
				else
					port[i]->BRR = pin[i];
			}
			Delay(1000);

			for ( i=0; i<8; i++)
			{
				if ((i%2)==0)
					port[i]->BRR = pin[i];
				else
					port[i]->BSRR = pin[i];
			}
				Delay(1000);
		}
	}
```

Jocul 2 de lumini va aprinde unul câte unul (pornind de la led-ul 1 până la led-ul 8) toate led-urile și apoi le va stinge în mod similar (în ordine inversă). Operația de aprindere sau stingere a unui led se va efectua la un interval de 500 milisecunde.

```
void joc2()
{
		int i;
		while (1)
		 {
			for (i=0; i<8; i++)
			{
				port[i]->BSRR = pin[i];
				Delay(500);
			}
			for (i=7;i>=0;i--)
			 {
				port[i]->BRR = pin[i];
				Delay(500);
			}
		}
	}
```

Jocul 3 de lumini va aprinde câte un led pornind de la poziția 1 până la poziția 8. Mutarea led-ului aprins de pe o poziție pe alta se va face la un interval de 500 milisecunde.

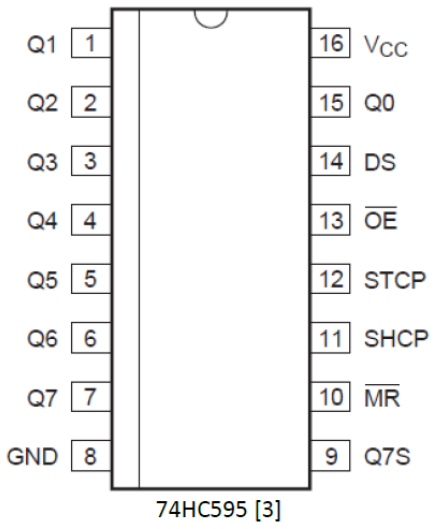
```
	void joc3()
	{
		uint8_t i=0;
		while (1)
		{
			port[i]->BSRR = pin[i];
			Delay(500);
			i++;
			port[i-1]->BRR = pin[i-1];
			Delay(500);
			if(i==8)
			    i=0;
		}
	}
```

Jocul 4 de lumini va aprinde sau va stinge total aleatoriu câte un led la un interval de 100 milisecunde.

```
void joc4()
 {
                    int pozitie;
                    int aprins;
                    while (1)
                     {
                              pozitie = rand()%8;
                              aprins = rand()%2;
                              if( aprins == 0)
                                 port[pozitie]->BRR = pin[pozitie];
                              else
                                port[pozitie]->BSRR = pin[pozitie];
                              Delay(100);
                     }
}
```

Proiect joc de lumini (conectare prin intermediul shift-register)

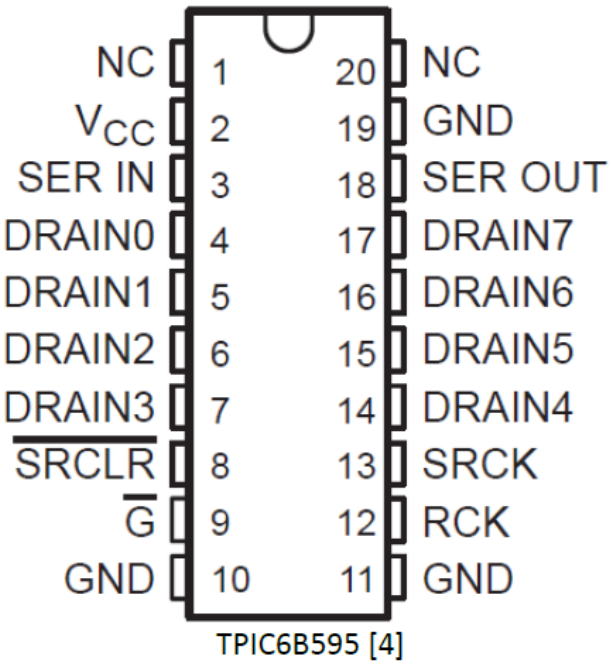
Cel de al doilea exemplu de interconectare utilizează un registru de shiftare pe 8 biți care va fi utilizat ca un registru cu încărcare serială. Astfel liniile de comandă a ansamblului de 8 leduri va fi redus de la 8 la 3. Comanda de aprindere a ledurilor nu va mai fi dată de pinii plăcii de dezvoltare ci de liniile de ieșire a registrului de shiftare.



74HC595 [3]

Ca registru de shiftare se poate utiliza un circuit 74HC595. Liniile de ieșire a circuitului care vor comanda cele 8 leduri sunt prezente pe pinii 15, 1, 2, 3, 4, 5, 6 și 7 ai circuitului (Q0-Q7). Pinii 8 și 16 ai circuitului vor trebui conectați la +5V şi la masă (Vcc și GND). Pinul 9 este semnalul de transport utilizat în cazul unei încascadări și nu va fi conectat în cazul montajului nostru. Pinul 14 (DS – serial data input) este linia de încărcare a registrului și în cazul nostru va fi conectat la pinul 2 al plăcii de dezvoltare (dataPin). Pinul 13 (/OE – Output Enable) va fi conetat la masă (activat întotdeauna). Pinii 11 și 12 (SHCP shift register clock input, STCP storage register clock input) vor fi conectați la pinii 3 și 4 ai plăcii Nucleo pentru generarea semnalului de ceas (clockPin) și a comenzii de încărcare (latchPin). Pinul 10 (/MR – Master Reset) va fi conectat la Vcc (inactiv).

Un circuit echivalent este circuitul TPIC6B595. Echivalențele de pini se observă imediat: DRAINx – Qx (linii de comandă pentru leduri), SER IN – DS (dataPin), SER OUT – Q7S (neconectat), , SRCK – SHCP (clockPin), RCK – STCP (latchPin) /SRCLR - /MR (inactiv – conectat la Vcc), /G - /OE (activat întotdeana – la masă). Pinii NC ai acestui circuit nu sunt conectați intern.



TPIC6B595 [4]

Există două diferențe majore între 74HC595 și TPIC6B595: comanda de aprindere la 74HC595 se dă pe ”1” logic iar la TPIC6B595 pe ”0” logic (**în schema cu TPIC6B595 se va conecta catodul ledului la pinul de comandă**);TPIC6B595 are o putere mai mare de comandă (150mA/pin) față de 74HC595 (35mA/pin).

Schemă de interconectare bazată pe TPIC6B595

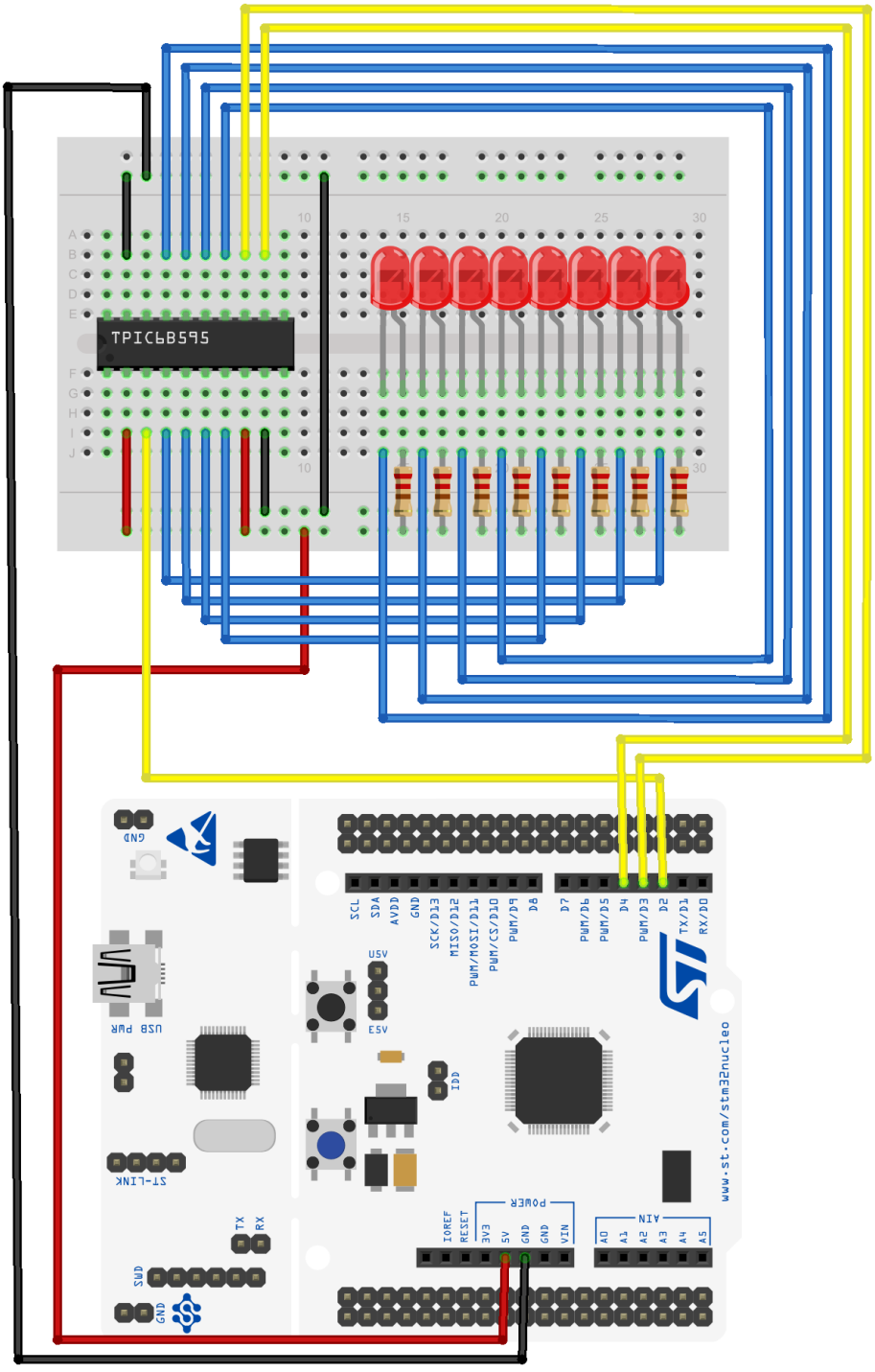


Figura 4. **Schemă de interconectare bazată pe TPIC6B595**

Implementarea jocurilor de led-uri folosind shift-register

Pentru configurarea pinilor de comanadă şi comunicaţie pentru registrul de shiftare (**pinul de clock pe PB_3, pinul latch pe PB_5 şi pinul de date pe PA_10**) se foloseşte funcţia *initGPIO_LED()*.

```
void initGPIO_LED()
{	// enable clock to GPIOA
	RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOA, ENABLE );
	// enable clock to GPIOB
	RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOB, ENABLE );

	// enable AFIO clock
	RCC_APB2PeriphClockCmd( RCC_APB2Periph_AFIO, ENABLE );
	// disable JTAG and JTAG-DP
	GPIO_PinRemapConfig( GPIO_Remap_SWJ_NoJTRST, ENABLE );
	GPIO_PinRemapConfig( GPIO_Remap_SWJ_JTAGDisable, ENABLE );

	// save pin speed and pin mode
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10 ;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_Init( GPIOA, &GPIO_InitStructure );

		// save pin speed and pin mode
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3 | GPIO_Pin_5;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_Init( GPIOB, &GPIO_InitStructure );
}
```

Funcția principală *main()* nu diferă față de exemplul precedent (conectare individuală) dar partea de comandă efectivă a ledurilor trebuie adaptată transmiterii prin registrul de shiftare.

Funcția pentru joc 1 de lumini va arăta în felul următor:

```
void joc1() {
	while(1) {
		GPIOB->BRR = GPIO_Pin_5; //Latch pin low
		shiftOut(MSB, 85);
		GPIOB->BSRR = GPIO_Pin_5; //latch pin high
		Delay(1000);
		GPIOB->BRR = GPIO_Pin_5; //Latch pin low
		shiftOut(MSB, 170);
		GPIOB->BSRR = GPIO_Pin_5; //latch pin high
		Delay(1000);
		}
```

```
}
```

Valorile transmise către registru sunt echivalente cu starea dorită a celor 8 leduri (170 în binar 10101010 – aprins/stins/aprins/stins/aprins/stins/aprins/stins iar 85 în binar 01010101). Pentru încărcarea în registru a valorilor se dezactivează semnalul de latchPin după care se utilizează funcția `shiftOut` pentru a transmite valoarea dorită. Activarea semnalului de latchPin conduce la memorarea și transmiterea la ieșiri a valorii transmise serial. În mod similar se pot rescrie și celelalte trei funcții (joc2, joc3, joc4):

```
void joc2() {
        int i;
        int afisare =0;
        while (1) {
                for (i=0; i<9; i++) {
                        afisare = afisare + (1<<i);
                        GPIOB->BRR = GPIO_Pin_5; //Latch pin low
                        shiftOut(MSB, afisare);
                        GPIOB->BSRR = GPIO_Pin_5; //latch pin high
                        Delay(500);
                }
                for(i=8; i>=0; i--) {
                        afisare = afisare - (1<<i);
                        GPIOB->BRR = GPIO_Pin_5; //Latch pin low
                        shiftOut(MSB, afisare);
                        GPIOB->BSRR = GPIO_Pin_5; //latch pin high
                        Delay(500);
                }
        }
}


void joc3() {
        int i;
        int afisare = 1;
        while (1) {
                for (i=0;i<8;i++) {
                        GPIOB->BRR = GPIO_Pin_5; //Latch pin low
                        shiftOut(MSB, afisare<<i);
                        GPIOB->BSRR = GPIO_Pin_5; //latch pin high
                        Delay(500);
                }
        }
}


void joc4() {
        int configuratie;
        while (1)
```

```
{
        configuratie = rand()%256;
        GPIOB->BRR = GPIO_Pin_5; //Latch pin low
        shiftOut(MSB, configuratie);
        GPIOB->BSRR = GPIO_Pin_5; //latch pin high
        Delay(100);
        }
}
```