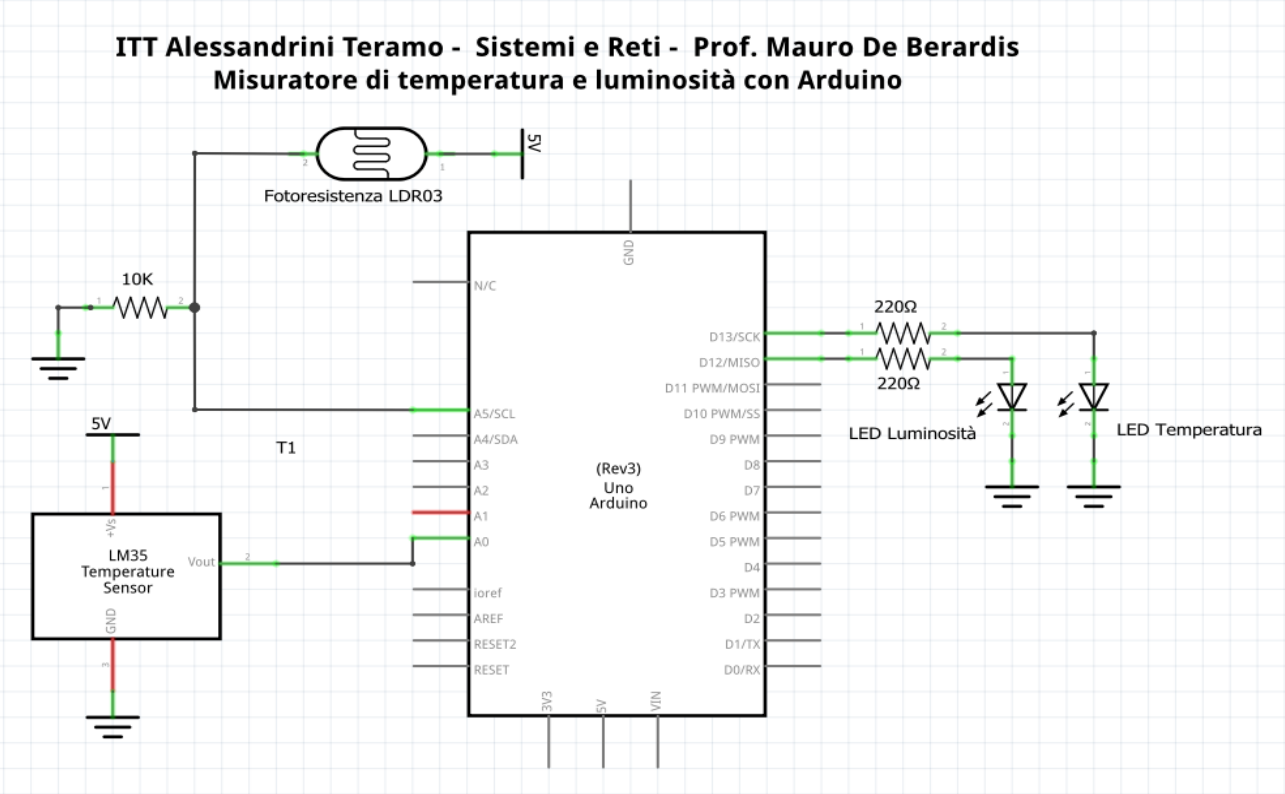
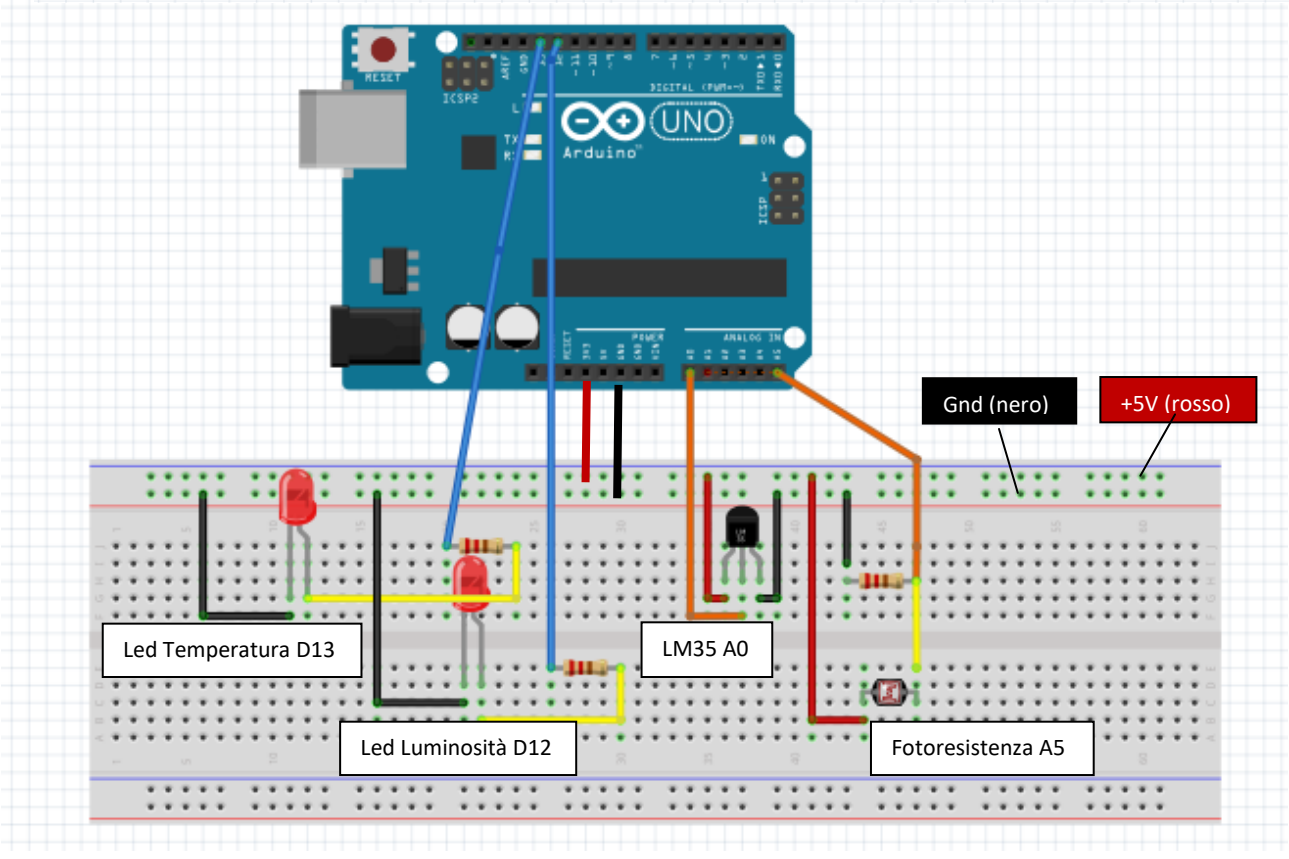
Arduino e Arduino + Visual C#: misuratore di temperatura e luminosità

Il progetto realizza un circuito, basato su Arduino, in grado di visualizzare i valori letti da un sensore di temperatura e da un sensore di luminosità. Si tratta di un'applicazione semplice e didattica realizzata:

1. utilizzando esclusivamente Arduino (Soluzione A)
2. interfacciando Arduino con Visual C# (Soluzione B). In tal caso si potrebbero utilizzare altri linguaggi, ugualmente diffusi e versatili, quali Java, Visual Basic e Pyton

Lo schema elettrico e i collegamenti sulla breadboard dell'applicazione che si vuole realizzare sono i seguenti:

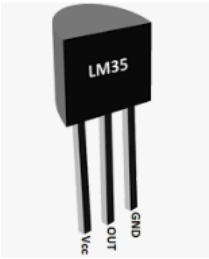




Componenti necessari:

- Arduino collegato ad un PC Windows X
- Breadboard con cavetteria
- 1 circuito integrato LM35
- 1 Fotoresistenza
- 1 Resistenza da 10 KOhm
- 2 LED preferibilmente di colore differente (rosso e verde ad esempio)
- 2 Resistenze da 220 ohm

Per misurare la temperatura utilizziamo il sensore integrato LM35, un circuito poco costoso e sufficientemente preciso. La tensione di uscita è proporzionale alla temperatura e non è necessario realizzare circuiti di controllo aggiuntivi.



Guardando l'integrato di fronte, il piedino a sinistra deve essere collegato all'alimentazione 5V, quello centrale, che fornisce la tensione di uscita proporzionale alla temperatura, va collegato all'ingresso analogico A0, il piedino di destra va collegato a massa.
Quando Arduino legge con AnalogRead l'ingresso analogico A0, lo converte in digitale con un ADC (Convertitore analogico digitale) con risoluzione di 10 bit. I livelli di quantizzazione che l'ADC può fornire, sono 2^{10} =1024, ovvero sono tutte le combinazioni di 10 bit comprese tra 0000000000 e 1111111111 (in decimale da 0 a 1023).
Poiché il valore massimo del segnale analogico in ingresso è di 5 Volt e tale valore è associato al livello 1023, con una semplice proporzione il valore di tensione letto viene riportato tra 0 e 5 volt.
Infine, tenendo conto che il sensore integrato LM35 presenta un'uscita lineare in tensione uguale a 10 mV/°C, si ottiene facilmente il valore della temperatura.

Anticipando una parte dello sketch dell'applicazione, le istruzioni per ottenere la temperatura sono le seguenti:

```
//Viene letto il valore della tensione di uscita  del sensore LM35 convertito in digitale
int ValoreLetto=analogRead(A0);
//Si ottiene la tensione   tra 0 e 5 volt utilizzando  la  proporzione 5V:1023=Vout:ValoreLetto
float Vout=5.0*ValoreLetto/1023.0;
//Si calcola la temperatura
float Temperatura =Vout*1000.0 / 10.0;
```

Per misurare la luminosità si utilizza una fotoresistenza, un dispositivo elettronico la cui resistenza varia in maniera inversamente proporzionale alla quantità di luce che lo colpisce.
Al buio il valore della resistenza è di qualche MOhm, in piena luce il valore scende a qualche KOhm.
Per la misura utilizziamo un partitore di tensione, con la fotoresistenza e una resistenza da 10 kOhm, collegato all'ingresso analogico A5.
In questo caso non ci preoccupiamo di convertire i valori forniti dal'ADC di Arduino in lumen o lux (lumen/mq) in quanto, al contrario che nel caso della temperatura, generalmente non abbiamo familiarità con queste unità di misura. Pertano assumiamo direttamente i valori letti con

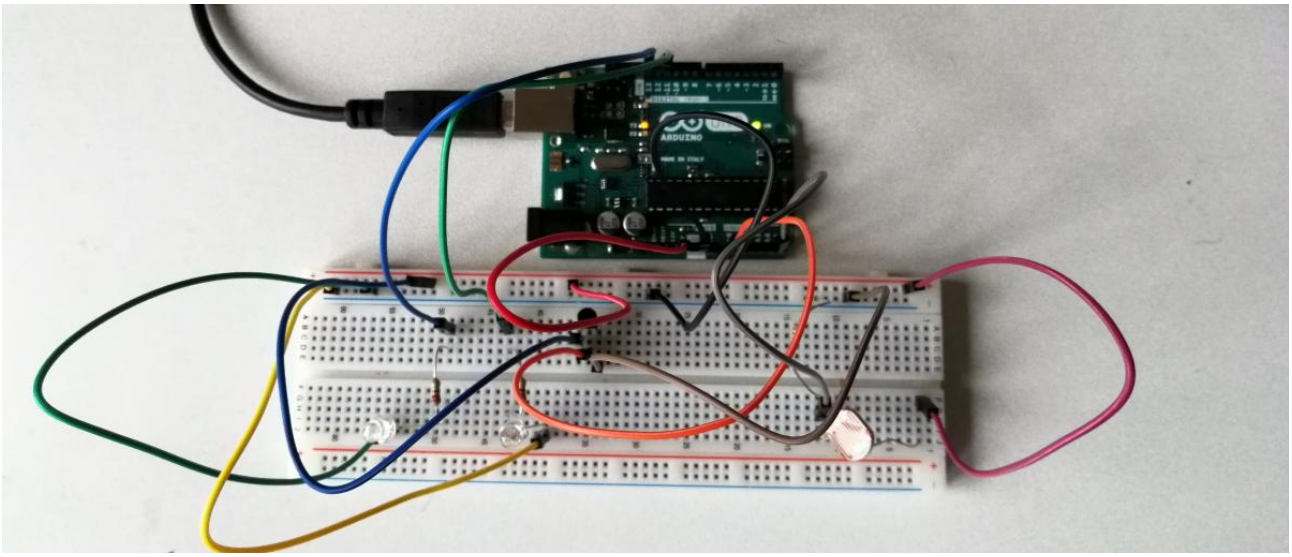
AnalogRead sul piedino A5 e convertiti dall'ADC a 10 bit, tenendo conto che tali valori tendono a 1023 nel caso di luce piena e prolungata e a 0 nel caso di buio totale e prolungato.

Il progetto è facilmente riproducibile in quanto vengono utilizzati componenti (sensore LM35 e fotoresistenza) molto comuni ed economici.

La Soluzione A è adatta agli studenti di informatica, elettrotecnica, elettronica del quarto/quinto anno degli Istituti Tecnici Tecnologici ma in generale a tutti gli studenti che, sempre più numerosi, imparano il coding con Arduino e amano creare dei manufatti utilizzando le tecnologie.

La soluzione B è invece adatta agli studenti di informatica che hanno competenze avanzate del linguaggio Visual C#.

Circuito di prova (utilizzato per entrambe le soluzioni)



Soluzione A (Arduino)

Funzionamento

Ogni n millisecondi (ad esempio ogni 500 millisecondi):

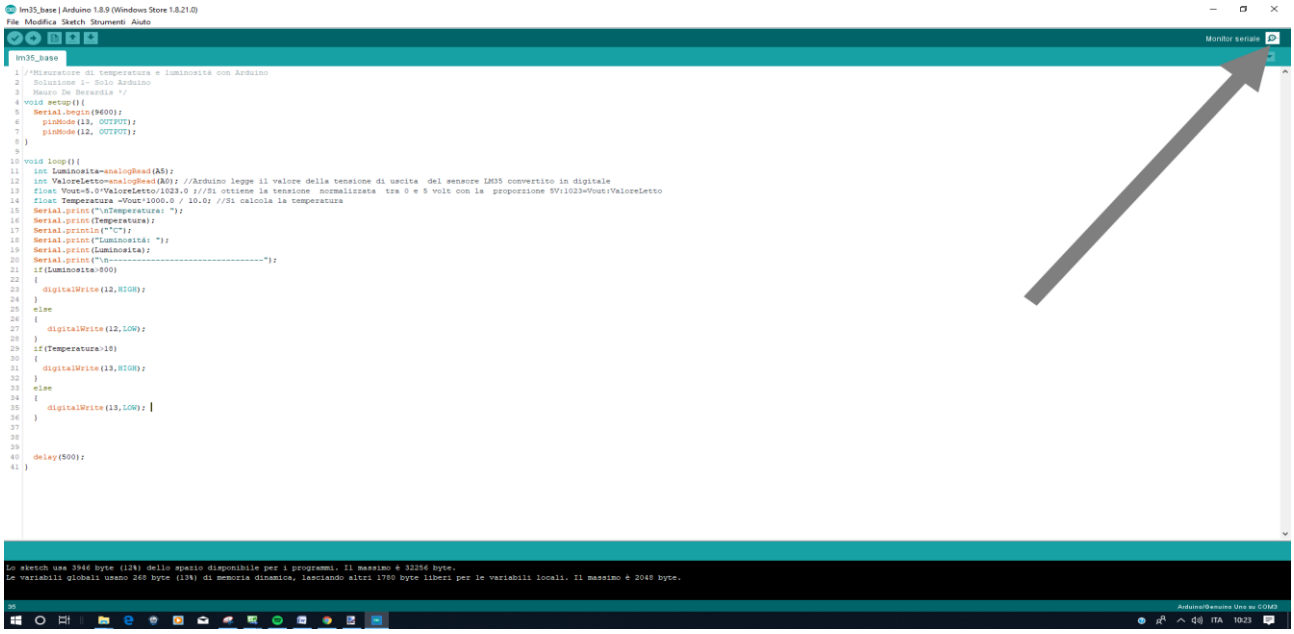
1. Arduino esegue le lettura degli ingressi analogici A0 e A5 e stampa i valori di temperatura e luminosità sul monitor seriale
2. Il led connesso al piedino D13 si accende quando la temperatura supera i 18 °C, il led connesso al piedino D12 si accende quando la luminosità supera il valore 800

Sketch Arduino (Sketch A)

```
// Misuratore di temperatura e luminosità con Arduino - Mauro De Berardis
// Soluzione A (Solo Arduino)

void setup(){
  Serial.begin(9600);
    pinMode(13, OUTPUT);
    pinMode(12, OUTPUT);
}
void loop(){
  // Lettura luminosità-----------------------------------------------
  // Arduino legge il valore della tensione del partitore Fotoresistenza-10K
  // convertito in digitale dall'ADC a 10bit
  int Luminosita=analogRead(A5);

  // Lettura temperatura----------------------------------------------
  // Arduino legge il valore della tensione di uscita  del sensore LM35
  // convertito in digitale dall'ADC a 10bit
  int ValoreLetto=analogRead(A0);
  //Si ottiene la tensione  normalizzata  tra 0 e 5 volt con la  proporzione
  //5V:1023=Vout:ValoreLetto
  float Vout=5.0*ValoreLetto/1023.0 ;
  float Temperatura =Vout*1000.0 / 10.0; //Si calcola la temperatura

  // Stampa dei valori di temperatura e luminosità sul monitor seriale
  Serial.print("\nTemperatura: ");
  Serial.print(Temperatura);
  Serial.println("°C");
  Serial.print("Luminosità: ");
  Serial.print(Luminosita);
  Serial.print("\n---------------------------------");

  //Controllo dei led su D13 e D12
  if(Temperatura>18)
  {
    digitalWrite(13,HIGH);
  }
  else
  {
     digitalWrite(13,LOW);
  }
  if(Luminosita>800)
  {
    digitalWrite(12,HIGH);
  }
  else
  {
     digitalWrite(12,LOW);
  }
  delay(500);  //il loop viene ripetuto ogni 500 millisecondi
}
```

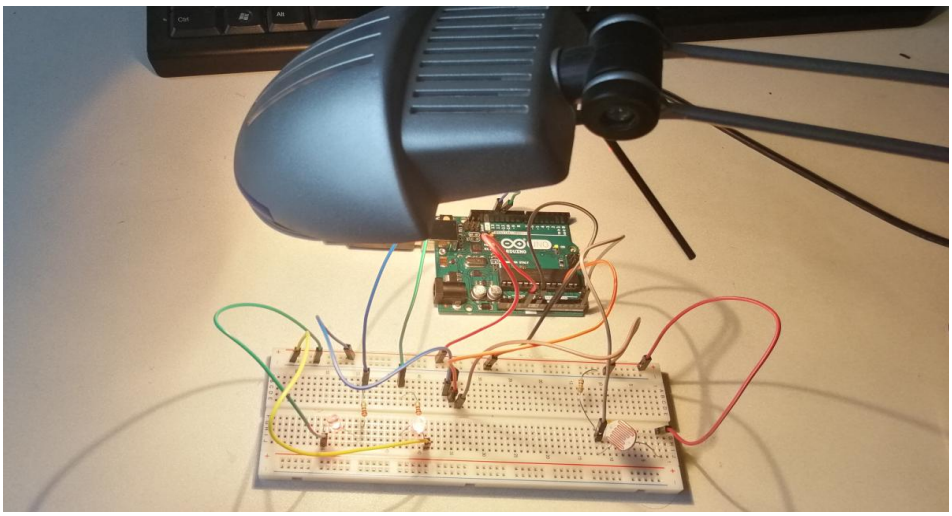
Prova della soluzione A

Una volta caricato lo sketch su Arduino, cliccando sull'icona del monitor seriale (come indicato in figura), visualizziamo i valori di temperatura e luminosità misurati da Arduino.



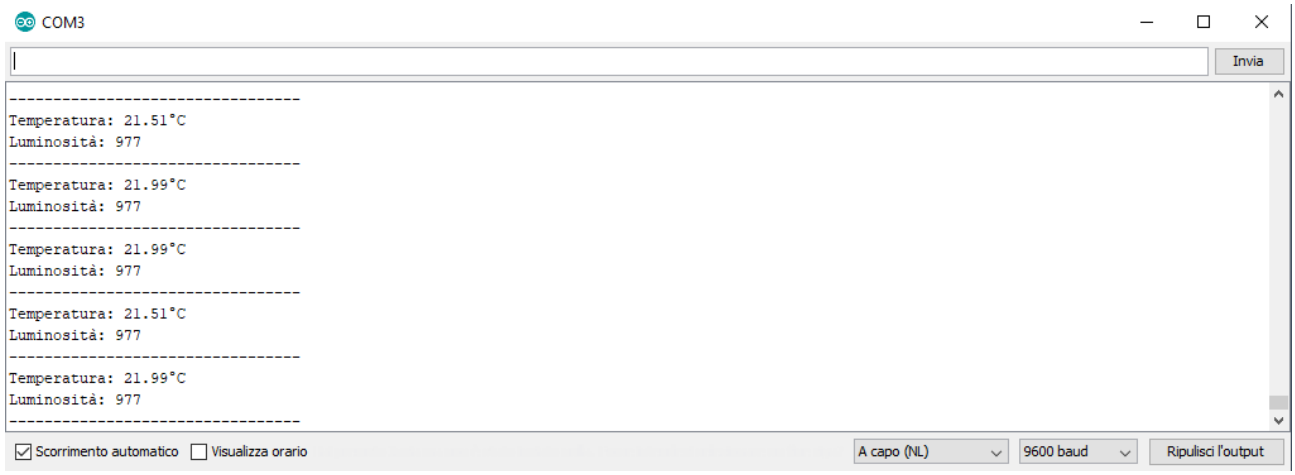
Sulla COM3 si leggono i valori misurati ogni 500 millisecondi: in una giornata fredda e grigia, la temperatura è al di sotto dei 18°C e la luminosità è al di sotto di 800. I due led sono pertanto spenti.

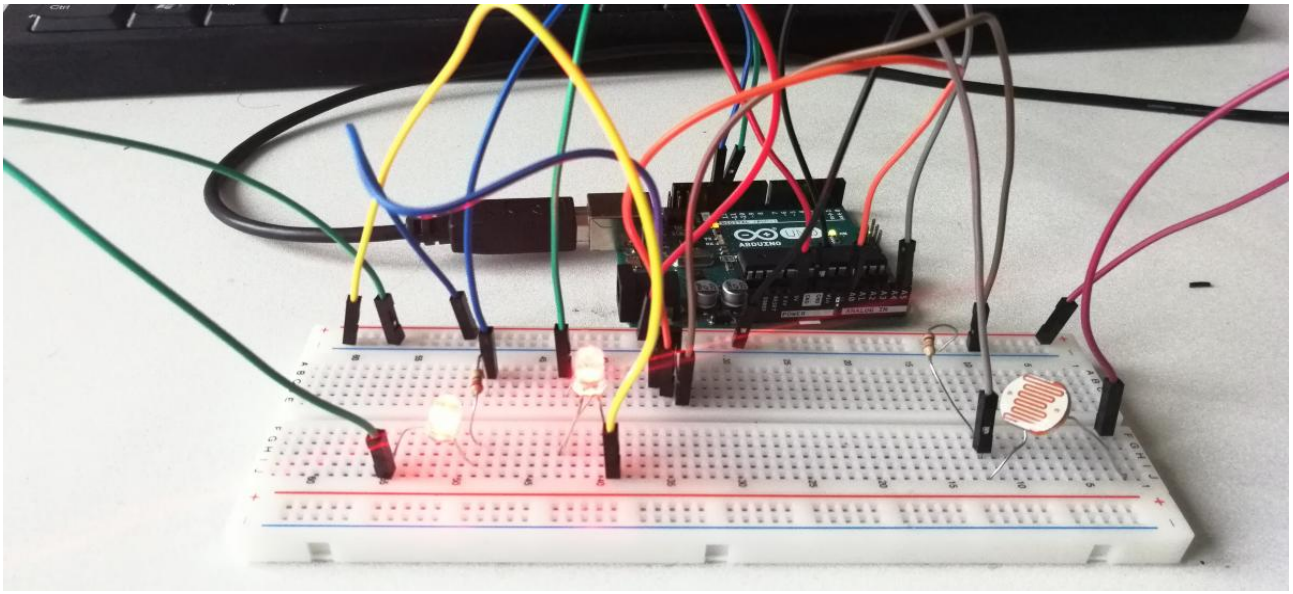
Con una lampada illuminiamo la fotoresistenza e allo stesso tempo, per l'effetto Joule, riscaldiamo il sensore di temperatura.



La luminosità sale rapidamente e quando supera il valore 800, si accende il led su D12. La temperatura aumenta più gradualmente rispetto alla luminosità e quando supera i 18°C, si accende il led su D13.

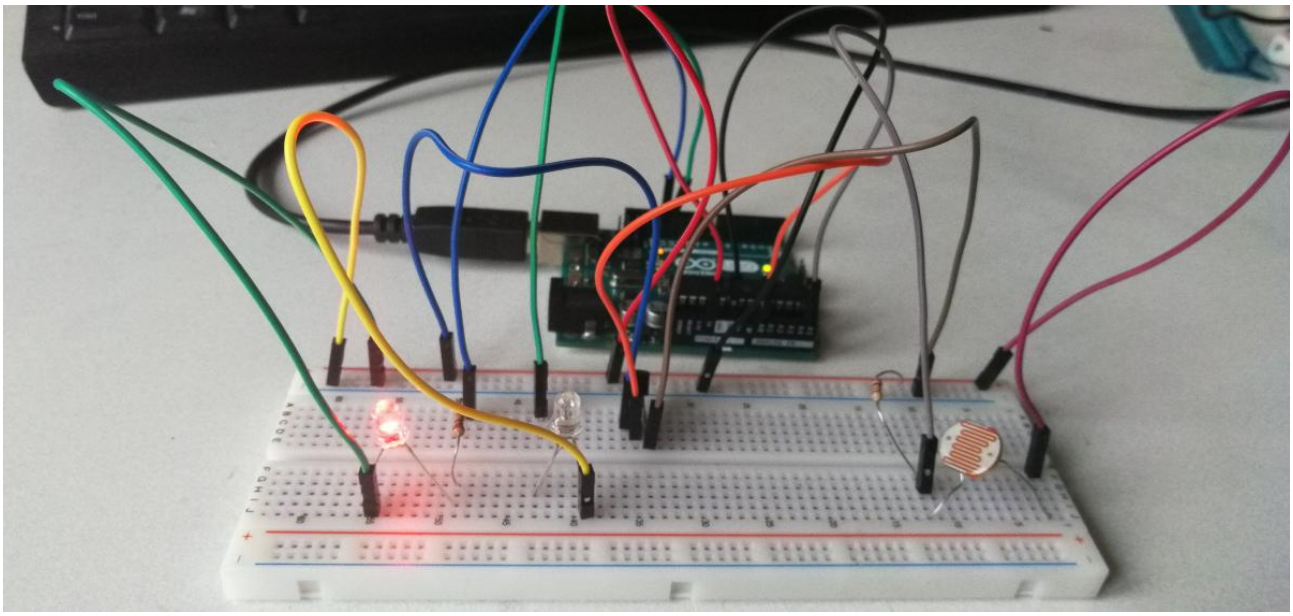
Dopo una decina di minuti, i due led sono accesi e sul monitor seriale leggiamo i seguenti valori:





Spegnendo la lampada, in pochi secondi la luminosità si abbassa al di sotto del valore di soglia 800 e il led su D12 si spegne. La temperatura varia più lentamente e comincia gradualmente ad abbassarsi.

Dopo pochi minuti, con il led della luminosità su D12 spento e il led della temperatura su D13 ancora acceso, sul monitor seriale leggiamo i seguenti valori:



Soluzione B (Arduino + Visual C#)

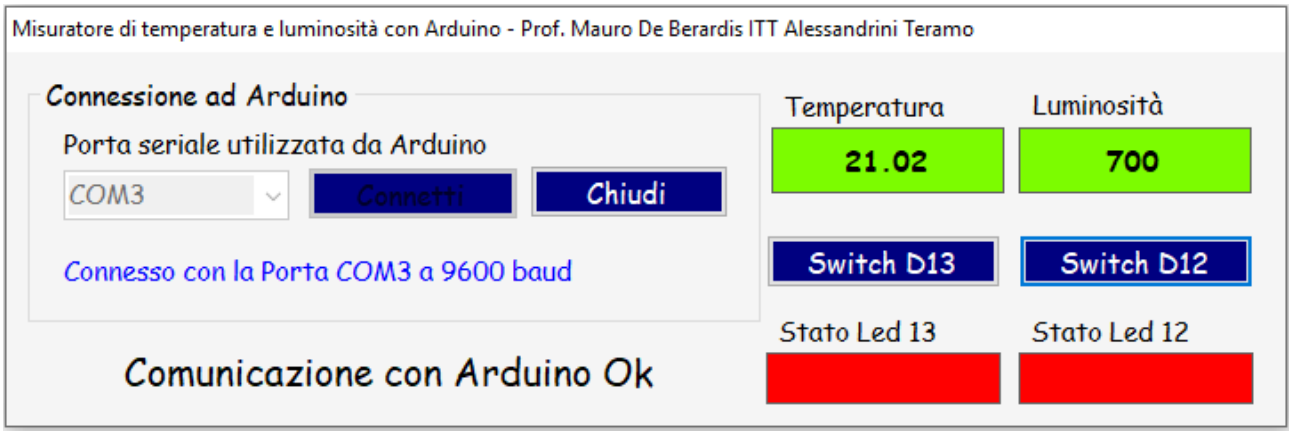
Funzionamento

Ogni n millisecondi (ad esempio ogni 500 millisecondi):

1. **Arduino** esegue la lettura degli ingressi analogici A0 e A5 e invia i valori letti sulla porta seriale tramite una stringa che possa essere letta dall'applicazione Visual C#. Allo stesso tempo rimane in attesa di eventuali comandi dall'applicazione Visual C# per accendere o spegnere i led sui piedini D13 e D12
2. L'applicazione **Visual C#** riceve e legge la stringa inviata da Arduino, visualizza la temperatura e la luminosità correnti, e invia i comandi di accensione/spegnimento dei due led di Arduino.

Il progetto è volutamente semplice ma lascia intuire come interfacciare Arduino con un linguaggio di alto livello (tipo Visual C#), consenta elaborazioni complete e sofisticate dei dati, grafici compresi, che sarebbero impossibili da realizzare solo con Arduino.

La costruzione del Form è semplice ed immediata. Tutti i controlli utilizzati sono facilmente riconoscibili e il codice Visual C# è ampiamente commentato.



- Attraverso la comboBox **comboSeriali** e il bottone **bConnetti,** l'applicazione si connette alla porta seriale utilizzata da Arduino (la porta si può determinare facilmente aprendo il monitor della soluzione 1) e legge le stringhe inviate da Arduino, le elabora e visualizza i valori di temperatura e luminosità
- I bottoni **SwitchD13** e **SwitchD12** consentono, se l'applicazione è connessa ad Arduino, di accendere/spegnere i led collegati ai piedini D13 e D12 di Arduino. Lo stato ON/OFF dei due Led di Arduino, viene replicato sul form tramite le label **lD13** e **lD12.**

Per provare la soluzione, occorre mandare in esecuzione lo sketch di Arduino (Sketch B) e quindi l'applicazione Visual C#.

Se si tenta di caricare lo sketch su Arduino e l'applicazione Visual C# è già in esecuzione, si verifica un errore.

Sketch Arduino (Sketch B)

```
// Misuratore di temperatura e luminosità con Arduino - Mauro De Berardis
// Soluzione B (Arduino + Visual C#)
void setup(){
  Serial.begin(9600);
    pinMode(13, OUTPUT);
    pinMode(12, OUTPUT);
}

void loop(){
  // Lettura luminosità-----------------------------------------------
  // Arduino legge il valore della tensione del partitore Fotoresistenza-10K
  // convertito in digitale dall'ADC a 10bit
  int Luminosita=analogRead(A5);

  // Lettura temperatura----------------------------------------------
  // Arduino legge il valore della tensione di uscita  del sensore LM35
  // convertito in digitale dall'ADC a 10bit
  int ValoreLetto=analogRead(A0);
  //Si ottiene la tensione  normalizzata  tra 0 e 5 volt con la  proporzione
  //5V:1023=Vout:ValoreLetto
  float Vout=5.0*ValoreLetto/1023.0 ;
  float Temperatura =Vout*1000.0 / 10.0; //Si calcola la temperatura

   // Invia stringa sulla seriale-------------------------------------------
  Serial.println((String)Temperatura+"#"+String(Luminosita)+"#"+"Arduino");

  // Legge ed esegue i comandi eventualmente  inviati dall'applicazione Visual c#
  // sulla seriale per accendere/spegnare i led su D13 e D12----------------------
  String rx="";
  if (Serial.available() > 0)
    {
       rx = Serial.readString();

       if(rx=="ON#ON")
        {
          digitalWrite(13, HIGH);
          digitalWrite(12, HIGH);
        }
       if(rx=="ON#OFF")
        {
          digitalWrite(13, HIGH);
          digitalWrite(12, LOW);
        }
       if(rx=="OFF#ON")
        {
          digitalWrite(13, LOW);
          digitalWrite(12, HIGH);
        }
       if(rx=="OFF#OFF")
        {
          digitalWrite(13, LOW);
          digitalWrite(12, LOW);
        }
    }
  delay(500); //il loop viene ripetuto ogni 500 millisecondi
}
```

Codice Visual C#

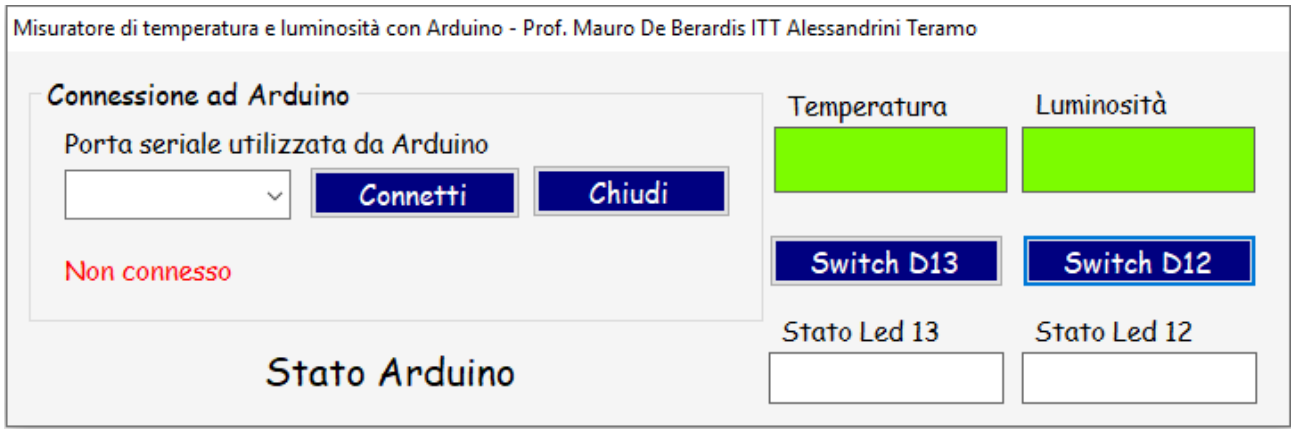
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.IO.Ports;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        SerialPort Porta;
        String D13 = "OFF";
        String D12 = "OFF";
        private delegate void MioDelegato(string s);
        private void comboPorteSeriali_DropDown(object sender, EventArgs e)
        {
            comboPorteSeriali.Items.Clear();
            for (int i = 1; i <= 10; i++)
            {
                comboPorteSeriali.Items.Add("COM" + i.ToString());
            }
            // la porta a cui è connesso Arduino è quella in cui vengono
            // monitorati i dati dello Sketch A
        }

        private void comboPorteSeriali_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.KeyChar = (char)0;//impedisce di scrivere nella comboBox
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            lConnessione.ForeColor = Color.Red;
            lConnessione.Text = "Non connesso";
        }
        private void bConnetti_Click(object sender, EventArgs e)
        {
            if (comboPorteSeriali.Text == "")
            {
                MessageBox.Show("Selezionare la porta", "Attenzione");
                return;
            }
            Porta = new SerialPort();
            Porta.PortName = comboPorteSeriali.Text;
            try
            {
                Porta.Open();
            }
            catch (Exception)
            {
             MessageBox.Show("La porta non esiste oppure è occupata", "Attenzione");
                lConnessione.ForeColor = Color.Red;
                lConnessione.Text = "Non connesso";
                comboPorteSeriali.Text = "";
                return;
            }
```

```
            lConnessione.ForeColor = Color.Blue;
            lConnessione.Text = "Connesso con la Porta ";
             lConnessione.Text+=Porta.PortName + " a "+ Porta.BaudRate.ToString()+ " baud";
            Porta.DataReceived += DatiArduino;
  // All'evento DataReceived dell'oggetto SerialPort viene richiamato il metodo DatiArduino
  // che però gira su un altro thread. Pertanto bisogna  ricorrere a un delegato.
  // Il Delegato punta ad un metodo e deve essere dichiarato.
        }
        // Procedura eseguita quando vengono ricevuti i dati sulla seriale
       void DatiArduino(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)
        {
           string stringa = "";
           try
            {
                if (Porta.IsOpen)
                {
                    stringa = Porta.ReadLine();
                    this.BeginInvoke(new  MioDelegato(ElaboraDatiLetti),stringa);
                    // BeginInvoke richiama in maniera asincrona e in un thread separato
                    // il metodo ElaboraDatiLetti tramite il delegato MioDelegato
                }
           }
            catch (Exception)
            {
                MessageBox.Show("Errore! Restart applicazione","Attenzione");
                Application.Restart();
            }
        }
       void ElaboraDatiLetti(string s)
        {
           // Se la stringa è stata inviata da Arduino, (per la stringa è stato scelto il
           // formato (String)Temperatura+'#'+(String)Luminosità+'#'+"Arduino") i valori di
           // temperatura e luminosità vengono 'splittati' e visualizzati nelle Label
           // lTemperatura e lLuminosità.I dati potrebbero essere stampati su una ListBox,
           // memorizzati su un file di testo oppure
            // su una tabella MySql per successive elaborazioni (medie, grafici ecc.)
            if (s.Contains("Arduino"))
            {
                string[] v = s.Split('#');

                lArduino.Text = "Comunicazione con Arduino Ok";
                bConnetti.Enabled = false;
                comboPorteSeriali.Enabled = false;
                lTemperatura.Text = v[0];
                lLuminosità.Text = v[1];
            }
            else
            {
                lArduino.Text = "Nessuna comunicazione con Arduino";
            }
        }
        private void SwitchD13_Click(object sender, EventArgs e)
        {
            // Se l'applicazione non è collegata ad Arduino è impossibile inviare
            // comandi da Visual C#
            if (lConnessione.Text == "Non connesso")
            {
                MessageBox.Show("Arduino non connesso", "Attenzione");
                return;
            }
            // Il bottone permette di inviare sulla seriale il comando per 'switchare'
            // il pin D13 di Arduino
```

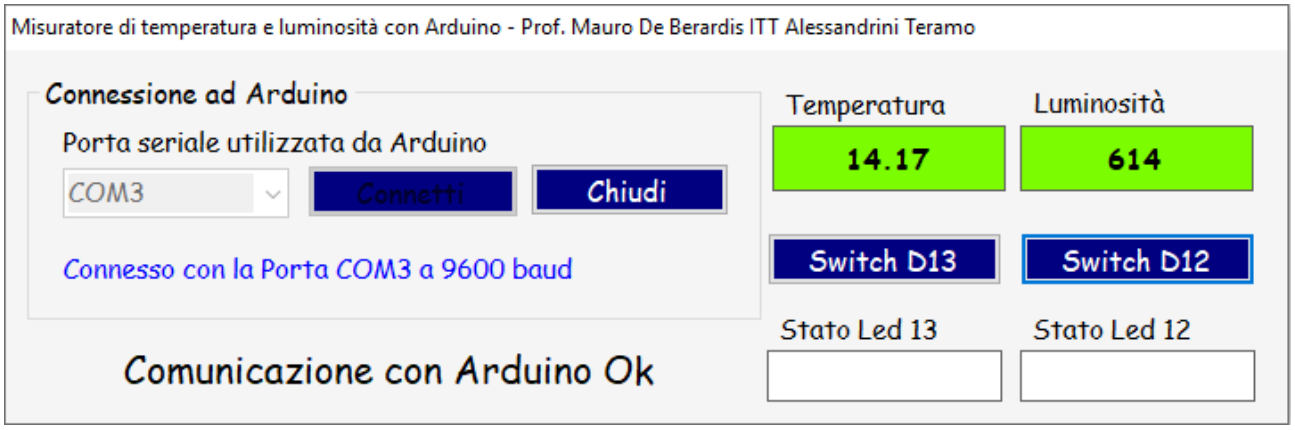
```
            if (Porta.IsOpen)
             {
                 if (D13 == "OFF")
                 {
                     D13 = "ON";
                     lD13.BackColor = Color.Red;
                 }
                 else
                 {
                     D13 = "OFF";
                     lD13.BackColor = Color.White;
                 }
                 Porta.Write(D13+'#'+D12);
             }
        }

        private void SwitchD12_Click(object sender, EventArgs e)
        {
            // Se l'applicazione non è collegata ad Arduino è impossibile inviare
            // comandi da Visual C#
            if (lConnessione.Text == "Non connesso")
               {
                   MessageBox.Show("Arduino non connesso","Attenzione");
                   return;
               }
             // Il bottone permette di inviare sulla seriale il comando per 'switchare'
             // il pin D12 di Arduino
            if (Porta.IsOpen)
             {
                 if (D12 == "OFF")
                 {
                     D12 = "ON";
                     lD12.BackColor = Color.Red;
                 }
                 else
                 {
                     D12 = "OFF";
                     lD12.BackColor = Color.White;
                 }
                 Porta.Write(D13+'#'+D12);
             }
         }
        private void bChiudi_Click(object sender, EventArgs e)
        {
            if (lConnessione.Text == "Non connesso")
            {
                Application.Exit();
            }
            else
            {
                Porta.Write("OFF#OFF");
                lD13.BackColor = Color.White;
                lD12.BackColor = Color.White;
                Porta.Close();
                Application.Exit();
            }
        }
    }
}
```

Prova della soluzione B

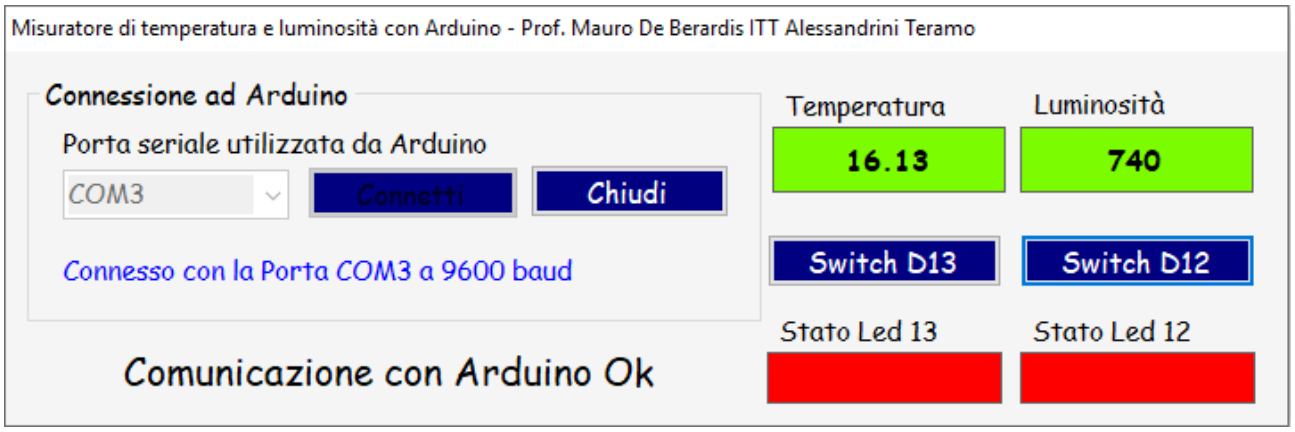
Dopo aver caricato lo sketch B su Arduino, eseguiamo l'applicazione Visual C#



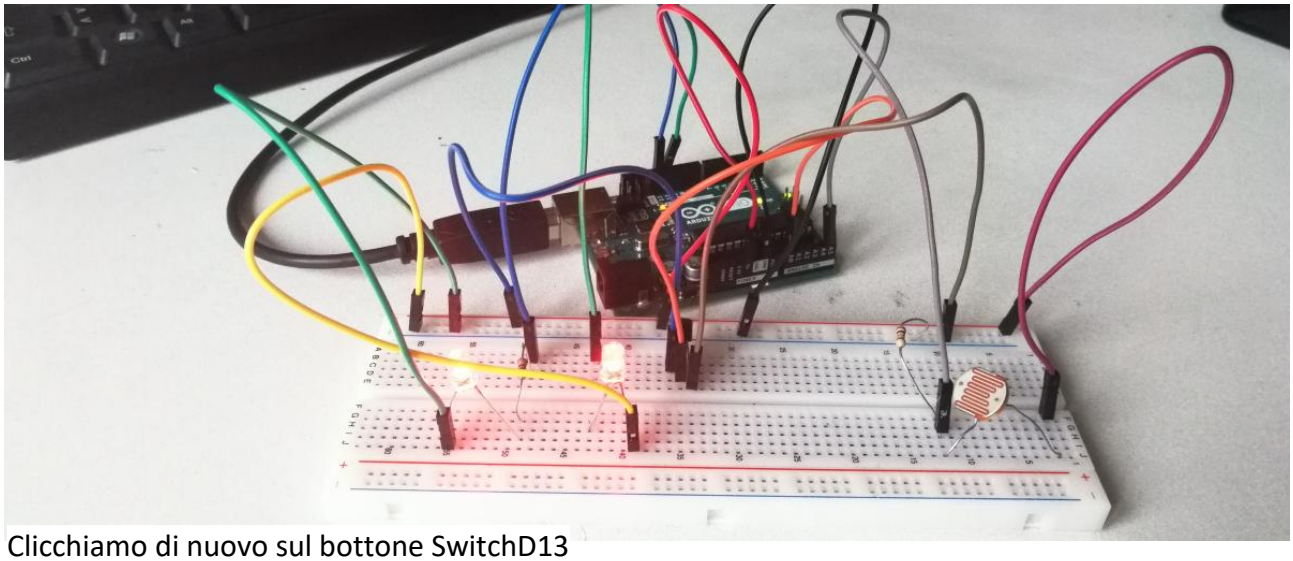
Apriamo la ComboBox, scegliamo la porta e ci connettiamo ad Arduino.
L'applicazione Visual C# legge i dati presenti sulla porta seriale trasmessi da Arduino ogni 500 millisecondi e li visualizza.



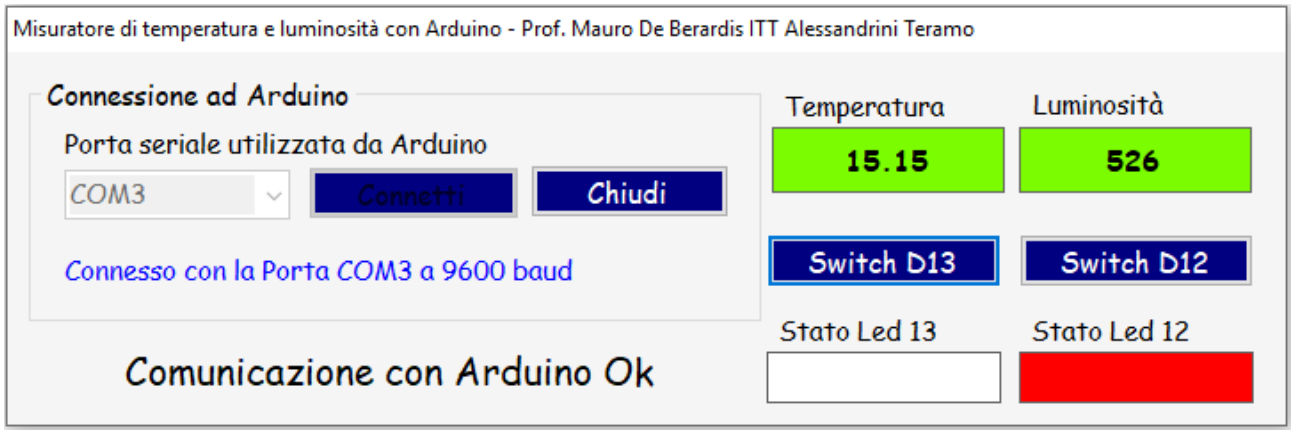
Tramite i bottoni SwitchD13 e SwitchD12 possiamo comandare ad Arduino di accendere o spegnere i led connessi sui piedini D13 e D12: in tal caso lo sketch di Arduino (Sketch B) legge la seriale ed esegue i comandi ricevuti. Ad esempio comandiamo di accendere entrambi i led.



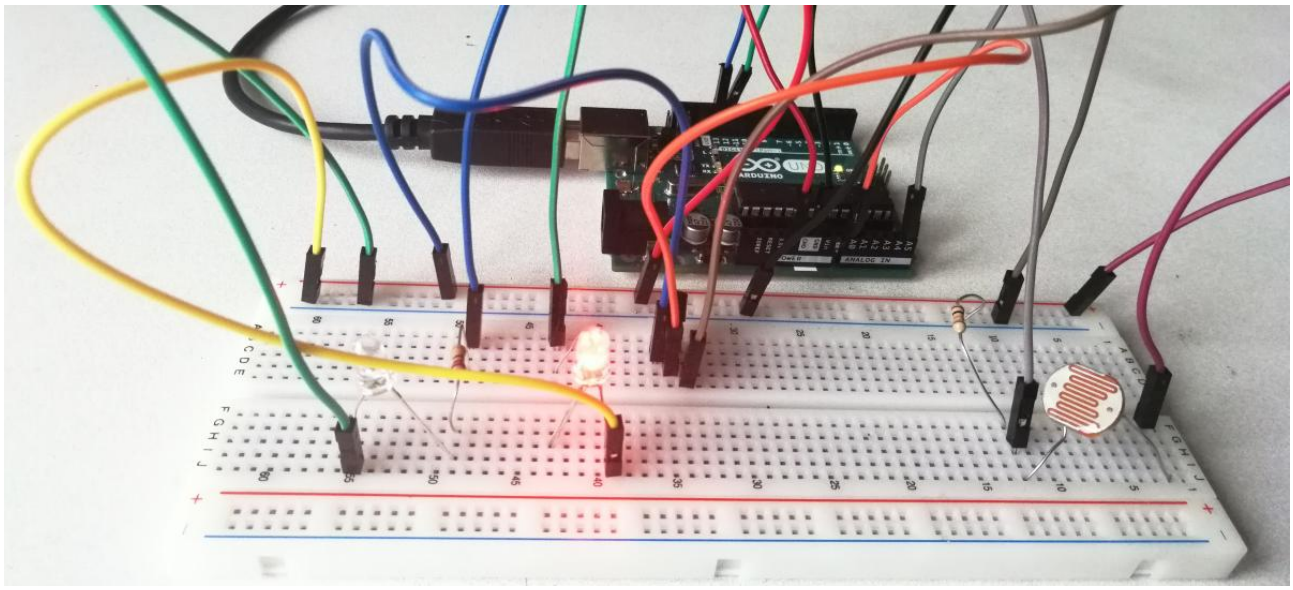
Lo stato diventa ON (rosso) per entrambi i led e, con un breve ritardo dovuto al fatto che i dati vengono trasmessi ogni 500 millisecondi, Arduino accende i due led.



Clicchiamo di nuovo sul bottone SwitchD13



Lo stato del Led 13 diventa OFF (bianco) e Arduino spegne il led su D13



La prova della Soluzione B è illustrata nel seguente video:
https://youtu.be/wbS6bOywXZs