6-09-2023 PROGETTO

Un processore z64 controlla la centralina di un'automobile autonoma per l'arresto di emergenza del veicolo. Il sistema è composto da una periferica TELECAMERA e RADAR . La periferica TELECAMERA viene interrogata in maniera sincrona dal processore, per acquisire delle immagini di dimensione 800x600 pixel in bianco e nero—ciascun pixel ha dimensione un byte. L'acquisizione delle immagini viene realizzata utilizzando il DMAC di sistema.

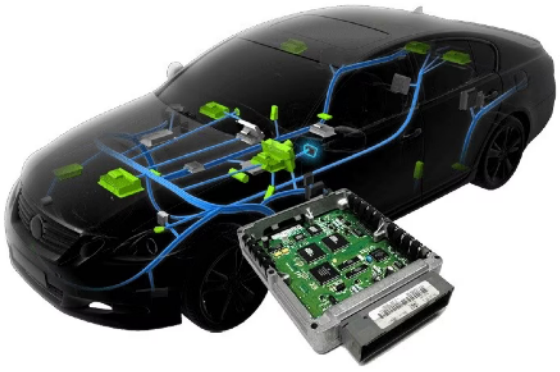
Le immagini acquisite sono delle mappe di profondità: un pixel nero (codificato come 0x00) rappresenta un punto in cui non ci sono oggetti nelle vicinanze, mentre un pixel bianco (codificato come 0xFF) rappresenta un punto in cui è presente un oggetto a distanza inferiore a 20 metri. Il processore utilizza queste immagini per pilotare la periferica sincrona ARRESTO che effettua la frenata di emergenza, qualora il numero di pixel bianchi sia maggiore di 1/4 dei pixel totali.

La periferica RADAR , invece, invia una richiesta di interruzione al processore z64 ogni volta che viene rilevato un oggetto in prossimità. Nel caso in cui il sistema rilevi un tale oggetto, il processore attiva la periferica ARRESTO per effettuare la frenata di emergenza.

Realizzare:

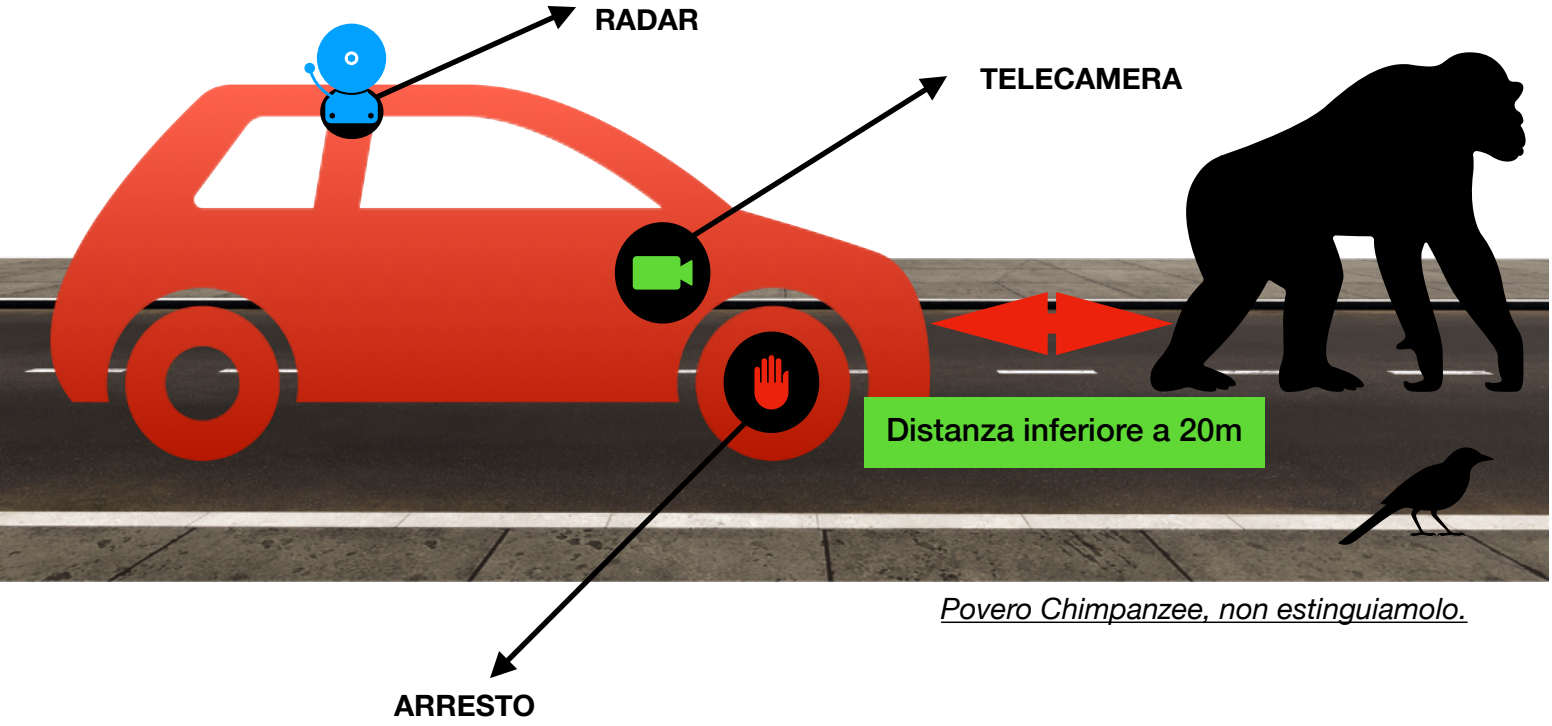
- Le interfacce delle periferiche TELECAMERA , RADAR e ARRESTO ;
- Tutto il software necessario al funzionamento del sistema, comprensivo di eventuali driver.

S V O L G I M E N T O



La periferica TELECAMERA acquisisce delle immagini di una certa risoluzione con pixel in bianco e nero e viene interrogata in Busy Waiting dal processore Z64. Queste immagini le utilizza il processore per attivare la periferica ARRESTO che effettua la frenata di emergenza qualora il numero di pixel bianchi sia maggiore dei pixel neri.

Per fugare ogni tentativo dubbioso circa la comprensione della specifica è riportato un disegno illustrativo:



Povero Chimpanzee, non estinguiamolo.

È, di norma, il radar che invia una richiesta di interruzione del flusso corrente perché c'è un oggetto in prossimità che è stato rilevato. Inoltre la prossimità di tale oggetto (codificato come un pixel bianco) si trova a distanza inferiore a 20 metri.

La telecamera acquisirà in un ciclo indefinito le immagini da trasferire al radar e quest'ultimo (leggendo dalla telecamera le immagini) attiverà o non attiverà l'arresto dell'impianto frenante.

TELECAMERA è un'interfaccia in Busy Waiting di input poiché il processore dovrà leggere all'interno di tale periferica per trasmettere le informazioni al radar ed eventualmente attivare l'arresto del veicolo.

RADAR è un'interfaccia bidirezionale di interrupt di input ed output poiché il processore decide di scrivere le informazioni relative ai pixel (nel reg.interf) in modo tale che possa elaborarle ed eventualmente attivare l'ARRESTO (tramite driver), un'interfaccia di output che riceve il valore 1 = ARRESTO o 0 = CAMMINA.

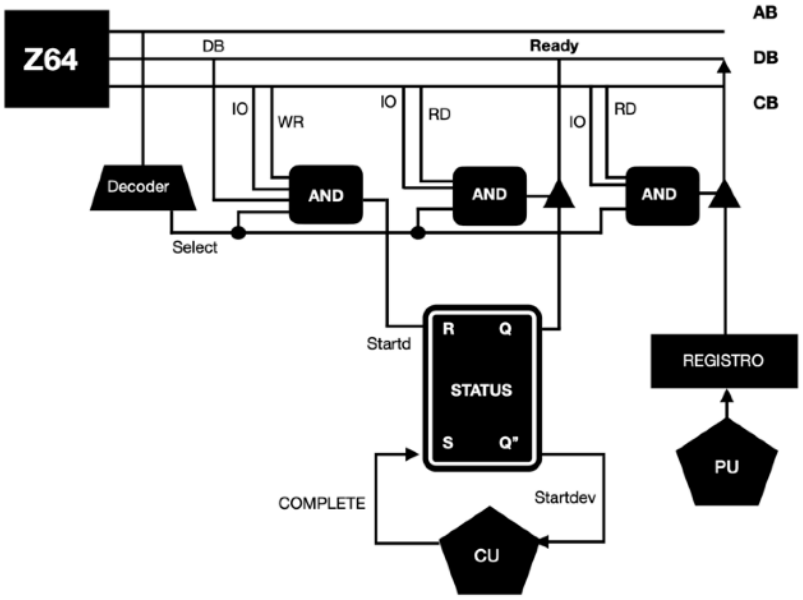
Di seguito viene implementata una possibile soluzione del codice con le relative interfacce.

Piccola nota: viene dato per assodato il fatto che la telecamera acquisisca delle immagini di dimensione 800 x 600 pixel di dimensione un byte, in bianco e nero - al contempo - il programma si limita semplicemente a leggere questo valore da un apposito registro di interfaccia.

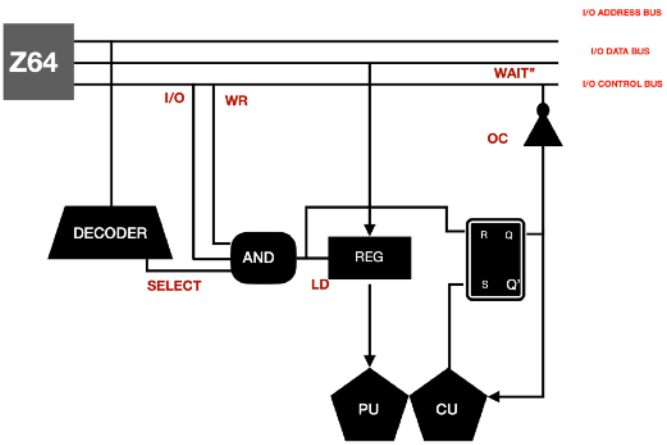
```
.org 0x800
.data
        .equ $STATUS_TELECAMERA, 0x0001
        .equ $REG_TELECAMERA, 0x0002 #ci sono i pixel di dimensione un byte
        .equ $STATUS_RADAR, 0x0003
        .equ $REG_RADAR, 0x0004
        .equ $IRQ_RADAR, 0x000a
        .equ $STATUS_ARRESTO, 0x0005
        .equ $REGISTRO_ARRESTO, 0x0006 #1 impianto frenante attivo, 0 cammina
        pixel_i_esimo_acquisito: .byte 0
        vettore_pixel: .fill 4800,1
        numero_pixel_bianchi: .quad 0
        pixel_temporaneo: .byte 0
        un_quarto_dei_pixel_totali: .quad 0
        pixel_totali: .quad 0
.text
main:
    .start:
        movq $480000, pixel_totali
        xorq %rbx, %rbx #rbx = 0, registro indice
        xorq %rdx, %rdx #rdx = 0, registro base
        movq $vettore_pixel, %rdx
        sti #abilito IF = 1
  .acquisizione:
        outb %al, $STATUS_TELECAMERA #richiedo l'acquisizione di un nuovo pixel | strlen("pixel") = 1 byte
  .bw:
        inb $STATUS_TELECAMERA, %al
        btb $0, %al
        jnc .bw
        inb $REG_TELECAMERA, %al #leggo il pixel i-esimo dalla telecamera
        movb %al, pixel_i_esimo_acquisito #pixel i-esimo
        movb pixel_i_esimo_acquisito, (%rdx, %rbx, 1) #operando in memoria
        addq $1, %rbx
        cmpq $480000, %rbx
        jz .fine
        jmp .acquisizione
   .fine:
        #ora devo leggere tutto l'array e contare i pixel bianchi
        xorq %rbx, %rbx #rbx = 0, registro indice
     .loopcycle
        movb (%rdx, %rbx, 1), pixel_temporaneo
        #è bianco?
        cmpb $255, pixel_temporaneo
        jz .oggetto_critico_minore_20_metri
      .ret:
        add $1, %rbx
        cmpq $480000, %rbx
        jnz .loopcycle
        jmp .setting #quando ha finito di scandire l'array
    .oggetto_critico_minore_20_metri:
        #conto il numero dei pixel bianchi
        addq $1, numero_pixel_bianchi
        jmp .ret
    .setting:
        #vediamo quanti sono i pixel_bianchi rispetto ad un quarto dei pixel totali
        shrq $4, pixel_totali #pixel_totali viene cancellato e viene scritto il valore di 1/4 dei pixel totali
        movq pixel_totali, un_quarto_dei_pixel_totali
        #il num dei pixel bianchi è maggiore di 1/4 dei pixel totali?
        movq $480000, pixel_totali #reimposto i pixel totali
        cmpq un_quarto_dei_pixel_totali, numero_pixel_bianchi #dest > sorg? Si o no? Se "si" ferma tutto;
        jns .ferma_tutto #sf = 0
        jmp .start
     .ferma_tutto:
        outb %al, $STATUS_RADAR #IL RADAR è pronto per sollevare una richiesta di interruzione
        #la CU del radar setta subito il flip flop così viene interrotto il flusso d'esecuzione corrente
        #questa situazione creata ad hoc in base alla quale è la CU a settare in tempo breve il flip flop è una supposizione.
        #quindi sicuro parte il driver!
   .driver 0:
        outb %al, $IRQ_RADAR #cancello la causa di interruzione
        outb $1, $REG_RADAR #scrivo 1 dentro il registro del radar (bidirezionale) = frena
        call frena
        iret
      frena:
        inb $REG_RADAR, %al #c'è il valore 1 perché bisogna fermarsi
        outb %al, $REGISTRO_ARRESTO #scrivo ufficialmente 1 sull'arresto: IMPIANTO FRENANTE BLOCCATO
        outb %al, $STATUS_ARRESTO #Attivo la periferica arresto (periferica di output) INFORMANDOLA che c'è un dato che
deve essere consumato
       .loop:
        inb $STATUS_ARRESTO, %al
        btb $1, %al
        jc .loop
        #se il ciclo è finito il dato è stato completamente consumato perciò è qui che vi è il vero blocco
        #si presume che il contenuto del registro nell'arresto si riazzeri (dato per assodato)
        ret
```

Di seguito le varie interfacce dei dispositivi.

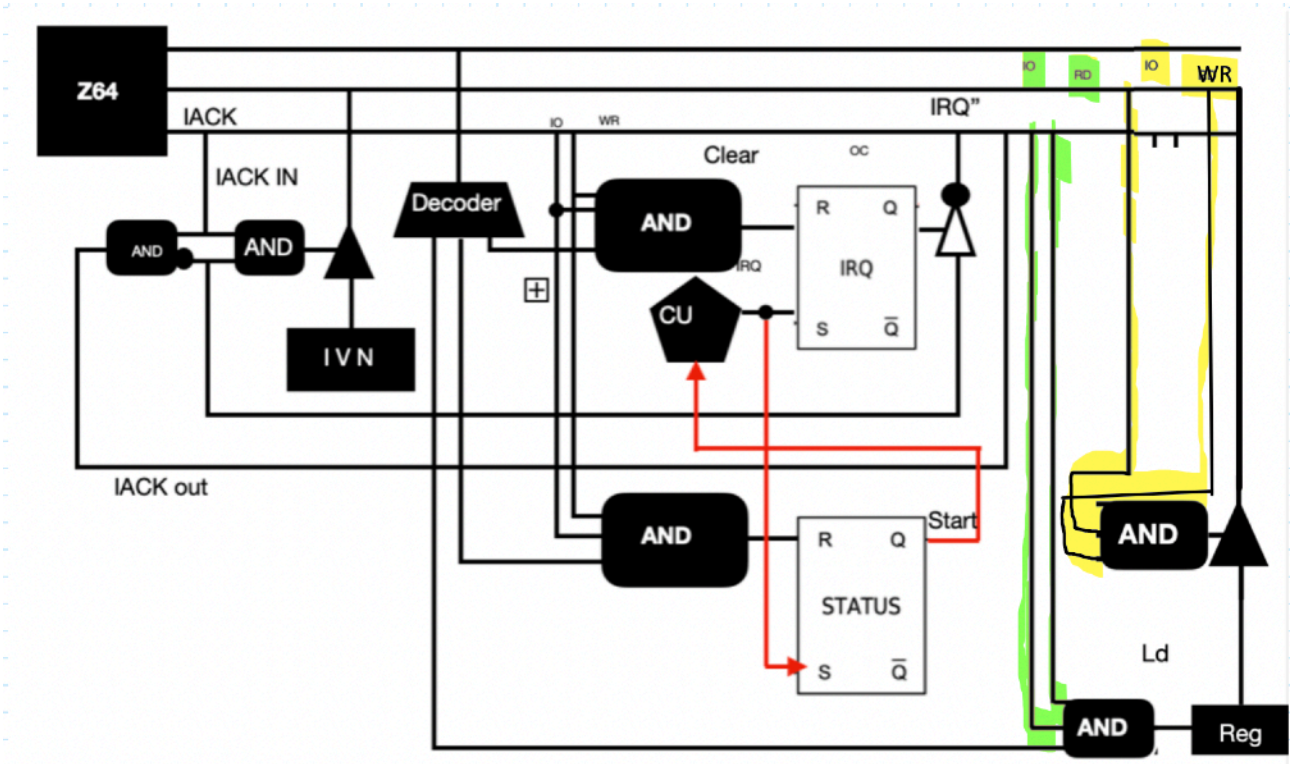
TELECAMERA - BUSY WAITING INPUT



ARRESTO - OUTPUT



RADAR - INPUT E OUTPUT + IVN



SE DENTRO A REG È PRESENTE 1 BISOGNA ATTIVARE L'IMPIANTO FRENANTE.