Electronic Prototyping

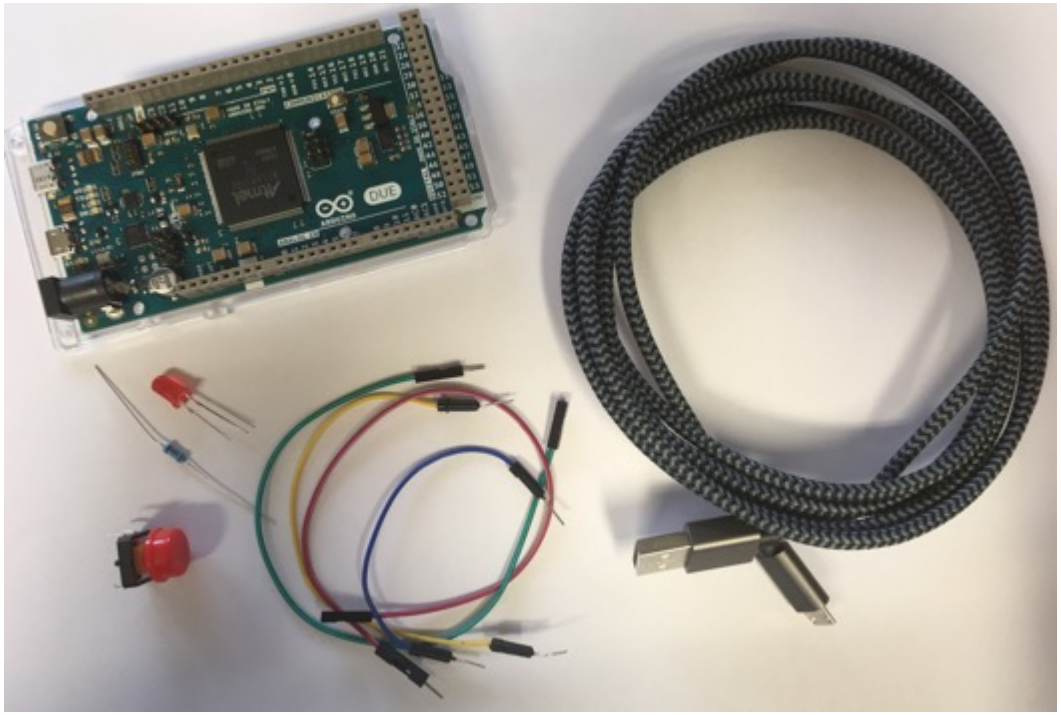
BUTTON and LED

Lesson 4

Turn on a led with a button

What are we going to use?

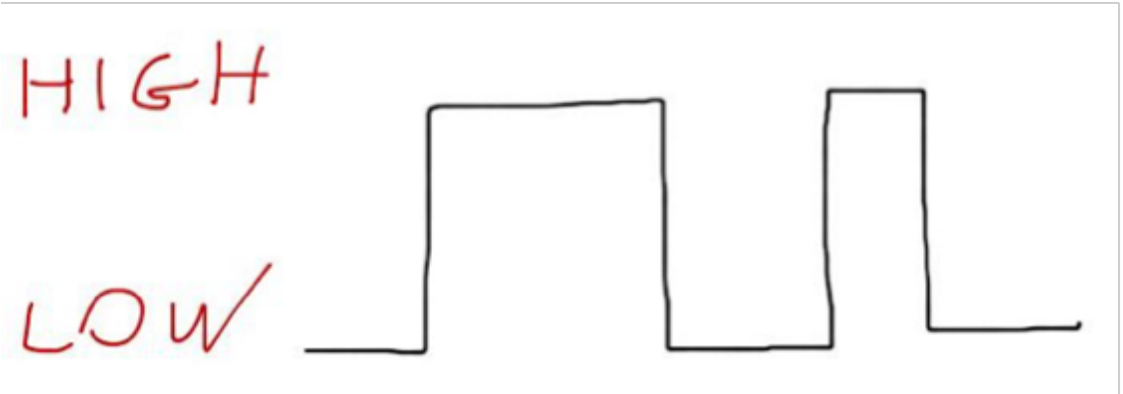
- Arduino DUE
- Breadboard
- USB Cable
- LED
- Button
- 10k resistor
- 220 resistor
- cables



Interruttori e pulsanti

Interruttori e pulsanti consentono di effettuare interruzioni e connessioni del passaggio di corrente all' interno di un circuito, ma Arduino per comprendere che un componente o un suo piedino è connesso o non connesso ha necessità di leggere una tensione elettrica e nello specifico:

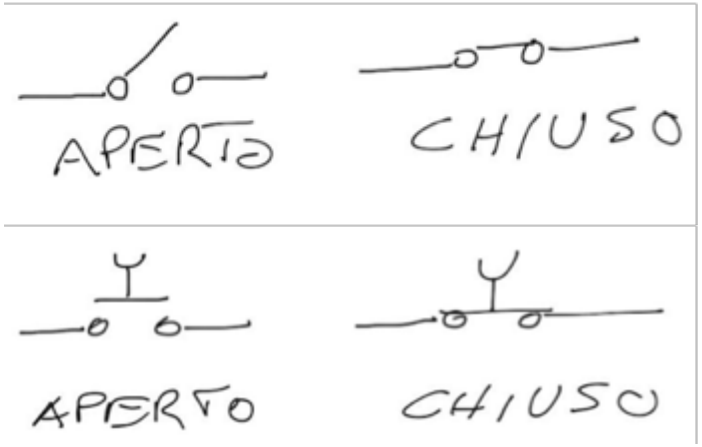
- livello logico **ALTO** > **HIGH** > equivalente a 5 volt = Vcc
- livello logico **BASSO** > **LOW** > equivalente a 0 volt = GND



Interruttori e pulsanti

Interruttori e pulsanti vengono definiti **CHIUSI** (resistenza tra i terminali minori di 1 Ohm equivalente ad un cortocircuito), quando consentono il passaggio di corrente.

Se il passaggio di corrente non è consentito si definisce il collegamento **APERTO** (resistenza tra i terminali > 10 MOhm)



Pulsante

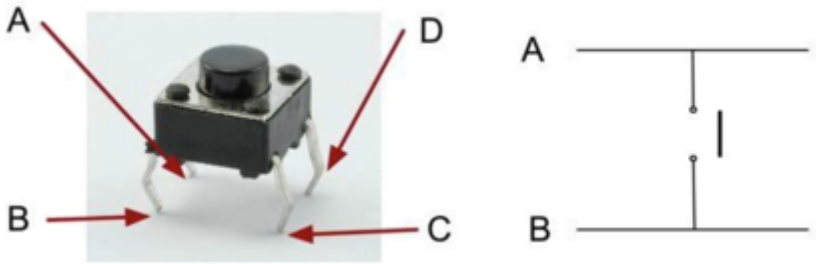
Un pulsante è un tipo di interruttore meccanico temporaneo in cui abbiamo sostanzialmente due contatti e una molla.

Possono esserci due specie di pulsanti:

- **Normalmente aperto**
- **Normalmente chiuso**

Pulsante normalmente APERTO

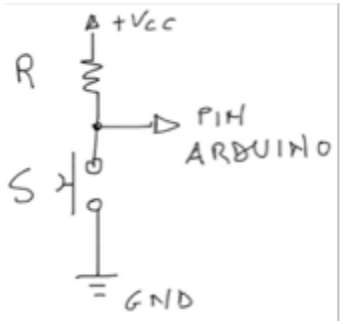
pulsante normalmente aperto (NO: *normally open*): nella posizione di riposo i contatti sono separati e quindi il circuito è aperto (non scorre corrente, quindi ha lo stato LOW), premendo il pulsante si attiva il contatto e si chiude il circuito (stato HIGH). Nel rilasciare di nuovo il pulsante, la molla spinge il pulsante verso l'alto e separa i contatti (stato LOW);



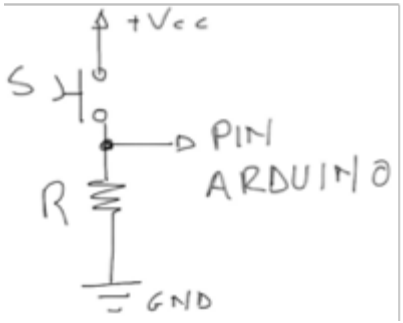
Collegamento pulsante Normalmente APERTO (N.O.)

Dire un pulsante è **normalmente aperto**, vuol dire che quando non premiamo il pulsante questo interrompe il circuito (non permette il passaggio di corrente), possiamo avere due tipologie di collegamento N.O:

1. Con resistenza **pull-up** in cui la resistenza è collegata direttamente a +Vcc e il pulsante a GND
- Pulsante **premuto** > pin Arduino collegato a GND (0 V) > livello logico di uscita **0**
- Pulsante **rilasciato** > pin Arduino collegato a +Vcc (5 V) > livello logico di uscita **1**



2. Con resistenza **pull-down** in cui la resistenza è collegata direttamente a GND e il pulsante a +Vcc
- Pulsante **premuto** > pin Arduino collegato a +Vcc (5 V) > livello logico di uscita 1
- Pulsante **rilasciato** > pin Arduino collegato a GND (0 V) > livello logico di uscita 0



Pulsante normalmente CHIUSO

pulsante normalmente chiuso (NC: *normally closed*):

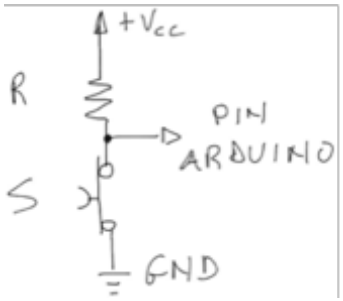
- nella posizione di riposo i contatti sono attaccati e quindi il circuito è chiuso (scorre corrente, stato HIGH),
- premendo il pulsante si disattiva il contatto e si apre il circuito (stato LOW).
- Nel rilasciare di nuovo il pulsante, la molla spinge il pulsante verso l'alto e unisce i contatti (stato HIGH).

Forse è poco intuitivo il fatto che, premendo il pulsante, la tensione venga cambiata da HIGH a LOW (premendo il pulsante togliamo l'alimentazione al circuito) ma i "giochi" si fanno nel codice dello sketch, dove possiamo cambiare completamente la logica del circuito.

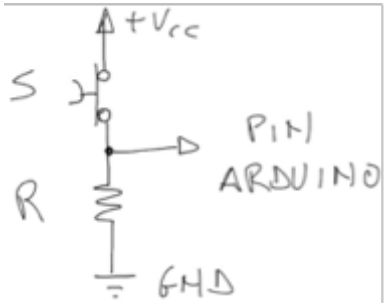
Collegamento pulsante Normalmente CHIUSO (N.C.)

Dire un pulsante è **normalmente chiuso**, vuol dire che quando non premiamo il pulsante questo non interrompe il circuito (permette il passaggio di corrente), possiamo avere due tipologie di collegamento N.C:

1. Con resistenza **pull-up** in cui la resistenza è collegata direttamente a +Vcc e il pulsante a GND
- Pulsante **premuto** > pin Arduino collegato a +Vcc (5 V) > livello logico di uscita **1**
- Pulsante **rilasciato** > pin Arduino collegato a GND (0 V) > livello logico di uscita **0**

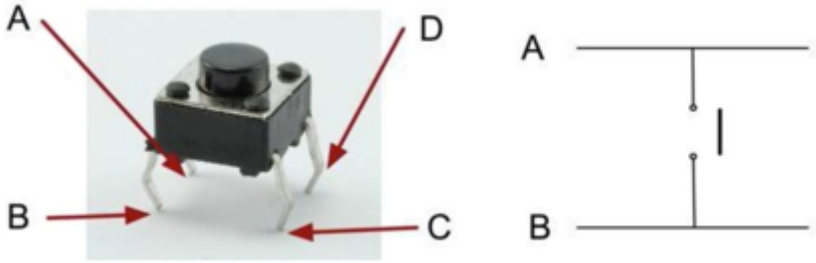


2. Con resistenza **pull-down** in cui la resistenza è collegata direttamente a GND e il pulsante a +Vcc
- Pulsante **premuto** > pin Arduino collegato a GND (0 V) > livello logico di uscita **0**
- Pulsante **rilasciato** > pin Arduino collegato a +Vcc (5 V) > livello logico di uscita **1**



Pulsante

Nel kit abbiamo pulsanti **normalmente aperti**, quindi in stato di “riposo” non scorre corrente e lo stato è LOW.



Internal PULL-UP Resistor

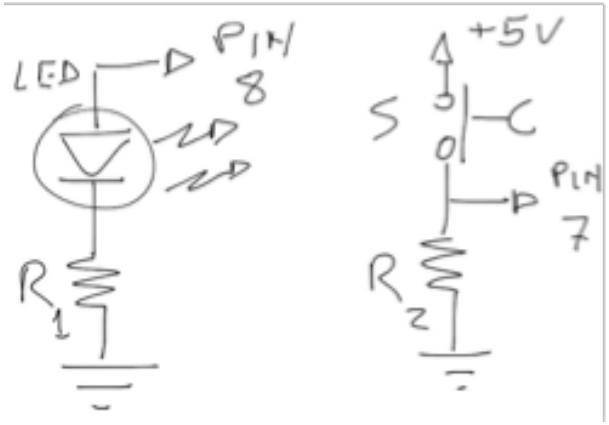
- Digital Pins of Arduino can be configured as
  - **OUTPUT**,
  - **INPUT** or
  - **INPUT_PULLUP**

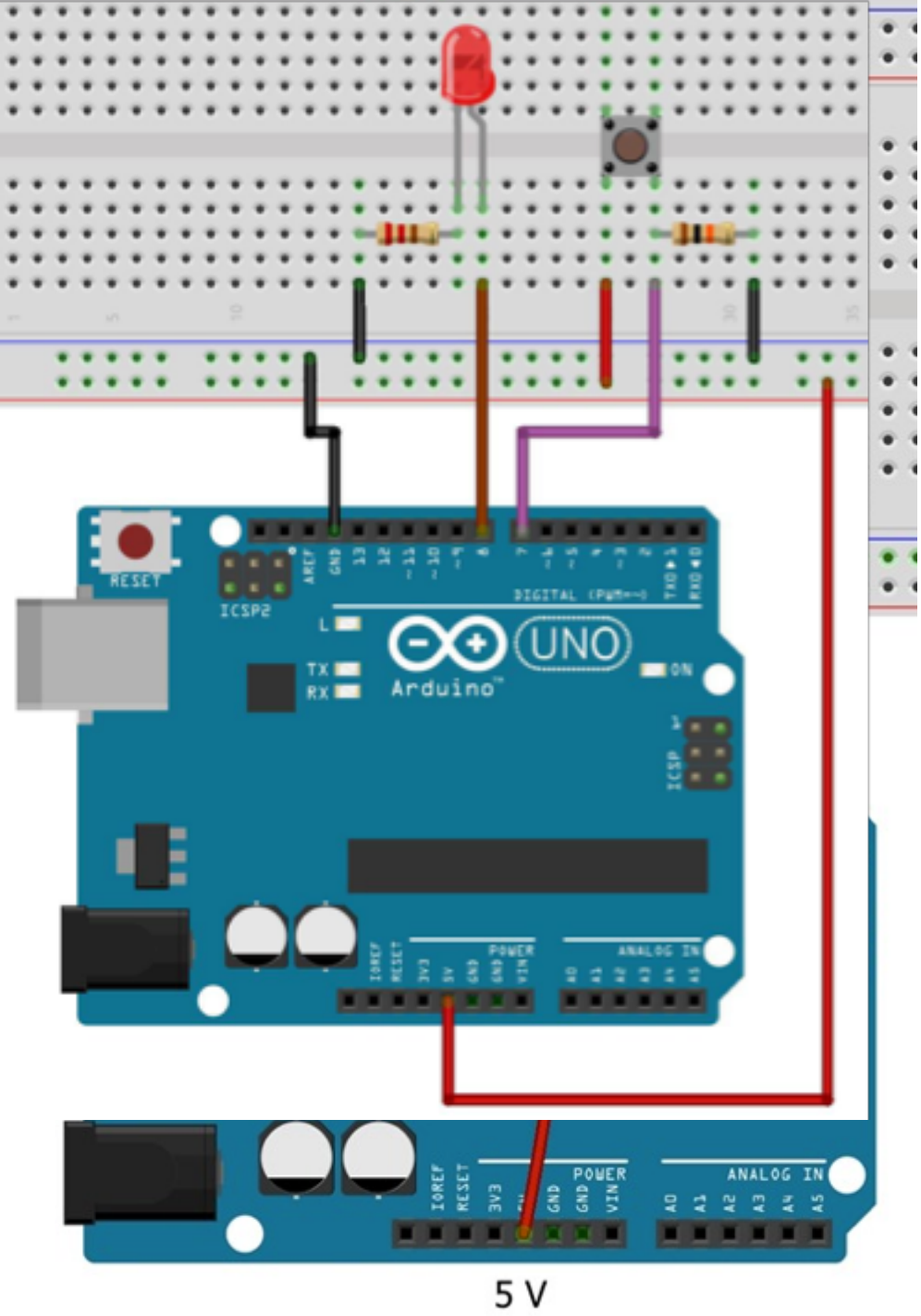
  mode using **pinMode()** function.
- INPUT_PULLUP mode is used to enable the Internal PULL-UP Resistor.
- The value of Internal PULL-UP resistor of Arduino Uno is about 20-50kΩ
- The value of Internal PULL-UP resistor of Arduino DUE is about 100kΩ

Lettura pulsante

Per controllare lo stato di un pulsante utilizzeremo l'istruzione **digitalRead(),** questa istruzione legge il valore su uno specifico pin digitale che può assumere due valori, **HIGH** o **LOW**, detto in modo meno informatico e più elettronico, verifica se su un determinato pin è applicata una tensione di +5V (definito HIGH) o 0V (definito LOW).

Quindi con digitalRead() possiamo leggere uno stato di un sensore e memorizzare questo stato nella memoria di Arduino per fare qualcosa.





Collegamento pulsante

- 5 Volt (alimentazione)
- a massa attraverso un resistore da 10 kiloOhm.
- Dato che abbiamo anche bisogno di rilevare se il pulsante è premuto o rilasciato, abbiamo anche una connessione al pin 6.

Example 1: Arduino Code

- Turn on a led with a button:
  - If Button pressed → Turn on a LED
  - Else → Turn off a LED

Open Example 1 in the shared folder

Example 2: Arduino Code

- Use the button as a switch to turn on a LED
  - Button press → turn on the led keeping it on when it is released
  - Button press the second time → turn off the led

Open Example 2 in the shared folder

Example 2bis: Arduino Code

- Use the button as a switch to turn on a LED
  - Button press → turn on the led keeping it on when it is released
  - Button press the second time → turn off the led

Debouncing

https://www.youtube.com/watch?v=rvIRQ390mtc

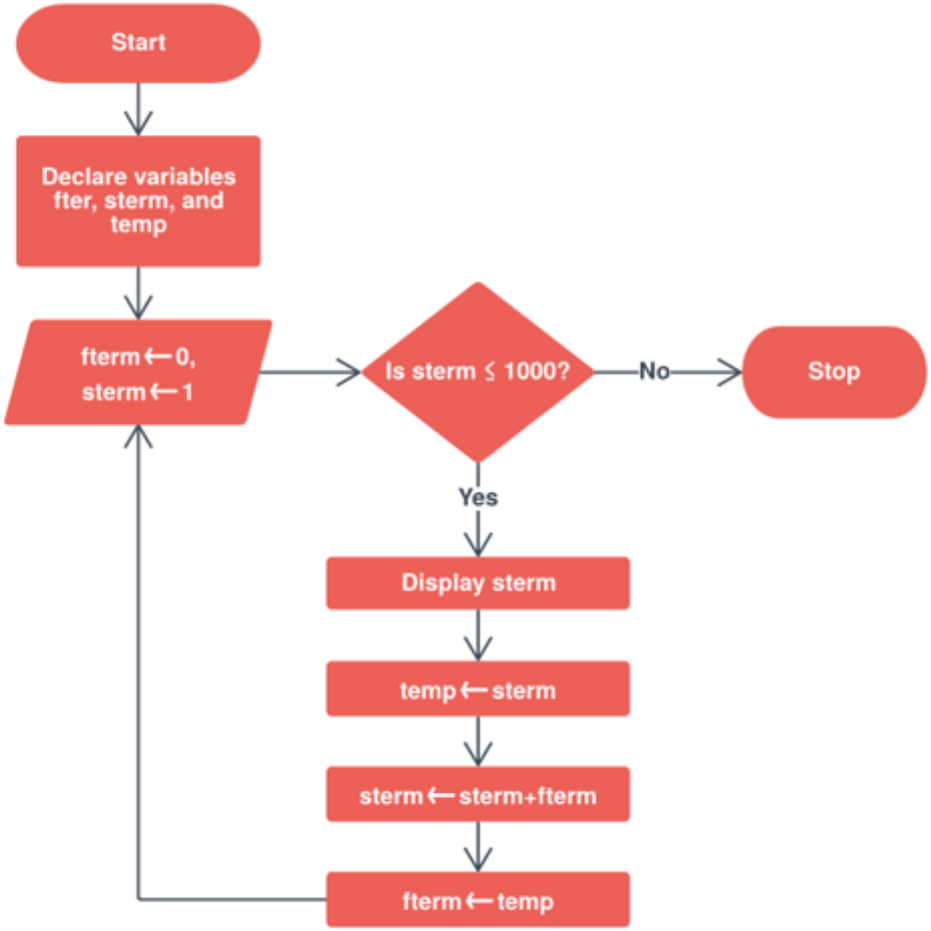
Open Example 2bis in the shared folder

Exercise

Create a program that performs this function:

- **when a button is pressed the LED diode flashes,**
- **when the button is pressed a second time the LED stops blinking and it is always turned on**

Flow chart


Start
Declare variables fter, sterm, and temp
fterm ← 0, sterm ← 1
Is sterm ≤ 1000?
No
Stop
Yes
Display sterm
temp ← sterm
sterm ← sterm+fterm
fterm ← temp

Flow Chart elements

Terminator
Indicates the beginning or end of a program flow in your diagram.

Process
Indicates any processing function.

Decision
Indicates a decision point between two or more paths in a flowchart.

Delay
Indicates a delay in the process.

Data
Can represents any type of data in a flowchart.

Document
Indicates data that can be read by people, such as printed output.

Multiple documents
Indicates multiple documents.

Subroutine
Indicates a predefined (named) process, such as a subroutine or a module.

Preparation
Indicates a modification to a process, such as setting a switch or initializing a routine.

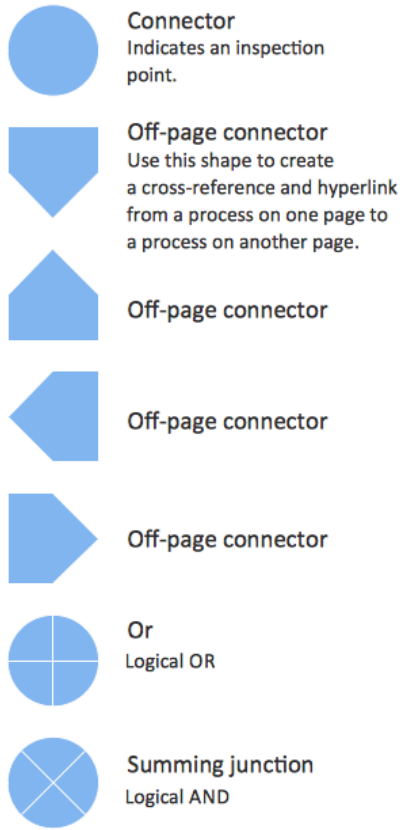
Display
Indicates data that is displayed for people to read, such as data on a monitor or projector screen.

Manual input
Indicates any operation that is performed manually (by a person).

Manual loop
Indicates a sequence of commands that will continue to repeat until stopped manually.

Loop limit
Indicates the start of a loop. Flip the shape vertically to indicate the end of a loop.

Stored data
Indicates any type of stored data.



Connector
Indicates an inspection point.

Off-page connector
Use this shape to create a cross-reference and hyperlink from a process on one page to a process on another page.

Off-page connector

Off-page connector

Off-page connector

Or
Logical OR

Summing junction
Logical AND

Collate
Indicates a step that organizes data into a standard format.

Sort
Indicates a step that organizes items list sequentially.

Merge
Indicates a step that combines multiple sets into one.

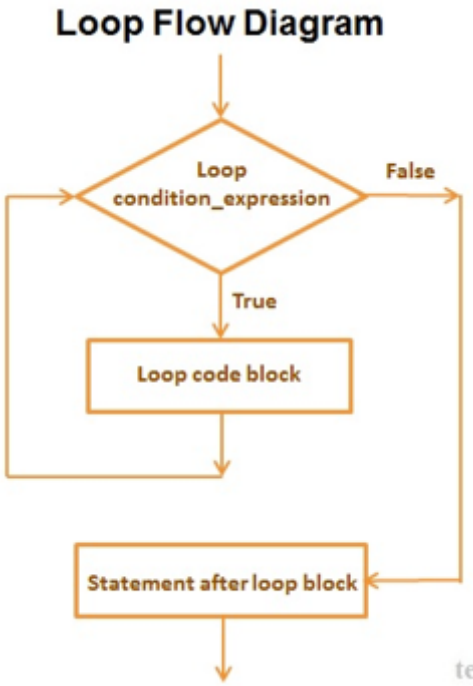
Database
Indicates a list of information with a standard structure that allows for searching and sorting.

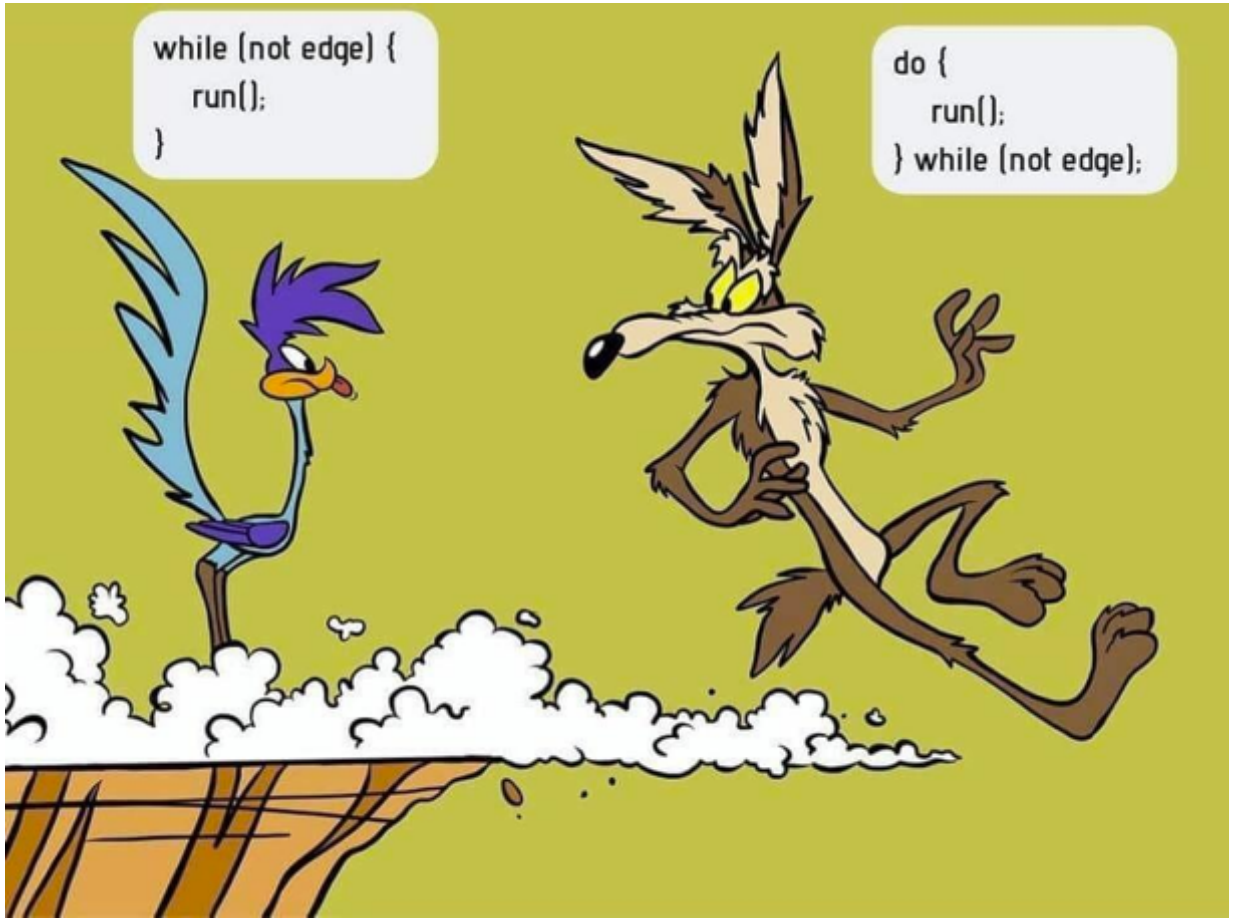
Internal storage
Indicates an internal storage device.

Control structure

- Loops
  - For
  - While
  - Do … while
  - If … else

- https://www.arduino.cc/reference/en/#structure




while (not edge) {
run();
}
do {
run();
} while (not edge);