Sprawozdanie z projektu

Systemy Wbudowane 2020/21

Konrad Walas, Marcin Filipek

Wprowadzenie - Temat projektu

Temat - Zamek szyfrowy

Zadanie polegało na zbudowaniu urządzenia, które udostępnia interfejs pozwalający na udzielaniu dostępu do zamka po wpisaniu poprawnego kodu.

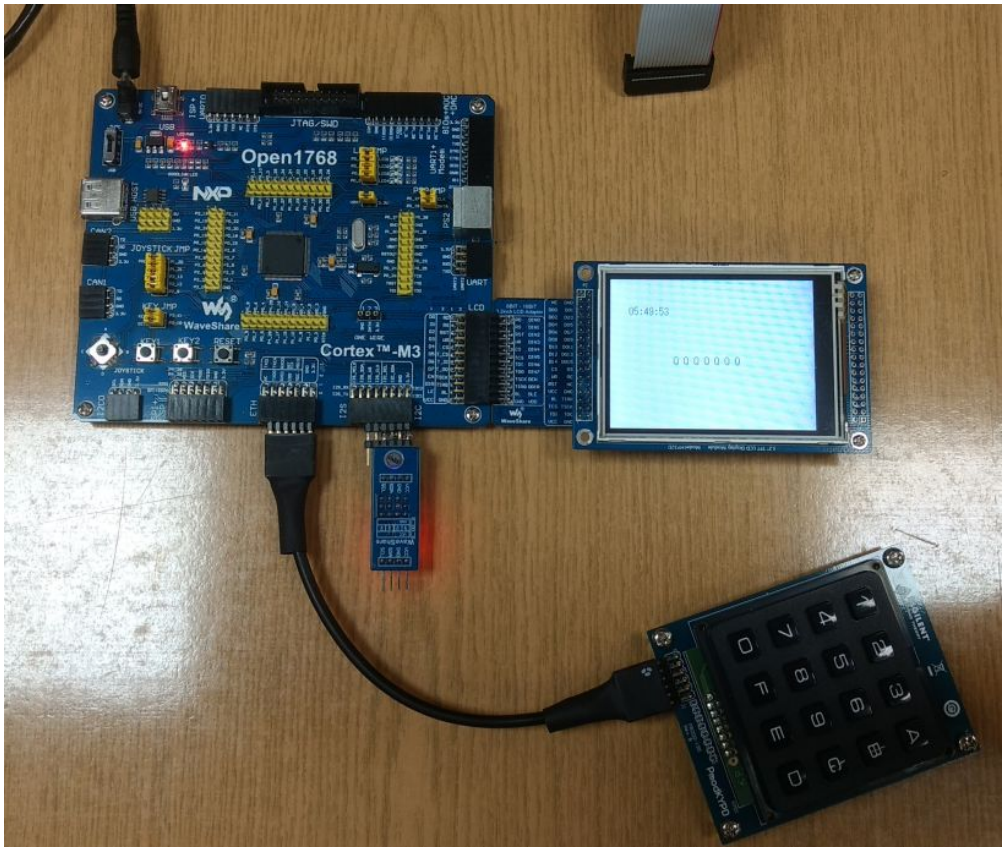
Użytkownik ma do dyspozycji 16-przyciskową klawiaturę dzięki której komunikuje się z urządzeniem. Na wyświetlaczu pokazana jest aktualna data oraz ukazane jest wpisywany kod. Użytkownik jest w stanie zaprogramować kod dostępu za pomocą przycisku serwisowego. Po wpisaniu poprawnego kodu zamek zostaje otwarty. Po wpisaniu kodu w systemie zostaje odnotowana godzina wejścia, wraz z informacją, czy próba była udana. Po pewnym czasie zamek blokuje się samoczynnie. Ustawiony kod jest przechowywany w zewnętrznej pamięci nieulotnej, dzięki czemu obowiązuje po ponownym włączeniu zasilania. Kod, który nie został do końca wpisany po określonym czasie zostaje automatycznie wyczyszczony.

Opis techniczny mikrokontrolera wraz z użytymi komponentami

Użyte części

1. Zestaw Waveshare Open1768 z mikrokontrolerem LPC 1768
2. Klawiatura 16-przyciskowa Pmod KYPD
3. Kabel do wygodnego podłączenia klawiatury do mikrokontrolera
4. Wyświetlacz LCD z zestawu Waveshare Open1768
5. Pamięć FRAM FM24CLXX wraz z rezystorami podciągającymi

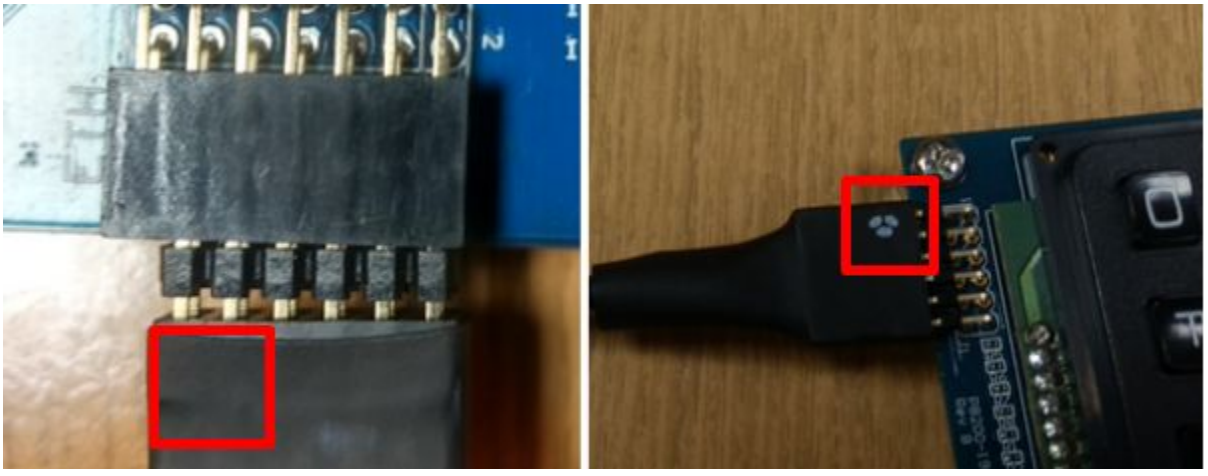
Sposób podłączenia urządzenia



Rys. 1: Ogólny schemat podłączenia wszystkich podzespołów.

Klawiatura

Klawiatura podłączona jest kablem do gniazda ETH (port 1, piny 4,9,15,17,0,10,8,16) na mikrokontrolerze. Należy pamiętać o odpowiednim dopasowaniu kabla, tak aby odpowiednie piny się zgadzały. W tym celu kabel powinien być wpięty po prawej stronie gniazda i powinien być obrócony o 180° względem panelu klawiatury.



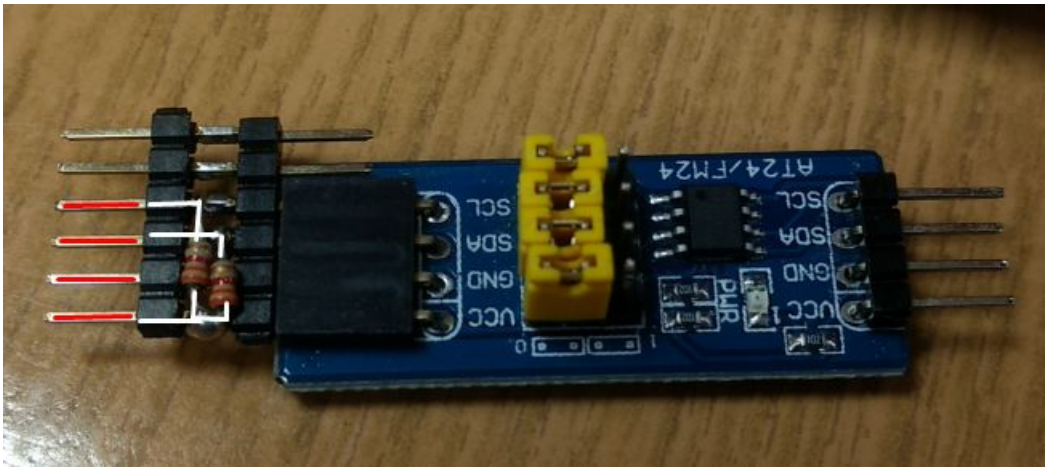
Rys. 2: Schemat podłączenia klawiatury. Należy zwrócić uwagę na obrócenie go (znaczek na kablu) tak, aby pin zasilania klawiatury podłączony był do zasilania z płytki.

Ekran LCD

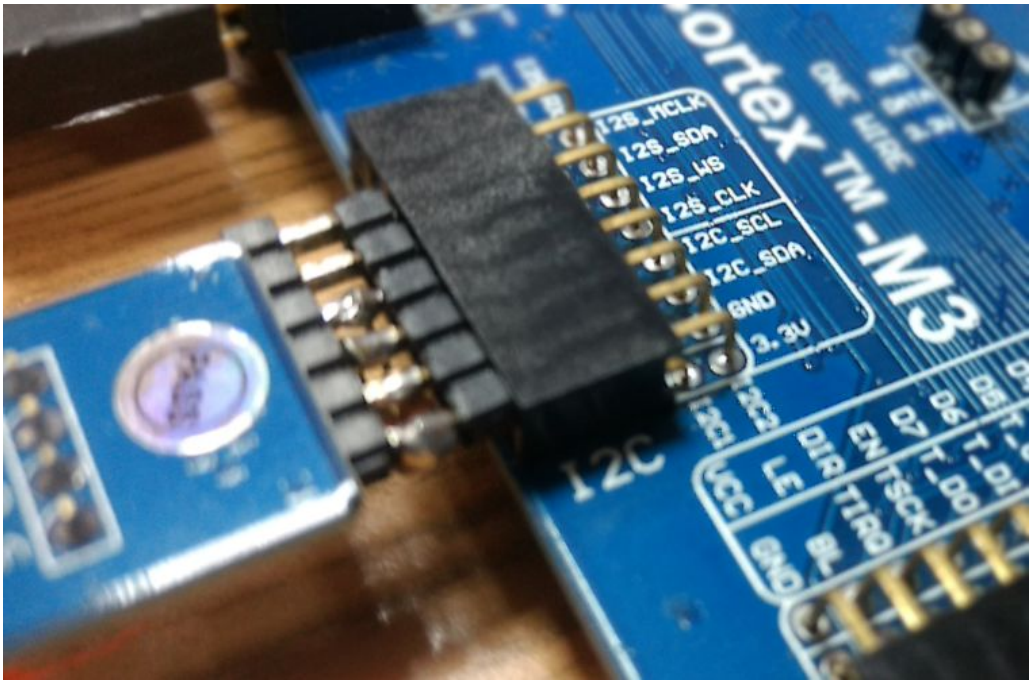
Ekran LCD powinien być wpięty do gniazda LCD. Nie ma tutaj konieczności stosowania dodatkowych kabli.

Pamięć FRAM

Pamięć skonfigurowana jest pod I2C2 (port 0 piny 10 i 11) . Należy pamiętać o rezystorach podciągających i odpowiednim ułożeniu pinów. W tym przypadku będzie ona “obrócona do góry dnem” i wpięta po prawej stronie w górnym rzędzie.



Rys. 3: Schemat podłączenia rezystorów.



Rys. 4: Podłączenie układu do płytki Open1768

Opis interfejsu

Opis interakcji, które może podjąć użytkownik oraz odpowiadającego zachowania programu

Wpisanie kodu

- Podczas działania urządzenia użytkownik może w każdym momencie wpisać kod
- Kod składa się z cyfr 0-9 oraz liter E oraz F
- Jeżeli wpisany kod jest poprawny, to zamek zostanie otworzony. Symbolizuje to zapalenie się diody wbudowanej w płytkę mikrokontrolera. Po ustalonym czasie dioda zgaśnie - symbolizuje to zamknięcie się zamka.
- Po czasie 5 sekund kod, który nie został wpisany do końca zostaje wyczyszczony
- Wpisywany kod nie jest widoczny, zamiast tego wyświetlane są symbole gwiazdki
- Po 5 sekundach odblokowany zamek zostaje automatycznie zablokowany

Wciśnięcie przycisku serwisowego

- Użytkownik może wcisnąć przycisk serwisowy
- Rolę tę pełni wbudowany w mikrokontroler przycisk “key2”
- Wciśnięcie przycisku pozwala na ustalenie nowego kodu zamka. Wpisywane wtedy symbole pokazują się na ekranie
- Należy wpisać wszystkie 9 symboli, wówczas kod zostanie zapisany
- Kod zostaje zapisany do pamięci nieulotnej, i obowiązuje po zresetowaniu urządzenia
- Podczas użycia przycisku serwisowego wszystkie pozostałe funkcje programu zostają zatrzymane, należy więc dokończyć jego wpisywanie, żeby program działał poprawnie

Rys. 5: Przycisk serwisowy “key2” na płytce.

Aktywacja/deaktywacja logów

- Po wciśnięciu przycisku A na klawiaturze z prawej strony ekranu wyświetlone zostaną logi
- Po wciśnięciu przycisku B logi zostaną ukryte

- Każdy wpis zawiera godzinę wpisania kodu, oraz informację czy wpisany kod był poprawny
    - P - wpisany został poprawny kod
    - N - wpisany został niepoprawny kod
- Jeżeli kod zostanie wpisany przy włączonych logach, trzeba je ręcznie odświeżyć wciskając przycisk A

Wygląd wyświetlacza

Na wyświetlaczu znajdują się następujące elementy:

- Aktualna godzina - lewy górny róg
- Pola na wpisany kod - środek ekranu na dole
- Miejsce na logi wpisywania kodu - prawa strona ekranu

Opis kodu

Pseudokod

```
def TIMER0_IRQHandler():
  'zgaś diodę LED sygnalizującą otwarty zamek'
  'usuń sygnał przerwania'

def TIMER1_IRQHandler():
  'wyczyść wpisany kod'
  'usuń sygnał przerwania'

def RTC_IRQHandler(){
  'pobierz aktualną godzinę i wypisz ją na ekran'
  'usuń sygnał przerwania'
}

def EINT0_IRQHandler()
{
  'pobierz od użytkownika nowy kod'
  'zapisz kod do pamięci FRAM'
  'usuń sygnał przerwania'
}

def SysTick_Handler(){
  'jeżeli jest dodatni, obniż wartość licznika I2C'
  'usuń sygnał przerwania'
}
```

```
def main():
  'inicjalizacja'
  'wczytaj dane z pamięci FRAM'
  'przygotuj ekran LCD'

  while True:
    managePressedKey()

def managePressedKey():
  pressedKey = scanKeypad()

  'zadbaj o włączenie timera czyszczącego kod'
  'poczekaj określony czas, żeby pozbyć się zakłóceń'
  'zadbaj, aby wciśnięty przycisk zarejestrowano tylko raz'
  'obsłuż specjalne przyciski, w tym pokazywanie i chowanie logów'

  'wypełnij kolejny symbol w tablicy wpisywanego kodu'

  if 'wpisano ostatni symbol'
    if 'hasło jest poprawne':
      'zapal diodę LED sygnalizującą otwarty zamek i odnotuj poprawnie
       wpisane hasło w logach'
    else:
      'odnotuj źle wpisane hasło w logach'
    'wyczyść wpisywane hasło'

def scanKeypad():
  'podaj stan wysoki na piny z kolumnami'

  for i in kolumny:
    'podaj stan niski na pin z i-tą kolumną'

    for j in rzędy:
      if 'pin w j-tej kolumnie ma wartość niską':
        'zwróć klawisz znajdujący się w i-tej kolumnie i j-tym wierszu'

    'podaj stan wysoki na pin z i-tą kolumną'
```

Wykorzystane elementy mikrokontrolera

SysTick

SysTick używany jest przez funkcje I2C do oczekiwania na status po wykonaniu operacji. Jeżeli oczekiwany status nie pojawi się w rejestrze na czas, funkcja zwraca błąd.

Inicjalizacja

```
SysTick_Config(SystemCoreClock / 1000);
```

RTC

Wykorzystujemy RTC do pobierania aktualnej godziny, w celu jej wyświetlenia na ekranie, oraz do zarejestrowania w logach.

Inicjalizacja

```
LPC_RTC->CCR = 1; // włacz RTC
LPC_RTC->ILR = 1; // włączenie generowania przerwań
LPC_RTC->CIIR = 1; // przerwanie co sekundę
NVIC_EnableIRQ(RTC_IRQn);
```

Przerwanie

Przerwanie wywoływane jest co sekundę, aby wypisać aktualną datę na ekran.

```
void RTC_IRQHandler(void){
  'pobierz aktualną godzinę i wypisz ją na ekran'
  LPC_RTC->ILR = 1; // czyszczenie informacji o przerwaniu
}
```

LCD

Używana jest biblioteka obsługująca ekran LCD.

Inicjalizacja

W inicjalizacji ustawiany jest odpowiedni kierunek ekranu.

```
 lcdWriteReg(0x03, 0x1030 ^ (1<<4 | 1<<5));
```

Diody LED

W projekcie jest używana dioda LED służąca do oznaczania, czy wpisano dobry kod, a co za tym idzie, czy zamek jest otwarty, czy zamknięty.

Inicjalizacja

W projekcie użyto metod z biblioteki Board_LED.h.

```
 LED_Initialize();
 LED_Off(2);
 LED_Off(3);
```

UART

UART w projekcie używany jest jedynie w celu debugowania. Ostateczna wersja projektu nie wymaga tego komponentu.

Inicjalizacja

```
void UART_Init(){
  PIN_Configure(0, 2, 1, PIN_PINMODE_TRISTATE, 0);
  PIN_Configure(0, 3, 1, PIN_PINMODE_TRISTATE, 0);
  LPC_UART0->LCR = (1<<7) | 3;
  LPC_UART0->DLL = 13;
  LPC_UART0->DLM = 0;
  LPC_UART0->FDR = 1 | (15<<4);
  LPC_UART0->LCR = 3;
}
```

Wbudowane Przyciski

Używany jest przycisk “key2” jako przycisk serwisowy służący do ustawiania poprawnego kodu. W tym celu wykorzystane zostało przerwanie zewnętrzne EINT0 pochodzące od przycisku “key2”, generowane na zboczu opadającym (wciśnięcie przycisku).

Inicjalizacja

```
 void EXTINTInit()
 {
   PIN_Configure(2, 10, 1, PIN_PINMODE_PULLUP, 0);
   LPC_SC->EXTMODE = 1;  // ustawienie przerwania zboczem
   LPC_SC->EXTPOLAR = 0; // ustawienie zboczem opadającym
   NVIC_EnableIRQ(EINT0_IRQn);
   LPC_SC->EXTINT = 1;
 }
```

Przerwanie

```
void EINT0_IRQHandler(void)
{
  'pobierz od użytkownika nowy kod'
  'zapisz kod do pamięci FRAM'
  LPC_SC->EXTINT = 1; // czyszczenie informacji o przerwaniu
}
```

Timer

Wykorzystywane są dwa timery - jeden do obsługi automatycznego zablokowania zamka po pewnym czasie tego jak jest otwarty. Drugi do obsługi automatycznego czyszczenia się kodu, który nie został do końca wypełniony.

Inicjalizacja

```
void TIM0Init(){
  LPC_TIM0->PR = 0;
  LPC_TIM0->MR0 = (SystemCoreClock/4 * 5) - 1;
  LPC_TIM0->MCR |= 3; //ustaw generowanie przerwania dla mr0
  NVIC_EnableIRQ(TIMER0_IRQn);
}

void TIM1Init(){
  LPC_TIM1->PR = 0;
  LPC_TIM1->MR0 = (SystemCoreClock/4 * timeToClearPasscode) - 1;
  LPC_TIM1->MCR |= 3; //ustaw generowanie przerwania dla mr0
  NVIC_EnableIRQ(TIMER1_IRQn);
}
```

Przerwania

```
void TIMER0_IRQHandler(void){
  'zgaś diodę LED sygnalizującą otwarty zamek'
  LPC_TIM0->IR = 1; // usuń sygnał przerwania
}

void TIMER1_IRQHandler(void){
  'wyczyść wpisany kod'
  LPC_TIM1->IR = 1; // usuń sygnał przerwania
}
```