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.

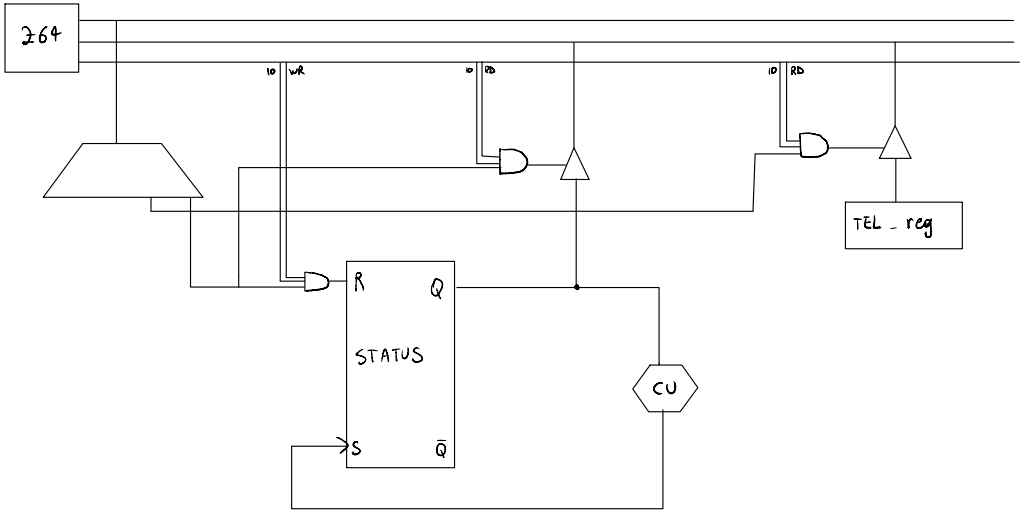
Le immagini acquisite sono delle *mappe di profondità*: un pixel nero (codificato come `0x00`) rappresenta un punto in cui non ci sono oggetti nelle vicinanze, mentre un pixel bianco (codificato come `0xFF`) rappresenta un punto in cui è presente un oggetto a distanza inferiore a 20 metri. Il processore 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.

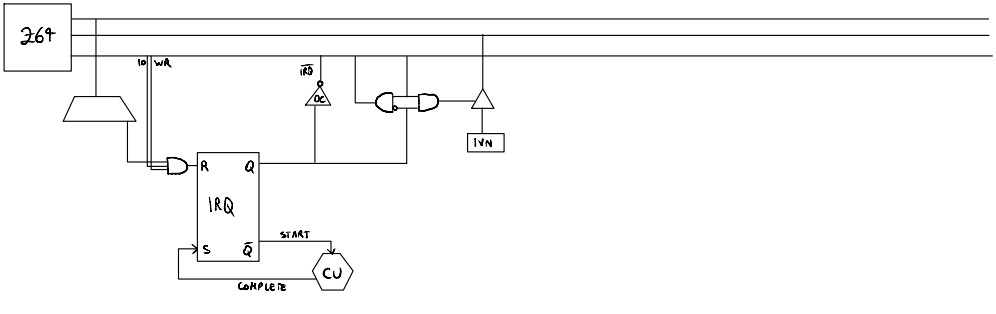
Realizzare:

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

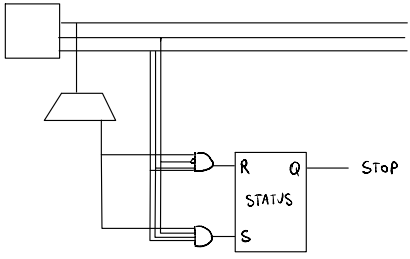
INTERFACCIA TELECAMERA



INTERFACCIA RADAR



INTERFACCIA ARRESTO



CODICE

```
.org 0x800
.data
  . equ $TEL_ status, 0x0
  . equ $TELECAMERA, 0x1
  . equ $RADAR_ irq, 0x2
  . equ $ARRESTO_ status, 0x3
  array: .Fill 480000, 1
.text
main:
   sti
    .loop:
     outb %al, $TEL_status
     .bw
      inb $TEL_status, %al
      btb $0, %al
      Jnc .bw
      movq array, %rdi
      mov q $480000, %rcx
      movW $TELECAMERA , %dx
      cld
      insb
     xorq %r8 , %r8
     xorq %r10, %r10
     . ciclo
       movb array(, %r8, 1), %rgb
       cmpb $0x FF, % rgb
       Jnz . nulla
       addq $1, %r10
       .nulla
       addq $1, %r8
       cmpq $480000, %r8
       Jnz . ciclo
       cmpq $120000, %r10
       Jc . fine
       movb $1, % di
       Jmp . call
       .fine
       movb $0, % di
       .call
       call arresto
       Jmp . loop
```

```
arresto :
   movb % di, % al
   outb % al, $ARRESTO_status
   ret
```

```
. driver 0
  push % rax
  movb $1, % al
  outb % al, $ARRESTO_status
  o ut b % al, $RADAR_ irq
  pop % rax
  iret
```