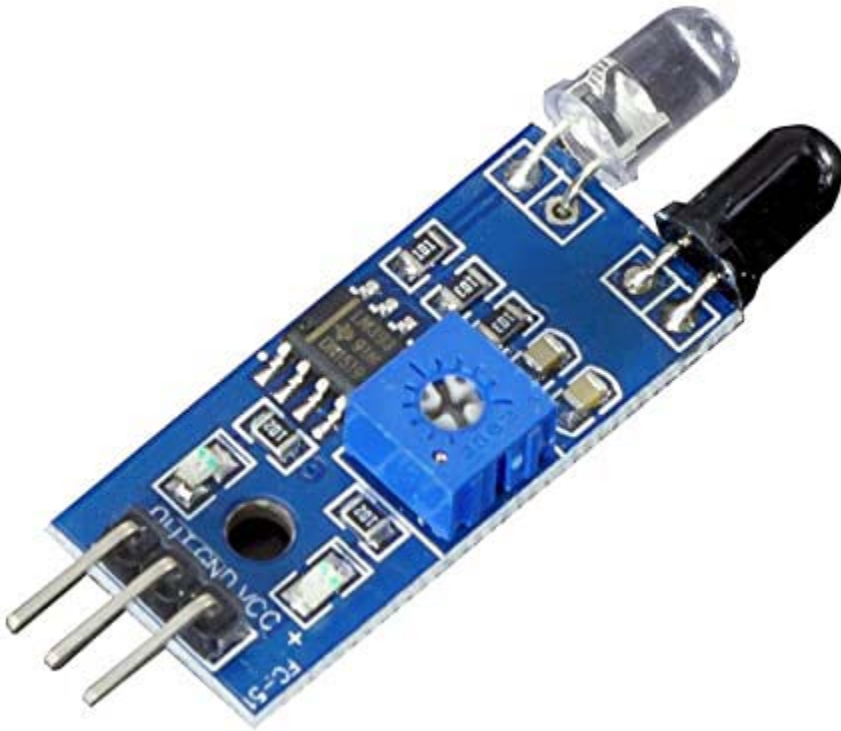


Contapezzi

# Fotocellula



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

La fotocellula è formato da:

- Un elemento trasmettitore che è un fotodiode a infrarossi
- Un elemento ricevitore che può essere un fotodiode 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

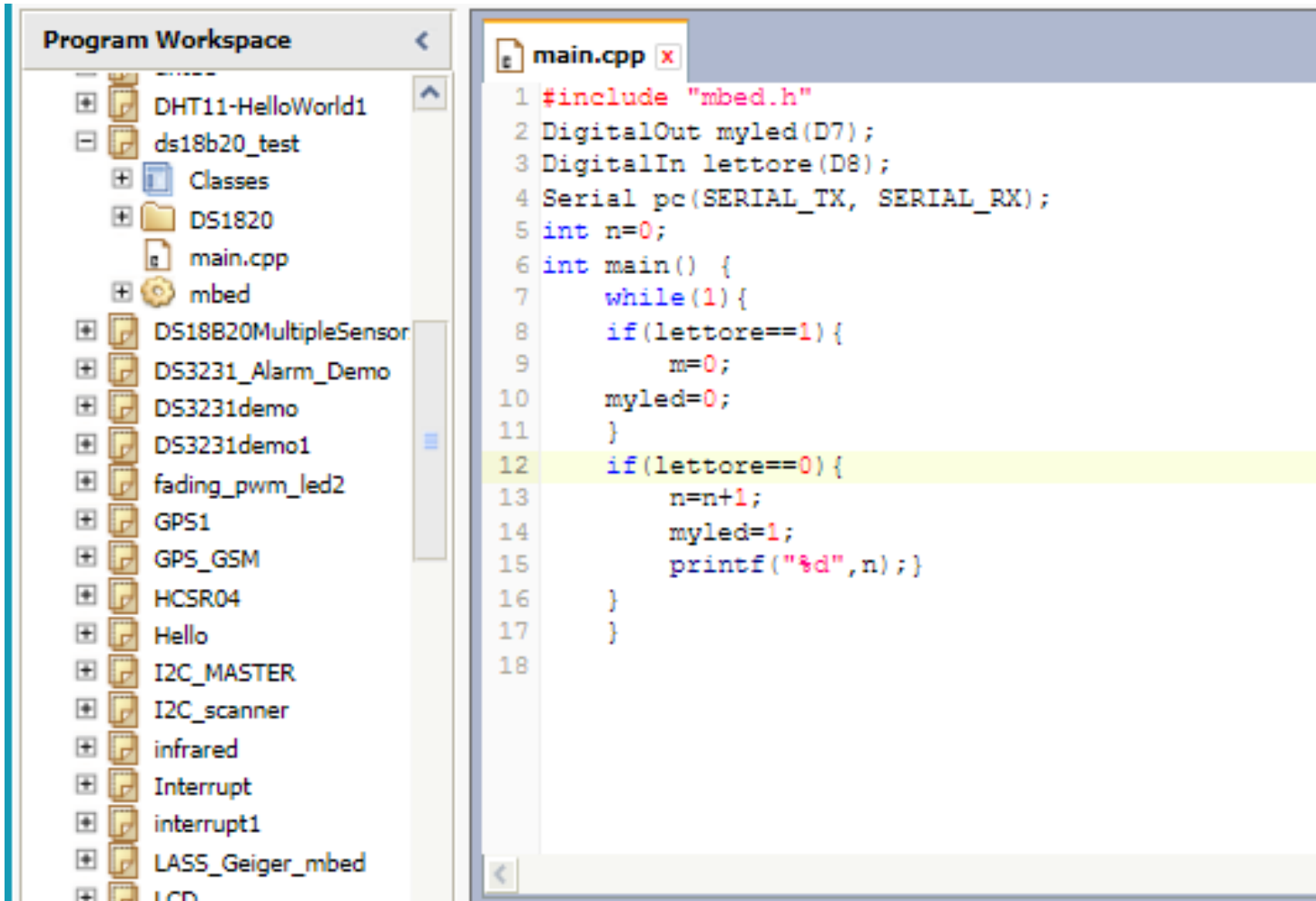
- Il problema di questo progetto è che incrementa troppo velocemente e capita che conta due volte lo stesso pezzo
- Si potrebbe mettere un ritardo ma non sappiamo con quale velocità può transitare un oggetto
- Si inserisce allora una variabile di ridondanza  $m$  che incrementa sempre al passaggio dell'oggetto
- 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);}
```

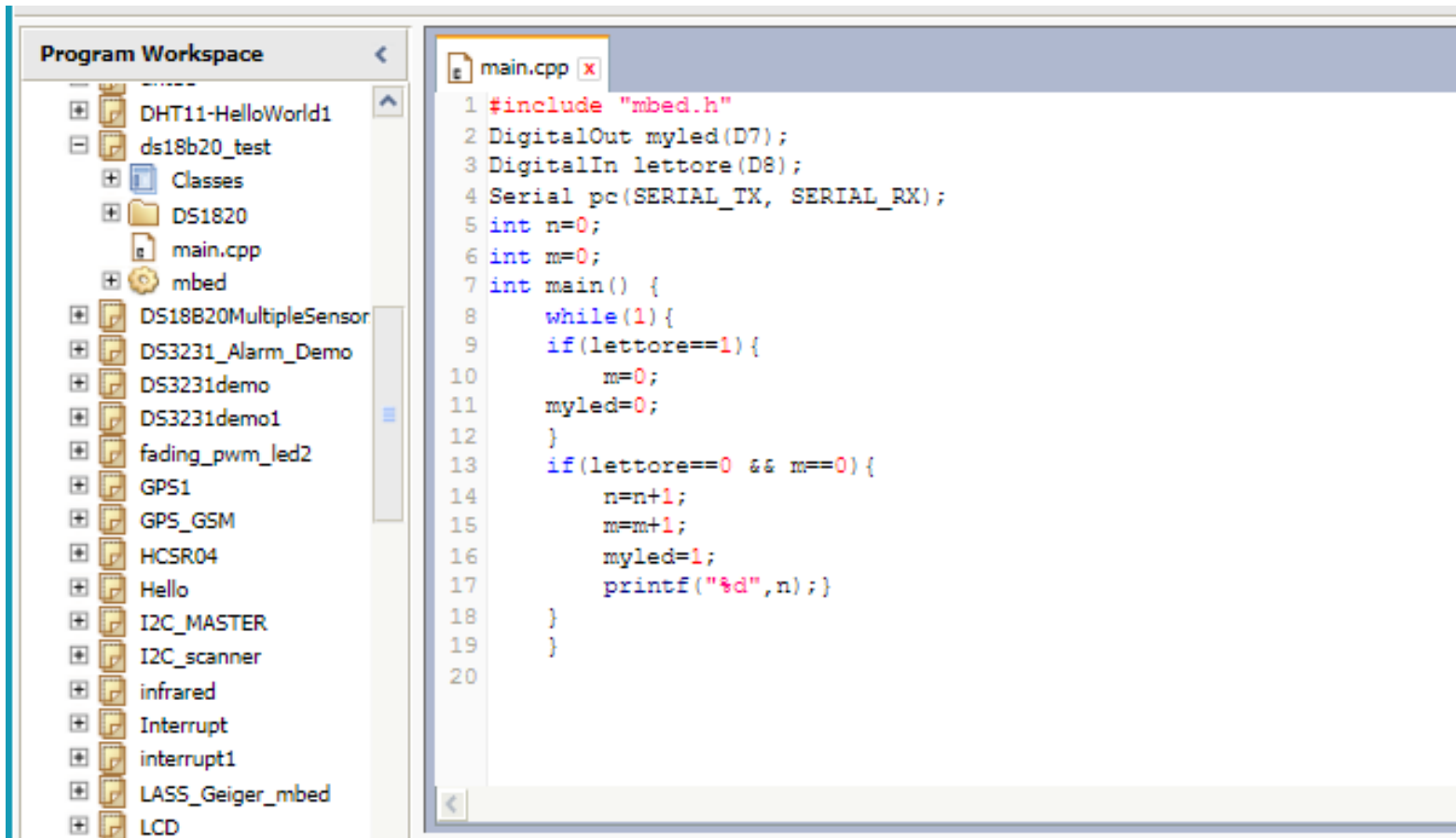
# Semplice contatore STM32



The image shows a screenshot of an IDE's Program Workspace and a code editor. The workspace on the left lists various projects, with 'ds18b20\_test' expanded to show 'main.cpp'. The code editor on the right displays the contents of 'main.cpp', which is a simple counter program. The code includes the 'mbed.h' header, initializes a digital output pin (D7) and a digital input pin (D8), and implements a main loop that checks the state of the input pin. When the input is high (1), it resets a counter 'n' to 0 and turns the LED off. When the input is low (0), it increments the counter 'n' and turns the LED on. The current line of code, 'if (lettore==0) {', is highlighted in yellow.

```
1 #include "mbed.h"
2 DigitalOut myled(D7);
3 DigitalIn lettore(D8);
4 Serial pc(SERIAL_TX, SERIAL_RX);
5 int n=0;
6 int main() {
7     while(1){
8         if(lettore==1){
9             m=0;
10            myled=0;
11        }
12        if(lettore==0){
13            n=n+1;
14            myled=1;
15            printf("%d",n);
16        }
17    }
18 }
```

# Contatore con ridondanza (STM32)



The image shows a screenshot of an IDE's Program Workspace and a code editor. The workspace on the left lists various projects, with 'ds18b20\_test' expanded to show 'main.cpp'. The code editor on the right displays the following C++ code:

```
1 #include "mbed.h"
2 DigitalOut myled(D7);
3 DigitalIn lettore(D8);
4 Serial pc(SERIAL_TX, SERIAL_RX);
5 int n=0;
6 int m=0;
7 int main() {
8     while(1){
9         if(lettore==1){
10             m=0;
11             myled=0;
12         }
13         if(lettore==0 && m==0){
14             n=n+1;
15             m=m+1;
16             myled=1;
17             printf("%d",n);
18         }
19     }
20 }
```

# Contatore

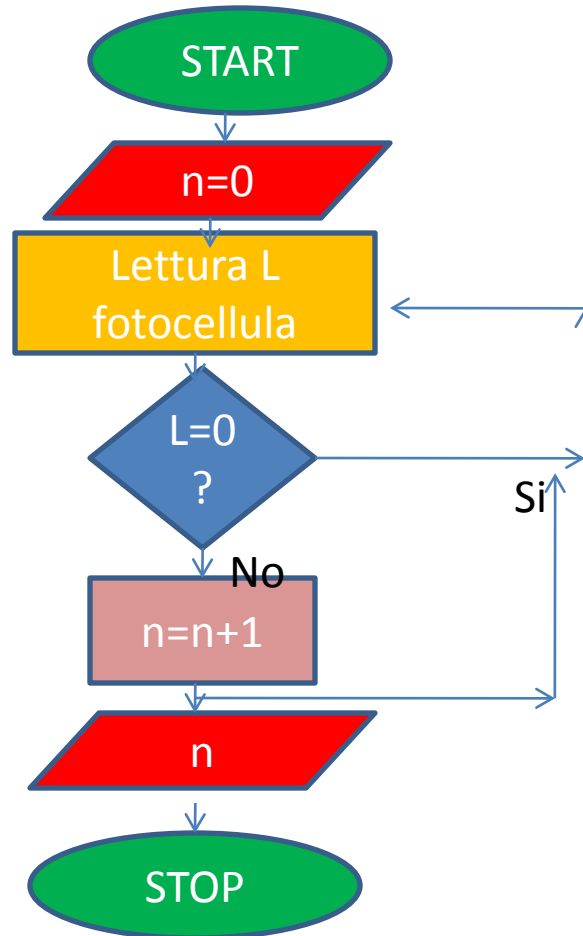
```
import RPi.GPIO as gpio #libreria
import time             #libreria
gpio.setwarnings(False) #tutti i pin impegnati vengono #disimpegnati
gpio.setmode(gpio.BCM) #il nome dei pin è quello del #costruttore e non sequenziale
#ma del costruttore
gpio.setup(18, gpio.OUT) #pin 18 (nome BCM 12 come sequenza) 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

