Sisteme Incorporate

Sistem de autentificare cu parolă pe placa Nucleo-64

Documentație

Hațegan Simina – Elena

Ilia Delia – Alexandra

Cuprins

1. Introducerea

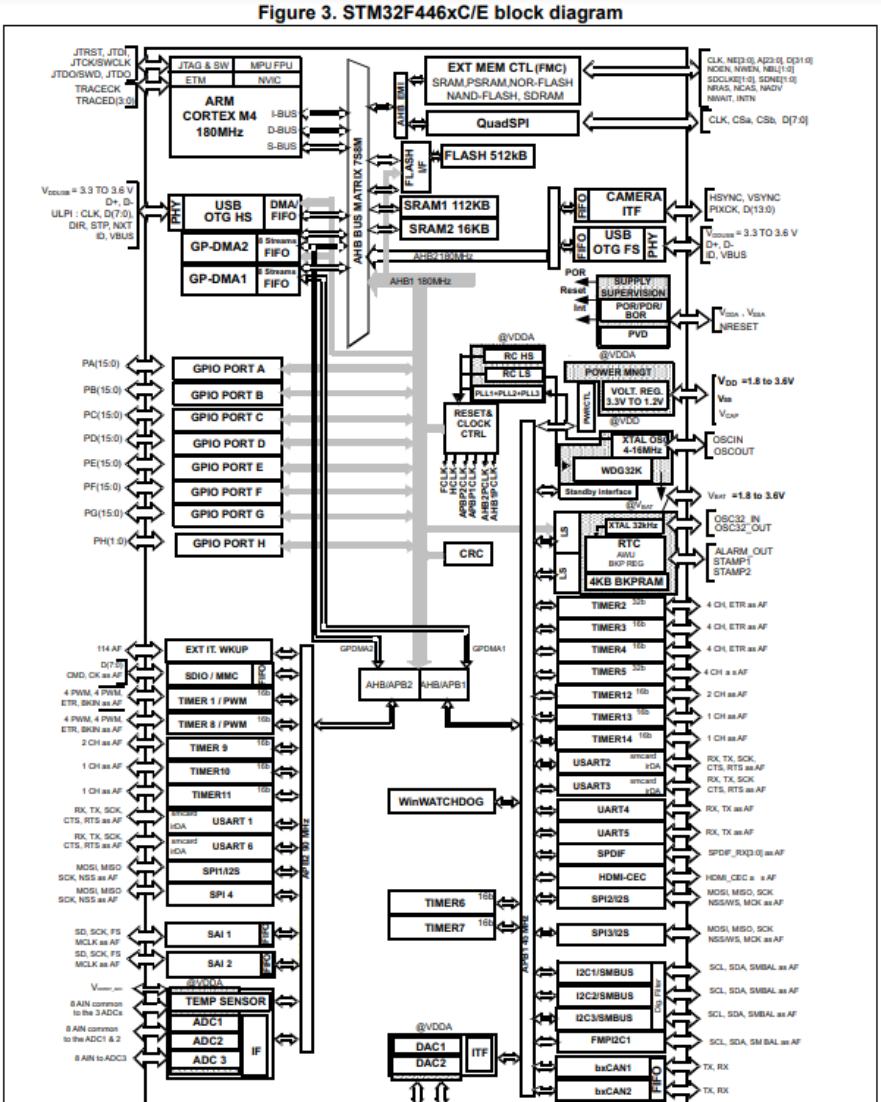
Tema lucrării:
Implementarea unui sistem de autentificare cu parolă pe placa Nucleo-64.
Cerințe:

- La inițializarea sistemului, acesta se află în starea „locked” (nu permite comanda actuatorilor de pe placă sau citirea senzorilor); utilizatorului îi este cerută introducerea parolei prin afișarea unui mesaj pe LCD;
- Utilizatorul poate introduce parola folosind tastatura plăcii Nucleo și are la dispoziție 3-5 încercări;
- La o introducere greșită a parolei, toate LED-urile LED0 – LED3 se aprind în culoarea roșie și numărul de încercări rămase este decrementat;
- La o introducere corectă a parolei, toate cele 4 LED-uri mentionate anterior se sting și se afișează pe LCD un mesaj corespunzător, sistemul schimbându-și starea în „unlocked”;

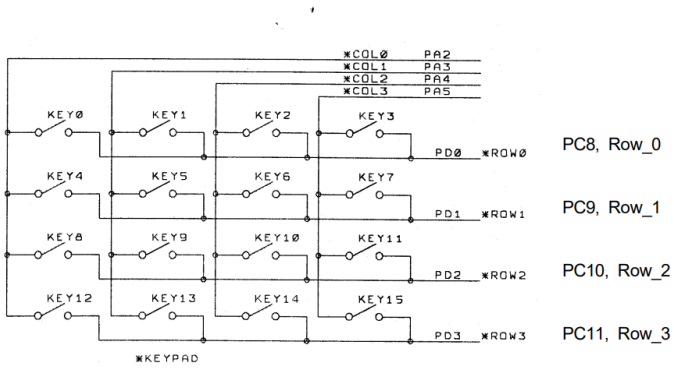
2. Conexiuni Hardware

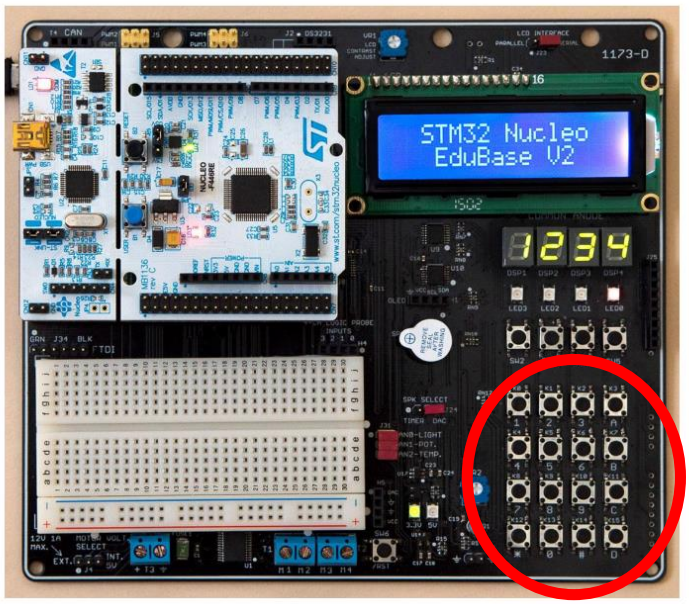
Pentru a implementa acest sistem de blocare protejat prin parola, s-a utilizat microcontroler-ul STM32F446RE.

Figure 3. STM32F446xC/E block diagram

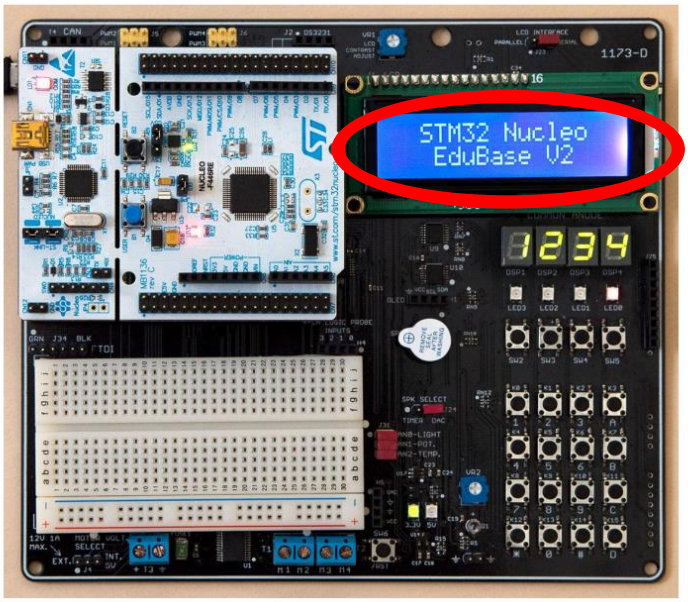


Sistemul folosește un keypad 4x4 pentru introducerea parolei. Liniile sunt conectate la pinii PC8, PC9, PC10 și PC11, iar coloanele sunt conectate la pinii PB12, PB13, PB14, PB15.

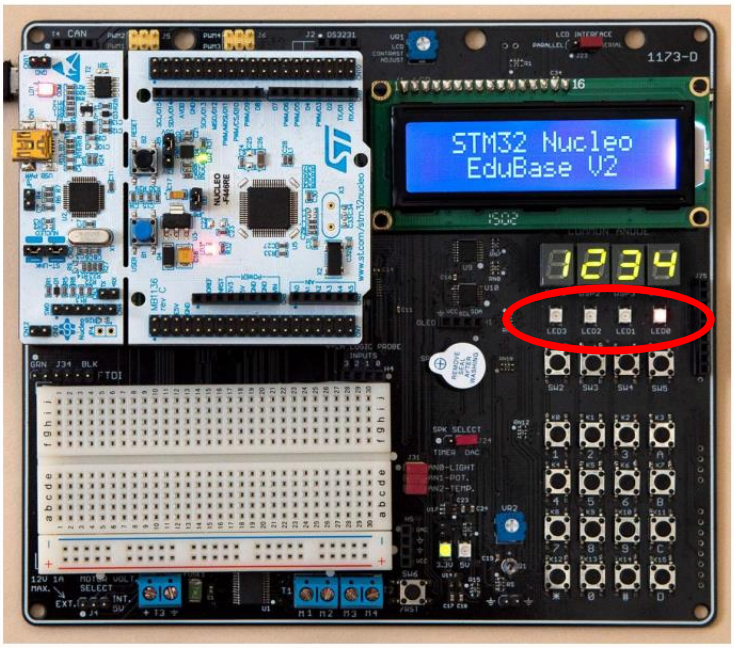




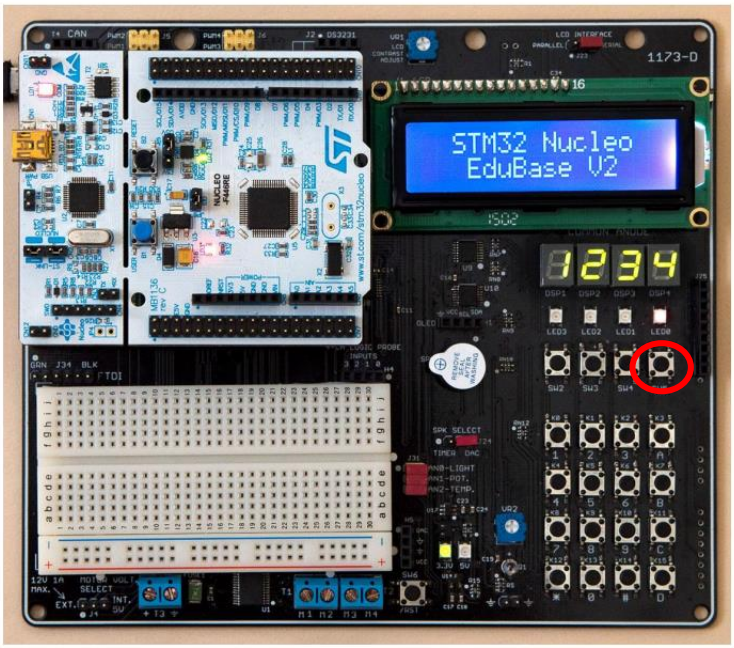
LCD pentru afișarea mesajelor. Conectarea se face prin interfața SPI1 (Serial Peripheral Interface).



Led-urile sunt folosite pentru a indica starea sistemului. Ele sunt conectate la pinii PB4, PB5, PB6, PB7. Pentru a aprinde un LED, trebuie ca pinul corespunzător al portului B să fie programat ca ieșire și setat pe HIGH.



Butonul SW5 este folosit pentru a trimite parola sistemului pentru a o verifica și este conectat la pinul PC8. Când o tastă este apăsată, intrarea portului C devine HIGH.



3. Componente Software

3.1 Definiții și variabile globale

```
#define RS 1
#define EN 2

#define NUMAR_MAXIM_CARACTERE_PE_RAND 16
#define MAX_PASSWORD_LENGTH 20
#define PASSWORD "223356"

char user_password[MAX_PASSWORD_LENGTH + 1];

int pozitie_cursor = 0;
int caractere_scrise = 0;
int numar_incercari = 3;
int isLocked=0;

const char keypad_values[4][4] = {
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'*', '0', '#', 'D'}
};

typedef enum {
    LOCKED,
    UNLOCKED
} SystemState;

SystemState system_state = LOCKED;
```

3.2 Funcții de inițializare butoane

3.2.1 Configurarea butoanelor

```
void configure_buttons(void) {

    GPIOB->MODER &= ~(0xFF000000);
    GPIOB->MODER |= 0x55000000;

    GPIOC->MODER &= ~(0xFF00);
    GPIOC->PUPDR &= ~(0xFF00);
    GPIOC->PUPDR |= 0x5500;

}
```

3.2.2 Verificarea Apăsării Butonului

```
int is_SW5_pressed(void) {
return (GPIOC->IDR & (1 << 8));
}

void wait_for_SW5_press(void) {
while (is_SW5_pressed()) {}
while (!is_SW5_pressed()) {}
delayMs(10);
}
```

3.2.3 Configurarea Pinului PC8

```
void configure_PC8(void){

RCC->AHB1ENR |=  2;
RCC->AHB1ENR |=  4;
GPIOB->MODER &= ~0x0000ff00;
GPIOB->MODER |=  0x00005500;
GPIOC->MODER &= ~0x00FF0000;

}
```

3.3 Funcții pentru keypad

3.3.1 Inițializarea keypad-ului

```
void keypad_init(void) {
RCC->AHB1ENR |= 4;
GPIOC->MODER &= ~0x00FF0000;
RCC->AHB1ENR |= 2;
GPIOB->MODER &= ~0xFF000000;
}
```

3.3.2 Obținerea valorii de la keypad

```
char get_keypad_value(int row, int col) {
return keypad_values[row][col];
}

char keypad_getkey(void) {
int row, col;
outputEnableCols(0xF);
writeCols(0xF);
delay();
row = readRows();
writeCols(0x0);
outputEnableCols(0x0);
```

```
if (row == 0) return 0;
for (col = 0; col < 4; col++) {
outputEnableCols(1 << col);
writeCols(1 << col);
delay();
row = readRows();
writeCols(0x0);
if (row != 0) break;
}


outputEnableCols(0x0);
if (col == 4)
return 0;
if (row == 0x01) return 0 + col;
if (row == 0x02) return 4 + col;
if (row == 0x04) return 8 + col;
if (row == 0x08) return 12 + col;
return 0;
}
```

3.3.3 Activarea coloanelor

```
void outputEnableCols(char n) {
   GPIOB->MODER &= ~0xFF000000;
      if (n & 1)
       GPIOB->MODER |=  0x01000000;
   if (n & 2)
       GPIOB->MODER |=  0x04000000;
   if (n & 4)
       GPIOB->MODER |=  0x10000000;
   if (n & 8)
       GPIOB->MODER |=  0x40000000;
}
```

3.3.4 Setarea valorii pe coloane

```
void writeCols(char n) {

    GPIOB->BSRR = 0xF0000000;

    GPIOB->BSRR = ((uint32_t)n << 12) | (n & 1);

}
```

3.3.5 Citirea valorilor de pe rânduri

```
int readRows(void) {
        return (GPIOC->IDR & 0x0F00) >> 8;
}
```

3.4 Funcții pentru LED-uri

```
void writeLEDs(char n) {
        GPIOB->BSRR = 0x00F00000;
        GPIOB->BSRR = n << 4;
}

void update_LEDs(void) {
   if (system_state == LOCKED) {
      GPIOB->BSRR = 0x000000FF;   // Turn on all LEDs in red
   } else {
      GPIOB->BSRR = 0x00F00000;   // Turn off all LEDs
   }
}
```

3.5 Funcții pentru LCD

3.5.1 Scrierea unui nibble.

```
void LCD_nibble_write(char data, unsigned char control) {
   data &= 0xF0;
   control &= 0x0F;
   SPI1_write(data | control);
   SPI1_write(data | control | EN);
   delayMs(10);
   if (pozitie_cursor < NUMAR_MAXIM_CARACTERE_PE_RAND) {
      SPI1_write(data);
      pozitie_cursor++;
   } else if (pozitie_cursor < (NUMAR_MAXIM_CARACTERE_PE_RAND * 2)) {
      SPI1_write(data | RS | 0x40);
      pozitie_cursor++;
   }
   caractere_scrise++;}
```

3.5.2 Comanda pentru LCD

```
void LCD_command(unsigned char command) {
   LCD_nibble_write(command & 0xF0, 0);
   LCD_nibble_write(command << 4, 0);
   if (command < 4)
      delayMs(2);
   else
      delayMs(1);
}
```

3.5.3 Date pentru LCD

```
void LCD_data(char data) {
   GPIOA->BSRR = RS;
   LCD_nibble_write(data & 0xF0, RS);
   LCD_nibble_write(data << 4, RS);
   delayMs(10);
}
```

3.5.4 Inițializarea LCD-ului

```
void LCD_init(void) {
   RCC->AHB1ENR |= 1;
   RCC->AHB1ENR |= 4;
   RCC->APB2ENR |= 0x1000;
   GPIOA->MODER &= ~0x0000CC00;
   GPIOA->MODER |= 0x00008800;
   GPIOA->AFR[0] &= ~0xF0F00000;
   GPIOA->AFR[0] |= 0x50500000;
   GPIOA->MODER &= ~0x03000000;
   GPIOA->MODER |= 0x01000000;
   SPI1->CR1 = 0x31F;
   SPI1->CR2 = 0;
```

```
    SPI1->CR1 |= 0x40;
    delayMs(20);
    LCD_nibble_write(0x30, 0);
    delayMs(5);
    LCD_nibble_write(0x30, 0);
    delayMs(1);
    LCD_nibble_write(0x30, 0);
    delayMs(1);
    LCD_nibble_write(0x20, 0);
    delayMs(1);
    LCD_command(0x28);
    LCD_command(0x06);
    LCD_command(0x01);
    LCD_command(0x0F);
    pozitie_cursor = 0;
    caractere_scrise = 0;
}
```

3.6 Funcții pentru Sistem

3.6.1 Blocarea și Deblocarea Sistemului

```
void lock_system(void) {
    LCD_command(0x01);
    LCD_command(0x02);
    LCD_data('S');
    LCD_data('y');
    LCD_data('s');
    LCD_data('t');
    LCD_data('e');
    LCD_data('m');
    LCD_data(' ');
    LCD_data('b');
    LCD_data('l');
    LCD_data('o');
    LCD_data('c');
    LCD_data('k');
    LCD_data('e');
    LCD_data('d');
    LCD_data('!');
    GPIOB->BSRR = 0x000000FF;
```

```
    LCD_command(0x01);
    delayMs(200);
}

void unlock_system(void) {
    LCD_command(0x01);
    LCD_command(0x02);
    LCD_data('S');
    LCD_data('y');
    LCD_data('s');
    LCD_data('t');
    LCD_data('e');
    LCD_data('m');
    LCD_data(' ');
    LCD_data('u');
    LCD_data('n');
    LCD_data('l');
    LCD_data('o');
    LCD_data('c');
    LCD_data('k');
    LCD_data('e');
    LCD_data('d');
    LCD_data('!');
    GPIOB->BSRR = 0x00F00000;
}
```

3.6.2 Verificarea Parolei

```
void check_password(char *input) {
    LCD_command(0x01);
    if (strcmp(input, PASSWORD) == 0) {
        unlock_system();
        numar_incercari = 3;
        LCD_data('c');
        LCD_data('o');
        LCD_data('r');
        LCD_data('r');
        LCD_data('e');
        LCD_data('c');
        LCD_data('t');
    } else {
        LCD_data('w');
        LCD_data('r');
        LCD_data('o');
        LCD_data('n');
        LCD_data('g');
        numar_incercari--;
        GPIOB->BSRR = 0x000000FF;
        LCD_command(0x01);
        delayMs(200);
```

```
            if (numar_incercari == 0) {
               lock_system();
               isLocked = 1;
            } else {
               LCD_data('t');
               LCD_data('r');
               LCD_data('y');
               LCD_data(' ');
               LCD_data('a');
               LCD_data('g');
               LCD_data('a');
               LCD_data('i');
               LCD_data('n');
            }
         }
      }
```

3.6.3 Funcții de Întârziere

```
      void delayMs(int n) {
         int i;
         for (; n > 0; n--)
            for (i = 0; i < 3195; i++) ;
      }

      void delay(void) {
          int j;

          for (j = 0; j < 300; j++);

      }
```

3.6.4 Scriere prin SPI

```
   void SPI1_write(unsigned char data) {
      while (!(SPI1->SR & 2)) {}
      GPIOA->BSRR = 0x10000000;
      SPI1->DR = data;
      while (SPI1->SR & 0x80) {}
      GPIOA->BSRR = 0x00001000;
   }
```

4. Program principal

```
      int main(void) {

         LCD_init();
         configure_buttons();
               configure_PC8();
```

```
lock_system();
delayMs(100);

LCD_command(0x01);

delayMs(100);


// Afisare mesaj de introducere a parolei
LCD_data('E');
LCD_data('n');
LCD_data('t');
LCD_data('e');
LCD_data('r');
LCD_data(' ');
LCD_data('p');
LCD_data('a');
LCD_data('s');
LCD_data('s');
LCD_data('w');
LCD_data('o');
LCD_data('r');
LCD_data('d');
LCD_data(':');


delayMs(100);

LCD_command(0x01);

delayMs(200);

char input_password[MAX_PASSWORD_LENGTH + 1];

int numar_caractere_introduse = 0;

//initializare led-uri pentru display

RCC->AHB1ENR |=  2;

GPIOB->MODER &= ~0x0000ff00;

GPIOB->MODER |=  0x00005500;

char a[20];

while(1) {

    if (system_state == LOCKED && isLocked==0) {

            char key = keypad_getkey();

            delayMs(10);

            if (key >= 1 && key <= 15) {


           switch (key) {
              case 1:
```

```
            user_password[numar_caractere_introduse] = '2';
            numar_caractere_introduse++;
            LCD_data('2');
 break;
 case 2:
           user_password[numar_caractere_introduse] = '3';
           numar_caractere_introduse++;
           LCD_data('3');
 break;
 case 3:
           user_password[numar_caractere_introduse] = 'A';
           numar_caractere_introduse++;
           LCD_data('A');
 break;
 case 5:
           user_password[numar_caractere_introduse] = '5';
           numar_caractere_introduse++;
           LCD_data('5');
 break;
 case 6:
           user_password[numar_caractere_introduse] = '6';
           numar_caractere_introduse++;
           LCD_data('6');
 break;
 case 7:
           user_password[numar_caractere_introduse] = 'B';
           numar_caractere_introduse++;
           LCD_data('B');

 break;

 case 9:

           user_password[numar_caractere_introduse] = '8';
           numar_caractere_introduse++;
           LCD_data('8');
 break;

 case 10:

           user_password[numar_caractere_introduse] = '9';
           numar_caractere_introduse++;
           LCD_data('9');
 break;
 case 11:
           user_password[numar_caractere_introduse] = 'C';
           numar_caractere_introduse++;
           LCD_data('C');
 break;
 case 13:
           user_password[numar_caractere_introduse] = '0';
           numar_caractere_introduse++;
           LCD_data('0');
```

```
                break;
                case 14:
                        user_password[numar_caractere_introduse] = '#';
                        numar_caractere_introduse++;
                        LCD_data('#');
                break;
                case 15:
                        user_password[numar_caractere_introduse] = 'D';
                        numar_caractere_introduse++;
                        LCD_data('D');
                  break;
                  default:
                  break;
                }
        }
        if (is_SW5_pressed()) {
                  check_password(user_password);
                  numar_caractere_introduse = 0;
                  writeLEDs(key);
                }
        }

}
```

5. Concluzie

În main, se inițializează toate componentele hardware de care este nevoie în program (led-uri, tastatura, butoane, lcd). Se afișează un mesaj de început, acela de a introduce parola (“Enter the password”).

În bucla while(1), se verifica starea în care se afla sistemul, care initial este blocat. Se introduce o parolă, caracter cu caracter, butoanele apăsate fiind detectate de funcţia keypad_getkey(). Pe ecran se afișează caracterele introduce, sub formă de “*”. Apoi, se verifică dacă butonul de trimitere a parolei (SW5) este apăsat. Urmează verificarea parolei, comparand-o pe cea salvată în sistem cu cea introdusă de utilizator. În cazul în care parola este greșită, se vor aprinde toate ledurile şi va apărea mesajul “Wrong”, “Try again”, iar daca numărul de încercări depășește numărul 3, atunci sistemul va fi blocat, “System blocked” . Dacă se introduce parola corectă sistemul va fi deblocat și va apărea pe ecran mesajul “System unlocked”.

6. Bibliografie

EduBaseV2_NucleoF446_user_guide_ver1.16-2.pdf
https://web.archive.org/web/20210512184345/http://www.microdigitaled.com/ARM/Edubase/06_stm32_keypad_LED.txt
https://web.archive.org/web/20210512174047/http://www.microdigitaled.com/ARM/Edubase/05_stm32_LCD.txt
STM32F446RE_Datasheet.pdf