151021
numer indeksu
Paweł Tarasiuk
imię i nazwisko

Kierunek Informatyka
Specjalność Inżynieria oprogramowania i analiza danych
Rok akademicki 2011/12
Semestr 7

Systemy wbudowane

Gra "refleks"

przy wykorzystaniu płytki Eduboard LPC2148

1 Opis projektu

Przygotowany projekt stanowi grę zręcznościową, która wymaga od gracza szybkich reakcji joystickiem na bodziec, jakim jest ukazywanie się znanych symboli na matrycy LED. Program wykonywany jest na płytce Eduboard LPC2148 - wystarczy że jest ona podłączona do zasilania i wgrany jest przygotowany w ramach projektu program. Jeżeli płytka jest podłączona do partu COM, możliwe jest odczytywanie dodatkowych informacji logujących za pomocą przygotowanego w ramach projektu programu działającego po stronie komputera (przesyłanie komunikatów odbywa się za pomocą przygotowanego w ramach projektu prostego protokołu z sumami kontrolnymi).

Program na płytkę został napisany w języku C i skompilowany za pomocą GNU ARM z kompilatorem z linii 3.4. Terminal do odczytywania komunikatów także został napisany w C, zaś skompilowany został za pomocą kompilatora GNU pod platformą Cygwin.

Poniższa tabela opisuje zakres projektu:

Zakres funkcjonalności	**Zastosowanie**
Joystick i przycisk P0.14	Odbieranie ruchów gracza i informacji o gotowości do gry
Matryca LED	Wyświetlanie symboli, na które ma reagować gracz
Wyświetlacz LCD	Wyświetlanie komunikatów, w tym: wyniku końcowego
Dioda RGB	Sygnalizowanie liczby żyć wynikającej z pomyłek gracza
Silnik	Wykonywanie ruchu na wiwat, gdy gracz wygra
UART	przesyłanie opcjonalnych komunikatów przez port COM

Szczegółowe rozważanie organizacji pracy zespołowej, podziału obowiązków oraz harmonogramu jest niemożliwe z przyczyn formalnych, ze względu na specyficzne warunki powstawania projektu.

2 Przebieg gry

Po uruchomieniu zaprogramowanego urządzenia oraz po zakończeniu gry, ekran LCD jest podświetlony i wyświetla napis informujący o tym, że wciśnięcie przycisku $P0.14$ spowoduje rozpoczęcie gry. Gra polega na tym, że na matrycy LED wyświetlane są przez pewien czas pseudolosowe wzory, aż w wyniku losowania program wykona instrukcję wyświetlającą zapytanie do użytkownika - to znaczy, symbol jednej z czterech strzałek na płaszczyźnie, albo kojarzący się z lotką strzały symbol wektora wskazującego "za diagram". Zapytanie jest wyświetlane do czasu, aż gracz wykona odpowiadający mu ruch joystickiem - jednocześnie zapamiętywane są stany regularnie inkrementowanego przez timer licznika w chwili wyświetlenia zapytania oraz uzyskania odpowiedzi. Po 10 poprawnych odpowiedziach, wynikiem ostatecznym jest średni czas reakcji gracza na zapytania - wynik zostaje wypisany na wyświetlaczu LCD, który po zakończonej grze znów jest podświetlony. Zakończenie gry sukcesem jest "nagradzane" hałaśliwym ruchem silnika. Podczas gry raz może dojść do sytuacji, że gracz wykona ruch niepasujący do zapytania (albo ruch w momencie, kiedy żadnego zapytania nie ma, a matryca wyświetla obraz pseudolosowy). Drugi błąd oznacza zakończenie gry i porażkę. Liczbę "żyć" gracza (2, 1 lub 0) sygnalizuje dodatkowo napis na niepodświetlonym ekranie LCD (aby gracz mógł się skoncentrować na matryce LED, nie zaś na podświetleniu wyświetlacza) oraz kolor diody RGB: zielony na początku gry, żółty po jednym błędzie, oraz czerwony w przypadku porażki.

Jeżeli przez RS232 płytka zostanie podłączona do portu COM komputera, na którym uruchomiony zostanie przygotowany w ramach projektu terminal, możliwe będzie wygodne odczy-

tywanie dodatkowych informacji o czasach reakcji na poszczególne zapytania, liczbie żyć oraz wyniku końcowym. Aby od razu przekonać się, że połączenie działa, na początku każdej rozgrywki wyświetlany jest ponadto komunikat powitalny.

3 Opis algorytmu

Program po uruchomieniu czeka na wciśnięcie przycisku P0.14 (w standardowy sposób badając jego stan przez GPIO). Automatycznie przy starcie programu tworzona jest także procedura odpowiadająca za zmiany w danych służących do generowania pseudolosowych ciągów bitów.

3.1 Liczby pseudolosowe

Zastosowana metoda jest podobna do Blum-Blum-Shub, problemem jest jednak brak prawdziwie losowego ziarna. Dlatego w 16-bitowej zmiennej "radnom" przechowywana jest pewna liczba, która wykorzystywana jest w następujący sposób:

- Przy generowaniu losowego bitu, jest ona podnoszona do kwadratu modulo 64829 (16-bitowy iloczyn dwóch możliwie dużych liczb pierwszych). Większe liczby losowe generowane są jako ciągi tworzonych w ten sposób losowych bitów.
- Jeden z tworzonych procesów odpowiada za psucie zmiennej random co pewien czas w możliwie chaotyczny sposób - aby ciągi bitów nie powtarzały się zbyt często (operacje wykonywane są na zbyt małych liczbach, aby metoda Blum-Blum-Shub działała dobrze) i całość była mniej przewidywalna.

3.2 GPIO

Obsługa przycisku P0.14 do startowania gry jest bardzo prosta - po ustawieniu odpowiedniego bitu $IODIR$ na 0 (odczyt), badana jest wartość bitu $IOPIN$ o odpowiedniej pozycji - gdy wynosi ona 0, to znaczy, że przycisk jest wciśnięty.

W przypadku joysticka zastosowany został ciekawszy sposób badania wielu bitów naraz (konkretnie: P0.16 do P0.20). W każdej iteracji zapamiętywana jest poprzednia wartość $IOPIN$ w 32-bitowej zmiennej $oldIOPIN$, po czym zapisywana w zmiennej jest wartość wyrażenia `(oldIOPIN&~IOPIN)&0x001f0000`. Jeżeli nie jest ona zerem, to stan joysticka zmienił się w porównaniu z poprzednią iteracją (pewne bity zmieniły stan z 1 na 0) - badając poszczególne bity wyznaczonej w ten sposób zmiennej łatwo ustalić, który ruch został wykonany.

3.3 Timer

W projekcie wykorzystany został TIMER1, który ustawiany jest przy wykorzystaniu mechanizmu przerwań. Zarządzaniem przerwaniami zajmuje się Vectored Input Controller (VIC), który posiada 32 sloty przypisane urządzeniom mogącym generować przerwania (zaliczają się do tego timery).

Timer jest ustawiony tak, aby generował przerwania typu IRQ. Funkcja ustawiająca timer jest wywoływana tak, aby co 2 ms wykonywana była procedura odpowiedzialna za aktualizację stanu matrycy LED oraz podbicie specjalnego licznika, którego wartości są porównywane w celu ustalenia ile czasu upływa od zapytania do reakcji gracza.

3.4 Matryca LED

Do obsługi wyświetlacza diodowego wykorzystywany jest szeregowy interfejs urządzeń peryferyjnych (SPI).

Za pomocą linii CS (Chip Select) wybierany jest odpowiedni układ (w tym wypadku - kolumna 8 diod), po rozpoznaniu którego przesyłana jest porcja danych (dla wyświetlacza diodowego - bajt którego bity opisują stan poszczególnych diod).

Wywoływana raz na 2 ms procedura ustawiająca stan diod za każdym razem ustawia jedną kolumnę, zatem odświeżenie całego wyświetlacza trwa 16 ms.

3.5 Wyświetlacz LCD

Wyświetlacz wierszowy LCD pozwala wyświetlić dwa wiersze po 15 znaków tekstu. Do kontroli wyświetlacza wykorzystany został mechanizm GPIO. Sterownik wyświetlacza LCD posiada następujące rejestry:

- ControlWR – sterowanie (P0.22)
- DataWR – zapis danych do sterownika (P1.16 – P1.23)
- ControlRD – sterowanie i status wyświetlacza (P1.24)
- DataRD – odczyt danych ze sterownika (P1.25)

Pin P0.30 pozwala ponadto sterować podświetleniem wyświetlacza.

W ramach projektu do wypisywania na wyświetlaczu całych napisów przygotowana została jedna funkcja, która w argumentach przyjmuje napis oraz informację o tym, czy wyświetlacz ma pozostać podświetlony (zawsze jest podświetlany podczas pojawiania się informacji, aby zasygnalizować że stało się coś nowego). Szczególnej obsługi wymagał znak przejścia do nowej linii, który powoduje że dalsze pisanie odbywa się w drugim wierszu wyświetlacza.

3.6 Silnik

Silnikiem zamontowanym na płytce można sterować za pomocą modulacji szerokości impulsów (PWM).

Inicjalizacja silnika polega na ustawieniu funkcji PWM4 i PWM6 dla pinów P0.8 i P0.9, a następnie wyłączenie silnika poprzez ustawienie stanu niskiego na pinie P0.10. Następnie możliwe jest ustalenie następujących ustawień:

- Prescale Counter ustawiony na 0 będzie oznaczał inkrementację PWMTC przy każdym takcie zegara
- Match Control Register pozwala ustawić, aby PWMTC był resetowany, gdy PWMMR0 = PWMTC
- W rejestrze PWMMR0 ustawiana jest szerokość impulsu: 0x1000
- Początkowe wartości PWM_MR4 i PWM_MR6 to 0, aby stosunek sygnału wysokiego do niskiego wynosił 0 i silnik się nie obracał wcale.
- Za pomocą Latch Enable Register należy zatwierdzić zmiany wartości PWM_MR4 i PWM_MR6.

- Ustawiony za pomocą PWM_PCR tryb "single edge" będzie oznaczał że stan wysoki będzie się utrzymywał aż do resetu licznika.
- Poprzez rejestr PWM_TCR dokonywane jest uruchomienie licznika PWM Timer Counter i włączenie trybu PWM.

Później wystarczy operować szybkością obrotów silniczka za pomocą rejestrów PWM_MR4 i PWM_MR6 i zatwierdzać zmiany za pomocą PWM_LER.

3.7 Dioda RGB

Dioda RGB, podobnie jak silnik - obsługiwana jest przez PWM. Mechanizm postępowania oraz sam kod odpowiadający za obsługę PWM dla diody jest łudząco podobny do tego, który jest związany z silnikiem. Ustawienia pinów dla poszczególnych rejestrów PWM to: PWM2 dla P0.7 określa poziom światła czerwonego, PWM4 dla P0.7 - niebieskiego, zaś PWM6 dla P0.9 - zielonego.

Należy zauważyć, że związek między diodą a silnikiem jest bardzo duży, gdyż reagują na PWM w ten sam sposób - dlatego obroty silniczka w jedną lub drugą stronę muszą być związane ze świeceniem diody odpowiednio na niebiesko i na zielono, zaś gwałtowne zmiany kolorów diody powinny być wykonywane przy wyłączonym silniku (tak jest to zrobione w projekcie). Niebieskie zabarwienie diody podczas ruchu silnika po wygranej grze nie kłóci się jednak w żaden sposób z instrukcją rozgrywki i znaczeniami kolorów diody - można przyjąć, że to element "fanfar" dla zwycięzcy.

3.8 Komunikacja z terminalem

Do przesyłania danych dla terminala przez port COM wykorzystane są funkcje zawarte w przygotowanym przez Embedded Artists frameworku, takie jak uproszczony w stosunku do standardu C, lecz mający podobną składnię printf. Interfejs RS232 zawarty na płytce połączony jest z UART#0.

W ramach inicjalizacji UART odpowiednie piny GPIO przestawiane są na funkcje UART: RxD0 i TxD0 odpowiedzialne odpowiednio za odbieranie i przesyłanie danych:

```
PINSEL0=(PINSEL0&0xfffffff0)|0x00000005;
```

Chcąc korzystać z UART należy wyznaczyć współczynnik będący pomnożonym przez 16 ilorazem częstotliwości zegara procesora (60 MHz) przez szybkośś transmisji (38400).

```
#define UART_DLL_VALUE(unsignedshort)((PCLK/(CONSOL_BITRATE*16.0))+0.5)
```

Następnie wartość umieszczana jest w rejestrach DLL i DLM; wcześniej ustawiamy bit DLAB w rejestrze LCR:

```
UART_LCR=0x80;
UART_DLL=(unsignedchar)(UART_DLL_VALUE&0x00ff);
UART_DLM=(unsignedchar)(UART_DLL_VALUE>>8);
```

Na koniec ustalany jest format danych opisywany przez bity w rejestrze LCR. Ustawienie dla 8 bitów danych i 1 bitu stopu, bez parzystości to:

```
UART_LCR=0x03;
```

Znaki mogą być później wysyłane przy wykorzystaniu rejestru UART_THR.

4 Analiza skutków awarii

Awarie o krytycznych skutkach:

- Joystick i przycisk, których stany odczytywane są przez GPIO - ich awaria uniemożliwiłaby komunikację z graczem.
- Awaria znacznej części diod z matrycy LED - gdyby symbole przestały być czytelne, łatwo można by je było pomylić z wzorami losowymi, nawet gdyby pozostały rozróżnialne między sobą.
- Timer - jego awaria spowodowałaby brak odświeżania wyświetlacza diodowego oraz brak pomiarów czasu.
- Jednoczesna awaria wyświetlacza LCD i transmisji przez port COM - uniemożliwiłaby odczytanie wyników czasowych, przez co gracz mógłby uznać grę za bezsensowną.

Awarie o poważnych skutkach:

- Awaria nieznacznej części diod z matrycy LED - komfort gry byłby znacznie zaburzony, gdyby symbole były wyświetlane niedokładnie - w szczególności, gracz miałby prawo myśleć, że to losowy wzór ułożył się w kształt podobny do symbolu zapytania i przez to go z początku zignorować, pogarszając swój wynik.
- Wyświetlacz LCD - gracz musiałby wiedzieć o konieczności rozpoczęcia gry przyciskiem P0.14. Ponadto odczytanie wyników rozgrywki byłoby możliwe tylko przez port COM (co jest mniej wygodne, bo wymaga komputera).

Awarie o niegroźnych skutkach:

- Dioda RGB - stanowi tylko dekorację a sygnalizowane przez nią informacje można odczytać także w inny sposób.
- Silnik - brak kręcenia się na wiwat - po prostu bez dźwięku silnika po wygranej grze zaburzony zostałby klimat gry.

Bibliografia

- LPC214x User Manual - `http://ics.nxp.com/support/documents/microcontrollers/pdf/user.manual.lpc2141.lpc2142.lpc2144.lpc2146.lpc2148.pdf`.
- LPC2148 Education Board User's Guide - `http://www.zsk.p.lodz.pl/~morawski/SCR&ES/Guides&Schematics/LPC2148_Education_Board_Users_Guide-Version_1.2_Rev_B.pdf`.
- `http://downloads.sourceforge.net/project/lpc21isp/lpc21isp/1.83/lpc21isp_183.tar.gz` - lektura kodu źródłowego była pomocna przy tworzeniu własnego terminala.