

DOMOTICA CON ARDUINO & JETSON

Dispensa di Elettronica di Base

Argomenti trattati:

- Grandezze elettriche fondamentali: tensione, corrente, resistenza, potenza
- Circuiti in serie e in parallelo
- Sensori di temperatura: principi fisici e funzionamento
- Digitalizzazione della luce e segnali luminosi
- Comunicazione seriale: UART, I2C, SPI
- Progetto: Arduino + sensori + Nvidia Jetson

Corso di Domotica — Anno Accademico 2024/2025

1. Grandezze Elettriche Fondamentali

L'elettronica moderna si basa su pochi concetti fondamentali che è essenziale padroneggiare prima di affrontare qualsiasi progetto pratico. In questo capitolo esploreremo le quattro grandezze principali: tensione, corrente, resistenza e potenza.

1.1 La Tensione Elettrica (V)

La tensione elettrica, misurata in Volt (V), rappresenta la differenza di potenziale elettrico tra due punti di un circuito. Possiamo immaginarla come la "pressione" che spinge gli elettroni nel conduttore, analogamente alla pressione dell'acqua in un tubo.

Analogia idraulica

Pensa a un serbatoio d'acqua posto in alto: più è alto, maggiore è la pressione all'uscita. Allo stesso modo, spinge più elettroni attraverso il circuito. La tensione non si consuma: è sempre misurata tra DUE punti del circuito.

Nel progetto Arduino-Jetson incontreremo principalmente:

- 5V — tensione di alimentazione di Arduino Uno
- 3.3V — tensione logica di molti sensori moderni
- 12V / 19V — alimentazione della scheda Jetson

$$V = E / Q$$

La tensione (V) è l'energia (E) per unità di carica (Q). Unità: Volt (V) = Joule (J) / Coulomb (C)

1.2 La Corrente Elettrica (I)

La corrente elettrica, misurata in Ampere (A), rappresenta il flusso di cariche elettriche attraverso un conduttore nell'unità di tempo. È la quantità di elettroni che "scorrono" in ogni secondo.

$$I = Q / t$$

La corrente (I) è la carica (Q) che passa in un tempo (t). Unità: Ampere (A) = Coulomb (C) / secondo (s)

Distinguiamo due tipi fondamentali di corrente:

- **DC** — Corrente Continua (DC, Direct Current): gli elettroni scorrono sempre nella stessa direzione. Usata da Arduino, Jetson, batterie, sensori.
- **AC** — Corrente Alternata (AC, Alternating Current): la direzione del flusso si inverte periodicamente (50 Hz in Europa). È la corrente della rete elettrica domestica.

Dispositivo	Tensione	Corrente tipica
Arduino Uno	5V DC	~50 mA (senza carichi)
LED	1.8–3.3V	10–20 mA
Sensore DS18B20	3–5.5V	~1 mA
Nvidia Jetson Nano	5V DC	~2 A (tipico)

Telecamera USB	5V DC	400–800 mA
----------------	-------	------------

1.3 La Resistenza Elettrica (R)

La resistenza elettrica, misurata in Ohm (Ω), è la proprietà di un materiale di opporsi al passaggio della corrente. Un resistore è un componente che dissipa energia trasformandola in calore.

La legge di Ohm

La relazione fondamentale dell'elettronica è: $V = R \times I$ (oppure $I = V/R$, oppure $R = V/I$). Questa legge dice che la tensione è proporzionale alla corrente che lo attraversa. Conoscendo due delle tre grandezze, la terza si calcola sempre.

$$V = R \times I$$

Legge di Ohm: tensione (V), resistenza (R) in Ohm, corrente (I) in Ampere

I resistori fisici sono identificati da una serie di bande colorate. Ogni colore corrisponde a una cifra numerica:

Colore	Valore	Colore	Valore	Multiplic.
Nero	0	Verde	5	$\times 100 \text{ k}\Omega$
Marrone	1	Blu	6	$\times 1 \text{ M}\Omega$
Rosso	2	Viola	7	—
Arancio	3	Grigio	8	—
Giallo	4	Bianco	9	—

Esempio pratico: per proteggere un LED collegato a 5V (Arduino), con caduta di tensione del LED di 2V e corrente desiderata 10 mA, il resistore necessario è: $R = (5V - 2V) / 0.01A = 300 \Omega$
 → si usa il valore standard più vicino: 330 Ω .

1.4 La Potenza Elettrica (P)

La potenza, misurata in Watt (W), rappresenta l'energia consumata (o prodotta) nell'unità di tempo. Sapere calcolare la potenza è fondamentale per scegliere componenti che non si surriscaldino o si danneggino.

$$P = V \times I = R \times I^2$$

Potenza in Watt. Esempio: un LED a 3V, 20mA consuma $P = 3 \times 0.02 = 0.06 \text{ W}$

Attenzione alla dissipazione

Ogni resistore ha una potenza massima ammissibile (tipicamente 0.25W o 0.5W per i componenti comuni). Se il valore, il componente si surriscalda e si danneggia. Controllate sempre che $P = R \times I^2$ sia inferiore alla potenza massima.

2. Circuiti in Serie e in Parallelo

Quando colleghiamo più componenti insieme, possiamo farlo in due modi fondamentali: in serie (uno dopo l'altro) o in parallelo (fianco a fianco). Le regole che governano questi collegamenti sono diverse e producono effetti molto diversi.

2.1 Collegamento in Serie

In un circuito serie, i componenti sono collegati uno dopo l'altro, formando un unico percorso per la corrente. Come perline su un filo: se la catena si spezza in un punto, tutto si interrompe.

Regole per il collegamento in serie

- 1) La corrente è uguale in tutti i componenti: $I_{\text{totale}} = I_1 = I_2 = I_3$
- 2) La tensione totale è la somma delle tensioni: $V_{\text{totale}} = V_1 + V_2 + V_3$
- 3) La resistenza totale è la somma delle resistenze: $R_{\text{tot}} = R_1 + R_2 + R_3$

Applicazione pratica: quando collegate più LED in serie con Arduino a 5V, la tensione si divide tra tutti i LED. Con 2 LED da 2V ciascuno: $V_{\text{resistore}} = 5 - 2 - 2 = 1V$. La corrente è la stessa per tutti.

$$R_{\text{serie}} = R_1 + R_2 + R_3$$

La resistenza totale in serie è sempre **MAGGIORE** della resistenza dei singoli componenti.

2.2 Collegamento in Parallelo

In un circuito parallelo, tutti i componenti condividono gli stessi due nodi di connessione. Come le lampadine di casa: ognuna ha la stessa tensione di rete, e se una si fulmine le altre continuano a funzionare.

Regole per il collegamento in parallelo

- 1) La tensione è uguale su tutti i componenti: $V_{\text{totale}} = V_1 = V_2 = V_3$
- 2) La corrente totale è la somma delle correnti: $I_{\text{totale}} = I_1 + I_2 + I_3$
- 3) La resistenza totale si calcola con la formula inversa: $1/R_{\text{tot}} = 1/R_1 + 1/R_2 + 1/R_3$

$$1/R_{\text{par}} = 1/R_1 + 1/R_2$$

La resistenza totale in parallelo è sempre **MINORE** della resistenza dei singoli componenti.

Caso speciale con due resistori uguali in parallelo: $R_{\text{tot}} = R / 2$. Con tre resistori uguali: $R_{\text{tot}} = R / 3$.

Proprietà	Serie	Parallelo
Corrente	Uguale ovunque	Si divide tra i rami
Tensione	Si divide tra i componenti	Uguale su tutti
Resistenza totale	Somma (cresce)	Formula inversa (diminuisce)

Se un componente si guasta	Tutto il circuito si interrompe	Gli altri continuano a funzionare
----------------------------	---------------------------------	-----------------------------------

2.3 Il Partitore di Tensione

Il partitore di tensione è uno schema circuitale fondamentale, basato su due resistori in serie, che permette di ottenere una tensione inferiore a quella di alimentazione. È utilizzatissimo per adattare segnali tra dispositivi che operano a tensioni diverse (es. da 5V di Arduino a 3.3V di Jetson).

$$V_{out} = V_{in} \times R2 / (R1+R2)$$

Formula del partitore: R2 è il resistore collegato a massa

Esempio: adattare un segnale da 5V (Arduino TX) a 3.3V (Jetson RX). Scegliamo $R1=2k\Omega$ e $R2=3.3k\Omega$: $V_{out} = 5 \times 3.3 / (2+3.3) = 5 \times 0.623 = 3.11V \approx 3.3V$. In pratica si usano valori standard $R1=2.2k\Omega$ e $R2=3.3k\Omega$.

3. Misurazione della Temperatura: Principi Fisici

La temperatura è una delle grandezze fisiche più misurate nei sistemi domotici. Esistono diversi principi fisici su cui si basano i sensori di temperatura; in questo capitolo esploreremo i più importanti, con particolare attenzione ai sensori usati con Arduino.

3.1 Effetto della Temperatura sulla Resistenza

I materiali conducono l'elettricità in modo diverso a seconda della loro temperatura. Questo fenomeno è alla base di due importanti categorie di sensori.

RTD — Resistance Temperature Detector

Nei metalli (come il platino), la resistenza AUMENTA all'aumentare della temperatura in modo quasi lineare. Il sensore PT100 è un esempio classico: ha una resistenza di 100 Ω a 0°C, e aumenta di circa 0.385 Ω per ogni grado Celsius.

$$R(T) = R_0 \times (1 + \alpha \times T)$$

R_0 = resistenza a 0°C, α = coefficiente di temperatura (α_{plati})

NTC — Negative Temperature Coefficient (Termistore)

Nei semiconduttori (ceramiche speciali), la resistenza DIMINUISCE esponenzialmente all'aumentare della temperatura. I termistori NTC sono economici, precisi in un range ristretto, e molto usati nei sistemi domotici.

$$R(T) = R_0 \times e^{(B(1/T - 1/T_0))}$$

B = costante del materiale (tipicamente 3000-5000 K)

Conversione Kelvin ↔ Celsius

La scala Kelvin è quella assoluta: 0 K = -273.15°C. Per convertire: $T[K] = T[°C] + 273.15$. Lo zero assoluto (0 K) è la temperatura più bassa possibile, corrispondente all'assenza totale di agitazione molecolare.

3.2 Il Sensore Digitale DS18B20

Il DS18B20 è il sensore di temperatura più utilizzato con Arduino. Contiene internamente un NTC, un ADC (convertitore analogico-digitale) e un'interfaccia digitale 1-Wire. Fornisce direttamente la temperatura in gradi Celsius come numero digitale.

Caratteristica	Valore
Range di temperatura	-55°C a +125°C
Precisione	±0.5°C (-10°C a +85°C)
Risoluzione configurabile	9, 10, 11, 12 bit
Risoluzione a 12 bit	0.0625°C (1/16 di grado)
Alimentazione	3V - 5.5V DC
Protocollo	1-Wire (un solo filo dati)

Resistenza pull-up necessaria	4.7 kΩ tra DQ e VCC
-------------------------------	---------------------

Il protocollo 1-Wire permette di collegare più sensori DS18B20 sullo stesso filo (ciascuno ha un ID univoco a 64 bit). La libreria Arduino "OneWire" e "DallasTemperature" gestiscono automaticamente la comunicazione.

3.3 Effetto Seebeck e Termocoppie

Quando due metalli diversi sono giunti assieme e i due giunti si trovano a temperature diverse, si genera una piccola tensione elettrica proporzionale alla differenza di temperatura. Questo è l'effetto Seebeck, alla base delle termocoppie.

$$V = S \times \Delta T$$

S = coefficiente di Seebeck ($\mu\text{V}/^\circ\text{C}$, dipende dalla coppia di metalli), ΔT = dif

Le termocoppie di tipo K (Nichel-Cromo / Nichel-Alluminio) sono le più comuni: generano circa $41 \mu\text{V}/^\circ\text{C}$ e coprono il range -200°C a $+1350^\circ\text{C}$. Non adatte per misure di precisione a bassa temperatura nei sistemi domotici, ma eccellenti per alte temperature.

4. Digitalizzazione dei Segnali Luminosi

Le telecamere trasformano la luce (un fenomeno fisico continuo) in dati digitali (numeri). Comprendere questo processo è fondamentale per lavorare con la telecamera controllata dalla Jetson nel nostro progetto.

4.1 La Natura della Luce Visibile

La luce visibile è radiazione elettromagnetica con lunghezze d'onda comprese tra circa 380 nm (violetto) e 780 nm (rosso). L'occhio umano percepisce colori diversi perché ha tre tipi di fotorecettori (coni), sensibili rispettivamente al rosso (R), verde (G) e blu (B).

Colore	Lunghezza d'onda	Frequenza
Violetto	380 – 450 nm	667 – 789 THz
Blu	450 – 495 nm	606 – 667 THz
Verde	495 – 570 nm	526 – 606 THz
Giallo	570 – 590 nm	508 – 526 THz
Arancio	590 – 625 nm	480 – 508 THz
Rosso	625 – 780 nm	384 – 480 THz

4.2 Il Sensore di Immagine: CCD e CMOS

Una telecamera digitale contiene un sensore a stato solido (CCD o CMOS) formato da milioni di fotositi (pixel), ciascuno dei quali converte i fotoni incidenti in una carica elettrica proporzionale all'intensità luminosa.

Come funziona un pixel

Ogni fotosito è essenzialmente un piccolo condensatore. I fotoni (particelle di luce) colpiscono il materiale semiconduttore (silicio) e liberano elettroni per effetto fotoelettrico. Più luce arriva, più elettroni vengono liberati, e maggiore sarà la carica accumulata. Questa carica viene poi convertita in una tensione e infine in un numero digitale.

Il sensore rileva solo intensità luminosa (livelli di grigio). Per ottenere informazioni sul colore, si usa il filtro di Bayer: una griglia di filtri colorati (R, G, B) posta sopra i fotositi, con il doppio dei filtri verdi rispetto ai rossi e ai blu (perché l'occhio umano è più sensibile al verde).

4.3 Il Modello di Colore RGB

Il modello RGB (Red, Green, Blue) è il sistema di rappresentazione del colore più diffuso nei dispositivi digitali. Ogni pixel è rappresentato da tre valori, uno per ogni canale di colore.

Pixel = (R, G, B)

Con 8 bit per canale: valori da 0 a 255. Totale colori: $256^3 = 16.777.216$

Colore	R	G	B
Bianco	255	255	255
Nero	0	0	0
Rosso puro	255	0	0
Verde puro	0	255	0
Blu puro	0	0	255
Giallo	255	255	0
Grigio 50%	128	128	128

4.4 Risoluzione, Profondità di Bit e Dimensione

La qualità di un'immagine digitale dipende da tre parametri fondamentali:

- **Risoluzione:** Risoluzione spaziale: numero di pixel (es. 1920×1080 = Full HD = 2.073.600 pixel)
- **Profondità:** Profondità di colore (bit depth): quanti bit per pixel. A 8 bit/canale (24 bit totali) abbiamo 16,7M di colori. A 10 bit/canale i moderni sensori catturano sfumature impercettibili all'occhio umano.
- **Frame rate:** Frame rate (FPS): quanti fotogrammi al secondo. 30 FPS è lo standard video; 60 FPS per riprese fluide.

$$\text{Dati/sec} = W \times H \times \text{BPP} \times \text{FPS}$$

W=larghezza, H=altezza in pixel, BPP=bit per pixel, FPS=frequenza di fotogrammi al secondo.
Esempio 1080p RGB 30fps: $1920 \times 1080 \times 24 \times 30 = 1.474.560.000$ bit/sec

Questi dati grezzi vengono compressi da codec come H.264, H.265 o MJPEG per ridurre lo spazio di archiviazione e la banda necessaria alla trasmissione. La Nvidia Jetson dispone di acceleratori hardware dedicati per la codifica e decodifica video.

5. Protocolli di Comunicazione Seriale

La comunicazione tra dispositivi elettronici avviene attraverso protocolli standardizzati. In questo capitolo analizziamo i tre protocolli seriali più usati nel mondo Arduino/Jetson: UART, I2C e SPI.

5.1 Concetti Fondamentali

Prima di entrare nei dettagli, è importante comprendere la distinzione tra comunicazione seriale e parallela. Nella comunicazione parallela, più bit vengono trasmessi simultaneamente su più fili distinti. Nella comunicazione seriale, i bit vengono inviati uno alla volta su un unico filo di dati, riducendo il numero di connessioni necessarie.

Protocollo	Fili dati	Velocità tipica	Dispositivi
UART	2 (TX/RX)	9600 – 921600 baud	2 (punto-punto)
I2C	2 (SDA/SCL)	100 kbps – 400 kbps	Molti (bus)
SPI	4 (MOSI/MISO/SCK/CS)	1 – 50 Mbps	Molti (bus)

5.2 UART — Universal Asynchronous Receiver/Transmitter

UART è il protocollo di comunicazione seriale più semplice e più antico. È "asincrono" perché i due dispositivi non condividono un segnale di clock: entrambi devono essere configurati con la stessa velocità (baud rate) in anticipo.

Come funziona UART

Un frame UART è composto da: 1 bit di START (sempre 0), 5-9 bit di dati, 1 bit opzionale di parità (per rilevare errori), 1-2 bit di STOP (sempre 1). Il ricevitore rileva il fronte di discesa del bit di START e campiona i bit dati a intervalli regolari determinati dal baud rate configurato.

Nel nostro progetto, la comunicazione tra Arduino e Jetson avviene via UART attraverso il cavo USB. Dal punto di vista software, appare come una porta seriale virtuale (COM su Windows, /dev/ttyACM0 o /dev/ttyUSB0 su Linux).

Parametro	Configurazione tipica
Baud rate	9600, 115200 (comuni)
Bit di dati	8 (quasi sempre)
Parità	None (N)
Bit di stop	1
Notazione compatta	"8N1" (la più comune)

Codice Arduino per inviare dati di temperatura via UART:

```
Serial.begin(9600); float temp = sensors.getTempCByIndex(0);
Serial.println(temp); // Invia es. "23.50\n"
```

Codice Python su Jetson per ricevere i dati:

```
import serial ser = serial.Serial('/dev/ttyACM0', 9600) while True:
line = ser.readline().decode().strip() temp = float(line)
print(f'Temperatura: {temp}°C')
```

5.3 I2C — Inter-Integrated Circuit

I2C (pronuncia: "I-squared-C") è un protocollo bus sincrono a due fili sviluppato da Philips (ora NXP) negli anni '80. Permette di collegare molti dispositivi usando solo due linee condivise: SDA (Serial Data) e SCL (Serial Clock).

Architettura Master/Slave

I2C usa un modello master-slave: il master (es. Arduino) gestisce il clock e avvia tutte le comunicazioni. Ogni dispositivo slave ha un indirizzo univoco a 7 bit (da 0x08 a 0x77, cioè fino a 112 dispositivi sullo stesso bus). Il master invia l'indirizzo del dispositivo con cui vuole comunicare, seguito dai dati.

Le linee SDA e SCL sono di tipo open-drain: richiedono resistori di pull-up (tipicamente 4.7 kΩ per 100 kbps, oppure 2.2 kΩ per 400 kbps) collegati alla tensione di alimentazione.

Modalità	Velocità	Note
Standard Mode	100 kbps	Compatibilità massima
Fast Mode	400 kbps	Più comune oggi
Fast Mode Plus	1 Mbps	Per periferiche moderne
High Speed Mode	3.4 Mbps	Raro, uso industriale

Struttura di un frame I2C per scrittura:

- START condition: SDA va a 0 mentre SCL è a 1
- 7 bit di indirizzo del dispositivo slave (es. 0x68 per un MPU-6050)
- 1 bit R/W: 0 per scrittura, 1 per lettura
- ACK dal dispositivo slave (1 bit a 0 = ricevuto)
- 8 bit di dati + ACK, ripetuti quante volte necessario
- STOP condition: SDA va a 1 mentre SCL è a 1

Su Arduino, la libreria Wire gestisce l'intero protocollo I2C. Sensori I2C comuni nel mondo domotico: BME280 (temperatura, umidità, pressione, indirizzo 0x76), MPU-6050 (accelerometro e giroscopio, 0x68), RTC DS3231 (orologio in tempo reale, 0x57).

5.4 SPI — Serial Peripheral Interface

SPI è un protocollo sincrono full-duplex sviluppato da Motorola negli anni '80. Usa quattro linee: MOSI (Master Out Slave In), MISO (Master In Slave Out), SCK (clock) e CS/SS (Chip Select, una per ogni dispositivo slave).

Full-Duplex vs Half-Duplex

SPI è full-duplex: i dati possono viaggiare contemporaneamente in entrambe le direzioni (trasmissione e ricezione simultanee). I2C e UART sono half-duplex (o simplex): i dati viaggiano in una direzione alla volta. SPI è più veloce ma richiede più fili.

SPI è usato quando serve alta velocità: lettori di schede SD, display TFT, ADC veloci, moduli RF. Il vantaggio principale rispetto a I2C è la velocità (fino a 50 Mbps o oltre); lo svantaggio è che ogni dispositivo aggiuntivo richiede una linea CS dedicata.

5.5 1-Wire: Il Protocollo del DS18B20

Il protocollo 1-Wire, sviluppato da Dallas Semiconductor (ora Maxim Integrated), usa un singolo filo per dati e alimentazione. È più lento degli altri protocolli (~16 kbps) ma è perfetto per reti di sensori dove la semplicità di cablaggio è prioritaria.

Ogni dispositivo 1-Wire ha un indirizzo seriale univoco a 64 bit programmato in fabbrica. Questo permette di identificare esattamente ogni sensore anche quando decine di DS18B20 sono collegati sullo stesso filo (configurazione multidrop).

6. Conversione Analogica/Digitale e Digitale/Analogica

Il mondo fisico è analogico: temperature, pressioni, luminosità, suoni variano con continuità nel tempo e nei valori. I microcontrollori e i computer, al contrario, lavorano esclusivamente con numeri interi in formato binario. La conversione tra questi due "mondi" è il cuore di ogni sistema di acquisizione e controllo, ed è un processo che avviene continuamente nel nostro progetto domotico.

6.1 Segnali Analogici e Digitali a Confronto

Un segnale analogico è una grandezza fisica che varia con continuità: può assumere infiniti valori all'interno di un intervallo. Un segnale digitale è invece una sequenza di valori discreti (tipicamente 0 e 1) che rappresentano la versione "quantizzata" del segnale analogico.

Caratteristica	Segnale Analogico	Segnale Digitale
Valori possibili	Infiniti (continui)	Finiti (discreti)
Esempi	Temperatura, tensione, audio	Bit, byte, numeri interi
Rumore	Si accumula e degrada	Immune fino a una soglia
Elaborazione	Circuiti analogici (op-amp)	Microcontrollori, CPU
Precisione	Teoricamente infinita	Limitata dal numero di bit

💡 Perché convertire?

Un sensore di temperatura NTC produce una tensione analogica variabile (es. 0–5V) proporzionale alla temperatura. Arduino non può "leggere" direttamente questa tensione: può solo contare impulsi digitali. Il convertitore ADC interno ad Arduino trasforma quella tensione in un numero (0–1023 per un ADC a 10 bit) che il programma può elaborare.

6.2 Conversione Analogica/Digitale (ADC)

Un convertitore Analogico/Digitale (ADC, Analog-to-Digital Converter) trasforma una tensione analogica continua in un numero intero digitale. Il processo avviene in tre fasi fondamentali: campionamento, quantizzazione e codifica.

6.2.1 Campionamento (Sampling)

Il campionamento consiste nel misurare il valore del segnale analogico a intervalli di tempo regolari, chiamati periodo di campionamento (T_s). La frequenza di campionamento ($f_s = 1/T_s$) determina quante "fotografie" del segnale vengono scattate ogni secondo.

💡 Teorema di Nyquist-Shannon

Per ricostruire correttamente un segnale analogico dalla sua versione digitale, la frequenza di campionamento deve essere ALMENO il doppio della frequenza massima presente nel segnale: $f_s \geq 2 \times f_{\text{max}}$. Campionare a frequenza insufficiente produce il

fenomeno dell'aliasing: frequenze inesistenti nel segnale originale appaiono nel segnale digitalizzato, distorcendolo irrimediabilmente.

$$f_s \geq 2 \times f_{\max}$$

Teorema di Nyquist: la frequenza di campionamento deve essere almeno doppia rispetto alla banda del segnale

Esempi pratici di frequenze di campionamento: audio CD 44.100 Hz (f_{\max} audio = 20 kHz), audio telefono 8.000 Hz (f_{\max} voce = 4 kHz), sensori di temperatura con Arduino 1–10 Hz (la temperatura varia lentamente).

6.2.2 Quantizzazione

La quantizzazione è il processo che assegna a ogni campione un valore numerico intero scelto tra un insieme finito di livelli. L'intervallo di tensione misurabile (detto full-scale range, FSR) viene diviso in 2^N livelli uguali, dove N è il numero di bit dell'ADC.

$$Q = \text{FSR} / 2^N$$

Q = passo di quantizzazione (risoluzione in tensione),
FSR = fondo scala, N = numero di bit

Bit (N)	Livelli (2^N)	Passo Q (0–5V)	Esempio dispositivo
8 bit	256	19.6 mV	Sensori semplici, audio telefonia
10 bit	1024	4.9 mV	Arduino Uno / Nano
12 bit	4096	1.2 mV	Arduino Due, STM32, sensori precisi
16 bit	65536	76 μ V	Audio Hi-Fi, strumentazione
24 bit	16.777.216	0.3 μ V	Misure scientifiche, audio studio

L'ADC di Arduino Uno è a 10 bit con riferimento a 5V: il passo di quantizzazione è $5V / 1024 \approx 4.88$ mV. La funzione `analogRead()` restituisce un intero da 0 a 1023. Per convertire in tensione: $V = (\text{analogRead}(\text{pin}) / 1023.0) \times 5.0$

6.2.3 Errore di Quantizzazione

La quantizzazione introduce inevitabilmente un errore: il valore reale viene arrotondato al livello discreto più vicino. L'errore massimo è pari a $\pm Q/2$ (metà del passo di quantizzazione). Questo errore è chiamato rumore di quantizzazione e diminuisce aumentando il numero di bit.

$$\text{SNR} \approx 6.02 \times N + 1.76 \text{ dB}$$

Il rapporto segnale/rumore (SNR) di un ADC ideale aumenta con il numero di bit aggiunto. Un ADC a 10 bit ha $\text{SNR} \approx 62$ dB.

6.2.4 Architetture ADC

Esistono diverse tecnologie per realizzare un ADC, con diversi compromessi tra velocità e precisione:

- **Flash:** ADC Flash (comparatori paralleli): velocissimo (GHz), costoso, bassa risoluzione (8 bit). Usato in oscilloscopi digitali.
- **SAR:** ADC SAR (Successive Approximation Register): buon compromesso velocità/risoluzione (fino a 16 bit, fino a pochi MHz). È il tipo usato in Arduino e nella maggior parte dei microcontrollori.
- **Sigma-Delta:** ADC Sigma-Delta ($\Sigma\Delta$): altissima risoluzione (16–24 bit), bassa velocità (audio, misure lente). Usato in bilance di precisione e microfoni digitali.
- **Pipeline:** ADC Pipeline: alta risoluzione e alta velocità (12–16 bit, centinaia di MHz). Usato in ricevitori radio e oscilloscopi di fascia alta.

6.3 L'ADC di Arduino in Dettaglio

Arduino Uno integra un ADC SAR a 10 bit con multiplexer a 6 canali (pin A0–A5). Il tempo di conversione tipico è circa 100 μ s (corrispondente a una frequenza di campionamento massima di circa 10.000 campioni/secondo).

Parametro ADC Arduino Uno	Valore
Risoluzione	10 bit (0 – 1023)
Tensione riferimento default	5V (= VCC)
Tensione riferimento interno	1.1V (maggiore precisione per segnali piccoli)
Canali analogici	6 (A0 – A5)
Frequenza campionamento max	~10.000 sps (samples per second)
Impedenza ingresso raccomandata	< 10 k Ω

Esempio: lettura di un sensore NTC collegato ad A0 con partitore di tensione. Il resistore fisso del partitore è $R_{ref} = 10 \text{ k}\Omega$, il sensore NTC ha $R_{25} = 10 \text{ k}\Omega$ a 25°C:

```
int raw = analogRead(A0);           // 0 - 1023 float V_out = raw * (5.0 /
1023.0); // conversione in Volt float R_ntc = 10000.0 * V_out / (5.0 -
V_out); // dalla formula del partitore // Poi si applica la formula di
Steinhart-Hart per ottenere T in °C
```

6.4 Conversione Digitale/Analogica (DAC)

Un convertitore Digitale/Analogico (DAC, Digital-to-Analog Converter) esegue il processo inverso: trasforma un numero intero digitale in una tensione analogica continua. È fondamentale per generare segnali audio, controllare velocità di motori, regolare luminosità di LED con precisione, e produrre qualsiasi forma d'onda arbitraria.

$$V_{out} = (D / 2^N) \times V_{ref}$$

D = valore digitale in ingresso, N = bit di risoluzione, V_{ref}

Esempio: con un DAC a 8 bit ($N=8$) e $V_{ref}=5V$, il valore digitale $D=128$ produce $V_{out} = (128/256) \times 5V = 2.5V$. Il valore $D=255$ produce $V_{out} = (255/256) \times 5V \approx 4.98V$ (mai esattamente 5V con un DAC ideale a N bit).

6.4.1 Architetture DAC

Come per gli ADC, esistono diverse architetture di DAC:

- **R-2R:** DAC a resistori pesati (R-2R ladder): usa una rete di soli due valori di resistenza (R e 2R) per sommare correnti pesate in base ai bit. Economico e relativamente preciso.
- **Sigma-Delta:** DAC Sigma-Delta: produce un treno di impulsi ad alta frequenza con duty cycle proporzionale al valore desiderato. Il filtro passa-basso estrae il valore medio analogico. Usato negli audio DAC Hi-Fi.
- **Correnti commutate:** DAC a correnti commutate: alta velocità e precisione, usato in applicazioni professionali e RF.

6.4.2 PWM come DAC Approssimato

Arduino Uno non dispone di un DAC vero. Tuttavia, può simularne il comportamento tramite PWM (Pulse Width Modulation): un segnale digitale che alterna rapidamente tra 0V e 5V con un rapporto ciclico (duty cycle) variabile. Se la frequenza PWM è sufficientemente alta e si aggiunge un filtro passa-basso, la tensione media risultante è proporzionale al duty cycle.

$$V_{\text{media}} = (\text{duty_cycle} / 255) \times 5V$$

Arduino PWM: duty_cycle da 0 (0%) a 255
analogWrite(pin, valore)

PWM su Arduino

I pin PWM di Arduino Uno sono: 3, 5, 6, 9, 10, 11 (contrassegnati con ~ sulle schede). La frequenza PWM è circa 980 Hz sui pin 5 e 6. Per ottenere una vera tensione analogica è necessario un filtro RC passa-basso con una frequenza di taglio (es. R=10kΩ, C=10μF → f_taglio ≈ 1.6 Hz).

Valore analogWrite()	Duty Cycle	Tensione media (filtrata)
0	0%	0.0 V
64	25%	1.25 V
128	50%	2.5 V
192	75%	3.75 V
255	100%	5.0 V

6.5 Il Percorso Completo del Segnale nel Progetto

Comprendere il percorso completo che un segnale fisico compie nel nostro sistema aiuta a progettare e diagnosticare correttamente ogni componente:

#	Stadio	Cosa succede	Tipo di segnale
1	Fenomeno fisico	La temperatura reale cambia nell'ambiente	Analogico (continuo)
2	Sensore NTC / DS18B20	La variazione di resistenza produce una variazione di tensione	Analogico (elettrico)

3	ADC interno (DS18B20)	La tensione viene campionata e quantizzata a 12 bit	Digitale (binario)
4	Protocollo 1-Wire	I bit vengono trasmessi serialmente ad Arduino	Digitale (seriale)
5	Arduino (CPU)	Il valore viene elaborato e formattato come stringa ASCII	Digitale (numero)
6	UART via USB	La stringa viene trasmessa alla Jetson bit per bit	Digitale (seriale)
7	Python su Jetson	La stringa viene decodificata e convertita in float	Digitale (elaborato)
8	Decisione logica	Se $T > \text{soglia}$ → scatta foto o avvia video	Digitale (controllo)

Ogni conversione introduce una piccola perdita di informazione (errore di quantizzazione) o un ritardo (latenza di campionamento). Progettare bene un sistema significa scegliere ADC con risoluzione adeguata, frequenza di campionamento sufficiente per il fenomeno fisico monitorato, e protocolli di comunicazione con latenza accettabile per l'applicazione.

7. Il Progetto: Arduino + Sensori + Jetson

Nei capitoli precedenti abbiamo acquisito tutti i concetti teorici necessari. Ora vediamo come questi si integrano nel progetto pratico del corso: un sistema domotico basato su Arduino (acquisizione dati temperatura) e Nvidia Jetson (elaborazione video e intelligenza artificiale).

7.1 Architettura del Sistema

Componente	Funzione	Protocollo	Tensione
DS18B20 → Arduino	Lettura temperatura	1-Wire	5V
Arduino → Jetson	Trasmissione dati	UART (USB)	5V/USB
Jetson CPU	Logica e decisioni	Software Python	—
Jetson → Telecamera	Scatta foto / video	USB / MIPI CSI	5V

Il flusso di dati è il seguente: il sensore DS18B20 misura la temperatura e la trasmette ad Arduino via protocollo 1-Wire. Arduino elabora la lettura e invia il valore via UART (porta USB-seriale) alla Jetson. Un programma Python in esecuzione sulla Jetson legge i dati dalla porta seriale e, in base alla logica implementata (es. temperatura > soglia), comanda la telecamera per scattare una foto o avviare la registrazione di un video.

7.2 Schema di Collegamento Hardware

Componenti necessari:

- Arduino Uno (o Nano)
- Sensore di temperatura DS18B20 (versione impermeabile consigliata per applicazioni esterne)
- Resistore 4.7 k Ω (pull-up per il bus 1-Wire)
- Cavo USB A-B (Arduino Uno) o micro-USB (Arduino Nano)
- Nvidia Jetson Nano / Orin Nano
- Telecamera USB compatibile o modulo camera MIPI

Collegamento del DS18B20 ad Arduino: il pin VDD del sensore va a 5V di Arduino, GND a GND, e il pin DQ (dati) al pin digitale 2 di Arduino. Il resistore da 4.7 k Ω si collega tra il pin DQ e 5V (pull-up).

Attenzione alle tensioni

La Jetson Nano opera a 3.3V sui suoi pin GPIO. Se volete collegare segnali digitali direttamente tra Arduino e Jetson, è necessario usare un level shifter o un level shifter dedicato per non danneggiare la Jetson. La comunicazione via USB-seriale è già gestita a livello di sistema.

7.3 Note Finali e Approfondimenti

Questo documento ha introdotto i concetti fondamentali dell'elettronica necessari per sviluppare il progetto. Gli studenti che volessero approfondire possono esplorare i seguenti argomenti: il teorema di Thevenin e Norton per l'analisi di circuiti complessi, i condensatori e la loro risposta

in frequenza, i transistor BJT e MOSFET come switch digitali e amplificatori, i convertitori A/D e D/A, i filtri passivi e attivi per il condizionamento del segnale.

Per la parte software, si consiglia di esplorare le librerie Python OpenCV per la gestione avanzata della telecamera, Jetson.GPIO per il controllo dei pin GPIO della Jetson, e i framework di machine learning come TensorFlow Lite o PyTorch ottimizzati per la Jetson tramite TensorRT.