Tomasz Adamczyk
Bartosz Bielawski
Tomasz Włostowski

Warszawa, 18 czerwca 2008 r.

Inteligentne Systemy Robotyczne (ISR)

Projekt

Temat: Sterownik do czujnika położenia pcBIRD

1 Zadanie do wykonania

Projekt polegał na napisaniu zestawu programów obsługujących urządzenie pcBIRD. pcBIRD jest systemem pomiarowym służącym do precyzyjnego określania położenia czujnika. Urządzenie składa się z trzech części:

- nadajnika podłączanego do karty ISA,
- odbiornika podłączanego do karty ISA,
- karty ISA umieszczonej w komputerze klasy PC.

Projekt składa się z następujących części:

- programu serwera pozwalającego na łatwy dostęp do danych pomiarowych za pomocą TCP/IP,
- przykładowej biblioteki klienckiej,
- przykładowego programu z graficznym interfejsem użytkownika (GUI),
- klasy czujnika wirtualnego systemu MRROC++.

Dodatkowym założeniem jest to, by napisane programy działały pod kontrolą systemu Linux.

2 Realizacja projektu

Dzięki zastosowaniu karty zgodnej ze standardem ISA możliwa jest komunikacja za pomocą portów komputera klasy PC. Z poziomu oprogramowania do komunikacji stosowane są dwie instrukcje procesora rodziny x86:

- `in` – odczyt danych z portu,
- `out` – zapis danych do portu.

Długość słowa czytanego/zapisywanego zależna jest od operandów instrukcji. W języku C instrukcje in i out mają swoje bezpośrednie odpowiedniki – `inb/outb` dla słowa 8-bitowego i `inw/outw` dla słowa 16-bitowego. Komunikacja z kartą odbywa się tylko w porcjach 16-bitowych.

Komunikacja z kartą pcBIRD przebiega za pomocą dwóch portów do zapisu (konfiguracja i komendy) oraz dwóch do odczytu (stan i dane). Bity statusu określają, czy karcie można wydać kolejne polecenie oraz czy karta zrealizowała poprzednie zadanie i są gotowe dane do odczytu.

Ponieważ ciągłe sprawdzenia bitu gotowości powodowało by znaczne obciążenie procesora,

zdecydowano się na obsługę odczytów za pomocą przerwań. W ogólności przerwanie jest to odpowiedź systemu na zajście jakiegoś zdarzenia i jest z nim zazwyczaj związana jakaś funkcja, która to zdarzenie obsługuje. W przypadku pcBIRDa urządzenie może zgłaszać albo gotowość do przyjęcia polecenia albo gotowość danych do odbioru. Dzięki takiemu podejściu możliwe jest zwolnienie programu z ciągłego odpytywania poprzez porty, czy dane już nadeszły i przerzucenie tego obowiązku na wydajne mechanizmy systemowe. W celu użycia przerwań konieczne okazało się napisanie własnego modułu jądra.

2.1 Moduł jądra – obsługa przerwań od karty pcBIRD

Moduł jądra jest pewnym zbiorem funkcji, które mogą być dynamicznie dołączane do działającego już jądra. Obsługa przerwań w przestrzeni użytkownika (userspace) została dodana dopiero do jądra Linux w wersji 2.6.23 (testowane na 2.6.24.7), dlatego do działania program wymaga co najmniej Linuksa w takiej wersji. Moduły mogą być ładowane tylko przez superużytkownika (root).

Moduł obsługujący kartę pcBIRD nazywa się `pcbird_uio.ko`. Podczas ładowania modułu do jądra należy podać jako parametr numer portku na którym nasłuchuje karta oraz numer przerwania używanego do zgłaszania gotowości. Przykładowe ładowanie modułu, z portem numer 0x400 oraz przerwaniem 15 wygląda następująco.

```
#modprobe ./pcbird_uio.ko port=0x400 irq=15
```

Jeżeli moduł został poprawnie podłączony do jądra w logu jądra (polecenie `dmesg`) powinna pojawić się informacja:

```
PCBird: probe done [port = 0x400, irq = 15].
```

W przeciwnym wypadku (brak karty, zły numer portu) pojawi się następujący komunikat:

```
PCBird: probe failed.
```

Załadowanie modułu jądra powoduje stworzenie w katalogu `/dev` urządzenia `uio0`. Urządzenie to służy do zgłaszania przerwań programom pracującym w przestrzeni użytkownika. W momencie nadejeścia przerwania odczyt z tego pliku zwraca informację, ile przerwań zgłoszono już wcześniej. Możliwe jest dzięki temu sprawdzenie, czy żadne przerwanie nie zostało pominięte.

2.2 Program serwera `birdsv`

Program serwera ma dwojakie zadanie:

- odbiór danych z karty,
- reagowanie na komendy klientów i odsyłanie danych.

Ze względu na dwa zadania program składa się z dwóch wątków:

- wątek obsługi sprzętu, odczytujący dane i zapisujące je do tablicy wyników,
- wątek obsługujący interfejs TCP/IP.

Wątki oraz mutexy stosowane do ich synchronizacji pochodzą z biblioteki pthreads (POSIX Threads).

Do uruchomienia programu konieczne są uprawnienia superużytkownika, ponieważ program wymaga bezpośredniego dostępu do portów. W razie uruchomienia z innego konta zostanie zgłoszony błąd i program przerwie pracę.

Wątek sprzętowy w pętli odbiera dane z karty. Funkcją, która czeka na pojawienie się danych jest blokująca funkcja `read()` wykonywana na desktyptorze urządzenia `/dev/uio0`. Dzięki takiemu podejściu system operacyjny może opóźnić wykonanie wątku aż do momentu gdy pojawią się dane do przetworzenia.

Wątek obsługujący klientów wykonuje w nieskończonej pętli dwie czynności:

- obsługę przychodzących połączeń i żądań klientów,
- wysyłanie danych.

Każdy podłączony klient ma zdefiniowany, jeden z dwóch, tryb obsługi:

- żądanie pomiaru – pojedyncza odpowiedź,
- ciągłe wysyłanie danych (streaming).

Maksymalna liczbę podłączonych klientów może być zmieniona w kodzie źródłowym (MAX_CLIENTS).

Dane odsyłane są do klienta w następującej strukturze:

```
struct pos_ang {
    uint16_t x,y,z,a,b,g;
    struct timeval ts;
};
```

Znaczenie pól jest następujące:

- x, y, z – współrzędne kartezjańskie czujnika,
- a, b, g – kąty nachylenia osi czujnika (azimuth, elevation, roll),
- ts – pole opisujące czas pomiaru (typ używany w funkcji gettimeofday()).

Rozmiar tej struktury to 20 B, zachowanie końcówkowości możliwe jest dzięki użyciu funkcji `htons()` i `htonl()`, dlatego klient powinien użyć analogicznych funkcji po swojej stronie (`ntohs()` i `ntohl()`).

Podczas uruchamiania serwera możliwe jest podanie następujących parametrów:

- `-ioport n` – ustawia na którym porcie przebiega komunikacja z pcBIRDem (standardowo 0x200),
- `-tcpport n` – ustawia na którym porcie TCP/IP będzie nasłuchiwał serwer (standardowo 12345),
- `-nd` – zapobiega daemonizacji serwera (działa w pierwszym planie).

2.3 Biblioteka kliencka `birdclient`

Biblioteka kliencka ułatwia komunikację z serwerem, dzięki dostarczaniu pewnych podstawowych funkcji. Te funkcje to:

- `int pcbird_connect(const char *addr, unsigned short port);` - połączenie z serwerem, zwraca deskryptor gniazda,
- `void pcbird_disconnect(int fd);` - zakończenie połączenia z serwerem,
- `int pcbird_start_streaming(int fd);` - rozpoczęcie transmisji strumienia danych z serwera,
- `int pcbird_stop_streaming(int fd);` - zakończenie transmisji strumienia danych

z serwera,

- `int pcbird_get_single_position(int fd, pcbird_pos_t *p);` - odczyt pojedynczego pomiaru,
- `int pcbird_data_avail(int fd);` - sprawdzenie czy nadeszły dane z serwera,
- `int pcbird_get_streaming_position(int fd, pcbird_pos_t *p);` - nieblokujący odczyt

W dwóch funkcjach użyto wskaźnika do struktury `pcbird_pos_t`. Struktura ta wygląda następująco:

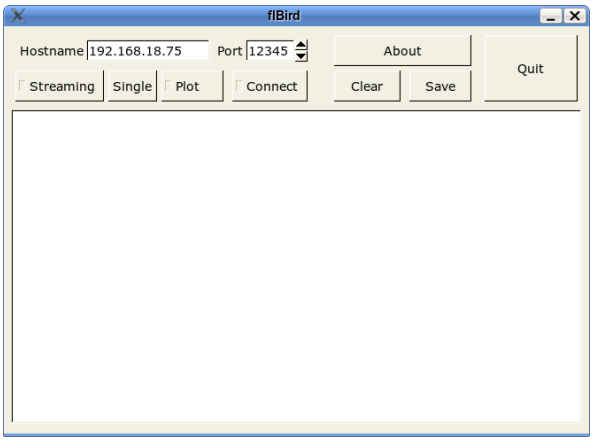
```
typedef struct {
    float x, y, z;  // pozycja
    float a, b, g;  // katy (a = azimuth, b = elevation, g = roll)
    float distance; // odleglosc
    uint32_t ts_sec, ts_usec; // timestamp
} pcbird_pos_t;
```

Podczas odbierania danych z serwera dane są przetwarzane do wygodniejszej postaci i zapisywane w takiej strukturze. Możliwe jest użycie funkcji `select()` lub `poll()` na deskryptorze gniazda w celu sprawdzenia, czy są gotowe dane do odczytu.

2.4 Klient z GUI – flBird

Program flBird jest graficznym klientem napisanego systemu. Wykorzystuje trzy biblioteki:

- bibliotekę FLTK do tworzenia GUI,
- bibliotekę SDL do rysowania wykresów położenia,
- bibliotekę kliencką birdclient.



Ilustracja 1: Wygląd głównego okna programu flBird

Funkcje pól edycyjnych i przycisków są następujące:

- pole hostname: nazwa hosta na którym uruchomiony jest serwer,
- pole port: podanie numeru portu na którym nasłuchuje serwer,
- przycisk Streaming – włączenie/wyłączenie trybu ciągłego przesyłania danych,

- przycisk Single – wykonanie pojedynczego pomiaru,
- przycisk Plot – włączenie/wyłączenie okna z wykresem (okno na rysunku 2),
- przycisk connect – połączenie/rozłączenie klienta z serwerem,
- przycisk Clear – czyszczenie okna komunikatów tekstowych,
- przycisk Save – zapis zawartości okna komunikatów tekstowych,
- przycisk Quit – zakończenie działania programu.

Ilustracja 2: Wygląd okna z wykresami

Okno pokazane na rysunku 2. zawiera 6 wykresów, znajdujących się w dwóch układach współrzędnych. Górny układ współrzędnych obrazuje położenie czujnika we współrzędnych kartezjańskich, dolny układ pokazuje ustawienie osi czujnika. Powyżej każdego wykresu znajduje się legenda kolorów używanych do oznaczania konkretnych współrzędnych.

2.5 Klasa ecp_mp_s_pcbird

System MRROC++ jest frameworkiem pozwalającym na tworzenie złożonych sterowników wielorobotowych składających się z modułów. Jednym z takich modułów są czujniki różnego rodzaju. System MRROC++ posiada zunifikowaną obsługę czujników za pomocą klas VSP. Na podstawie przykładowych klas oraz biblioteki klienckiej napisano klasę umożliwiającą komunikację systemu MRROC++ z kartą pcBIRD.

3 Podsumowanie

Zastosowanie precyzyjnego czujnika, jakim jest system pcBIRD, umożliwia lepszą kalibrację ruchów robota. Dzięki wykonanemu projektowi możliwe będzie włączenie tego czujnika do sieci czujników systemu MRROC++ tym samym zwiększając możliwości tego systemu.

Użyte podczas tworzenia projektu oprogramowanie jest darmowe, najczęściej na licencji GPL i pisane zgodnie ze standardami tworzenia software'u. Stwarza to możliwość powtórnego użycia kodu w systemach zgodnych z POSIX.

4 Bibliografia

- Dokumentacja systemu pcBIRD,
- dokumentacja jądra Linux 2.6.23,
- dokumentacja bibliotek FLTK i SDL.