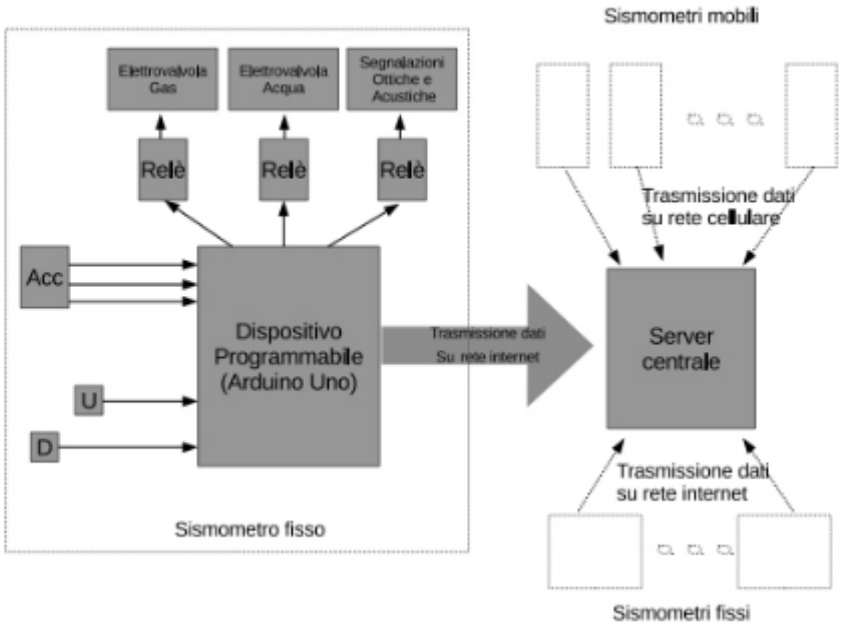
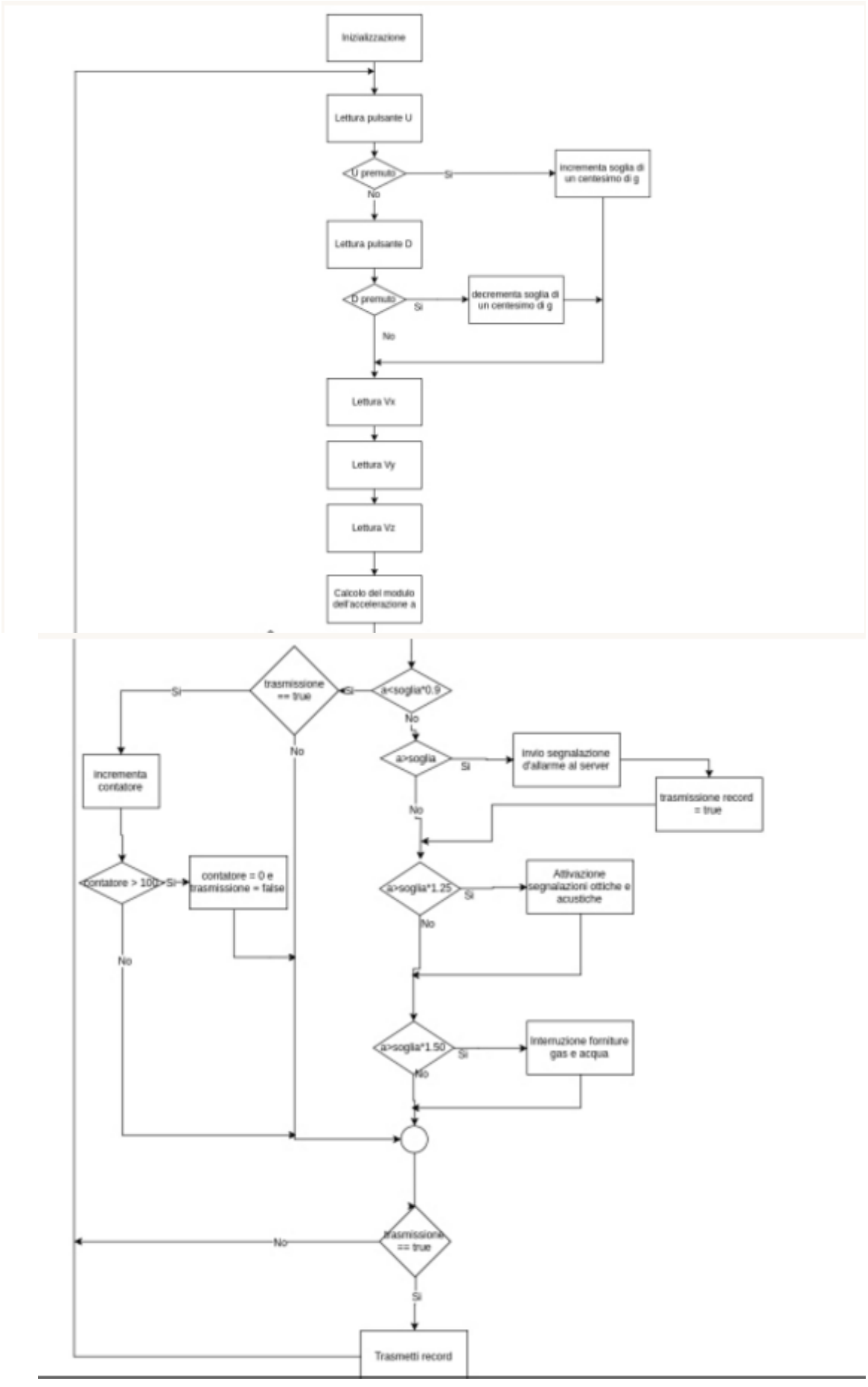
Il testo non precisa l'estensione della rete sperimentale della fase di strong motion dei terremoti. Si può quindi ragionevolmente considerare un'intera città divisa in zone, ognuna delle quali è in grado di gestire la propria fornitura di gas e acqua. La traccia neppure precisa il modo in cui le forniture di acqua e gas vengano ristabilite dopo un'eventuale interruzione. Pertanto si può assumere che questi servizi vengano riattivati manualmente. Si può ipotizzare lo stesso per quanto riguarda la disattivazione dell'avvisatore acustico e il segnalatore ottico. Per gestione in polling dei pulsanti U e D si intende che il loro stato viene acquisito a ogni iterazione del ciclo di esecuzione delle istruzioni del dispositivo programmabile. Si sceglie un microcontrollore programmabile Arduino Uno



Si consideri come accelerometro una scheda fornita di cinque pin: tre di uscita corrispondenti a Vx, Vy e Vz, un quarto per la tensione VCC e un ultimo per la tensione GND. Le tre tensioni vanno lette mediante convertitori analogico-digitale, i quali sono presenti sulle schede Arduino in corrispondenza dei pin A0-A5. Le tensioni generate, in quanto analogiche, sono segnali di natura continua. I segnali che si ottengono tramite campionamento, in quanto digitali, sono discreti e quantizzati. Inoltre, essendo l'estremo superiore dell'intervallo di frequenze di interesse 25Hz, la frequenza di campionamento, per il teorema del campionamento di Nyquist-Shannon, deve essere almeno pari a 50Hz, il doppio della frequenza del segnale misurato. Questa soglia evita che i segnali provenienti dall'accelerometro vengano distorti. Si può ragionevolmente richiedere una frequenza di campionamento di 250Hz per garantire una misura sufficientemente fedele.


Inizializzazione
Lettura pulsante U
U premuto
Si
No
incrementa soglia di un centesimo di g
Lettura pulsante D
D premuto
Si
No
decrementa soglia di un centesimo di g
Lettura Vx
Lettura Vy
Lettura Vz
Calcolo del modulo dell'accelerazione a
a<soglia*0.9
Si
No
trasmissione == true
Si
No
incrementa contatore
contatore > 100
Si
No
contatore = 0 e trasmissione = false
a>soglia
Si
No
Invio segnalazione d'allarme al server
trasmissione record = true
a>soglia*1.25
Si
No
Attivazione segnalazioni ottiche e acustiche
a>soglia*1.50
Si
No
Interruzione forniture gas e acqua
Trasmissione == true
No
Si
Trasmetti record

```
% codice Arduino per sismografo fisso
float K = 300;

// ANALOG IN
int axPin = A0;
int ayPin = A1;
int azPin = A2;

// DIGITAL
int upPin = 8;
int downPin = 7;

// ANALOG OUT
int releGas = 9;
int releAcqua = 10;
int releOtticAcust = 11;

double Vx = 1.5;
double Vy = 1.5;
double Vz = 1.5;

double ax = 0.0;
double ay = 0.0;
double az = 0.0;
double a = 0.0;
dougle aTH = 9.81/20; // soglia
double aTHincr = 9.81/100; // incremento della soglia

void setup()
{
    // imposta U e D come input
    pinMode(upPin,INPUT);
    pinMode(downPin,INPUT);

    // attiva la resistenza di pull-up
    digitalWrite(upPin, HIGH); // ora HIGH corrisponde a interruttore aperto
    digitalWrite(downPin, HIGH);

    // èRel
    pinMode(releGas, OUTPUT);
    pinMode(releAcqua, OUTPUT);
    pinMode(releOtticAcust, OUTPUT);
```

```
void loop()
{
    int up = digitalRead(upPin);
    if (up == LOW)
        aTH = aTH + aTHincr;
    ...
    int down = digitalRead(downPin);
    if (up == HIGH)
        aTH = aTH - aTHincr;
    ...

    Vx = analogRead(axPin);
    Vy = analogRead(ayPin);
    Vz = analogRead(azPin);

    // calcolo del modulo dell'accelerazione
    ax = (Vx-1.5)/K;
    ay = (Vy-1.5)/K;
    az = (Vz-1.5)/K;
    a = sqrt(ax*ax+ay*ay+az*az);

    ...

    if(a>aTH)
    {
        ...
```

Le modiche da apportare per gestire i pulsanti U e D con una tecnica di interrupt con il sistema descritto nella prima parte sono soptrattutto di tipo software. Infatti i piedini dei pulsanti U e D sono collegati rispettivamente a GND e sui pin 8 e 7, pertanto vengono già usati in modalità HIGH corrispondente a circuito aperto. Tuttavia per la tecnica di interrupt U e D vanno trasferiti su i pins 3 e 2, gli unici disponibili per la modalità interrupt su Arduino Uno. Le variazioni da apportare al software sviluppato nella prima parte sono riportate di seguito.

```
// DIGITAL
int upPin = 3;
int downPin = 2;

volatile double aTH = 9.81/20; // forza il compilatore a caricarla da RAM
double aTHincr = 9.81/100; // incremento della soglia
...

void setup()
{
    // imposta U e D come pins di interrupt
    pinMode(upPin,INPUT_PULLUP);
    pinMode(downPin,INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(upPin), incrementa, CHANGE);
    attachInterrupt(digitalPinToInterrupt(downPin), decrementa, CHANGE);
    ...
}

void loop()
{
    ...
}

void incrementa()
{
    aTH = aTH + aTHincr;
}

void decrementa()
{
    aTH = aTH - aTHincr;
}
```

Fonte: Redazione

Dai diagrammi di Bode si evince facilmente che il sistema in anello chiuso non è asintoticamente stabile poiché il margine di fase della funzione di trasferimento ad anello aperto è negativo nonostante il guadagno generalizzato sia positivo.

Dai diagrammi di Bode si evince facilmente che il sistema in anello chiuso non è asintoticamente stabile poiché il margine di fase della funzione di trasferimento ad anello aperto è negativo. Infatti in corrispondenza della frequenza di attraversamento la fase è minore di π.

Detta $F(s)$ la funzione di trasferimento a ciclo aperto sintetiziamo una rete compensatrice del tipo $kR(s)$ dove $R(s) = \frac{1+\tau s}{1+\frac{\tau}{m}s}$ è una rete anticipatrice elementare. La funzione di trasferimento a ciclo aperto compensata è quindi $F(s) = k\hat{F}(s)R(s)$.

Essendo il modulo di 40db circa, ovvero 100, scegliendo $k = \frac{1}{100}$ si ha un margine di fase prossimo a zero ma positivo per $k\hat{F}(s)$ e una frequenza di attraversamento di circa 1 kHz.

Con l'ausilio dei diagrammi di fase e modulo al variare di m e $\omega\tau$ di $R(s)$ si possono determinare dei valori per m e per $\omega\tau$ che soddisfino le specifiche del problema di controllo. Si sceglie quindi $m = 10$ e $\omega\tau = 2$ che, ponendo $\omega = 2\pi 1000\text{Hz}$ dà $\omega = \frac{1}{1000\pi}$.

Pertanto ponendo la rete $kR(s) = \frac{1}{100}\frac{1+\frac{1}{1000\pi}s}{1+\frac{1}{10000\pi}s}$ in cascata a $\hat{F}(s)$ si garantiscono un margine di fase di $\frac{\pi}{4}$ e una banda passante non inferiore a 1kHz.

Fonte: Redazione

QUESITO 4 →

Per risolvere il quesito è necessario innanzitutto ricavare la funzione di trasferimento dal disturbo $D(s)$ all'uscita $U(s)$. Si ha una funzione propria della variabile s:

$$F_d(s) = \frac{1}{1+\frac{k}{s+1}} = \frac{s+1}{s+k+1}.$$

Il sistema ad anello chiuso è asintoticamente stabile per $k > -1$.

Si calcola il modulo della funzione di trasferimento del disturbo in $s = j\omega = j$, dove j indica il numero immaginario:

$$|F(j)| = \frac{|j+1|}{|j+k+1|} = \frac{1^2+1^2}{1^2+(k+1)^2} = \frac{2}{k^2+2k+2}$$

Infine di pone:

$$|F(j)| < -14\text{dB} = 10^{-\frac{14}{20}}$$

e risolve trovando i valori di k ammissibili che soddisfano le specifiche del problema:

$$\frac{2}{k^2+2k+2} < -14\text{dB} = 10^{-\frac{14}{20}}$$

$$2 < (k^2+2k+2)10^{-\frac{14}{20}}$$

$$k^2+2k+2 > 2 \cdot 10^{\frac{14}{20}}$$

$$k^2+2k+2-2 \cdot 10^{\frac{14}{20}} > 0$$

dove $2-2 \cdot 10^{\frac{14}{20}} \simeq -8$. Si è giunti a un'equazione di secondo grado nella k le cui radici sono $-1 \pm \sqrt{1-2(1-10^{\frac{14}{20}})} \simeq -1 \pm 3$. Poiché il coefficiente del termine di secondo grado è positivo le soluzioni della disequazione sono $k < -1-\sqrt{1-2(1-10^{\frac{14}{20}})}$ e $k > -1+\sqrt{1-2(1-10^{\frac{14}{20}})} = k_{min}$ e, tenendo conto della condizione di stabilità $k > -2$ l'insieme ammissibile è $(k_{min}, +\infty)$.

In conclusione scegliendo $k > k_{min}$ si garantisce che l'ampiezza del disturbo additivo d(t), di [illegible] alla pulsazione $\omega = 1\text{rad/s}$, venga ridotta di almeno 14dB.

Fonte: Redazione

In una comunicazione I2C deve essere presente un Master ed uno, o più Slave, ossia un nodo principale (Master I2C) ed uno, o più nodi, secondario (Slave I2C)

Il compito del Master è di gestire tutta la comunicazione I2C sia inviando i messaggi a tutti gli slave connessi, sia richiedendo a ciascun slave di inviargli un messaggio.

Per eseguire il test puoi usare due arduino uno.

Occorre includi la libreria **Wire.h** necessaria per la gestione della Comunicazione I2C arduino to arduino;

```
Wire.begin();  //inizializza la comunicazione I2C con il metodo begin() della libreria Wire;
```

```
Wire.beginTransmission(8); // inizia una trasmissione verso l'indirizzo contenuto tra parentesi, device #8
```

```
Wire.write(flag); // con il metodo write() invii allo Slave 8, l'altro arduino in questo esempio, il valore contenuto tra parentesi, in questa linea il valore flag;
```

```
Wire.endTransmission();     // stop transmitting
```

COMUNICAZIONE I²C

Una comunicazione I²C si avvale di un bus seriale a due fili ed è sempre di tipo half-duplex. La struttura di un bus I²C è riportata in figura 5.

interfacce seriali SPI o I2C

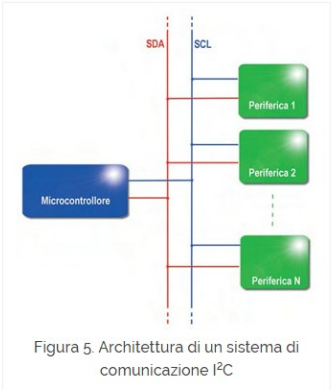


Figura 5. Architettura di un sistema di comunicazione I²C

I due segnali sono **SCL** (*SerialCLock*) e **SDA** (*SerialDAta*), sono entrambi bidirezionali e necessitano di una connessione verso il positivo di alimentazione mediante resistori di pull-up.
Le linee SCL ed SDA sono infatti di tipo open collector per consentire l'implementazione della funzione AND cablato (*wired AND*) direttamente sul bus.
A differenza di una comunicazione SPI, I²C è una **architettura multiMaster** in cui la periferica che inizia la trasmissione e genera quindi il segnale di sincronismo è designata come Master, mentre la periferica oggetto della transazione indirizzata dal Master è considerata come Slave.
Una periferica designata come Master per una certa trasmissione, potrà quindi essere anche uno Slave per una trasmissione successiva. La periferica Slave può in ogni momento rallentare o addirittura interrompere la comunicazione del Master, mantenendo a livello basso la linea SCL (operazione nota come *clock stretching* e possibile grazie al meccanismo dell'AND cablato in cui uno zero logico su una linea condivisa prevale sempre sul livello logico alto). Questa tecnica permette la corretta comunicazione dati tra una periferica Master veloce ed una periferica Slave relativamente lenta. Il meccanismo dell'AND cablato consente anche di gestire l'arbitraggio:
se due periferiche Master iniziano contemporaneamente una trasmissione dati, verrà data priorità a quella che trasmette per prima uno zero logico.
I dati possono essere trasferiti a velocità massime di 100Kbps in modalità *Standard*, 400Kbps in modalità *Fast* e 3.4Mbps in modalità *High-Speed*.
Nella modalità *Standard* gli Slave sono indirizzati con una modalità di indirizzamento a 7 bit, mentre nelle modalità *Fast* e *High-Speed* l'indirizzamento può essere a 7 o a 10 bit.
Fisicamente il carico capacitivo sul bus non deve eccedere i 400pF per non compromettere la velocità di comunicazione.
In figura 6 è riportato il diagramma temporale di una tipica comunicazione I²C.
Le periferiche sono *level-sensitive* per cui **il dato deve essere stabile per tutto l'intervallo di tempo in cui la linea SCL permane a livello alto. Solo quando la linea SCL è a livello basso è possibile commutare il bit sulla linea SDA a meno di due eccezioni:**

- **START: la periferica Master inizia una nuova comunicazione con una transizione da 1 a 0 mentre la linea SCL è a livello alto;**
- **STOP: la transizione da 0 a 1 da parte del Master mentre la linea SCL è alta segna il termine di una comunicazione.**

Nella figura 7 sono riportate le due tecniche di indirizzamento a 7bit e 10bit.
Nel caso di indirizzamento a 7 bit, a seguito della condizione di start (indicata con S in figura 7) viene trasmesso un byte i cui primi 7 bit sono l'indirizzo dello Slave mentre l'ottavo bit indica se lo Slave indirizzato dovrà ricevere o trasmettere dati.
Dopo l'invio del byte, solo una periferica Slave risponderà con un *acknowledgment* (*ACK*) a seguito del quale inizierà la trasmissione dati vera e propria.
Nel caso di indirizzamento a 10bit, a seguito della condizione di start viene inviato un byte i cui i primi 7 bit sono costituiti dalla sequenza 11110xx dove xx sono i due bit più significativi dell'indirizzo della periferica target.
L'ottavo bit determina ancora la direzione del flusso dati. L'invio di tale sequenza comporterà *l'acknowledgement* da parte di più periferiche (ACK1 in figura 7). Una volta ricevuti gli *ACK* si inviano i restanti 8 bit dell'indirizzo in modo che solo una periferica risponda con l'*ACK* (l'*ACK2* in figura 7).

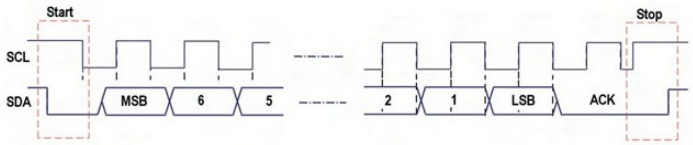


Figura 6. Temporizzazione di una comunicazione I²C

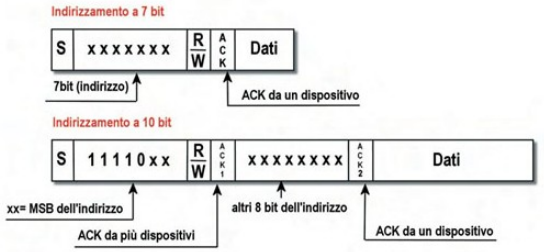


Figura 7. Modalità di indirizzamento a 7 bit e a 10 bit