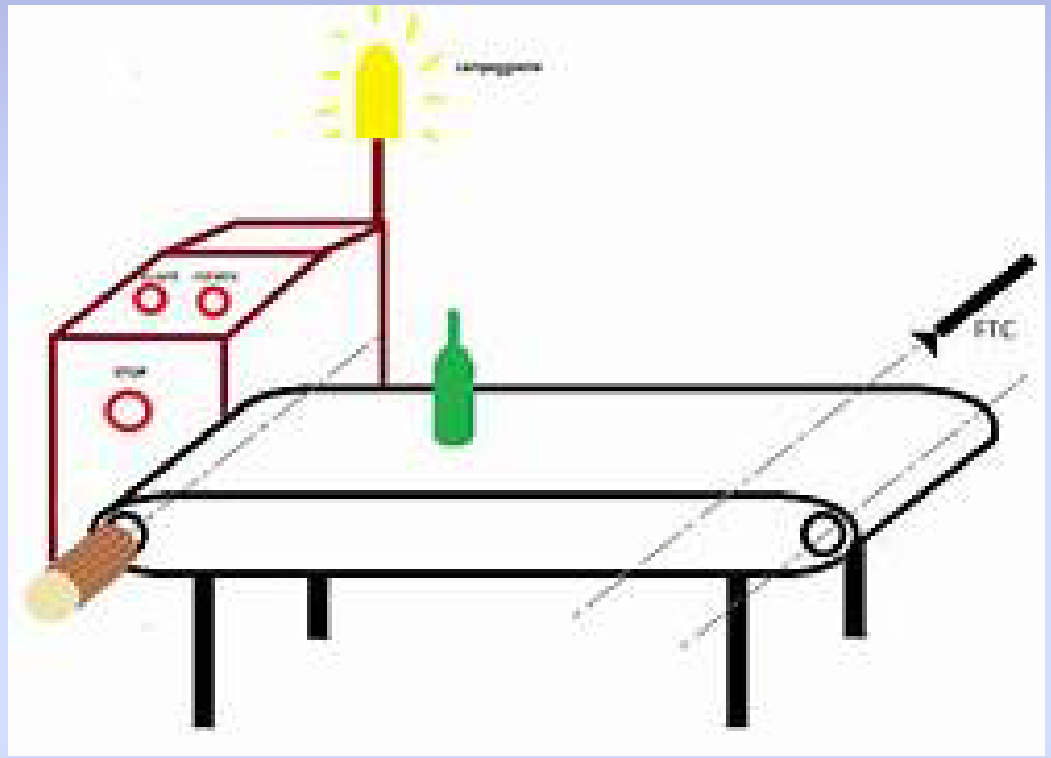
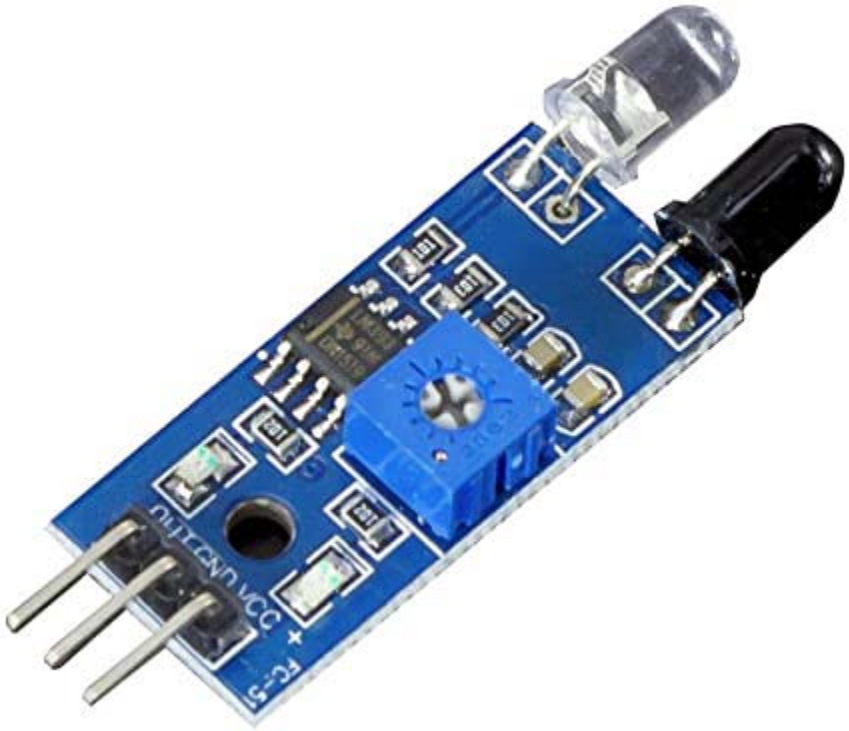



# Contapezzi

# Fotocellula




Nel file interruttore.pdf è stata accennata la funzionalità della fotocellula

La fotocellula è formato da:

- Un elemento trasmettitore che è un fotodiodo a infrarossi
- Un elemento ricevitore che può essere un fotodiodo con transistor in configurazione ON/OFF o solo un fototransistor

In alcuni casi, l'elemento trasmettitore è separato da quello ricevitore

In presenza di un ostacolo il segnale emesso torna indietro e sul pin digitale del micro compare un segnale OFF. Il micro incrementerà un intero ogni volta che arriva il segnale

# Un semplice contatore con la scheda arduino

contatore

```
int led=10, interruttore=7;
int n=0; //il contatore viene posto nullo all'inizio
void setup()
{
  pinMode(led, OUTPUT);
  pinMode(interruttore, INPUT);
  Serial.begin(9600);
}
void loop()
{int lettura=digitalRead(interruttore);
 if(lettura==0){
  n=n+1;//il contatore viene incrementato di una unità
 digitalWrite(led, HIGH);
   }
 if(lettura==1)
  digitalWrite(led, LOW);

 Serial.println(n);}
```

# Problemi

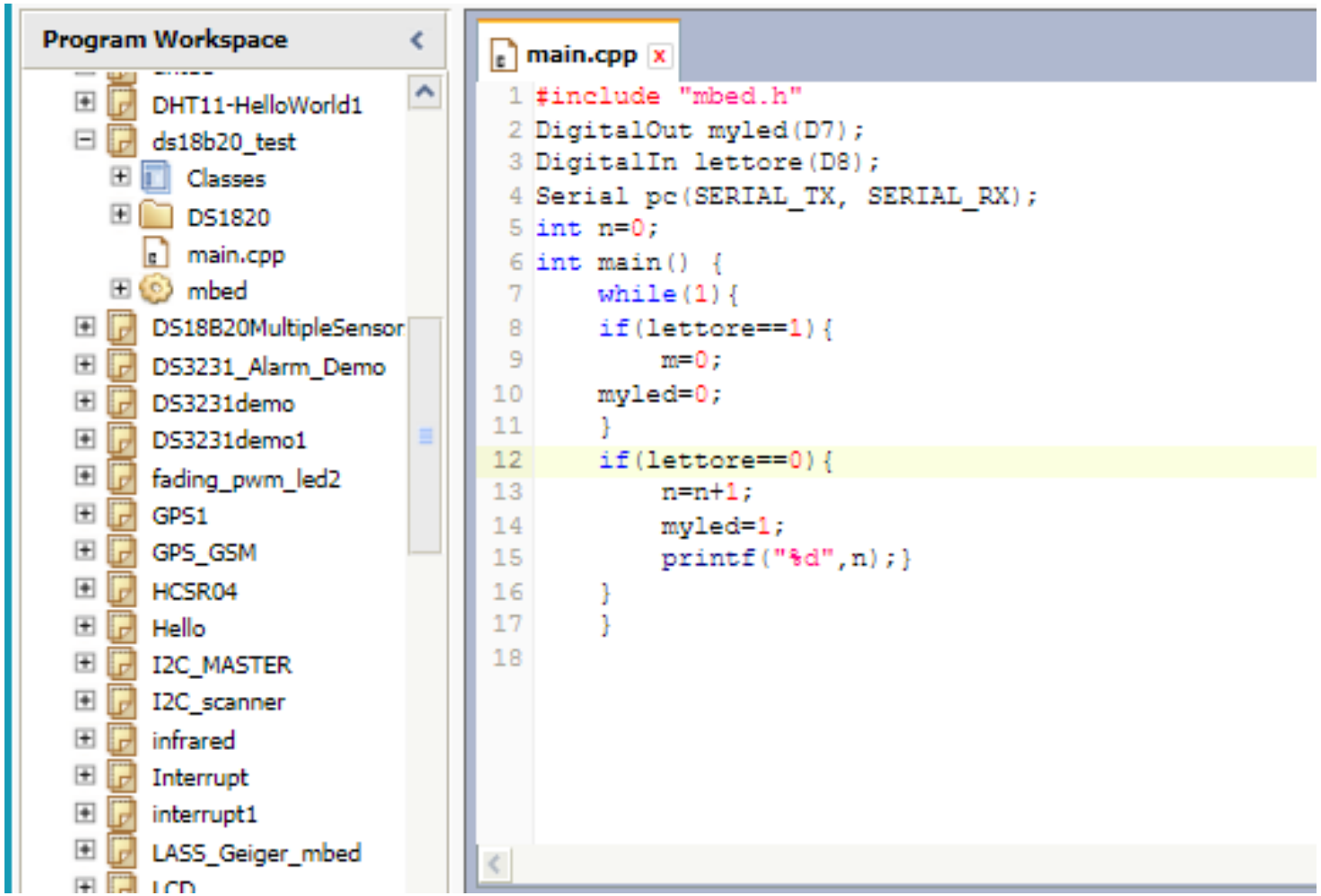
- Il problema di questo progetto è che incrementa troppo velocemente e capita che conta due volte lo stesso pezzo
- Si potrebbe mettere un ritardo ma non sappiamo con quale velocità può transitare un oggetto
- Si inserisce allora una variabile di ridondanza m che incrementa sempre al passaggio dell'oggetto
- Il contatore si incrementa solo se m=0
- Appena il pezzo non viene più rilevato, m viene posto a zero

# Contatore con ridondanza (Arduino)

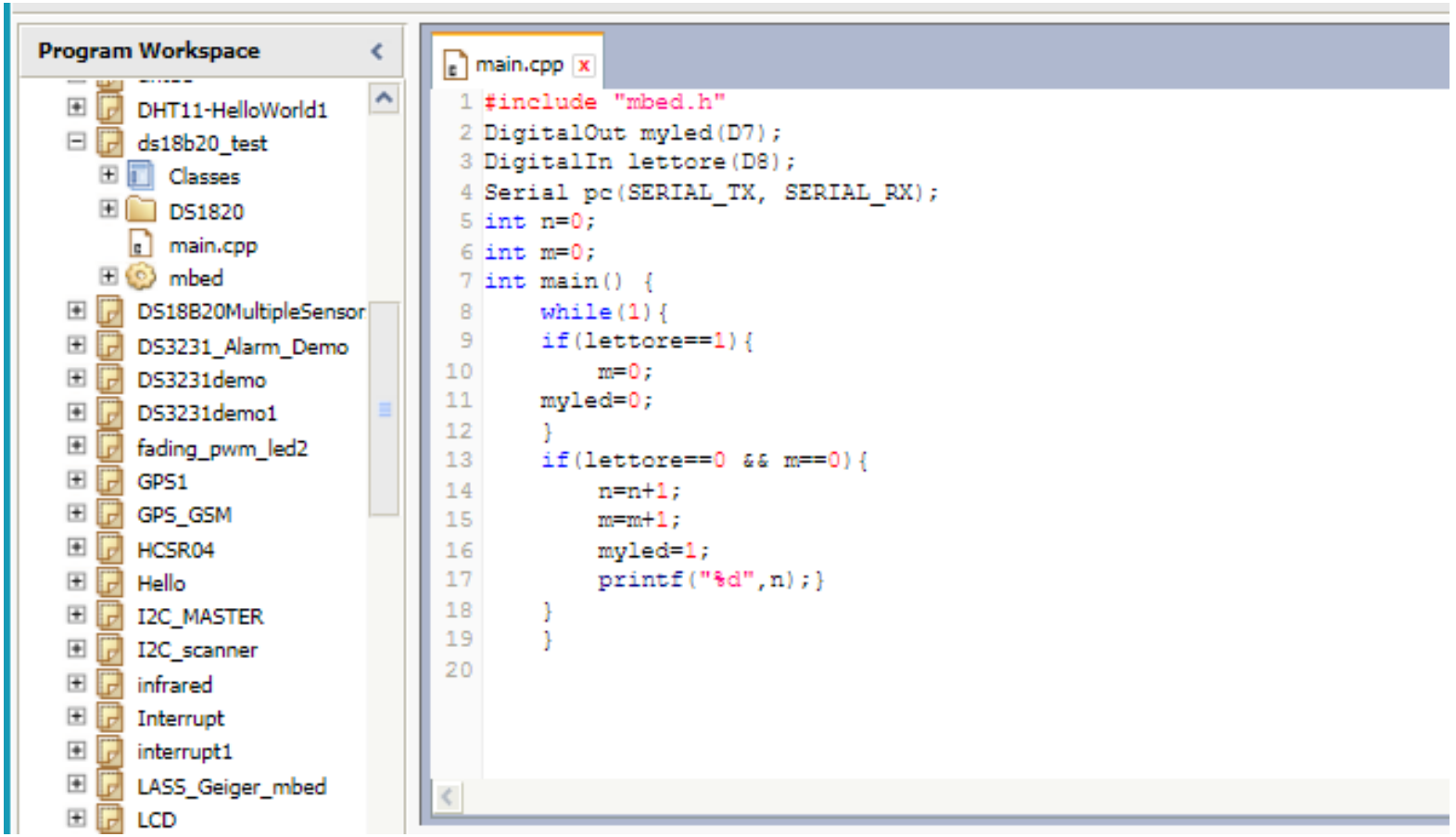
contatore §

```
int led=10, interruttore=7;
int n=0, m=0; //il contatore e la ridondanza vengono posti =0 all'inizio
void setup()
{
  pinMode(led, OUTPUT);
  pinMode(interruttore, INPUT);
  Serial.begin(9600);
}
void loop()
{int lettura=digitalRead(interruttore);
 if(lettura==0 && m==0){
  n=n+1;//il contatore viene incrementato di una unità
  m=m+1;//si incrementa il contatore di ridondanza
 digitalWrite(led, HIGH);
   }
 if(lettura==1){
  m=0; //m viene azzerato
  digitalWrite(led, LOW);
  }
 Serial.println(n);}
```

# Semplice contatore STM32



# Contatore con ridondanza (STM32)



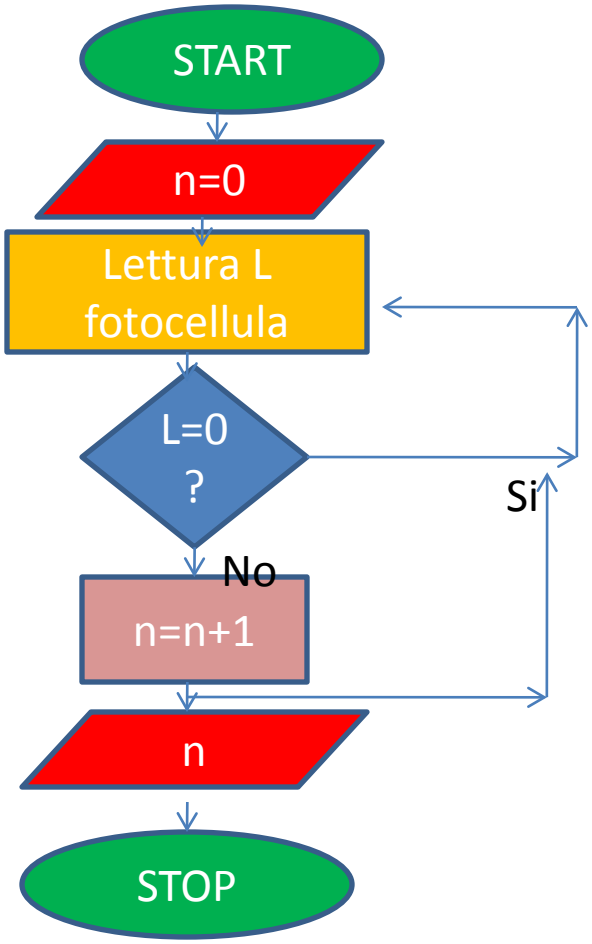
# Contatore

```
import RPi.GPIO as gpio  #libreria
import time                    #libreria
gpio.setwarnings(False)  #tutti i pin impegnati vengono #disimpegnati
gpio.setmode(gpio.BCM)  #il nome dei pin è quello del #costruttore e non sequenziale
#ma del costruttore
gpio.setup(18, gpio.OUT)    #pin 18 (nome BCM 12 come sequenza)  OUTPUT #(led)
gpio.setup(23,gpio.IN) #pin 21 (nome BCM 23 come sequenza) INPUT
n=0                  #variabile contatore =0
while True:
    val=gpio.input(23)
    if val==0:
        gpio.output(18,True)
        n=n+1
        print (n)
    if val==1:
         gpio.output(18,False)    #il led non si accende
```

# Contatore con ridondanza (Raspberry)

```
import RPi.GPIO as gpio
import time
gpio.setwarnings(False)
gpio.setmode(gpio.BCM)  #il nome del pin non è sequenziale ma del #costruttore
gpio.setup(18, gpio.OUT)    #pin 18 (nome BCM 12 come sequenza)  OUTPUT #(led)
gpio.setup(23,gpio.IN) #pin 21 (nome BCM 23 come sequenza) INPUT
n=0 #variabile contatore =0
m=0
while True:
    val=gpio.input(23)
    print(n)
    if val==0 and m==0:
        gpio.output(18,True)
        n=n+1
        m=m+1
    if val==1:
        m=0
        gpio.output(18,False)    #il led non si accende
```

# Flow chart contatore



# Contatore