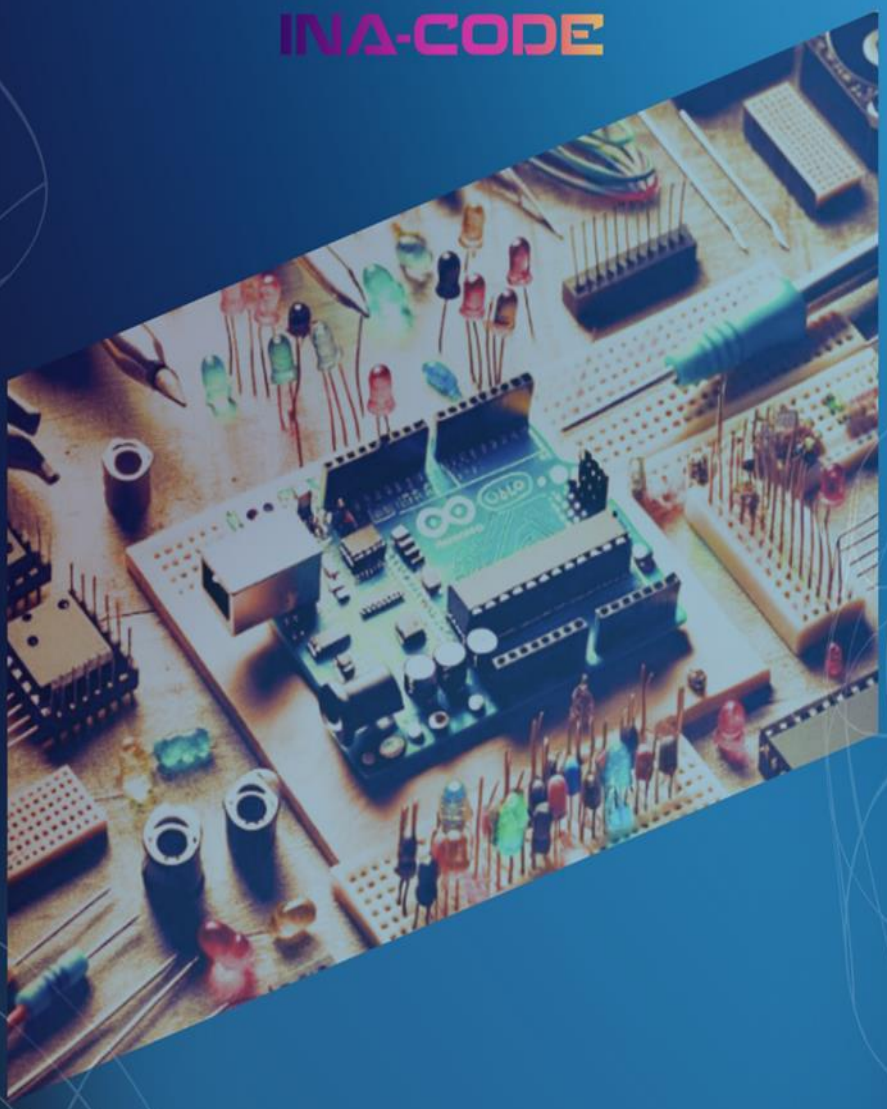
EXPERIMENT BOOKLET

INA-CODE



INNOVATIVE APPROACH FOR CODING IN DIGITAL ERA

EXPERIMENT BOOKLET

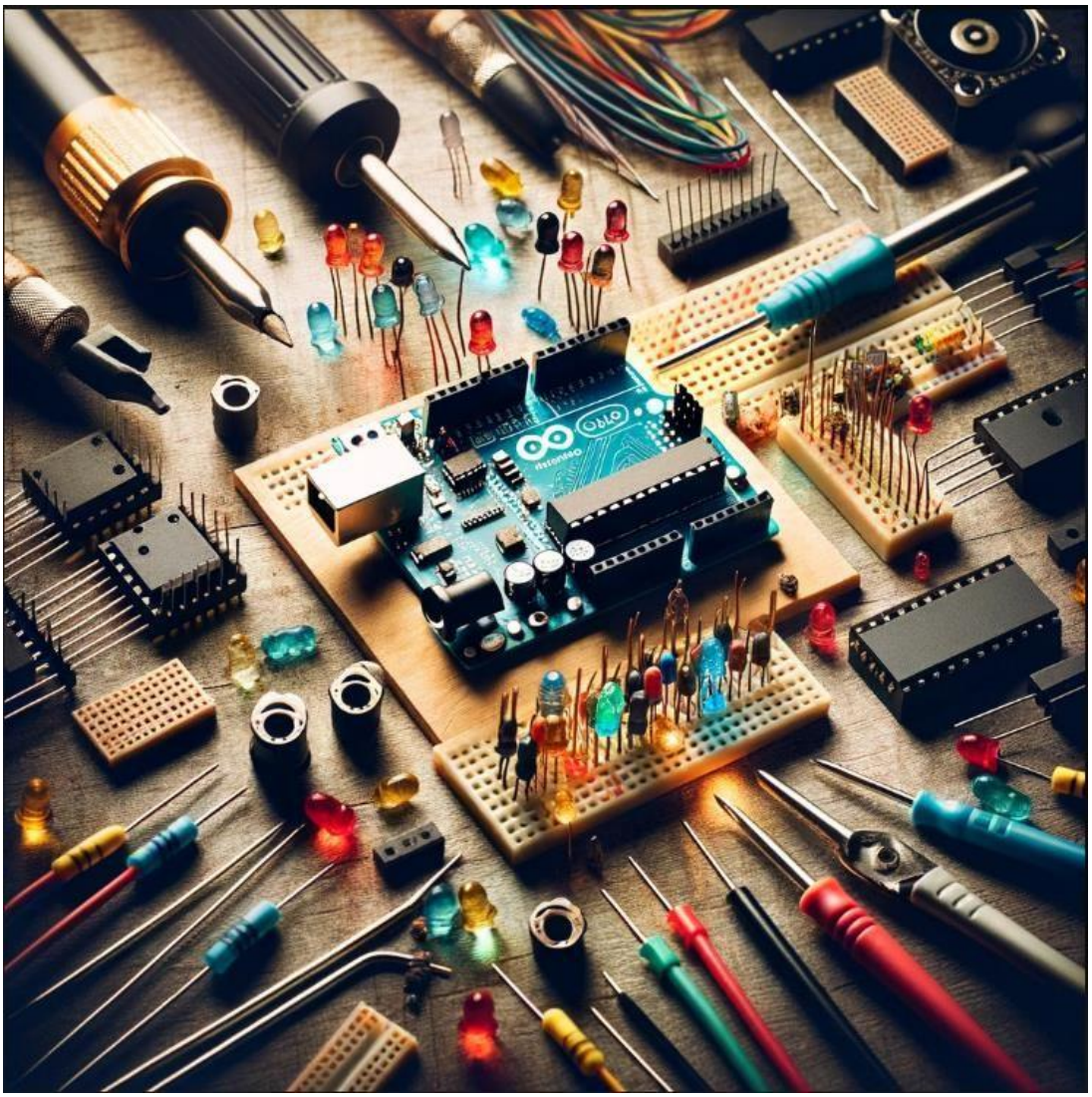
INNOVATIVE APPROACH FOR CODING IN DIGITAL ERA

2021-DE03-KA220-SCH-000024558

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them

Libretto degli esperimenti



Indice dei contenuti

Prefazione

Nel mondo di oggi, il ritmo e la portata della trasformazione digitale richiedono che i nostri sistemi educativi si adattino a questi cambiamenti. In questo contesto, il nostro lavoro, "Experiment Booklet: Teacher's Book & Student's Book including 25 Experiments", rappresenta un passo significativo verso il miglioramento delle competenze digitali di insegnanti e studenti. Questo opuscolo fa parte di un'iniziativa più ampia sostenuta dalla Commissione europea e dalle Agenzie nazionali, volta a colmare il divario digitale e a promuovere un panorama educativo digitale più inclusivo in tutta Europa.

La creazione di questo opuscolo è stata motivata dall'urgente necessità di dotare gli educatori degli strumenti e delle conoscenze necessarie per navigare nell'era digitale, in particolare sulla scia delle sfide evidenziate dalla pandemia COVID-19. Il manuale è una risorsa per gli insegnanti che possono integrare il coding e la robotica nei loro programmi di studio, migliorando così non solo i loro profili professionali, ma anche preparando gli studenti alle esigenze della forza lavoro del XXI secolo.

I nostri partner in questa impresa includono un consorzio di scuole, istituti di formazione professionale e università in tutta Europa, tutti uniti dall'obiettivo comune di far progredire l'istruzione digitale. L'opuscolo è stato progettato per essere pratico, accessibile e adattabile a vari contesti educativi, in modo da garantire che insegnanti e studenti provenienti da contesti diversi possano trarne beneficio.

Lo scopo di questo opuscolo va oltre il semplice insegnamento del coding e della robotica; mira a promuovere il pensiero critico, la creatività e la capacità di risolvere i problemi tra gli studenti. Incorporando questi esperimenti nel loro insegnamento, gli educatori possono offrire un'esperienza di apprendimento più coinvolgente e interattiva, incoraggiando gli studenti a esplorare le vaste possibilità della tecnologia digitale.

In conclusione, questo opuscolo non è solo un ausilio didattico; è un catalizzatore del cambiamento nei nostri sistemi educativi, che promuove una società più competente e resiliente dal punto di vista digitale. Siamo orgogliosi di contribuire a questo percorso di trasformazione e speriamo che ispiri educatori e studenti ad abbracciare le sfide e le opportunità dell'era digitale.

Numero di esperimenti: 1

1- Batterie

Obiettivo dell'esperimento

L'obiettivo di questo esperimento è spiegare come funzionano le batterie e quali tipi di batterie esistono. Gli studenti impareranno a riconoscere i diversi tipi di batterie.

Contesto teorico

Quando si sceglie una batteria, è necessario tenere presente la tensione e la capacità della stessa. Per comprendere meglio questi termini, possiamo fare un'analogia con l'acqua.

Immaginiamo di avere un serbatoio pieno d'acqua.

La tensione è la pressione nel serbatoio dell'acqua che spinge l'acqua verso il tubo. L'altezza dell'acqua aumenta la pressione ed è equivalente alla tensione nei circuiti elettrici.

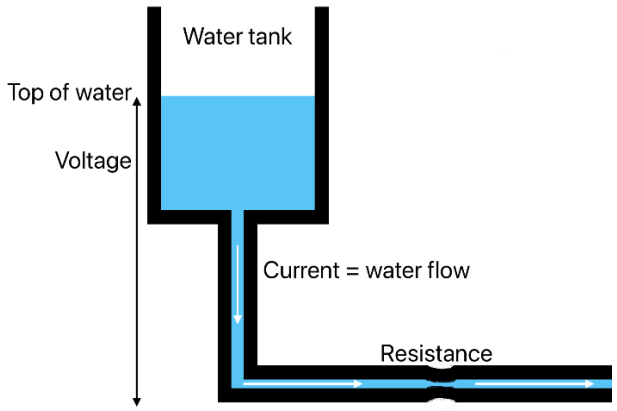
Sul fondo di un serbatoio si trova un tubo. L'acqua che scorre nel tubo rappresenta una corrente elettrica. Più alta è la pressione, più acqua si ottiene, cioè più alta è la tensione, più alta è la corrente.

Un'ostruzione può trovarsi all'estremità del tubo. Rappresenta una resistenza perché il flusso dell'acqua è ridotto. Quanto più piccolo è il diametro del tubo (maggiore resistenza), tanto minore sarà il flusso d'acqua. Questo è rappresentato nel circuito elettrico: maggiore è la resistenza, a parità di tensione, la corrente sarà minore.

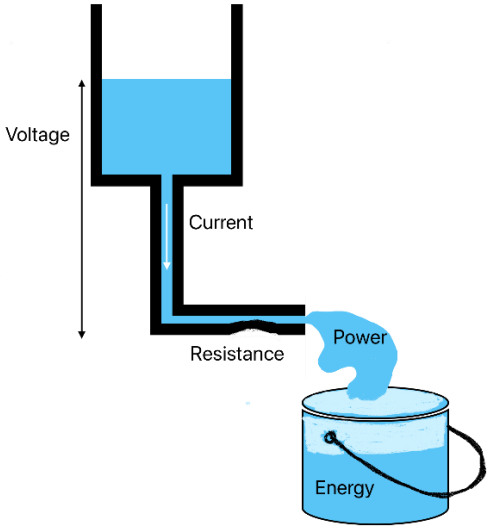
In parole matematiche:

$$U = I \cdot R$$

dove U rappresenta la tensione, I la corrente e R la resistenza.



L'analogia può essere estesa sia alla potenza che all'energia. La potenza è analoga alla portata dell'acqua. L'energia è la quantità di acqua che finisce nel secchio. La potenza si misura in watt (W) o kilowatt (kW) e l'energia in wattora (Wh), miliwattora (mWh) o kilowattora (kWh).



Nella figura sono riportati alcuni esempi di batterie.

Materiali richiesti

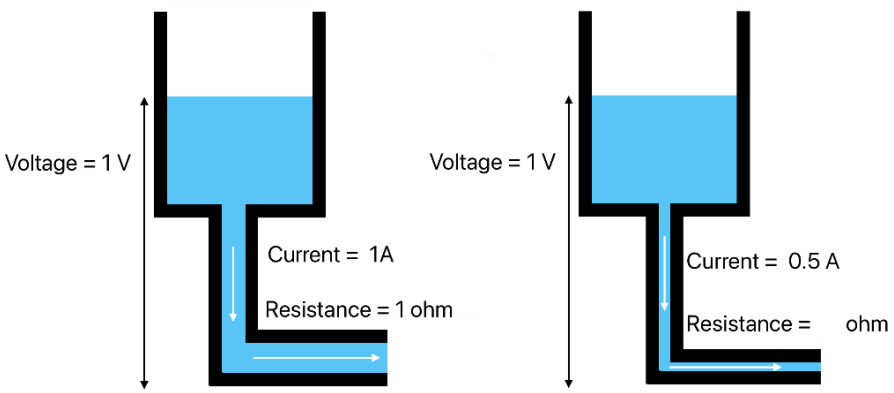
Batterie diverse.

Compito 1.

Trovate diverse batterie nella vostra classe e determinatene il nome del modello (ad esempio AA o AAA) e il voltaggio.

Compito 2.

Per la tensione e la corrente indicate nella figura, determinare la resistenza.



Soluzione 1.

Nella figura di sinistra, la resistenza è

$$R = \frac{U}{I} = \frac{1V}{1A} = 1\Omega$$

Nella figura di destra, la resistenza è

$$R = \frac{U}{I} = \frac{1\,V}{0.5} = 2\Omega$$

Numero di esperimenti: 2

2- Introduzione all'elettronica: resistenze, LED, fotoresistenze e sensori flessibili.

Obiettivo dell'esperimento

L'obiettivo di questo esperimento è spiegare come collegare LED, resistenze, fotoresistenze e sensori flessibili ai circuiti elettrici. Gli studenti impareranno a utilizzare questi elementi in modo sicuro per evitare che si brucino durante l'uso.

Contesto teorico

Un diodo a emissione luminosa o LED è un elemento elettronico semiconduttore che emette luce quando viene attraversato da corrente.

La parte più corta del diodo si chiama catodo e rappresenta il polo -. È collegato al polo - delle batterie o dell'alimentazione.

La parte più lunga del diodo si chiama anodo e rappresenta il polo +. È collegato al + della batteria o dell'alimentazione.

Se il diodo è collegato in modo errato, non passa corrente e non si accende.

Quando si collega un LED a un circuito, occorre considerare la corrente che lo attraversa. Se la corrente che attraversa il diodo è eccessiva, il diodo si brucia. La corrente massima raccomandata attraverso il LED è di 20 mA.

Per evitare che si bruci, è necessario collegare una resistenza al circuito elettrico con il LED. Il resistore da utilizzare si determina in base alla legge di Ohm:

$$U = I \cdot R$$

dove U rappresenta la tensione, I la corrente e R la resistenza.

Il LED ha una tensione di 3 V. Se utilizziamo una batteria da 9 V, abbiamo bisogno di una resistenza:

$$R = \frac{U}{I} = \frac{9V - 3V}{20mA} = \frac{6V}{0.02A} = 300\Omega$$

In genere, si utilizza un resistore con valori standard che si avvicinano maggiormente al valore calcolato. In questo esempio, si tratta di un resistore da 330Ω.

Compito 1

Determinare la resistenza da utilizzare se si collega il LED a una batteria da 4,5 V.

Soluzione 1

Dobbiamo usare:

$$R = \frac{U}{I} = \frac{4{,}5V - 3V}{20mA} = \frac{1.5V}{0.02A} = 75\Omega$$

Compito 2

Ricercate su Internet la caduta di tensione dei LED di altri colori e compilate una tabella.

Colore del LED	Tensione
Giallo	
Arancione	
Rosso	
Blu	
Verde	
Viola	

Soluzione 2.

Colore del LED	Tensione
Giallo	1.9-2.1V
Arancione	2.0-2.1V
Rosso	1.6-2.0V
Blu	2.7-3.2C
Verde	1.9-2.2V
Viola	2.8-4.0V

Compito 3

Prova il simulatore di circuiti elettronici: https://www.falstad.com/circuit/e-

voltdivide.html Il circuito mostrato ha due rami in parallelo. La tensione della batteria è

di 10 V.

Nel primo ramo sono presenti due resistenze da 10 $k\Omega$, per un totale di 20 $k\Omega$. La corrente che attraversa queste è $I = \frac{U}{R} = \frac{10V}{20k\Omega} = 0{,}5\,mA$.

Nel secondo ramo sono presenti quattro resistenze da 10 kΩ, per un totale di $40k\Omega$. La corrente che attraversa queste è $= \frac{U}{R} = \frac{10V}{40k\Omega} = 0{,}25\,mA$.

Nell'animazione, il ramo con corrente maggiore è rappresentato da punti che viaggiano più velocemente, mentre i punti del ramo con corrente minore viaggiano più lentamente.

Determinare i valori delle resistenze nell'altro ramo in modo che i valori della corrente siano uguali. Eseguire il test nel simulatore.

Soluzione 3

Nel secondo ramo devono essere presenti quattro resistenze da 5 $k\Omega$, per un totale di 20 $k\Omega$. La corrente attraverso Queste resistenze sarebbero $I = \frac{U}{R} = \frac{10V}{20k\Omega} = 0{,}5\,mA$.

Compito 4

Se nel primo ramo sono presenti due resistenze da $10k\Omega$, la tensione tra di esse è $\frac{10V}{2} = 5V$.

Quando una delle resistenze cambia, cambia anche la tensione tra di esse. Se la resistenza superiore è di $10k\Omega$ e quella inferiore è di $30k\Omega$, la tensione tra di esse sarà di 7,5 V.

Determinare il valore del resistore inferiore in modo che la tensione tra i due sia

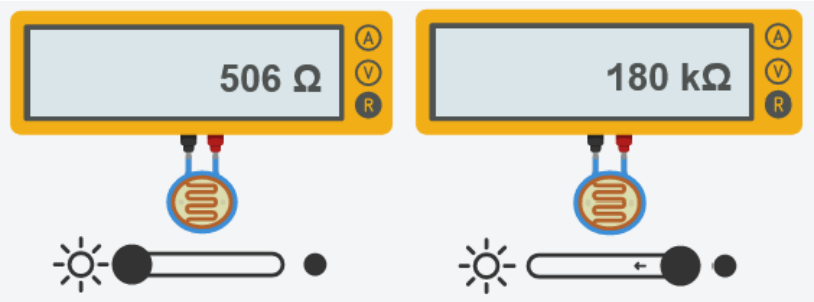
di 8V. Un resistore che cambia il suo valore è chiamato **potenziometro**.

Soluzione 4

Oltre alla resistenza superiore di $10k\Omega$, quella inferiore deve essere di $40k\Omega$.

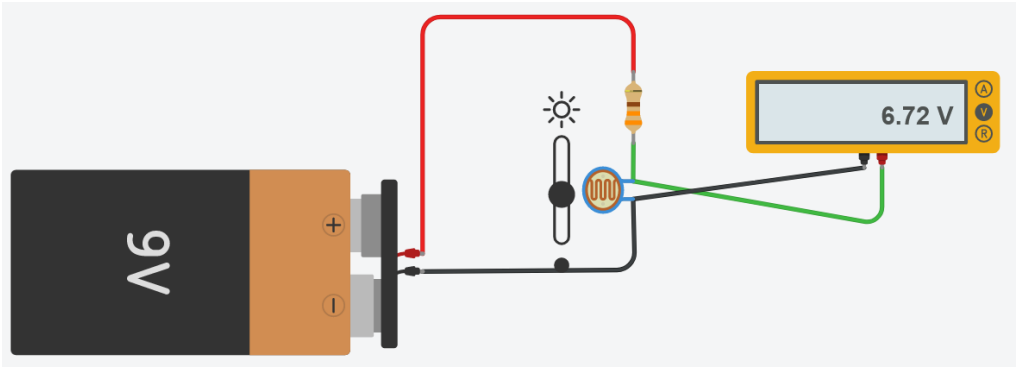
Compito 5

Il fotoresistore è un resistore utilizzato per misurare la luce. Più luce cade su di esso, la sua resistenza è più bassa. Meno luce cade su di esso, la sua resistenza è più alta. Il fotoresistore si comporta come un resistore variabile,
cioè un potenziometro. L'unica differenza è che non lo regoliamo con le nostre mani, ma con l'illuminazione della stanza.



Al variare della resistenza del fotoresistore, cambia anche la sua tensione. Quanto più alta è la resistenza del fotoresistore (assenza di luce), tanto più alta è la sua tensione.

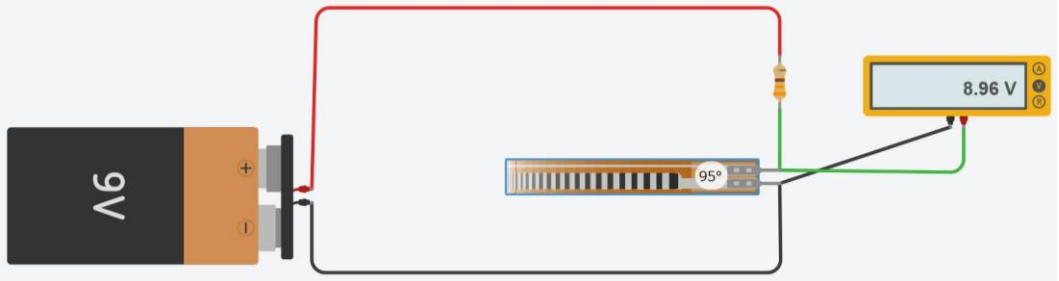
In TINKERCAD, collegate la batteria da 9 V, il resistore, il fotoresistore e il voltmetro, secondo l'esempio della figura. Cambiate la luce sul fotoresistore e osservate come cambiano la sua resistenza e la sua tensione.



Compito 5

Un sensore di flessione è un sensore che misura la quantità di deviazione o di flessione. Viene solitamente utilizzato nei dispositivi indossabili biotecnologici per il rilevamento della posizione e della mobilità delle articolazioni umane.

Collegare il sensore flessibile secondo lo schema e misurare la resistenza e la tensione.



Numero di esperimenti: 3

3- Introduzione all'elettronica: capacità, induttanza, relè, transistor e ponte H con motore in c.c.

Obiettivo dell'esperimento

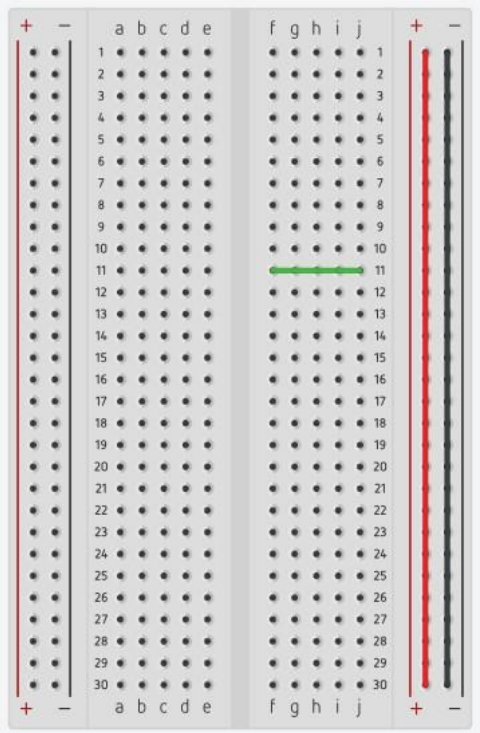
L'obiettivo di questo esperimento è spiegare come collegare capacità, induttanza, relè, transistor e motori CC ai circuiti elettrici. Gli studenti impareranno a utilizzare questi elementi in modo sicuro per evitare che si brucino durante l'uso.

Contesto teorico

Un **condensatore** è un elemento elettronico in grado di immagazzinare una carica elettrica. La grandezza fisica che descrive la capacità di immagazzinare carica è la capacità elettrica. La **capacità** si misura in **Farad (F)**.
Aumentando la tensione aumenta anche la quantità di carica immagazzinata. Allo stesso tempo, la capacità non cambia. Nella maggior parte dei casi, il condensatore ha due piastre conduttrici parallele che vengono elettrificate collegandole a una sorgente di tensione continua. Una piastra è collegata al polo positivo e l'altra al polo negativo della sorgente.

Quando una corrente elettrica attraversa la bobina, si crea un campo magnetico. Una **bobina** è un elemento elettronico che immagazzina l'energia di un campo elettromagnetico. La quantità di energia immagazzinata dipende dall'induttanza e dalla corrente che attraversa la bobina stessa. L'**induttanza** della bobina è una quantità utilizzata per descrivere la capacità di immagazzinare l'energia del campo magnetico. Maggiore è la corrente che attraversa la bobina, maggiore è l'induzione magnetica della bobina. L'unità di misura dell'induttanza **è l'Henri (H)**.

Protoboard è una scheda di costruzione utilizzata per collegare i circuiti elettrici dei prototipi. Cinque fori in una fila sono cortocircuitati. Se colleghiamo un elemento elettrico o un filo a due di essi, si comporteranno come se i contatti fossero direttamente collegati. Lo stesso vale per le colonne sotto i segni + e - all'estrema sinistra e all'estrema destra della scheda prototipo.



Compito 1

Considerate un semplice circuito elettrico che contiene solo una sorgente, un singolo resistore, un **condensatore** e un interruttore. Studiate cosa succede alla corrente quando l'interruttore viene acceso e spento. Aggiungete una lampadina al circuito e osservate quando si accende.

Link al simulatore: https://www.falstad.com/circuit/e-cap.html

Compito 2

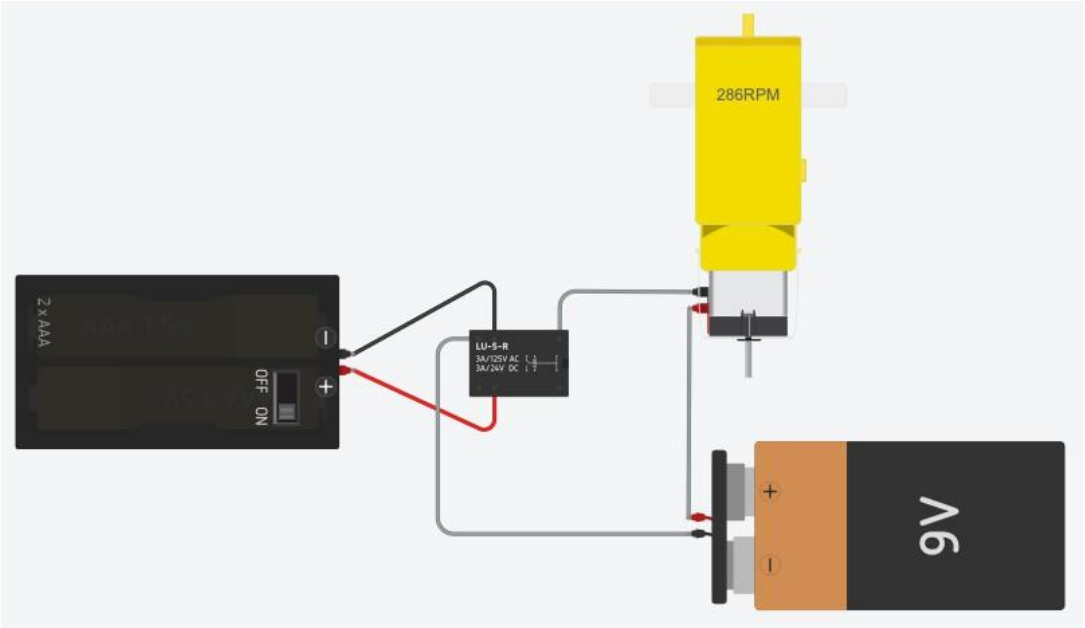
Considerate un semplice circuito elettrico che contiene solo una sorgente, un singolo resistore, un'**induttanza** e un interruttore. Studiate cosa succede alla corrente quando l'interruttore viene acceso e spento. Aggiungete una lampadina al circuito e osservate quando si accende.

Link al simulatore: https://www.falstad.com/circuit/e-induct.html

Compito 3

Il **relè** è un interruttore elettrico. Viene attivato dalla corrente del primo circuito elettrico per accendere o spegnere il secondo circuito elettrico. Quando si lavora con i relè, è obbligatorio utilizzare due alimentatori separati per due circuiti distinti.

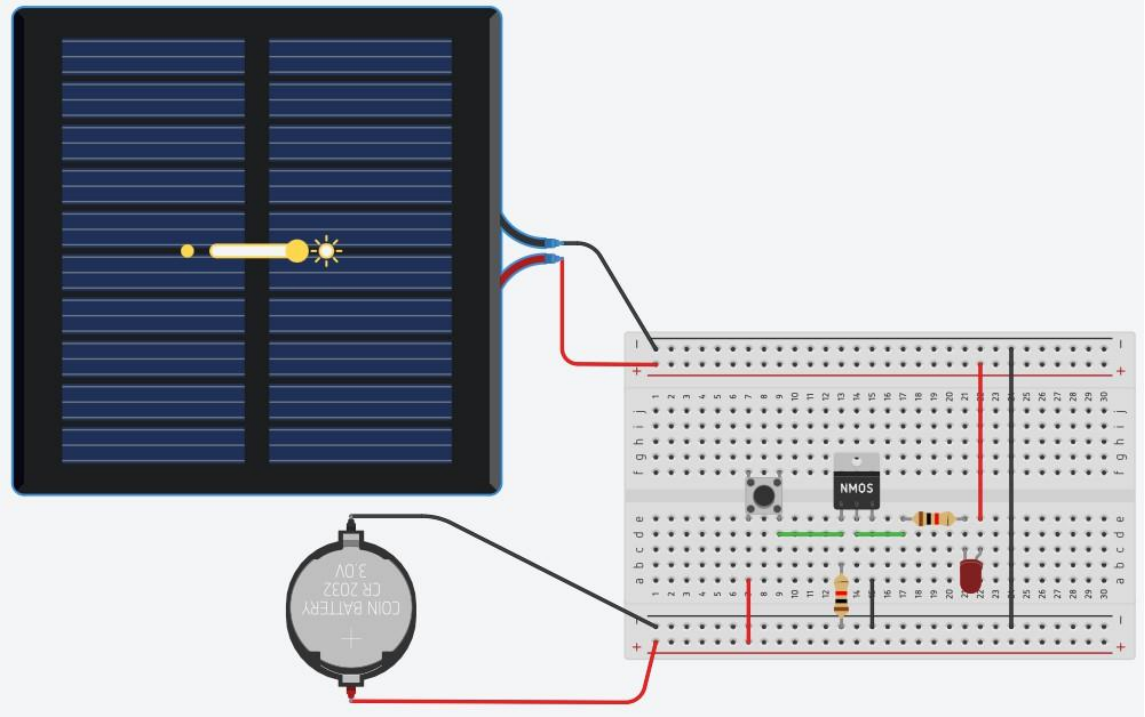
Collegare lo schema con il relè in TINKERCAD secondo la figura e verificare il funzionamento del relè.



Compito 4

Un **transistor NMOS** è un semiconduttore utilizzato come interruttore elettronico.

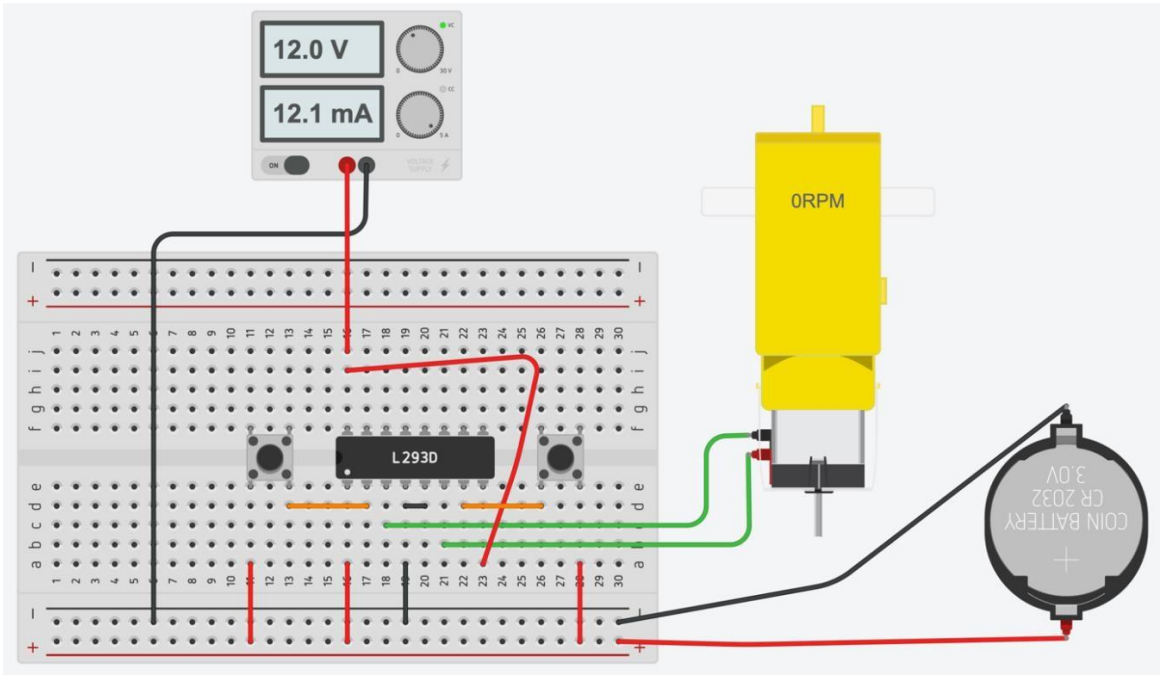
Collegare lo schema con NMOS in TINKERCAD secondo la figura e verificarne il funzionamento. Modificare i valori della cella solare e testarla. Modificate i valori dei resistori e testateli.



Compito 5

I motori CC hanno due fili, negativo (nero) e positivo (rosso). Quando il positivo è collegato all'alimentazione e il negativo a GND, l'albero del motore ruota. Quando i fili sono collegati al contrario, positivo a GND e negativo all'alimentazione, l'albero del motore ruota di nuovo, ma in direzione opposta. Per non dover collegare manualmente i fili ogni volta che la direzione viene cambiata, si utilizza il chip a ponte H.

Collegare lo schema con il ponte H e il motore in TINKERCAD secondo la figura e testarlo. Premere i pulsanti e osservare in quale direzione ruota l'albero del motore.



Numero di esperimenti: 4

4- Introduzione ad Arduino

Obiettivo dell'esperimento

L'obiettivo di questo esperimento è spiegare cos'è e a cosa serve un microcontrollore. Gli studenti impareranno a usare il simulatore per collegare l'elettronica ad Arduino.

Contesto teorico

Un microcontrollore è un computer piccolo ed economico su un singolo chip che può essere utilizzato per controllare le funzioni di elettrodomestici e uffici, robot, veicoli, elettronica di consumo e altro ancora. Il microcontrollore più famoso è Arduino UNO.

Compito 1

In Tinkercad, menu Starters, selezionare Arduino. Verranno aperti oltre 30 schemi dimostrativi e programmi con Arduino e l'elettronica.

Scegliete un programma dimostrativo e rispondete alle domande.

1. Quale elemento elettronico viene utilizzato?
2. A quale pin di Arduino è collegato l'elemento?
3. Avviare la simulazione. Che cosa ha fatto il programma?
4. Copiare lo schema.
5. Aprire il codice. Copiare il codice utilizzato. Provate a collegare gli stessi comandi dalle versioni Blocks e Text del programma.

Compito 2

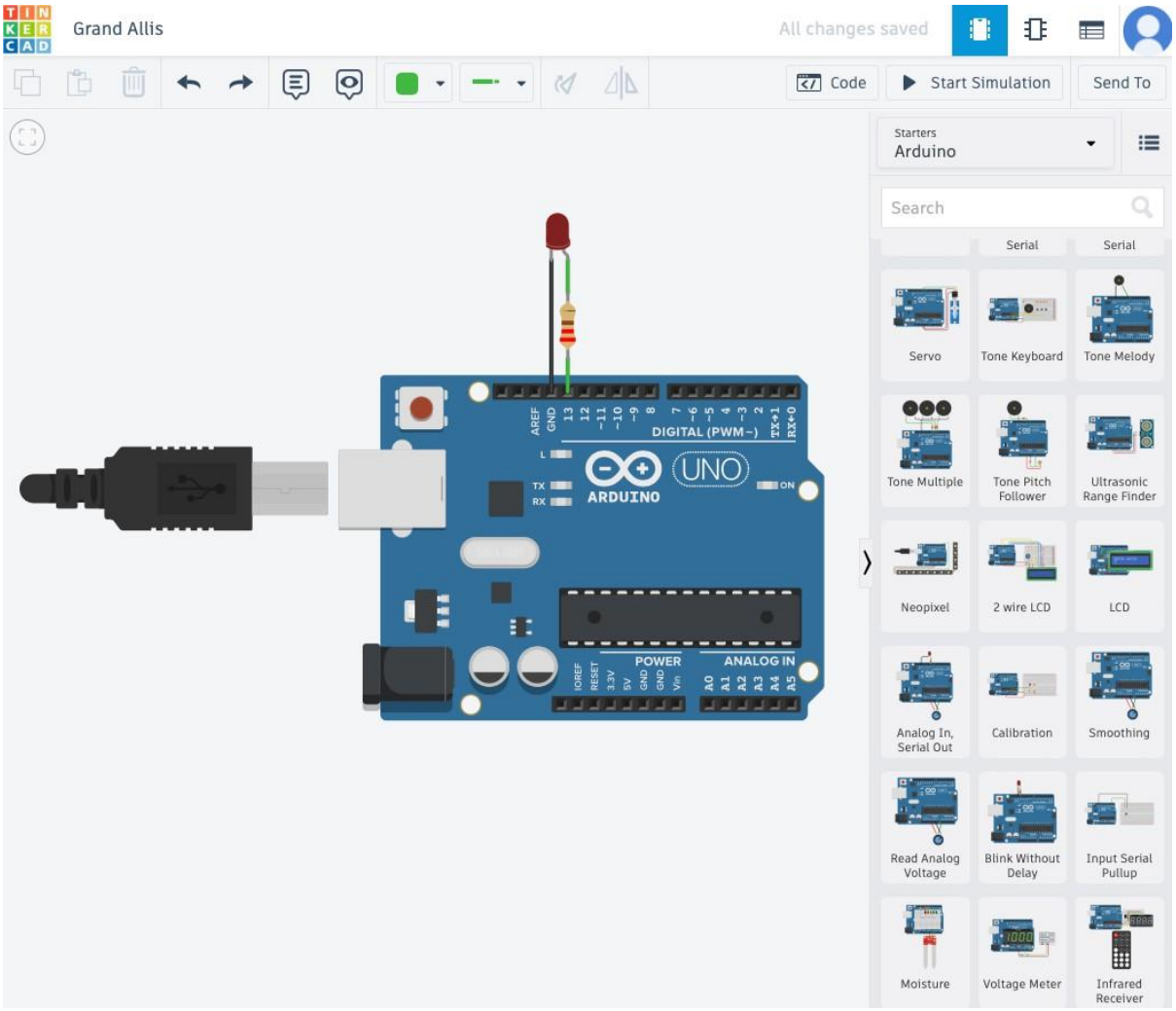
In Tinkercad, menu Starters, selezionare Arduino. Verranno aperti oltre 30 schemi dimostrativi e programmi con Arduino e l'elettronica.

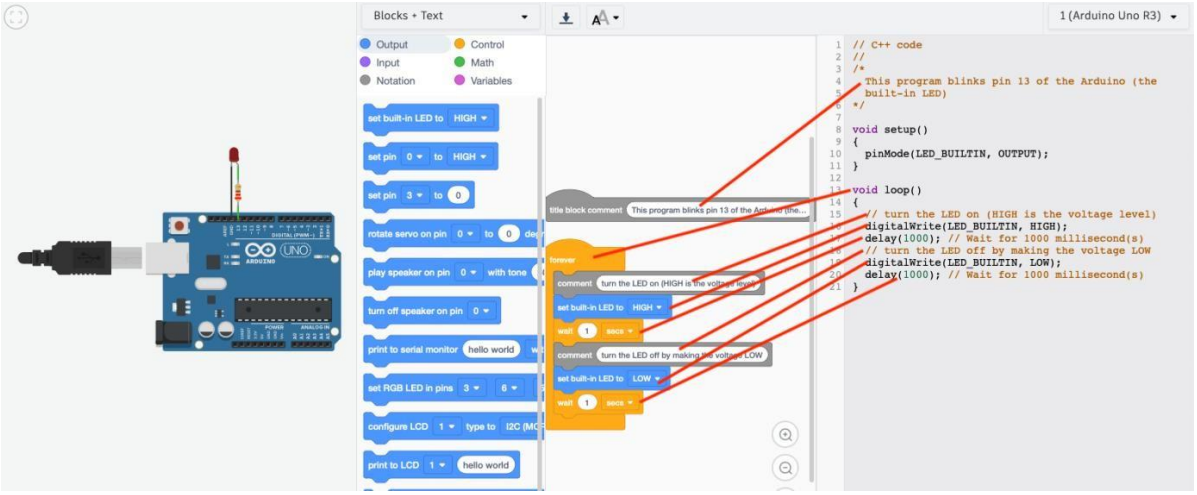
Scegliete un programma dimostrativo e rispondete alle domande.

1. Quale elemento elettronico viene utilizzato?
2. A quale pin di Arduino è collegato l'elemento?
3. Avviare la simulazione. Che cosa ha fatto il programma?
4. Copiare lo schema.
5. Aprire il codice. Copiare il codice utilizzato. Provate a collegare gli stessi comandi dalle versioni Blocks e Text del programma.

Possibile soluzione per il Compito 1 e il Compito 2

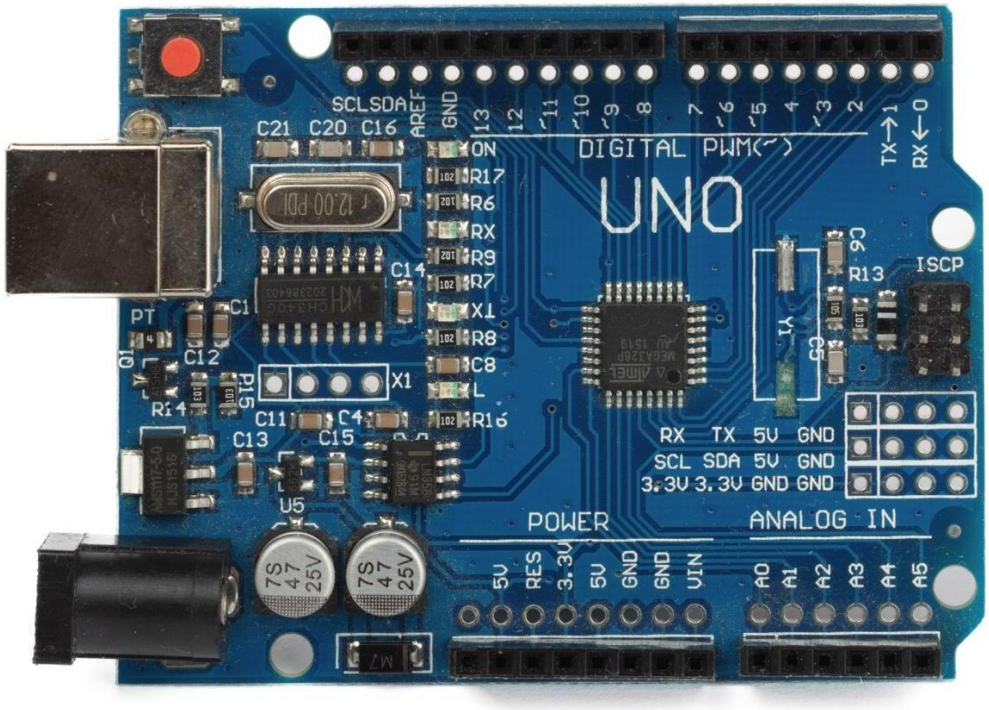
Un esempio è il demo Blink. Al pin 13 sono collegati un LED e un resistore. Il LED lampeggia ogni 1 s.


Grand Allis
All changes saved
Code
Start Simulation
Send To
Starters
Arduino
Search
Serial
Serial
Servo
Tone Keyboard
Tone Melody
Tone Multiple
Tone Pitch Follower
Ultrasonic Range Finder
Neopixel
2 wire LCD
LCD
Analog In, Serial Out
Calibration
Smoothing
Read Analog Voltage
Blink Without Delay
Input Serial Pullup
Moisture
Voltage Meter
Infrared Receiver
DIGITAL (PWM~)
POWER
ANALOG IN
ARDUINO
UNO


Blocks + Text
1 (Arduino Uno R3)
Output
Input
Notation
Control
Math
Variables
set built-in LED to HIGH
set pin 0 to HIGH
set pin 3 to 0
rotate servo on pin 0 to 0
play speaker on pin 0 with tone
turn off speaker on pin 0
print to serial monitor hello world
set RGB LED in pins 3 6
configure LCD 1 type to I2C
print to LCD 1 hello world
title block comment This program blinks pin 13 of the Arduino (the...
forever
comment turn the LED on (HIGH is the voltage level)
set built-in LED to HIGH
wait 1 secs
comment turn the LED off by making the voltage LOW
set built-in LED to LOW
wait 1 secs
// C++ code
//
/*
This program blinks pin 13 of the Arduino (the built-in LED)
*/
void setup()
{
pinMode(LED_BUILTIN, OUTPUT);
}
void loop()
{
// turn the LED on (HIGH is the voltage level)
digitalWrite(LED_BUILTIN, HIGH);
delay(1000); // Wait for 1000 millisecond(s)
// turn the LED off by making the voltage LOW
digitalWrite(LED_BUILTIN, LOW);
delay(1000); // Wait for 1000 millisecond(s)
}

Compito 3

Nella foto di Arduino UNO segnare le parti più importanti.



5- Piattaforme Arduino

. Che cos'è l'esperimento

Arduino Obiettivo

L'obiettivo è utilizzare Arduino o una tecnologia simile per mostrare un'applicazione pratica che coinvolga sensori, attuatori o altri componenti. Il progetto mira a risolvere un problema o a svolgere un compito specifico, dimostrando come queste tecnologie possano essere applicate in scenari reali.

Contesto teorico

I concetti fondamentali comprendono l'elettronica, la programmazione e i principi di progettazione del sistema rilevanti per l'esperimento. Gli argomenti trattati possono riguardare la teoria dei circuiti, la logica digitale, l'architettura dei microcontrollori e l'uso di C/C++ nello sviluppo di Arduino, fornendo una comprensione completa necessaria per la realizzazione di un progetto di successo.

Compito 1

Creare un semplice sistema di monitoraggio della temperatura utilizzando una scheda Arduino per comprenderne meglio le funzionalità.

Materiali richiesti

Arduino UNO R3, software Arduino IDE o portale di simulazione Tinkercad.

Circuito sperimentale

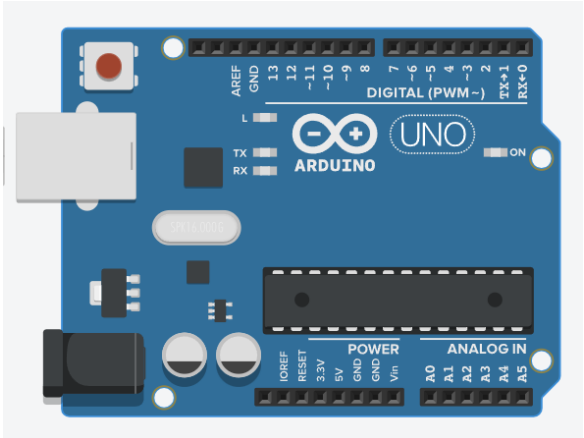


Figura 1 Scheda di sviluppo Arduino UNO R3

Collegare la scheda Arduino al computer tramite un cavo USB di tipo A/B e aprire l'IDE Arduino. Selezionare la porta (Strumenti→Porta) a cui è collegato Arduino. Aprire la schermata Serial Monitor dal menu Strumenti.

Soluzione: Fasi

della procedura

1- Impostare il circuito come descritto, assicurandosi che tutti i collegamenti siano sicuri.

2- Scrivere e caricare il codice su Arduino.

3- Osservare la schermata della porta seriale.

4- Osservare le modifiche sullo schermo.

Codici

```
// Definire il pin di ingresso a cui è collegato il sensore di
temperatura int sensorPin = A0;
void setup() {
  Serial.begin(9600); // Avvio della comunicazione seriale
}
void loop() {
  int reading = analogRead(sensorPin); // Leggere il valore del
  sensore float voltage = reading * 5,0;
  tensione /= 1024,0; // Convertire la lettura in tensione
  float temperaturaC = (tensione - 0,5) * 100; // Convertire la tensione in temperatura in gradi Celsius
  Serial.print("Temperatura: ");
  Serial.print(temperaturaC);
  Serial.println(" C");
    delay(1000); // Attendere un secondo prima di ripetere il ciclo
}
```

Spiegazione del Codice

Il codice legge l'uscita del sensore di temperatura su Arduino, la converte in tensione, calcola la temperatura in gradi Celsius e la visualizza continuamente sul Serial Monitor.

6- Linguaggio di programmazione e editor Arduino-1

Obiettivo dell'esperimento

L'obiettivo dell'esperimento con Arduino è scrivere un semplice programma per controllare vari LED, accendendoli e spegnendoli in sequenza a intervalli specifici. Questo esperimento mira a insegnare i costrutti di programmazione di base di Arduino e come interagire con il mondo fisico attraverso la programmazione.

Contesto teorico

Il linguaggio di programmazione utilizzato in Arduino è basato su Wiring, una versione personalizzata di C++ progettata per facilitare l'uso delle comuni operazioni di input/output. I punti chiave di questo linguaggio di programmazione sono la programmazione orientata agli oggetti (OOP), la portabilità, le alte prestazioni, la libreria standard, la versatilità, la programmazione multi-paradigma e la gestione della memoria. La programmazione di Arduino ruota attorno alle funzioni setup() e loop(). setup() viene chiamata una volta all'inizio del programma per impostare configurazioni come le modalità dei pin, mentre loop() contiene la logica principale del programma che si ripete nel tempo, controllando l'hardware come i LED in base alla logica del programma.

Materiali richiesti

Arduino UNO R3, software Arduino IDE o portale di simulazione Tinkercad.

Circuito sperimentale

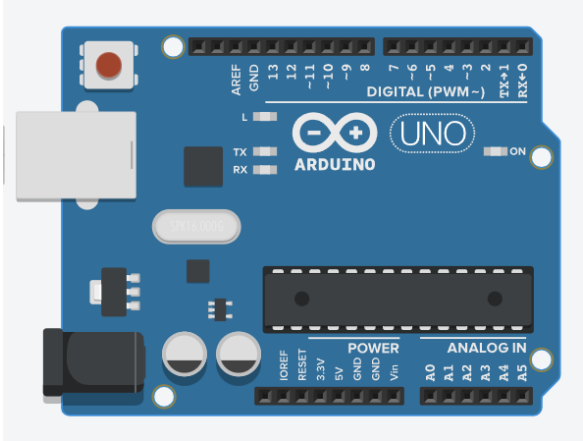


Figura 2 Scheda di sviluppo Arduino UNO R3

Collegare la scheda Arduino al computer tramite un cavo USB di tipo A/B e aprire l'IDE Arduino. Selezionare la porta (Strumenti→Porta) a cui è collegato Arduino. Aprire la schermata Serial Monitor dal menu Strumenti.

Fasi della procedura

1-Impostare il circuito come descritto, assicurandosi che tutti i collegamenti siano sicuri.

2-Scrivere e caricare il codice su Arduino.

3-Osservare la schermata della porta seriale.

4-Osservare le modifiche sullo schermo.

Codici

Primo esempio di codice: Controllo dei led

```
int led1 = 4, led2 = 5, led3 = 6, led4 = 7;
void setup() {
  Serial.begin(9600);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
}
void loop() {
  digitalWrite(led1, HIGH);
  delay(1000); // Attendere 1 secondo
 digitalWrite(led1, LOW);
 delay(500); // Attendere 0,5 secondi
 digitalWrite(led2, HIGH);
 delay(1000);
  digitalWrite(led2, LOW);
 delay(500);
 digitalWrite(led3, HIGH);
 delay(1000);
 digitalWrite(led3, LOW);
 delay(500);
 digitalWrite(led4, HIGH);
 delay(1000);
 digitalWrite(led4, LOW);
 delay(500);
}
```

Spiegazione del Codice

L'esempio di codice mostra come accendere e spegnere in sequenza quattro LED collegati ad Arduino, creando un semplice motivo visivo. Utilizza funzioni di base come pinMode(), digitalWrite() e delay() per controllare la tempistica e lo stato di ciascun LED.

Secondo esempio di codice: Lettura del valore del sensore

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
  delay(1); // Breve ritardo per la leggibilità
}
```

Spiegazione del Codice

L'esempio di codice si concentra sulla lettura dei valori di un sensore collegato al pin analogico A0 di Arduino e sulla stampa di questi valori sul monitor seriale. Mostra come utilizzare analogRead() e Serial.println() per l'acquisizione e la registrazione dei dati del sensore.

7- Linguaggio di programmazione e editor Arduino-2

Obiettivo dell'esperimento

L'obiettivo è esplorare le capacità di Arduino nell'interfacciarsi con vari sensori e componenti. Si tratta di creare progetti che dimostrino come Arduino possa essere utilizzato per applicazioni pratiche come il monitoraggio ambientale, la robotica e le installazioni artistiche interattive.

Contesto teorico

Le basi teoriche comprendono la comprensione dei principi dei microcontrollori, in particolare l'architettura e la programmazione di Arduino. Vengono trattati gli input/output digitali e analogici, le basi dell'elettricità e dei circuiti, i costrutti di programmazione (variabili, loop, condizionali) e i protocolli di comunicazione come Serial e I2C. L'approfondimento tocca anche l'importanza dell'hardware e del software open-source nel promuovere l'innovazione e l'educazione all'elettronica e alla programmazione.

Materiali richiesti

Arduino UNO R3, software Arduino IDE o portale di simulazione Tinkercad.

Circuito sperimentale

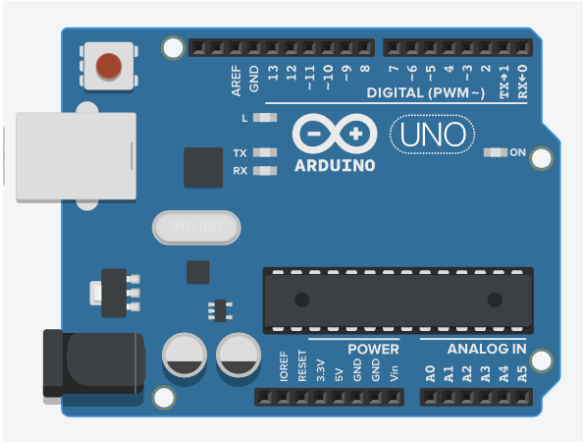


Figura 3 Scheda di sviluppo Arduino UNO R3

Collegare la scheda Arduino al computer tramite un cavo USB di tipo A/B e aprire l'IDE Arduino. Selezionare la porta (Strumenti→Porta) a cui è collegato Arduino. Aprire la schermata Serial Monitor dal menu Strumenti.

Compito 1.

Creare un progetto basato su Arduino che dimostri i principi della progettazione di circuiti elettronici e della programmazione utilizzando Arduino IDE.

Soluzione 1.

Progettare un circuito che interagisce con vari sensori e attuatori, programmati con Arduino per eseguire funzioni specifiche, come la misurazione delle condizioni ambientali o il controllo di luci o motori.

Numero di esperimenti: 8

8- Comandi per le operazioni matematiche nell'IDE Arduino: come utilizzare gli operatori aritmetici esperimento

Obiettivo dell'esperimento

L'obiettivo di questo esperimento è dimostrare come gli operatori aritmetici di base, quali addizione, sottrazione, moltiplicazione, divisione e resto, vengano utilizzati nell'ambiente di sviluppo integrato (IDE) di Arduino. Gli studenti impareranno a utilizzare questi operatori e a stampare il risultato sullo schermo della porta seriale di Arduino IDE.

Contesto teorico

Tutti i linguaggi di programmazione includono alcuni operatori aritmetici e anche l'IDE Arduino include operatori aritmetici di base. Questi operatori possono essere utilizzati in Arduino per eseguire alcune operazioni matematiche, come l'elaborazione dei dati di un sensore. In questa applicazione verranno mostrati 5 operatori aritmetici di base.

Operatori	Operatore Simbolo	Variabili	Esempio	Risultato
Aggiunta	+	Sen1 = 21	Risultato = Sen1 + Sen2	24
Sottrazione	-	Sen2 = 3	Risultato = Sen1 - Sen2	18
Moltiplicazione	*	Risultato = 0	Risultato = Sen1 * Sen2	63
Divisione	/		Risultato = Sen1 / Sen2	7
Resto	%		Risultato = Sen1 % Sen2	0

Materiali richiesti

Arduino UNO R3, software Arduino IDE o portale di simulazione Tinkercad.

Circuito sperimentale

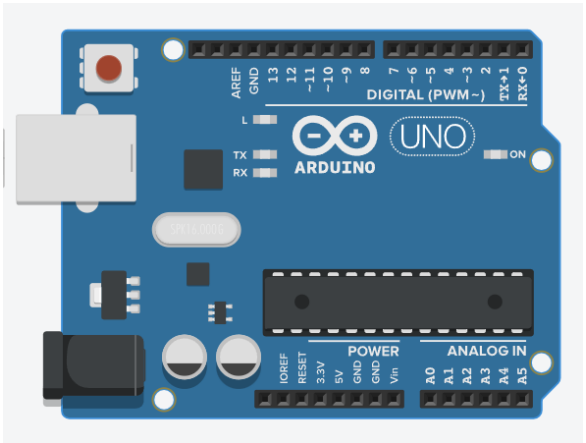


Figura 4 Scheda di sviluppo Arduino UNO R3

Collegare la scheda Arduino al computer tramite un cavo USB di tipo A/B e aprire l'IDE Arduino. Selezionare la porta (Strumenti→Porta) a cui è collegato Arduino. Aprire la schermata Serial Monitor dal menu Strumenti.

Fasi della procedura

1-Impostare il circuito come descritto, assicurandosi che tutti i collegamenti siano sicuri.

2-Scrivere e caricare il codice su Arduino.

3-Osservare la schermata della porta seriale.

4-Osservare le modifiche sullo schermo.

Codici

```
/* Definizione della variabile
globale */ int sensor1 = 21;
int sensor2 = 3;
int Result = 0;
void setup(){
 Serial.begin(9600);
 /Aggiungi un appuntamento per oggi
 Risultato = sensore1 + sensore2;
 Serial.print("L'addizione del sensore1 e del sensore2 è ");
 Serial.println(Risultato);
 ritardo(2000);
 //Sottrazione
 Risultato = sensore1 - sensore2;
 Serial.print("La sottrazione del sensore1 e del sensore2 è ");
 Serial.println(Risultato);
 ritardo(2000);
 //Moltiplicazione
 Risultato = sensore1 * sensore2;
 Serial.print("La moltiplicazione di sensore1 e sensore2 è ");
 Serial.println(Risultato);
 ritardo(2000);
 //Divisione
 Risultato = sensore1 / sensore2;
 Serial.print("La divisione del sensore1 e del sensore2 è ");
 Serial.println(Risultato);
 ritardo(2000);
 //Rimedio
 Risultato = sensore1 % sensore2; Serial.print("Il
 resto del sensore1 e del sensore2 è ");
 Serial.println(Risultato);
 ritardo(2000);
}

void loop() {
}
```

Spiegazione del Codice

Impostazione: Il monitor seriale è attivato alla velocità di 9600 baud. Per le variabili definite in precedenza, le relative operazioni matematiche vengono eseguite una sola volta.

Loop: -

Numero di esperimenti: 9

9- Comandi per le operazioni matematiche in Arduino IDE: come usare gli operatori aritmetici esperimento 2

Obiettivo dell'esperimento

L'obiettivo di questo esperimento è dimostrare come gli operatori aritmetici di base, quali addizione, sottrazione, moltiplicazione, divisione e resto, siano utilizzati nell'ambiente di sviluppo integrato (IDE) Arduino. Gli studenti impareranno a utilizzare questi operatori con le variabili dei numeri in virgola mobile.

Contesto teorico

Il comportamento degli operatori aritmetici varia a seconda del tipo di variabile numerica utilizzata. In particolare, l'operatore di divisione può produrre risultati in virgola mobile. Pertanto, quando l'operatore di divisione viene utilizzato su variabili intere, il risultato atteso potrebbe non essere ottenuto. Nelle applicazioni dei sensori con le schede Arduino, i numeri elaborati sono per lo più di tipo floating point. Inoltre, l'operatore modulo (remainder) è un operatore che può essere utilizzato solo con i numeri interi.

Operatori	Operatore Simbolo	Variabili	Esempio	Risultato
Aggiunta	+	Sen1 = 24,5	Risultato = Sen1 + Sen2	28.00
Sottrazione	-	Sen2 = 3,5	Risultato = Sen1 - Sen2	21.00
Moltiplicazione	*	Risultato = 0,0	Risultato = Sen1 * Sen2	85.75
Divisione	/	Sen3 = 12	Risultato = Sen1 / Sen2	7.00
Resto	%	Sen4 = 5	Risultato = Sen3 % Sen4	2.00

Materiali richiesti

Arduino UNO R3, software Arduino IDE o portale di simulazione Tinkercad.

Circuito sperimentale

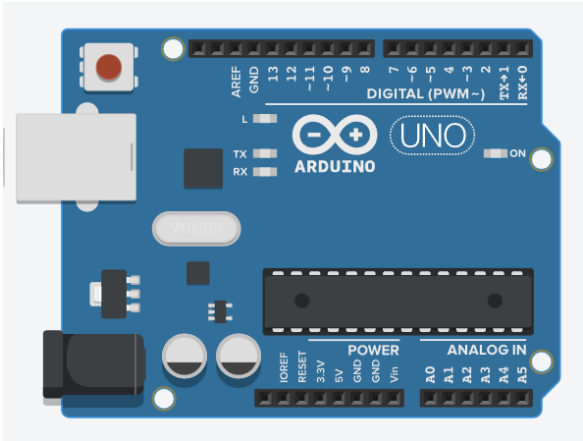


Figura 5 Scheda di sviluppo Arduino UNO R3

Collegare la scheda Arduino al computer tramite un cavo USB di tipo A/B e aprire l'IDE Arduino. Selezionare la porta (Strumenti→Porta) a cui è collegato Arduino. Aprire la schermata Serial Monitor dal menu Strumenti.

Fasi della procedura

1- Impostare il circuito come descritto, assicurandosi che tutti i collegamenti siano sicuri.

2- Scrivere e caricare il codice su Arduino.

3- Osservare la schermata della porta seriale.

4- Osservare le modifiche sullo schermo.

Codici

```
//Variabile globale
float sensore1 =
24,5; float sensore2
= 3,5; float
Risultato = 0,0; int
seed = 123;
vuoto setup()
{
  Serial.begin(9600);

  /Operazione di aggiunta
  Risultato = sensore1 + sensore2;
  Serial.print("L'addizione del sensore1 e del sensore2 è ");
  Serial.println(Risultato);
  ritardo(2000);

  /Operazione di sottrazione
  Risultato = sensore1 -
  sensore2;
  Serial.print("La sottrazione del sensore1 e del sensore2 è ");
  Serial.println(Risultato);
  ritardo(2000);

  /Operazione di multiplazione
  Risultato = sensore1 *
  sensore2;
  Serial.print("La moltiplicazione di sensore1 e sensore2 è ");
  Serial.println(Risultato);
  ritardo(2000);

  /Operazione di divisione
  Risultato = sensore1 / sensore2;
  Serial.print("La divisione del sensore1 e del sensore2 è ");
  Serial.println(Risultato);
  ritardo(2000);

  int sensore3 = 12, sensore4 = 5;
  /Operazione di rimescolamento
  (modulo) Risultato = sensore3 %
  sensore4;
  Serial.print("Il resto del sensore3 e del sensore4 è ");
  Serial.println(Risultato);
  ritardo(2000);

}
```

```
vuoto loop()
{
  randomSeed(seed);
  sensor1 = random(1, 300);
```

```
  sensor2 = random(1, 300);
  Serial.println(sensor1);
  Serial.println(sensor2); seed
  = sensor1 * sensor2; Result
  = sensor1 + sensor2;
  Serial.print("Aggiunta dei dati del sensore = ");
  Serial.println(Risultato);
  Risultato = sensore1 - sensore2;
  Serial.print("Sottrazione dei dati del sensore =
  "); Serial.println(Risultato);
  Risultato = sensore1 * sensore2;
  Serial.print("Moltiplicazione dei dati del
  sensore = "); Serial.println(Risultato);
  Risultato = sensore1 / sensore2;
  Serial.print("Divisione dei dati del sensore = ");
  Serial.println(Risultato);
  Risultato = (int)sensore1 % (int)sensore2;
  Serial.print("Resto dei dati del sensore = ");
  Serial.println(Risultato);
}
```

Spiegazione del Codice

Impostazione: Il monitor seriale è attivato alla velocità di 9600 baud. Per le variabili definite in precedenza, le relative operazioni matematiche vengono eseguite una sola volta.

Loop: È stato progettato un semplice gioco di simulazione. A tale scopo, si utilizzano i metodi integrati random() e randomSeed() dell'editor Arduino. Quando viene eseguita ogni funzione di loop, i dati del sensore 1 e del sensore 2 vengono prodotti di nuovo in modo casuale tra un determinato intervallo. In ogni processo, vengono ripetute 5 operazioni matematiche.

Numero di esperimenti: 10

10- Strutture di controllo in Arduino IDE: come utilizzare le strutture di controllo-1

Obiettivo dell'esperimento

L'obiettivo di questo esperimento è dimostrare come vengono utilizzate le strutture di controllo nell'ambiente di sviluppo integrato (IDE) Arduino. Gli studenti impareranno a utilizzare gli **operatori di confronto** e le ***istruzioni if*** ("*if*", "*if ... else*", "*if ... else if ... else*").

Contesto teorico

Tutti i linguaggi di programmazione hanno alcuni tipi di istruzioni di controllo per prendere una decisione su condizioni specifiche. Per esempio, utilizziamo queste istruzioni di controllo quando decidiamo l'operazione da eseguire in base a un certo valore di temperatura o quando decidiamo l'operazione da eseguire in base al voto ottenuto da uno studente.

Come in altri linguaggi, esistono varie strutture di controllo che utilizziamo nell'IDE Arduino. Queste strutture di controllo e dichiarazioni possono essere elencate come segue:

Dichiarazione di controllo	Descrizione
"Dichiarazione "if	Prende un'espressione tra parentesi e un'affermazione o un blocco di dichiarazioni. Se l'espressione è vera, l'istruzione o il blocco di istruzioni viene eseguito, altrimenti viene saltato.
"Dichiarazione "if ... else	Un'istruzione **if** può essere seguita da un'istruzione **else** opzionale, che viene eseguita quando l'espressione è falsa.
Dichiarazione "se... altrimenti se... altrimenti".	L'istruzione **if** può essere seguita da un'istruzione **else if** opzionale, che può essere utilizzata per verificare varie condizioni.
"Dichiarazione "switch case	In modo simile alle istruzioni **if**, **switch...case** controlla il flusso di programmi, consentendo ai programmatori di specificare diversi blocchi che devono essere eseguiti in varie condizioni.
Operatore condizionale "?	Si tratta di un operatore ternario che consente all'utente di effettuare u

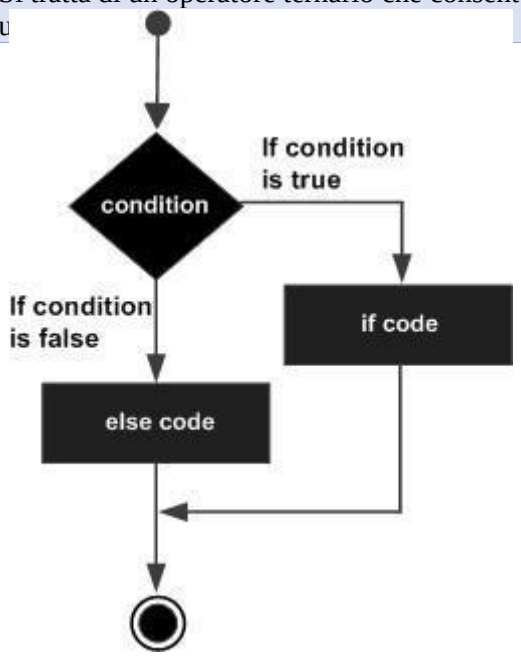


Figura 6 Diagramma di flusso delle istruzioni if

Alcuni operatori di confronto vengono utilizzati per confrontare due valori quando vengono scritte le istruzioni di controllo. Come risultato del confronto, si ottiene un risultato logico di **vero (1)** o **falso (0)**. Si supponga che la variabile portB sia 63 e la portC sia 127,

Operatore nome	Operatore simbolo	Descrizione	Esempio
pari a	= =	Verifica se il valore di due operandi è uguale o meno; in caso affermativo, la condizione diventa vera.	(portaB == portaC) è **falso**
non uguale a	! =	Verifica se il valore di due operandi è uguale o meno; se i valori non sono uguali, la condizione diventa vera.	(portaB != portaC) è **vero**
meno di	<	Verifica se il valore dell'operando di sinistra è inferiore al valore dell'operando di destra; in caso affermativo, allora diventa vera.	(portaB < portaC) è **vero**
maggiore di	>	Verifica se il valore dell'operando di sinistra è maggiore di rispetto al valore dell'operando di destra; in caso affermativo, la condizione diventa vera.	(portaB > portaC) è **falso**
inferiore o uguale a	< =	Verifica se il valore dell'operando sinistro è minore di superiore o uguale al valore dell'operando di destra; in caso affermativo, la condizione diventa vera.	(portaB <= portaC) è **vera**
maggiore o pari a	> =	Verifica se il valore dell'operando di sinistra è maggiore o uguale al valore dell'operando di destra, se sì, allora la condizione diventa vera.	(portaB >= portaC) è **falso**

Materiali richiesti

Arduino UNO R3, software Arduino IDE o portale di simulazione Tinkercad.

Circuito sperimentale

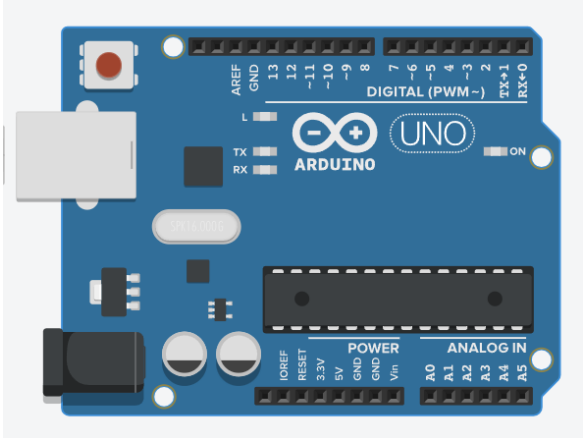


Figura 7 Scheda di sviluppo Arduino UNO R3

Collegare la scheda Arduino al computer tramite un cavo USB di tipo A/B e aprire l'IDE Arduino. Selezionare la porta (Strumenti→Porta) a cui è collegato Arduino. Aprire la schermata Serial Monitor dal menu Strumenti.

Fasi della procedura

1- Impostare il circuito come descritto, assicurandosi che tutti i collegamenti siano sicuri.

2- Scrivere e caricare il codice su Arduino.

3- Osservare la schermata della porta seriale.

4-Osservare le modifiche sullo schermo.

Codici

```
/* Definizione della variabile
globale */ int portB = 63;
int portC = 127;

void setup () {
  Serial.begin(9600);
  Serial.println(portB == portC);delay(1000);
  Serial.println(portB != portC);delay(1000);
  Serial.println(portB < portC);delay(1000);
  Serial.println(portB > portC);delay(1000);
  Serial.println(portB <= portC);delay(1000);
  Serial.println(portB >= portC);
}

void loop () {
  /* se il blocco */
  if(portB < portC) /* se la condizione è vera, eseguire la seguente istruzione*/
  portB++;
  ritardo(200);
  Serial.print("PORTB= ");Serial.println(portB);

  /* blocco if ... else */
  se(portaB == portaC){
    Serial.println("La portaB è uguale alla
    portaC"); portB++;
  }
  else{
    Serial.println("La portaB non è uguale alla portaC");
  }

  /* blocco if ... else if ... else */
  se(portaB > portaC){
    Serial.println("La portaB è maggiore della portaC");
  }
  else if (portB < portC){
    Serial.println("PortB è inferiore a
    PortC");
  }
  else{
    Serial.println("La portaB è uguale alla portaC");
  }

}
```

Spiegazione del Codice

Impostazione: Il monitor seriale è attivato alla velocità di 9600 baud. Per le variabili definite in precedenza, le relative operazioni di confronto vengono eseguite una sola volta.

Loop: Ciclo infinito che gestisce blocchi di codice all'interno in modo continuo.

Numero di esperimenti: 11

11- Strutture di controllo in Arduino IDE: come utilizzare le strutture di controllo-2

Obiettivo dell'esperimento

L'obiettivo di questo esperimento è dimostrare come vengono utilizzate le strutture di controllo nell'ambiente di sviluppo integrato (IDE) Arduino. Gli studenti impareranno a utilizzare **l'istruzione *switch case*** e l'***operatore condizionale*** "?

Contesto teorico

Simile alle istruzioni if, **switch...case** controlla il flusso dei programmi consentendo ai programmatori di specificare diversi codici che devono essere eseguiti in varie condizioni. In particolare, un'istruzione **switch** confronta il valore di una variabile con i valori specificati nelle istruzioni **case.**

Alla fine di ogni istruzione **case** viene scritta la parola chiave **break**. Altrimenti, ogni istruzione **switch** viene eseguita indipendentemente dalla condizione dell'istruzione **case**. Una parola chiave **default** viene utilizzata per l'esecuzione se non c'è corrispondenza tra i casi nello switch.

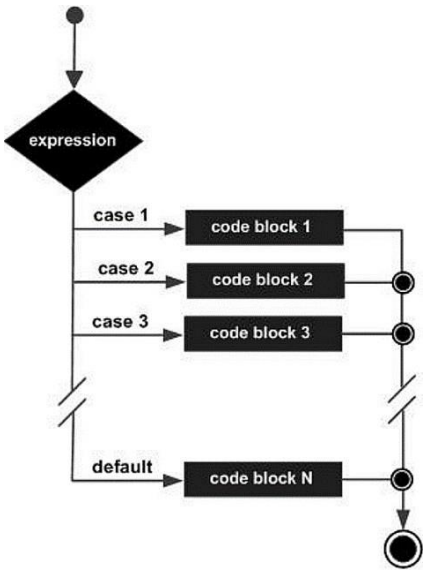


Figura 8 Diagramma di flusso dell'istruzione switch case

Materiali richiesti

Arduino UNO R3, software Arduino IDE o portale di simulazione Tinkercad.

Circuito sperimentale

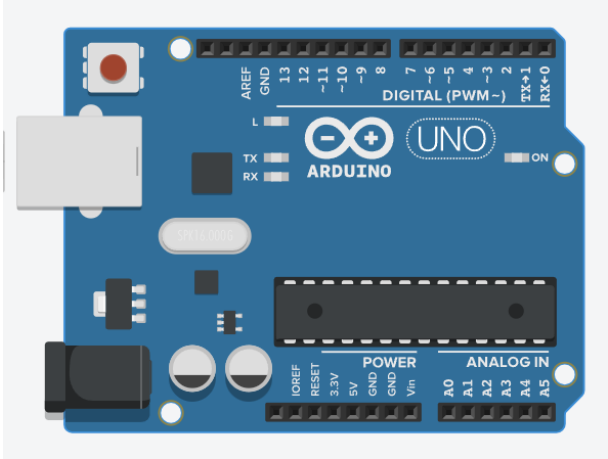


Figura 9 Scheda di sviluppo Arduino UNO R3

Collegare la scheda Arduino al computer tramite un cavo USB di tipo A/B e aprire l'IDE Arduino. Selezionare la porta (Strumenti→Porta) a cui è collegato Arduino. Aprire la schermata Serial Monitor dal menu Strumenti.

Fasi della procedura

1-Impostare il circuito come descritto, assicurandosi che tutti i collegamenti siano sicuri.

2-Scrivere e caricare il codice su Arduino.

3-Osservare la schermata della porta seriale.

4-Osservare le modifiche sullo schermo.

Codici - 1

```
/* Definizione variabile globale
*/ byte portC = 1;
bool left = true;
void setup(){
Serial.begin(9600);
}
void loop(){
 switch (portC) {
  caso 1: Serial.println("Il primo LED è in funzione.");
  break; caso 2: Serial.println("Il secondo LED è in
  funzione."); break; caso 4: Serial.println("Il terzo LED è
  in funzione."); break; caso 8: Serial.println("Il quarto
  LED viene azionato."); break; caso 16: Serial.println("Il
  quinto LED viene azionato."); break; caso 32:
  Serial.println("Il sesto LED viene azionato."); break; case
  64: Serial.println("Il settimo LED viene azionato."); break;
  case 128: Serial.println("Viene azionato l'ottavo LED.");
  break; default: Serial.println("Stato non valido!");
 }
 se(sinistra)
   portC = portC << 1; // Operatore di spostamento dei bit. Tutti i bit vengono spostati a sinistra una volta.
 altro
   portC = portC >> 1; // Operatore di spostamento dei bit. Tutti i bit vengono spostati a destra una volta.
 if(portC == 0 && left){ // Confronto multiplo con il blocco if
   portC = 128;
   left = false;
   Serial.println("**********************");
 }
```

```
else if(portC == 0 && !left){ // Confronto multiplo con else if blocco
  portC = 1;
```

```
    sinistra = vero;
    Serial.println("**********************");
  }
  ritardo(1000);
}
```

Spiegazione del Codice

Impostazione: Il monitor seriale è attivato alla velocità di 9600 baud.

Loop: Viene azionato un blocco di casi di commutazione. La variabile booleana viene modificata in base al valore massimo o minimo del bit, quindi la scansione viene eseguita da sinistra a destra o da destra a sinistra in modo continuo.

Codici - 2

```
/* Definizione della variabile
globale */ int portB = 63;
int portC = 127;
char c = 'k';

void setup() {
  Serial.begin(9600);

  /* Trova max(portaB, portaC): */
  Serial.println((portB>portC)?portB:portC);/*Il valore della portaB viene stampato se è maggiore della portaC.
Altrimenti viene stampato il valore della portaC.*/

  /* Convertire la lettera minuscola in maiuscola: */
  c = ( c >= 'a' && c <= 'z' ) ? ( c - 32 ) : c;
  Serial.println(c);
}

void loop() {

}
```

Spiegazione del Codice

Impostazione: Il monitor seriale è attivato alla velocità di 9600 baud. L'operatore condizionale viene utilizzato per determinare quale ingresso è maggiore di un altro. Anche la conversione di una determinata lettera minuscola in maiuscola in base al valore della tabella dei caratteri ASCII può essere eseguita facilmente con l'operatore condizionale.

Loop: -

Numero di esperimenti: 12

12- Strutture di controllo in Arduino IDE: come utilizzare le strutture di controllo-3

Obiettivo dell'esperimento

L'obiettivo di questo esperimento è dimostrare come vengono utilizzate le strutture di controllo nell'ambiente di sviluppo integrato (IDE) Arduino. Gli studenti impareranno a utilizzare i cicli ***for***, ***while*** e ***do while***.

Contesto teorico

L'istruzione **for** viene utilizzata per ripetere un blocco di istruzioni racchiuse tra parentesi graffe. Di solito si usa un contatore di incremento per incrementare e terminare il ciclo. L'istruzione **for** è utile per qualsiasi operazione ripetitiva. Questa struttura è spesso preferita, soprattutto nella scansione dei pin delle porte delle schede Arduino. L'intestazione del ciclo ***for è composta da*** tre parti:

```
for (inizializzazione; condizione; incremento) {

//dichiarazione/i;

}
```

I cicli "***while***" eseguono un ciclo continuo, all'infinito, finché l'espressione all'interno della parentesi () non diventa falsa. Il ciclo viene eseguito finché la condizione è soddisfatta. Per questo motivo, deve essere usato con attenzione per evitare loop infiniti.

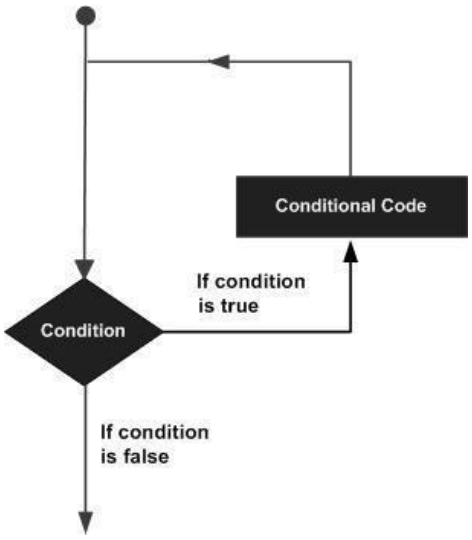


Figura 10 Funzionamento dei loop

Il ciclo "***do*** ... ***while***" è simile al ciclo ***while***, ma viene eseguito almeno una volta, indipendentemente dal fatto che la condizione sia soddisfatta o meno.

Materiali richiesti

Arduino UNO R3, 6 resistenze da 220 Ω, 6 LED, software Arduino IDE o portale di simulazione Tinkercad.

Circuito sperimentale

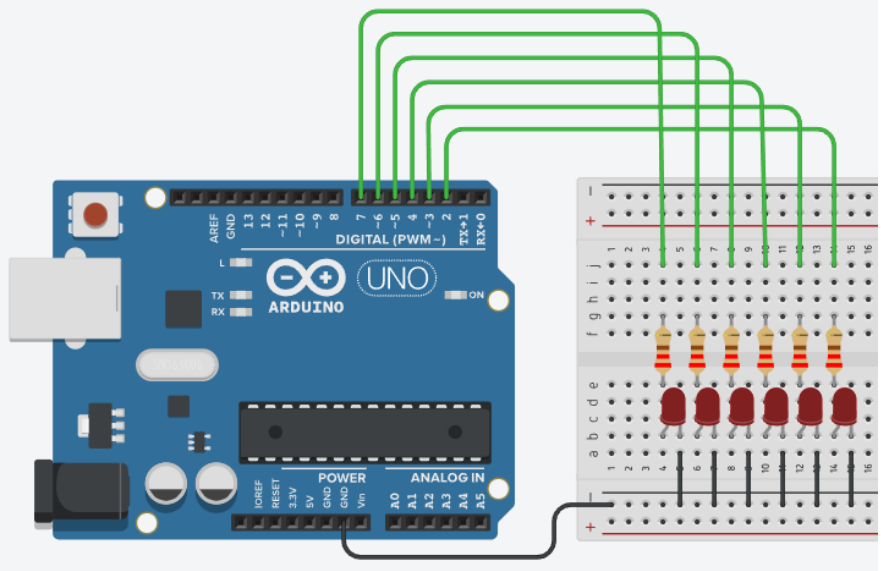


Figura 11 Scheda di sviluppo Arduino UNO R3

Collegare la scheda Arduino al computer tramite un cavo USB di tipo A/B e aprire l'IDE Arduino. Selezionare la porta (Strumenti→Porta) a cui è collegato Arduino. Aprire la schermata Serial Monitor dal menu Strumenti.

Fasi della procedura

1-Impostare il circuito come descritto, assicurandosi che tutti i collegamenti siano sicuri.

2-Scrivere e caricare il codice su Arduino.

3-Osservare il comportamento dei LED.

4-Osservare le modifiche sulla schermata della porta seriale.

Codici

```
// Variabili globali
// sono definiti gli array a due dimensioni
int sensors01[3][3]={3,-7,7,8,5,-4,2,3,0};
int sensors02[3][3]={4,3,0,2,-1,-4,8,9,5};
int sensore3=50;
int
sensore4=100;
int contatore =
1; void setup () {
  Serial.begin(9600);
}

 void loop () {
  PORTD = 4;
  ritardo(250);
  // inizia il ciclo for
  for(int i=0;i<6;i++){
    PORTD = PORTD << 1;
  ritardo(250);
  }
  PORTD = 128;
  // Inizia un altro ciclo for
  for(int i=0;i<6;i++){
```

```
PORTD = PORTD >> 1;
```

```
  ritardo(250);
 }

 //ciclo for annidato
 for(int i=0;i<=2;i++){//i variabile
   for(int j=0;j<=2;j++){// variabile j
     Serial.print(sensors01[i][j] + sensors02[i][j]);
     Serial.print("\t");//operatore di tabulazione
   }
   Serial.println();/Ritorno a capo
 }

 // ciclo while
 while(sensore3 <
 sensore4){
 Serial.print("Ho eseguito
 "); Serial.print(counter);
 Serial.println(". volte.");
 counter++;
   sensor3 += 10; // operatore di addizione unario
   delay(200);
 }

 //fare... mentre il ciclo
 fare{
   ritardo(200);
   Serial.println("Eseguo almeno una volta...");
 }while (sensore3 > sensore4);
}
```

Spiegazione del Codice

Impostazione: Il monitor seriale è attivato alla velocità di 9600 baud.

Loop: Vengono eseguiti due cicli ***for*** per pilotare i LED sulla PORTD della scheda Arduino. Il ciclo ***for*** annidato viene eseguito per mostrare come eseguire la scansione di array a due dimensioni. I cicli "***while***" e "***do...while***" vengono eseguiti per verificare il confronto di due valori.

Numero di esperimenti: 13

Nome dell'esperimento: Strutture di controllo in Arduino IDE: come usare le strutture di controllo 4 Obiettivo dell'esperimento

L'obiettivo di questo esperimento è dimostrare come vengono utilizzate le funzioni definite dall'utente e quelle incorporate nell'ambiente di sviluppo integrato (IDE) di Arduino. Gli studenti impareranno a dichiarare le funzioni definite dall'utente e a utilizzare alcune importanti funzioni incorporate.

Contesto teorico

Come avete sperimentato finora, abbiamo utilizzato alcune funzioni di base integrate nell'IDE Arduino, come setup(), loop() e delay(). Una funzione consente all'utente di strutturare i programmi in segmenti di codice per eseguire singoli compiti. L'ambiente di sviluppo Arduino richiede due funzioni principali, setup() e loop().

La sintassi più comune per definire una funzione è la seguente:

```
return_type nome_funzione(argomento1, argomento2, ...) {
    dichiarazioni;
    restituire tipo_valore;
}
```

return_type è un tipo di dati come, ad esempio, integer, float, char e così via. Non tutte le funzioni richiedono la restituzione di un valore. In tal caso, il valore *return_type* della funzione è definito come **void**. Gli argomenti sono qualsiasi variabile o valore con un tipo di dati adeguato. Il tipo di dati di un argomento deve corrispondere al tipo definito.

Una funzione viene dichiarata al di fuori di qualsiasi altra funzione, sopra o sotto le funzioni *loop* e *setup*. Se si scrive solo la funzione stessa, si deve dichiarare la funzione prima delle funzioni di *loop* o di *setup* (dove viene chiamata).

Se si desidera dichiarare una funzione dopo le funzioni di setup o di loop, è necessario scrivere un **prototipo di funzione** sopra le altre funzioni in cui viene chiamata. Il prototipo di funzione deve essere seguito da un punto e virgola (;).

Materiali richiesti

Arduino UNO R3, software Arduino IDE o portale di simulazione Tinkercad.

Circuito sperimentale

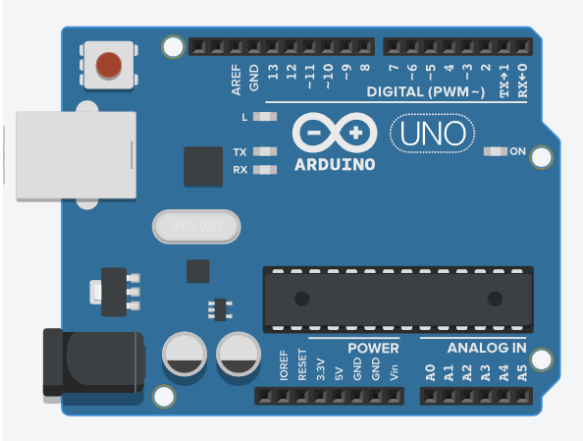


Figura 12 Scheda di sviluppo Arduino UNO R3

Collegare la scheda Arduino al computer tramite un cavo USB di tipo A/B e aprire l'IDE Arduino. Selezionare la porta (Strumenti→Porta) a cui è collegato Arduino. Aprire la schermata Serial Monitor dal menu Strumenti.

Fasi della procedura

1-Impostare il circuito come descritto, assicurandosi che tutti i collegamenti siano sicuri.

2-Scrivere e caricare il codice su Arduino.

3-Osservare la schermata della porta seriale.

4-Osservare le modifiche sullo schermo.

Codici

```
// Variabili globali
int sum_func(int x, int y);// un prototipo di funzione definito
dall'utente int sensor1=21;
int sensore2=45;

// Dichiarazione di una funzione definita dall'utente prima
che venga chiamata void printMessage(){
  Serial.println("Questa funzione è dichiarata sopra la funzione setup()");
  Serial.print("La differenza dei sensori è ");
  Serial.println(sensore1-sensore2);
}

void setup () {
  Serial.begin(9600);
  printMessage(); //viene richiamata la funzione definita dall'utente
}

void loop () {
  if(sum_func(sensore1, sensore2)>=50){
    Serial.print("La somma dei sensori è, ");
    Serial.println(sum_func(sensore1, sensore2));
  }
  else{
    Serial.println("La somma dei sensori è inferiore a 50");
  }
  sensore2 = 25;
  delay(1000); // I dati del sensore vengono letti ogni secondo.
```

```
}

// Declerazione della funzione dopo loop() dove viene
chiamata. int sum_func(int x, int y){
  int z = 0;
  z = x + y;
  restituir
  e z;
}
```

Spiegazione del Codice

Impostazione: Il monitor seriale è attivato alla velocità di 9600 baud. Viene richiamata una funzione definita dall'utente.

Loop: Viene richiamata una funzione di somma definita dall'utente. Viene eseguita una scansione ogni secondo.

Numero di esperimenti: 14

14- Stringhe in Arduino IDE: come utilizzare i letterali di stringa

Obiettivo dell'esperimento

L'obiettivo di questo esperimento è dimostrare come il tipo di dati stringa viene utilizzato nell'ambiente di sviluppo integrato (IDE) Arduino. Gli studenti impareranno i diversi metodi di dichiarazione del tipo di dati stringa.

Contesto teorico

Come in tutti gli altri linguaggi di programmazione, anche in Arduino IDE esiste un importante tipo di dati chiamato "stringa" sul quale non è possibile eseguire operazioni matematiche. Le stringhe sono un tipo di dati basato sul testo. Le stringhe di testo possono essere rappresentate in due modi. Si può utilizzare il tipo di dati ***String***, oppure si può creare una stringa da un array di tipo ***char*** e null-terminare ('\0').

- Sintassi delle stringhe

Tutte le seguenti dichiarazioni sono valide per le

stringhe. char Str1[10];
char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
char Str4[] = "arduino";
char Str5[7] = "arduino";
char Str6[10] = "arduino";

- Possibilità di dichiarare le stringhe
  - Dichiarare un array di caratteri senza inizializzarlo come in Str1
  - Dichiarare un array di caratteri (con un carattere extra) e il compilatore aggiungerà il carattere null richiesto, come in Str2
  - Aggiungere esplicitamente il carattere null, Str3
  - Inizializza con una costante di stringa tra virgolette; il compilatore dimensionerà l'array per adattarlo alla costante di stringa e a un carattere nullo finale, Str4.
  - Inizializza l'array con una dimensione esplicita e una costante di stringa, Str5
  - Inizializza l'array, lasciando dello spazio in più per una stringa più grande, Str6
- Terminazione nulla

In genere, le stringhe vengono terminate con un carattere null (codice ASCII 0). Questo permette alle funzioni (come **Serial.print()**) di capire dove si trova la fine di una stringa. Altrimenti, continuerebbero a leggere byte di memoria successivi che non fanno effettivamente parte della stringa.

- Virgolette singole o doppie

Le stringhe sono sempre definite all'interno di doppi apici ("Arduino") e i caratteri sono sempre definiti all'interno di apici singoli ('A').

- Avvolgimento di corde lunghe

È possibile avvolgere lunghe stringhe in questo modo:

char myString[] = "Questa è la prima riga".
" questa è la seconda
riga" " questa è l'ultima

riga";

- Array di stringhe

Gli array di stringhe sono utilizzati quando si lavora con grandi quantità di testi di messaggi, ad esempio nei progetti con display LCD. È possibile stampare gli stessi messaggi in diverse parti del progetto. In questi casi, sarebbe logico preparare una serie di messaggi.

```
char *sensorStrings[] = {"Questo è il primo sensore", "Questo è il primo sensore", "Questo è il primo sensore".
                         secondo sensore", "Questo è il terzo sensore", "Questo è il
                         quarto sensore", "Questo è il quinto sensore", "Questo è il
                         sesto sensore6".
                };
```

Nel codice sottostante, l'asterisco dopo il tipo di dato char "**char***" indica che si tratta di un array di "puntatori".

- Concatenazione di stringhe

Le stringhe create con il tipo di dati **char** non possono essere concatenate con l'operatore aritmetico. Per questa operazione è necessario utilizzare la struttura dati **String**.

Materiali richiesti

Arduino UNO R3, software Arduino IDE o portale di simulazione Tinkercad.

Circuito sperimentale

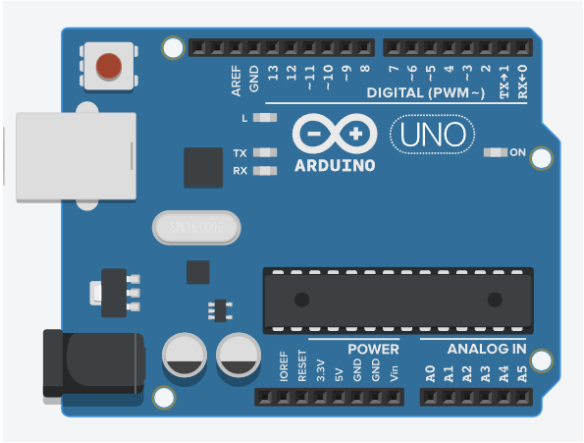


Figura 13 Scheda di sviluppo Arduino UNO R3

Collegare la scheda Arduino al computer tramite un cavo USB di tipo A/B e aprire l'IDE Arduino. Selezionare la porta (Strumenti→Porta) a cui è collegato Arduino. Aprire la schermata Serial Monitor dal menu Strumenti.

Fasi della procedura

1-Impostare il circuito come descritto, assicurandosi che tutti i collegamenti siano sicuri.

2-Scrivere e caricare il codice su Arduino.

3-Osservare la schermata della porta seriale.

4-Osservare le modifiche sullo schermo.

Codici

```
// Variabili globali
char Str1[10]; // inizializzato ma senza assegnazione del primo valore
char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'}; /* inizializzato con l'assegnazione del primo valore, la terminazione null
```

```
viene aggiunta automaticamente */
```

```
char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'}; /* inizializzato con l'assegnazione del primo valore, la terminazione
nulla viene aggiunta manualmente */
char Str4[] = "arduino"; //la dimensione dell'array viene creata dal
valore assegnato. char Str5[8] = "arduino"; //la dimensione dell'array e
l'assegnazione vengono effettuate insieme. char Str6[10] = "arduino";
//la dimensione dell'array è maggiore del valore assegnato. char
myString[] = "Questa è la prima riga"
" questa è la seconda
riga" " questa è l'ultima
riga";

char *sensorStrings[] = {"Questo è il primo sensore", "Questo è il secondo sensore", "Questo è il terzo sensore",
"Questo è il quarto sensore", "Questo è il quinto sensore", "Questo è il sesto sensore6"
                };

void setup () {
  Serial.begin(9600);
  Serial.println(Str1);delay(500);
  Serial.println(Str2);delay(500);
  Serial.println(Str3);delay(500);
  Serial.println(Str4);delay(500);
  Serial.println(Str5);delay(500);
  Serial.println(Str6);delay(500);
  Serial.println(myString);
}

void loop () {
  for (int i = 0; i < 6; i++) {
    Serial.println(sensorStrings[i]);
    delay(500);
  }
}
```

Spiegazione del Codice

Impostazione: Il monitor seriale è attivato alla velocità di 9600 baud. Vengono stampati diversi letterali di stringa.

Loop: Un array di stringhe viene stampato con la tecnica della scansione utilizzando il ciclo for.

Numero di esperimenti: 15

15- Stringhe in Arduino IDE: come utilizzare la struttura dati Stringa

Obiettivo dell'esperimento

L'obiettivo di questo esperimento è dimostrare come l'oggetto String() (struttura dati) viene utilizzato nell'ambiente di sviluppo integrato (IDE) Arduino. Gli studenti impareranno i diversi metodi di dichiarazione dell'oggetto String().

Contesto teorico

L'IDE Arduino utilizza una struttura di programmazione orientata agli oggetti e sono molti i tipi di oggetto utilizzati nelle piattaforme Arduino. L'oggetto String() è uno dei più popolari. String() è un tipo di struttura dati di classe. Esistono diverse versioni per costruire un'istanza della classe String.

- una stringa costante di caratteri, tra doppi apici (cioè un array di caratteri)
- un singolo carattere costante, tra virgolette singole
- un'altra istanza dell'oggetto String
- un numero intero costante o un numero intero lungo
- un numero intero costante o un numero intero lungo, utilizzando una base specificata
- una variabile intera o un numero intero lungo
- una variabile di tipo intero o intero lungo, utilizzando una base specificata
- un float o un double, utilizzando le cifre decimali specificate

Costruendo una stringa a partire da un numero, si ottiene una stringa che contiene la rappresentazione ASCII di quel numero. L'impostazione predefinita è la base dieci, quindi

```
String thisString = String(15);
```

dà la stringa "15". È tuttavia possibile utilizzare altre basi. Ad esempio,

```
String thisString = String(15, HEX);
```

fornisce la stringa "f", che è la rappresentazione esadecimale del valore decimale 15. O, se preferite, in binario,

```
Stringa thisString = String(15, BIN);
```

fornisce la stringa "1111", che è la rappresentazione binaria di 15.

- **Sintassi**

```
String(val)
String(val, base)
String(val, decimalPlaces)
```

- **Parametri**

val: una variabile da formattare come stringa. Tipi di dati consentiti: ***string***, ***char***, ***byte***, ***int***, ***long***, ***unsigned int***, ***unsigned long***, ***float***, ***double***.
base: (opzionale) la base in cui formattare un valore integrale.
decimalPlaces: solo se val è float o double. Le cifre decimali desiderate.

- Concatenazione di stringhe

Affinché le stringhe possano essere sommate, devono essere definite come oggetto String e devono ottenere un valore iniziale prima di iniziare a concatenare tipi di dati diversi.

Materiali richiesti

Arduino UNO R3, software Arduino IDE o portale di simulazione Tinkercad.

Circuito sperimentale

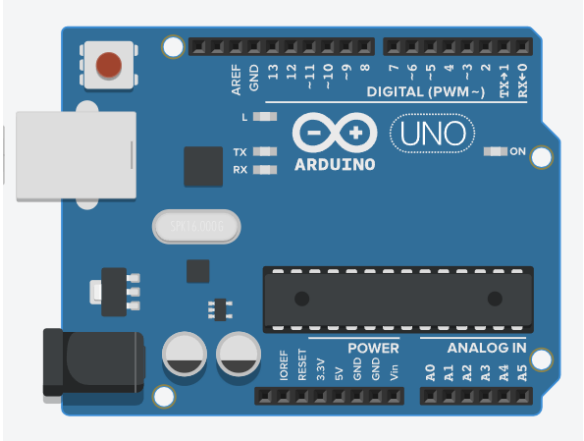


Figura 14 Scheda di sviluppo Arduino UNO R3

Collegare la scheda Arduino al computer tramite un cavo USB di tipo A/B e aprire l'IDE Arduino. Selezionare la porta (Strumenti→Porta) a cui è collegato Arduino. Aprire la schermata Serial Monitor dal menu Strumenti.

Fasi della procedura

1-Impostare il circuito come descritto, assicurandosi che tutti i collegamenti siano sicuri.

2-Scrivere e caricare il codice su Arduino.

3-Osservare la schermata della porta seriale.

4-Osservare le modifiche sullo schermo.

Codici

```
// Variabili globali
intSensor = 15;
floatSensor = 13,495;
String string = "Hello String"; // uso di una stringa costante
String stringCharacter = String('a'); // conversione di un carattere costante in una stringa
String stringObject = String("Questa è una stringa"); // conversione di una stringa costante in un
oggetto String StringConcatenation = String(stringa + " con altro"); // concatenazione di due
stringhe String stringInteger = String(15); // utilizzo di un intero costante
String stringDEC = String(intSensor, DEC); // usando un int e una base
String stringHEX = String(intSensor, HEX); // utilizzando un int e una base
(esadecimale) String stringBIN = String(intSensor, BIN); // utilizzando un int e una base
(binaria)
Stringa stringLongDEC = String(millis(), DEC);// usando un long e una base
String stringFloat = String(floatSensor, 2);// utilizzo di un float con precisione a due decimali

String stringOne, stringTwo, stringThree;
void setup () {
  Serial.begin(9600);
}
```

```
void loop () {
  Serial.println(stringa);delay(500);
```

```
  Serial.println(stringCharacter);delay(500);
  Serial.println(stringObject);delay(500);
  Serial.println(stringConcatenation);delay(500);
  Serial.println(stringInteger);delay(500);
  Serial.println(stringDEC);delay(500);
  Serial.println(stringHEX);delay(500).println(st
  ringHEX);delay(500);
  Serial.println(stringBIN);delay(500);
  Serial.println(stringLongDEC);delay(500);
  Serial.println(stringFloat);delay(500);
  stringLongDEC = String(millis(), DEC);
  Serial.println(stringLongDEC);

  stringOne = String("Hai aggiunto
  "); stringTwo = String("questa
  stringa"); stringThree = String();
  // aggiunta di un numero intero costante a una
  stringa: stringThree = stringOne + 123;
  Serial.println(stringThree);delay(500);
  // aggiunta di un intero lungo costante a una
  stringa: stringThree = stringOne + 123456789;
  Serial.println(stringThree);delay(500);
  // aggiunta di un carattere costante a una
  stringa: stringThree = stringOne + 'A';
  Serial.println(stringThree);delay(500);
  // aggiunta di una stringa costante a una stringa:
  stringThree = stringOne + "abc";
  Serial.println(stringThree);delay(500);
  stringThree = stringOne + stringTwo;
  Serial.println(stringThree);delay(500);
  // aggiunta di una variabile intera a una
  stringa: int sensorValue =
  analogRead(A0); stringOne = "Valore
  del sensore: "; stringThree = stringOne
  + sensorValue;
  Serial.println(stringThree);delay(500);
  // aggiunta di una variabile intera lunga a una
  stringa: stringOne = "valore millis(): ";
  stringThree = stringOne + millis();
  Serial.println(stringThree);delay(500);
  while(true); //ciclo infinito, non è stato fatto
  nulla.
}
```

Spiegazione del Codice

Impostazione: Il monitor seriale è attivato alla velocità di 9600 baud.

Loop: Viene richiamata una funzione di somma definita dall'utente. Viene eseguita una scansione ogni secondo.

Numero di esperimenti: 16

16- Operazioni di I/O digitale

Obiettivo dell'esperimento

L'obiettivo principale di questo esperimento è introdurre gli studenti alle operazioni di base di input/output (I/O) digitale utilizzando Arduino. I partecipanti impareranno a configurare e utilizzare i pin digitali della scheda Arduino per leggere gli ingressi (come i pulsanti) e controllare le uscite (come i LED).

Contesto teorico

La piattaforma Arduino è dotata di pin digitali che possono essere configurati come ingressi o uscite. Questi pin digitali sono utilizzati per leggere segnali digitali e controllare dispositivi fisici. Quando sono configurati come ingressi, possono rilevare la presenza o l'assenza di un segnale (ALTO o BASSO). Quando sono configurati come uscite, possono impostare il pin su ALTO (di solito 5V) o BASSO (0V), consentendo di controllare dispositivi come LED, motori, ecc.

Ingresso digitale: Lettura dello stato di un segnale digitale. Si usa comunemente con pulsanti, interruttori e sensori.

Uscita digitale: Impostazione di un pin digitale su HIGH o LOW per controllare dispositivi come LED o relè.

Le funzioni pinMode(), digitalWrite() e digitalRead() di Arduino sono fondamentali per le operazioni di I/O digitale:

pinMode(pin, modalità): Imposta un pin come INPUT, OUTPUT o

INPUT_PULLUP. digitalWrite(pin, valore): Scrive un valore HIGH o LOW su

un pin digitale. digitalRead(pin): Legge il valore da un pin digitale.

Materiali richiesti

Scheda Arduino UNO R3

Scheda per il pane

LED

Resistenza da 220

ohm Interruttore a

pulsante Fili di

ponticello Software

Arduino IDE

Circuito sperimentale

1-Collegare il LED a uno dei pin digitali attraverso una resistenza da 220 ohm per limitare la corrente.

2-Collegare un lato del pulsante a un altro pin digitale e l'altro lato a massa. Utilizzare una resistenza di pull-up o la modalità interna INPUT_PULLUP per garantire un segnale ALTO stabile quando il pulsante non è premuto.

Fasi della procedura

1- Assemblare il circuito come descritto, assicurandosi che tutti i collegamenti siano sicuri.

2- Scrivere il codice Arduino per controllare il LED con il pulsante.

3- Caricare il codice sulla scheda Arduino.

4- Premere e rilasciare il pulsante per testare il controllo del LED.

5- Osservare la risposta del LED al pulsante.

Codice di esempio

```
// Definire i numeri dei pin
        const int buttonPin = 2; // il numero del pin del pulsante const
        int ledPin = 13; // il numero del pin del LED
// Le variabili cambieranno:
        int buttonState = 0;     //variabile per la lettura dello stato del
        pulsante void setup() {
  // inizializzare il pin del LED come uscita:
          pinMode(ledPin, OUTPUT);
  // inizializzare il pin del pulsante come ingresso:
  pinMode(buttonPin, INPUT);
}
void loop() {
  // leggere lo stato del valore del pulsante:
  buttonState = digitalRead(buttonPin);
  // controlla se il pulsante è premuto.
  // se lo è, il buttonState è HIGH: if
  (buttonState == HIGH) {
  // Accensione del LED:
    digitalWrite(ledPin, HIGH);
  } else {
    // spegnere il LED:
    digitalWrite(ledPin, LOW);
  }
}
```

Spiegazione del Codice

- Configurazione: La funzione pinMode() configura il pin del LED come uscita e il pin del pulsante come ingresso.
- Loop: Il programma legge continuamente lo stato del pulsante utilizzando digitalRead(). Se il pulsante è premuto (lettura HIGH grazie alla configurazione pull-up), il LED si accende. Altrimenti, il LED viene spento. Lo stato del LED cambia in base alla pressione del pulsante.

Questo esperimento dimostra le operazioni di I/O digitale di base, che sono fondamentali per progetti Arduino più complessi che coinvolgono sensori, motori e altri componenti.

Numero di esperimenti: 17

17- Operazioni di I/O analogico

Obiettivo dell'esperimento

L'obiettivo di questo esperimento è quello di familiarizzare gli studenti con il concetto e l'applicazione delle operazioni di ingresso e uscita analogica utilizzando una scheda Arduino. I partecipanti impareranno a leggere i segnali analogici provenienti da sensori e a controllare dispositivi analogici come LED dimmerabili o motori con la modulazione di larghezza di impulso (PWM).

Contesto teorico

A differenza dei segnali digitali che sono o ALTI o BASSI, i segnali analogici possono rappresentare una gamma di valori. Le schede Arduino possono leggere i segnali analogici utilizzando convertitori analogico-digitali (ADC) e produrre uscite analogiche utilizzando la modulazione di larghezza di impulso (PWM).

Ingresso analogico: Le schede Arduino, come la Uno, dispongono di una serie di pin di ingresso analogici, contrassegnati da A0 ad A5. Questi pin possono leggere i segnali analogici provenienti da sensori, come la temperatura o la luce, e convertirli in un valore digitale che può essere elaborato dal microcontrollore.

Uscita analogica (PWM): Sebbene Arduino non sia in grado di generare un'uscita analogica vera e propria, può simularla attraverso il PWM. Il PWM controlla il duty cycle relativo di un segnale digitale per variare la potenza erogata a dispositivi come LED o motori.

Funzioni chiave per le operazioni analogiche:

- analogRead(pin): Legge il valore di un pin analogico e restituisce un valore compreso tra 0 (0V) e 1023 (5V).
- analogWrite(pin, valore): Scrive un valore analogico (onda PWM) su un pin per simulare l'uscita analogica. Il valore può essere compreso tra 0 (sempre spento) e 255 (sempre acceso).

Materiali richiesti

Scheda Arduino UNO R3

Lavagna per pane

Potenziometro (ad

esempio, 10kΩ) LED

Resistenza da 220

ohm Fili di ponticello

Software Arduino

IDE

Circuito sperimentale

1-Collegare il potenziometro a uno dei pin di ingresso analogico (ad esempio, A0). Collegare un lato a 5 V, il pin centrale ad A0 e l'altro lato a GND.

2-Collegare il LED a un pin digitale in grado di eseguire il PWM (ad esempio, 9, 10, 11 su Uno) attraverso una resistenza da 220 ohm per limitare la corrente.

Fasi della procedura

1-Impostare il circuito come descritto, assicurandosi che tutti i collegamenti siano sicuri.

2-Scrivere e caricare il codice Arduino che legge il valore analogico dal potenziometro e controlla di conseguenza la luminosità del LED.

3-Ruotare il potenziometro e osservare la variazione di luminosità del LED.

4-Comprendere la relazione tra la posizione del potenziometro e la luminosità del LED.

Codice di esempio

```
// Definire i numeri dei pin
const int potPin = A0; // il numero del pin del potenziometro
const int ledPin = 9; // il numero del pin del LED
// Variabile per memorizzare il valore del
potenziometro int potValue = 0;
void setup() {
  // inizializzare il pin del LED come uscita:
  pinMode(ledPin, OUTPUT);
  // inizializzare la comunicazione seriale a 9600 bit al
  secondo: Serial.begin(9600);
}
void loop() {
  // leggere il valore dal potenziometro:
```

```
  potValue = analogRead(potPin);
  // mappare il valore del potenziometro da 0-1023 a 0-
  255: int ledBrightness = map(potValue, 0, 1023, 0,
  255);
  // impostare la luminosità del LED:
  analogWrite(ledPin, ledBrightness);
  // stampare i risultati sul monitor seriale:
  Serial.print("Potenziometro: ");
  Serial.print(potValue);
  Serial.print("Luminosità LED: ");
  Serial.println(ledBrightness);
  delay(10); // piccolo ritardo per
  stabilità
}
```

Spiegazione del Codice

Impostazione: Inizializza il pin LED come uscita e avvia la comunicazione seriale per il monitoraggio.

Loop: La funzione analogRead() legge il valore analogico dal potenziometro, che viene poi mappato in un valore PWM adeguato (0-255). Questo valore viene utilizzato in analogWrite() per impostare la luminosità del LED. I valori del potenziometro e del LED vengono stampati sul monitor seriale per essere osservati.

Questo esperimento dimostra come leggere gli ingressi analogici e produrre uscite PWM, essenziali per interfacciarsi con un'ampia gamma di sensori e controllare vari attuatori in progetti Arduino più complessi.

INA-CODE

robyKode

web & mobile