ARDUINO

IMPARIAMO LA LOGICA

PROF. NACLERIO PASQUALE

Interrutore_con_stato §

```
int pinLed = 12;       //pin per il led
int pinSwitch = 11;  //pin per l'interruttore
bool lettura;         //creo un contenuotre dove mettere la lettura del pulsante
bool stato = LOW;      //imposto uno stato a basso;

void setup() {
  pinMode(pinLed, OUTPUT);  //settiamo il pin del led a ingresso
  pinMode(pinSwitch, INPUT); //settiamo il pin dell'interruttore a uscita
}

void loop() {
  lettura = digitalRead(pinSwitch);  //leggiamo il valore del pin 11 e salviamolo

  if(lettura == HIGH){                    //se la lettura è alta
    stato = !stato;                       //cambia lo stato
    delay(200);                           //aspetta qualche secondo
  }

  if(stato == HIGH){                   //se lo stato è alta

      digitalWrite(pinLed, HIGH);        //accendi il led

    }
    else{                                //altrimenti

      digitalWrite(pinLed, LOW);         //tieni spento il led

      }

}
```

ANALIZZIAMO LE OPERAZIONI LOGICHE

- Nel nostro codice abbiamo usato delle operazioni logiche
- Abbiamo usato l'uguaglianza
- Abbiamo usato la negazione

```
13
14  if(lettura == HIGH){                    //se la lettura è alta
15    stato = !stato;                       //cambia lo stato
16    delay(200);                           //aspetta qualche secondo
17  }
18
```

OPERAZIONI LOGICHE

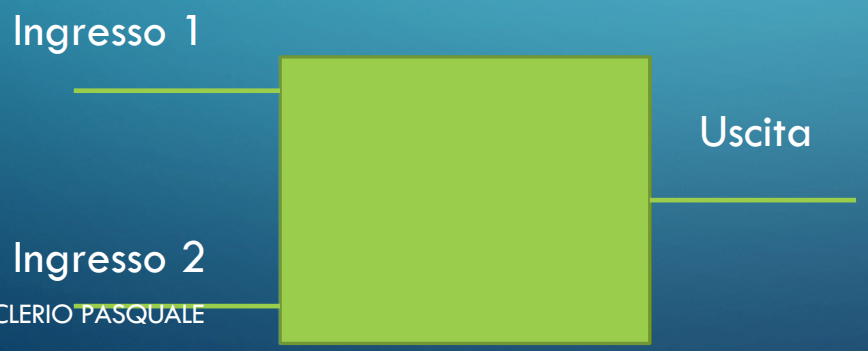
- Le operazioni logiche sono molto importanti nella programmazione
- Ci servono a prendere delle decisioni o a progettare delle decisioni
- Quando dobbiamo pensare a come si dovrebbe comportare un programma, dobbiamo progettare le scelte che questo programma deve fare.
- Esistono numerose operazioni logiche e sono tutte rappresentate con dei simboli ben precisi

PREMESSA

- Nella logica booleana esistono solo 2 stati possibili
- 1 | 0 HIGH | LOW ON | OFF
- Partendo da questo è possibile costruire un sistema logico e fare delle vere operazioni
- Prendiamo confidenza con quello che prende il nome di TABELLA DI VERITA'

TABELLA DI VERITA’

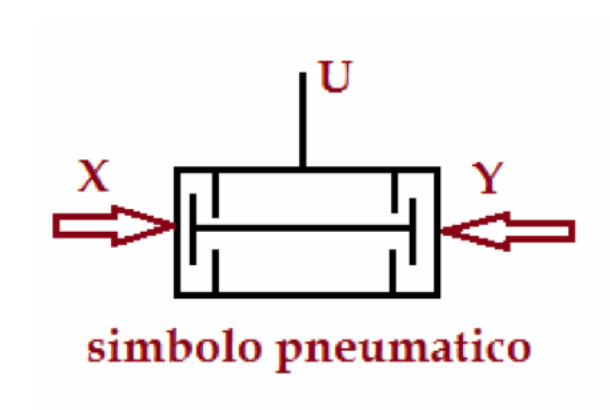
- La tabella di verità è una semplicissima tabella dove da un lato abbiamo tutti i possibili stati degli ingressi e dell’altra abbiamo tutte le possibili uscite
- Dobbiamo immaginare le operazioni booleane come delle vere e proprie scatole dove da una parte entrano gli ingressi e dall’altra abbiamo le uscite.



Ingresso 1	Ingresso 2	Uscita
0	0	0
0	1	0
1	0	0
1	1	1

VOI AVETE GIÀ VISTO QUESTE COSE !

- Nella pneumatica avete visto delle valvole che permettevano l'uscita dell'aria solo quando entrava aria solo dalle due porte.
- Posso avere ARIA = 1 NON ARIA = 0

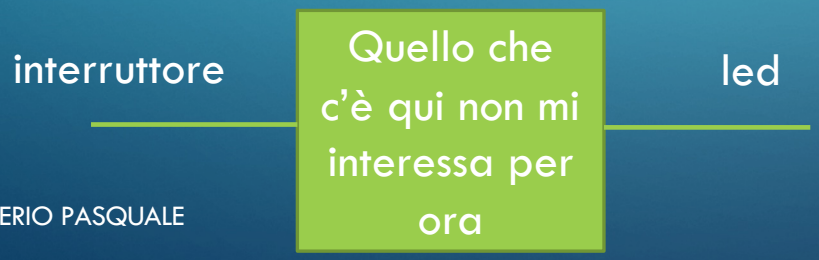


X	Y	U
non aria	non aria	non aria
non aria	aria	non aria
aria	non aria	non aria
aria	aria	aria

Ingresso 1	Ingresso 2	Uscita
0	0	0
0	1	0
1	0	0
1	1	1

TUTTO QUESTO VALE ANCHE PER NOI

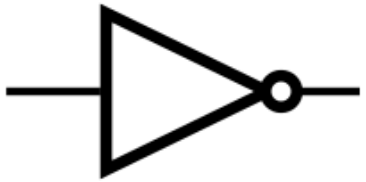
- Facciamo un esempio
- Io voglio che quando un interruttore è acceso = 1 allora voglio che il led sia spento = 0.
- Ovviamente vale anche il viceversa e quindi quando l'interruttore è spento = 0 il led sia accesso = 1



interruttore	led
0	1
1	0

PORTA NOT

- Quello che abbiamo appena fatto prende il nome di porta NOT e cioè è la negazione dell'ingresso.



INPUT	OUTPUT
A	Y
0	1
1	0

Nella programmazione è fatta così

```
Y = !A
```

Noi l'abbiamo già usata

```
15      stato = !stato;
```

PORTA OR

- Il led rimane spento solo se entrambi gli interruttori sono spenti

INPUT		OUTPUT
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

```
Y = A || B
```

PORTA AND

- Il led si accende solo quando entrambi gli interruttori sono accesi

INPUT		OUTPUT
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

```
Y = A && B
```

PORTA NOR

- Il led si accende solo quando entrambi gli interruttori sono spenti
- Notare che è una negazione di una porta OR

INPUT		OUTPUT
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

```
Y = !(A || B)
```

PORTA NAND

- Il led è sempre acceso e si spegne solo quando gli interruttori sono entrambi spenti
- È una negazione di una porta AND

INPUT		OUTPUT
A	*B*	*Y*
0	0	1
0	1	1
1	0	1
1	1	0

```
Y = !(A && B)
```

PORTA XOR

- Il led si accende solo quando SOLO un interruttore è acceso e l'altro è spento

INPUT		OUTPUT
A	B	*Y*
0	0	0
0	1	1
1	0	1
1	1	0

```
Y = (A && !B) || (!A && B)
```

PORTA XNOR

- Il led si accende solo quando entrambi gli interruttori sono accesi o spenti

INPUT		OUTPUT
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

```
Y = !((A && !B) || (!A && B))
```

PROF. NACLERIO PASQUALE

OVVIAMENTE ESISTONO ANCHE CON MOLTI PIÙ INGRESSI E MOLTE PIÙ USCITE

- Ad esempio una porta AND a 3 ingressi ed un uscita
- Noi la possiamo realizzare con 2 porte and a cascata

Simbolo funzionale

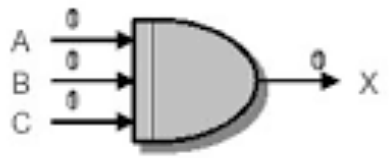
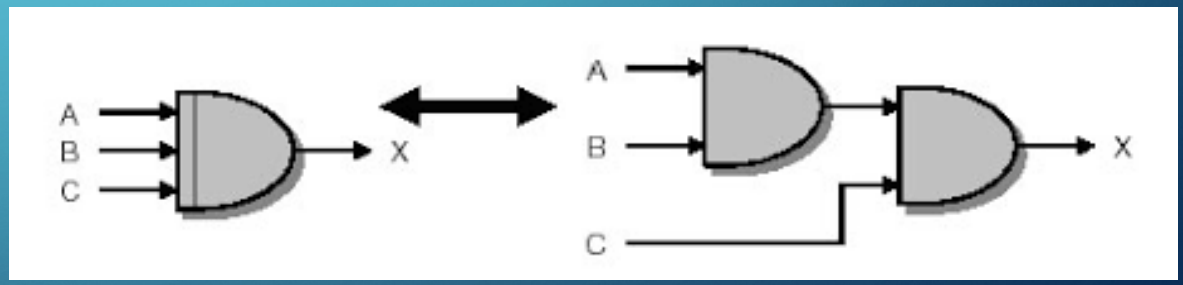


Tabella delle verità

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



X = (A && B) && C

LA LORO IMPORTANZA (ESEMPIO)

- Abbiamo costruito un sistema caldai con un sensore di temperatura e sensore di pressione.
- La caldaia se arriva a 100 °C raggiunge la sua massima temperatura che non deve superare. La pressione se troppo alta potrebbe far scoppiare. Quindi bisogna aprire la valvola di sfogo.
- Analizziamo i casi

CASI

- Temperatura BASSA con pressione BASSA allora valvola CHIUSA
- Temperatura BASSA con pressione ALTA allora valvola APERTA
- Temperatura ALTA con pressione BASSA allora valvola APERTA
- Temperatura ALTA con pressione ALTA allora valvola APERTA

- Temperatura 0 con pressione 0 allora valvola 0
- Temperatura 0 con pressione 1 allora valvola 1
- Temperatura 1 con pressione 0 allora valvola 1
- Temperatura 1 con pressione 1 allora valvola 1

T	P	V
0	0	0
0	1	1
1	0	1
1	1	1

MA È LA PORTA OR ECCO COME USARLA

T	P	V
0	0	0
0	1	1
1	0	1
1	1	1

INPUT		OUTPUT
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

```
34 if( temperatura || pressione){
35   digitalWrite(valvola, HIGH);
36 }
37 else{
38   digitalWrite(valvola, LOW);
39 }
```