Enginyeria de Sistemes TIC

Xarxes de Comunicació

Pràctica 1 - Detecció d'errors

Francisco del Águila López

Octubre 2014
Escola Politècnica Superior d'Enginyeria de Manresa
Universitat Politècnica de Catalunya

1 Objectiu

L'objectiu d'aquesta pràctica és definir un mòdul en C que serveixi per fer una detecció d'errors. Aquest mòdul es farà servir per implementar un protocol d'enllaç fiable.

2 Codis detectors d'errors

Existeixen diversos mecanismes de detecció de errors en transmissions digitals. A [Wiki-Error] es troben classificats de la següent manera:

1. Codis de repetició
2. Bits de paritat
3. Checksums
4. Comprovació de redundància cíclica (CRC)
5. Funcions de hash de xifrat
6. Codis correctors d'errors

Aquesta classificació es de menys a més eficaços. També estan ordenats de manera creixent en complexitat. En la última posició es troben codis que a més de detectar errors, també són capaços de corregir-los.

A la pràctica s'ha de desenvolupar un codi detector basat en un Checksum [Wiki-Check] i basat en un CRC [Wiki-CRC]. La complexitat de l'algoritme de Checksum és molt

reduïda. Aquest algoritme es dissenyarà similar a l'algoritme de Checksum que es fa servir al protocol TCP i al protocol IP de Internet. En el cas del disseny de l'algoritme de CRC es farà us de les llibreries de C dels AVR [AVR-libc], ja que existeix en particular la llibreria /util/crc16.h que facilitarà la feina.

3 Disseny del Checksum

3.1 Definició

El Checksum que es dissenyarà serà el que es fa servir a [Check-TCP/IP] amb la diferència que la mida serà de 1 Byte. Aquest algoritme consisteix en la suma de totes les paraules que formen el bloc de dades, en el nostre cas aquestes paraules són de 1 Byte. Un cop feta la suma, s'agafen els bits que formen el *carry* i es tornen a sumar. El resultat final, invertint els bits (complement a 1) és el Checksum.

Per comprovar el Checksum es fa la mateixa operació incloent el Checksum. Si el resultat final són tots els bits a "1" se suposa que no hi hagut errors i el Checksum és correcte.

Les característiques que defineixen el Checksum són:

- La mida del Checksum és de 1Byte. Això vol dir que la quantitat de bits redundants que s'afegeixen són 8.
- La mida del bloc de dades amb la que es calcularà el Checksum serà variable. Aquest bloc de dades serà el que formarà la trama del protocol d'enllaç.
- El bloc de dades que protegirà el Checksum està format per caràcters que poden ser transmesos en Morse. Això vol dir que només s'admeten les lletres de A-Z i números de 0-9. Els caràcters especials que no formen part del codi Morse internacional així com els accents, no estan permesos. De la mateixa manera, no s'admeten lletres minúscules.
- La codificació dels caràcters admesos és ASCII, ja que la conversió a codi Morse es realitza a la capa física.
- El resultat del càlcul del Checksum dóna lloc a 1Byte que pot prendre qualsevol valor. Per tant, pot ser que no sigui un possible caràcter vàlid. Cal un mecanisme que garanteixi sempre els caràcters vàlids.

Per garantir que el càlcul del Checksum es pugui codificar amb caràcters de Morse vàlids, es fa servir la codificació hexadecimal. Es a dir, aquest Byte es codificarà amb 2 caràcters ASCII en format hexadecimal. Els caràcters hexadecimals 0-F són caràcters vàlids per transmetre en Morse.

3.2 Implementació

La implementació de la funció que calcula el Checksum es farà en llenguatge C. La funció Checksum ha de cumplir les següents condicions:

- El nom de la funció és check_morse()
- El paràmetre d'entrada és un *string* que conté el bloc de dades que es vol protegir amb el Checksum.
- El resultat de la funció ha de ser els dos caràcters que codifiquen en hexadecimal el Byte que s'ha calculat.

4 Disseny del CRC

El funcionament d'un CRC està descrit a [Wiki-CRC]. Al igual que en el cas del Checksum, l'algoritme per afegir un CRC a un *stream* de bytes o per comprovar si s'han produït errors en aquest *stream* és el mateix. Poden haver diferents mides de bits afegits per garantir la detecció d'errors. Les mides més comuns són de 16 bits, però en el cas de la pràctica farem servir una mida de CRC de 8bits. A major mida de CRC, més capacitat de detecció d'errors.

La llibreria /util/crc16.h ofereix 3 funcions de càlcul per 3 variants de CRC de 16 bits i una funció de càlcul per el CRC de 8 bits. La funció

```
static __inline__ uint8_t _crc_ibutton_update (uint8_t __crc, uint8_t __data)
```

accepta com a paràmetres 2 bytes. __data és el byte que s'ha d'anar introduint corresponent al *stream* que es vol processar. __crc és el valor inicial o bé el valor calculat anterior del candidat a ser els bits corresponent al CRC. El valor retornat és el nou valor del CRC calculat. Això vol dir que aquesta funció s'ha d'anar cridant successivament fins processar tot el *stream* corresponent. El valor inicial amb el que s'ha de començar és en aquest cas tot zeros.

Per calcular el CRC s'ha de processar el *stream* de bytes. El valor final retornat és el valor del CRC.

Per comprovar el CRC s'ha de processar el *stream* de bytes que ja tindrà inclòs com últim byte el CRC correcte. Si el valor retornat final és tot zeros implica que no s'han detectat errors. Recordeu, que en el nostre cas, el *stream* que rebem té l'últim byte desdoblat en dos caràcters. Això és així degut al requeriment que aquest CRC s'ha de poder transmetre amb codificació Morse i per tant fent servir els caràcters vàlids Morse.

Al igual que en el cas del Checksum, per garantir que es pugui codificar amb caràcters de Morse vàlids, es fa servir la codificació hexadecimal. Es a dir, aquest Byte es codificarà amb 2 caràcters ASCII en format hexadecimal. Els caràcters hexadecimals 0-F són caràcters vàlids per transmetre en Morse.

4.1 Implementació

La implementació de la funció que calcula el CRC es farà en llenguatge C. La funció CRC ha de cumplir les següents condicions:

- El nom de la funció és crc_morse()
- El paràmetre d'entrada és un *string* que conté el bloc de dades que es vol protegir amb el CRC.
- El resultat de la funció ha de ser els dos caràcters que codifiquen en hexadecimal el Byte que s'ha calculat.

5 Estudi previ

1. Defineix el fitxer de capçalera del mòdul error_morse.h que conté les funcions check_morse() i crc_morse(). Aquest mòdul també ha de contenir les funcions de comprovació tant del Checksum com del crc test_check_morse() i test_crc_morse(). Aquestes funcions reben com a paràmetre un stream de bytes amb la redundància afegida i retornen un booleà indicant si el stream és correcte o no (s'han detectat errors o no).

6 Treball pràctic

1. Dissenya el mòdul error_morse. Es recomanable que definiu funcions privades del mòdul per transformar un byte en la seva representació ASCII en hexadecimal formada per 2 caràcters byte2hex(), així com la funció que fa l'operació inversa hex2byte().
2. Dissenya un programa que s'executi al teu PC de manera que utilitzant l'entrada estàndard calculi el Checksum d'un bloc de caràcters. S'ha d'ignorar els caràcters que no siguin vàlids. El delimitador de bloc de caràcter serà el retorn de carro. A la sortida estàndard s'ha de donar el bloc d'entrada amb el Checksum afegit. El delimitador serà també el retorn de carro.
3. Dissenya un programa que s'executi al teu PC de manera que utilitzant l'entrada estàndard comprovi que el Checksum d'un bloc de caràcters és correcte. El delimitador de bloc és el retorn de carro. La sortida s'ha d'indicar si és correcta o no.
4. Dissenya un programa que s'executi a l'AVR que faci el mateix que el programa 2. Considera que l'entrada i sortida estàndard és el port sèrie.
5. Dissenya un programa que s'executi a l'AVR que faci el mateix que el programa 3. Considera que l'entrada i sortida estàndard és el port sèrie.
6. Dissenya un programa que s'executi a l'AVR que faci el mateix que el programa 2 però fent servir la detecció amb CRC. Considera que l'entrada i sortida estàndard és el port sèrie.

7. Dissenya un programa que s'executi a l'AVR que faci el mateix que el programa 3 però fent servir la detecció amb CRC. Considera que l'entrada i sortida estàndard és el port sèrie.

Referències

[Wiki-Error] `http://en.wikipedia.org/wiki/Error_detection`

[Wiki-Check] `http://en.wikipedia.org/wiki/Checksum`

[Wiki-CRC] `http://en.wikipedia.org/wiki/Cyclic_redundancy_check`

[AVR-libc] `http://www.nongnu.org/avr-libc/user-manual/group__util_ _crc.html`

[Check-TCP/IP] `http://en.wikipedia.org/wiki/IPv4_header_checksum`