

Programmare Arduino con Scratch



Scratch per Arduino

Andare sul sito s4a.cat e scaricare il programma **S4A**

Download and Install

- 1 Download and install **S4A**
- 2 Install our firmware into your **Arduino**
- 3 Enjoy programming your **world!**

Installing S4A requires you to install software both in your PC and your **Arduino** board. Here you'll find the detailed steps to get it up and running.

Installing S4A into your computer

S4A works in the three major consumer operating systems. Download and install the one that fits your configuration:

- [Windows](#)
- [Mac](#)
- [Linux \(Debian\)](#)
- [Linux \(Fedora\) \(version 1.5\)](#)
- [Raspbian \(Debian for RaspberryPi\) \(version 1.5\)](#)



Scegliere il proprio sistema operativo

Secondo passo

Installing the Firmware into your Arduino

This firmware is a piece of software you need to install into your Arduino board to be able to communicate with it from S4A.

- Download and install the Arduino environment by following the instructions on <http://arduino.cc/en/Main/Software>. Take in account Arduino Uno requires at least version 0022.
- Download our firmware from [here](#)
- Connect your Arduino board to a USB port in your computer
- Open the firmware file (S4AFirmware.ino) from the Arduino environment
- In the Tools menu, select the board and the serial port where the board is connected
- Load the firmware into your board through the Upload button

Salvare il file **S4AFirmware16.ino** sul nostro computer, ci servirà per far comunicare arduino con scratch.

Aggiornare il firmware di Arduino

HOME SOFTWARE PRODUCTS EDUCATION RESOURCES COMMUNITY HELP

SOFTWARE

Access the Online IDE

ARDUINO WEB EDITOR
Start coding online with the Arduino Web Editor, save your sketches in the cloud, and always have the most up-to-date version of the IDE, including all the contributed libraries and support for new Arduino boards. The Arduino Web Editor is one of the Arduino Create platform's tools.

Try It Now
Getting Started

Download the Arduino IDE

ARDUINO 1.8.5
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for installation instructions.

Windows Installer
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

Scaricare dal sito arduino.cc l'ide per programmare arduino in linguaggio C.

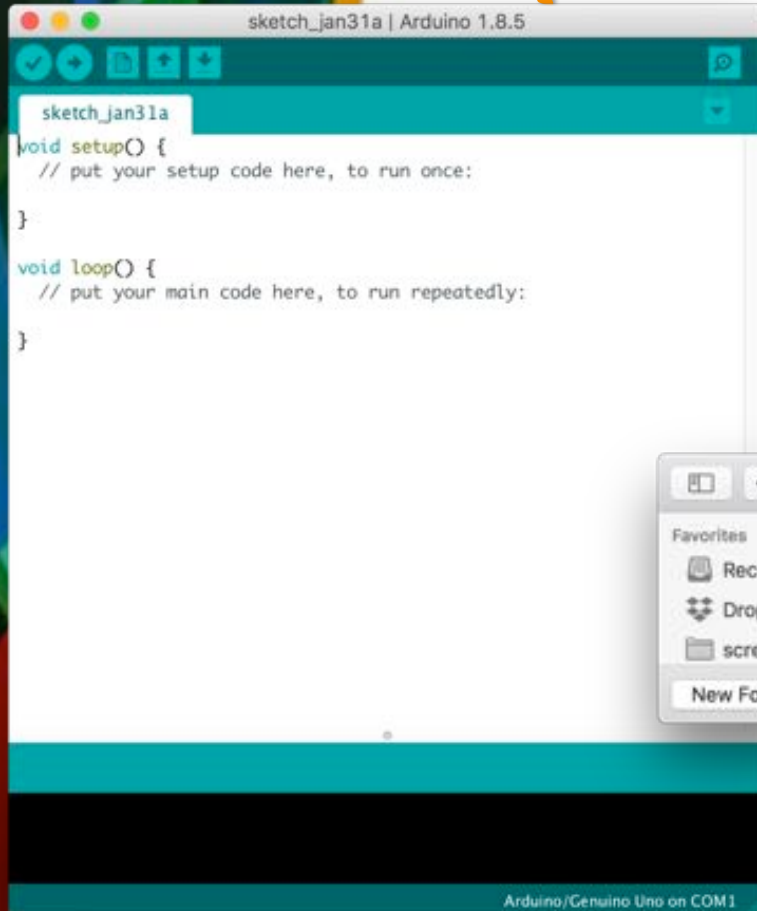
Scegliere il proprio sistema operativo

Far partire Arduino

File -> apri

E selezionare il file

S4AFirmware16.ino



```
SAASmartIO
SAASmartIO

// NEW IN VERSION 1.04 (by Jorge Gomez):
// Fixed variable type in pin structure: pin.state should be int, not byte
// Optimized speed of execution while receiving data from computer in readSerialPort()

// NEW IN VERSION 1.06 (by Jorge Gomez):
// Added new structure arduinoPin to hold the pins information:
// - This makes the code easier to read and modify (IMHO)
// - Allows to change the type of pin more easily to meet non standard use of SAA
// - Eliminates the need of having to deal with different kind of index access (ie: states[pin-K])
// - By using an enum to hold all the possible output pin states the code is now more readable
// Changed all functions using old style pin access: configurePin(), resetPins(), readSerialPort(), updateServoActuator() and sendDataActuator()
// Fixed possible overflow every 78 minutes (xk2 us) in pulse() while using write() (Changed for delayMicroseconds())
// Some minor coding style fixes

// NEW IN VERSION 1.06 (by Jorge Gomez):
// Fixed compatibility with Arduino Leonardo by avoiding the use of timers
// readSerialPort() optimized:
// - created state machine for reading the two bytes of the SAA message
// - updateActuator() is only called if the state is changed
// Memory use optimization
// Cleaning some parts of code
// Avoid using some global variables

// NEW IN VERSION 1.0:
// Refactored reset pins
// Merged code for standard and CR servos
// Merged patch for Leonardo from Peter Mueller (many thanks for this!)

// NEW IN VERSION 1.1:
// Changed pin 8 from standard servo to normal digital output

// NEW IN VERSION 1.4:
// Changed Serial.print() for Serial.write() in ScratchBoardSensorReport function to make it compatible with latest Arduino IDE (1.8)

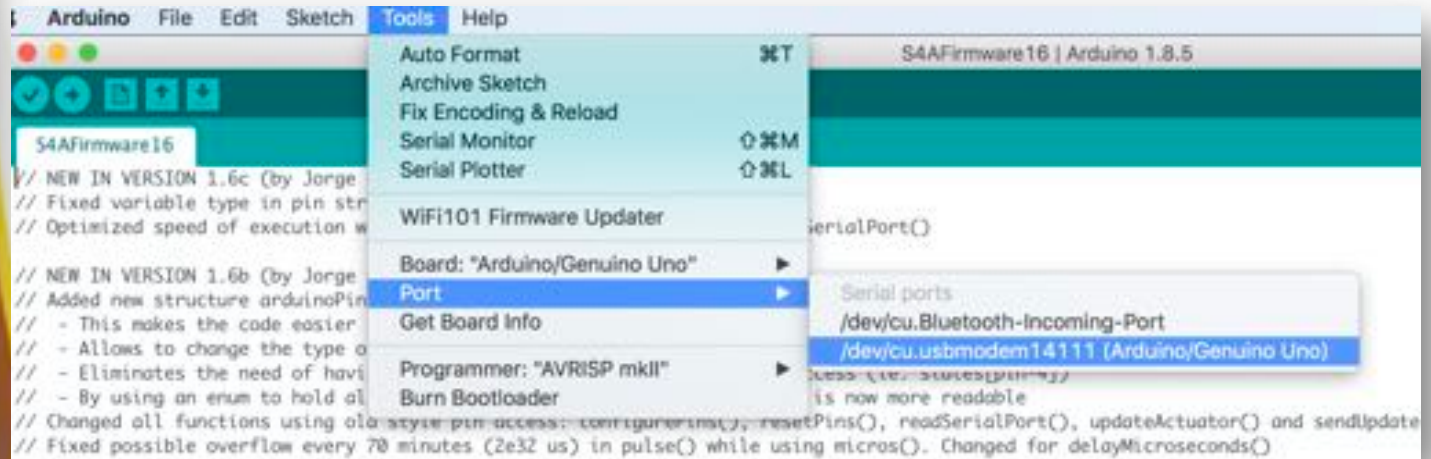
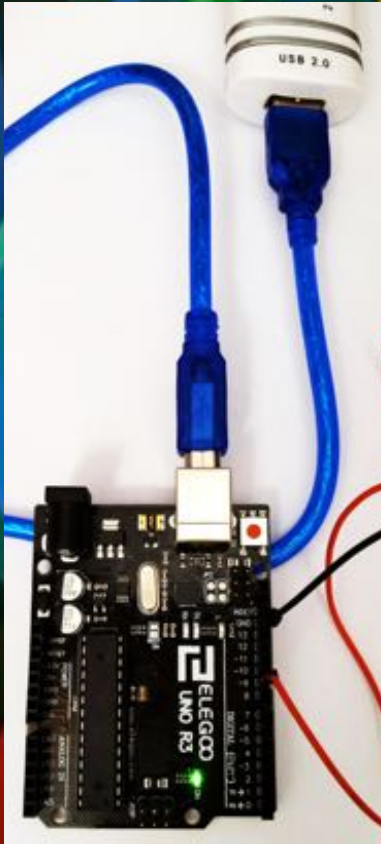
// NEW IN VERSION 1.3:
// Now it works on GNU/Linux. Also tested with MacOS and Windows 7.
// timer2 set to 20ms, fixing a glitch that made this period unstable in previous versions.
// readSerialPort() function optimized.
// pulse() modified so that it receives pulse width as a parameter instead using a global variable.
// updateServoMotors changes its name as a global variable had the same name.
// Some minor fixes.

typedef enum {
  INPUT, servomotor, pwm, digital }
pinType;

typedef struct pin {
  pinType type; //Type of pin
  int state; //State of an output
  //byte value; //Value of an input. Not used by now. TODO
};
```

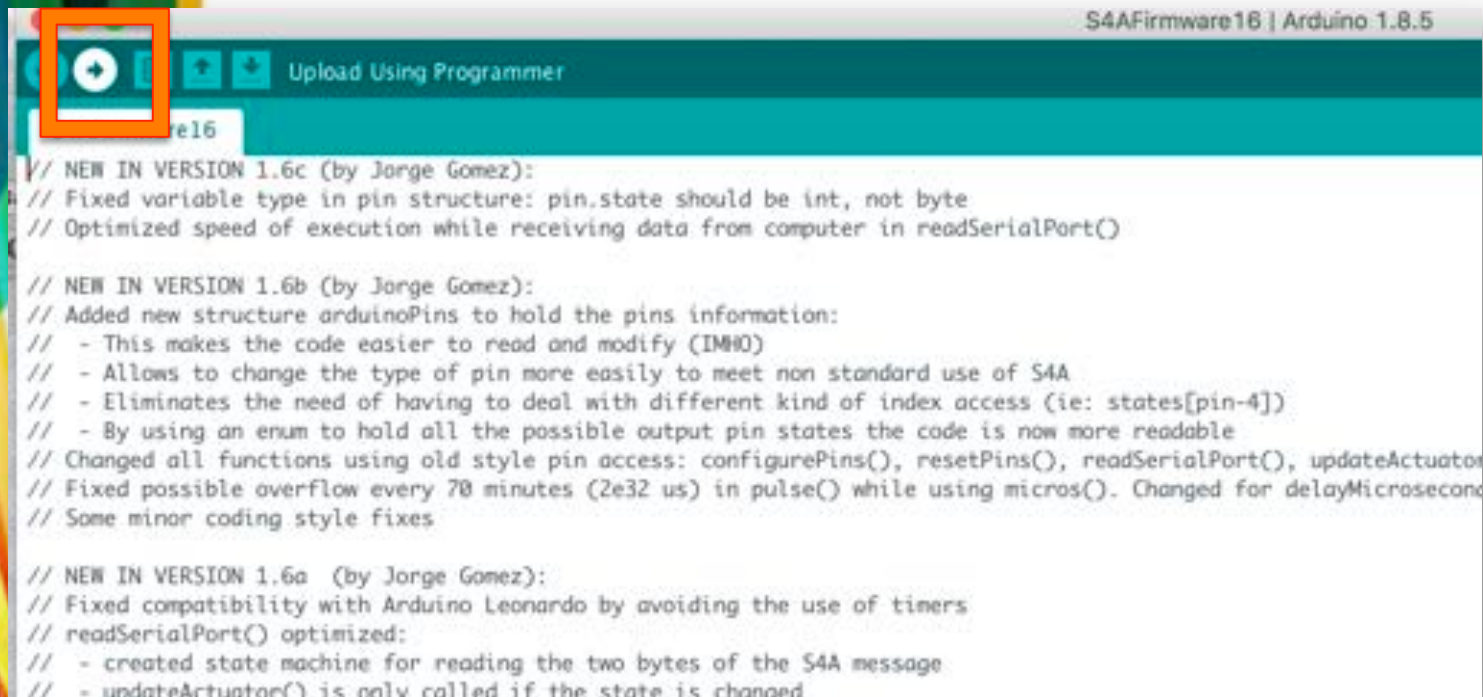
Collegare Arduino

Impostare la porta dall'IDE. **Tools**→ **Port** e selezionare la porta dove si trova Arduino, in genere viene indicato tra parentesi.



Aggiornare il firmware

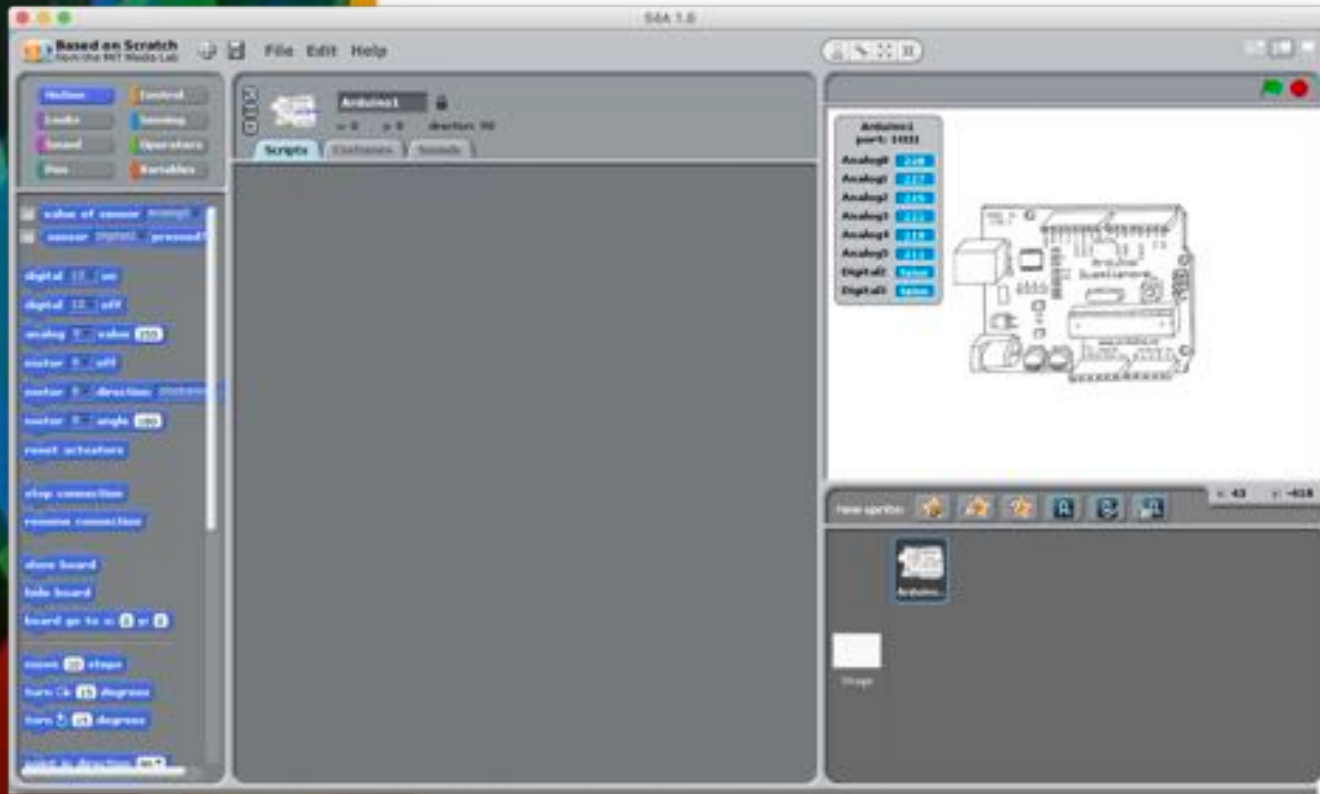
Fare upload del file **S4AFirmware16.ino** sulla scheda arduino



The screenshot shows the Arduino IDE interface with the file 'S4AFirmware16.ino' open. The 'Upload' button, represented by a green arrow icon, is highlighted with an orange rectangular box. The top of the IDE shows the title bar 'S4AFirmware16 | Arduino 1.8.5' and the menu bar with 'File', 'Edit', 'Tools', and 'Help'. Below the menu bar is a toolbar with various icons, including the 'Upload' button. The main text area displays the source code for the firmware, which includes comments about version updates and code changes.

```
// NEW IN VERSION 1.6c (by Jorge Gomez):  
// Fixed variable type in pin structure: pin.state should be int, not byte  
// Optimized speed of execution while receiving data from computer in readSerialPort()  
  
// NEW IN VERSION 1.6b (by Jorge Gomez):  
// Added new structure arduinoPins to hold the pins information:  
// - This makes the code easier to read and modify (IMHO)  
// - Allows to change the type of pin more easily to meet non standard use of S4A  
// - Eliminates the need of having to deal with different kind of index access (ie: states[pin-4])  
// - By using an enum to hold all the possible output pin states the code is now more readable  
// Changed all functions using old style pin access: configurePins(), resetPins(), readSerialPort(), updateActuator  
// Fixed possible overflow every 70 minutes (2e32 us) in pulse() while using micros(). Changed for delayMicroseconds  
// Some minor coding style fixes  
  
// NEW IN VERSION 1.6a (by Jorge Gomez):  
// Fixed compatibility with Arduino Leonardo by avoiding the use of timers  
// readSerialPort() optimized:  
// - created state machine for reading the two bytes of the S4A message  
// - updateActuator() is only called if the state is changed
```

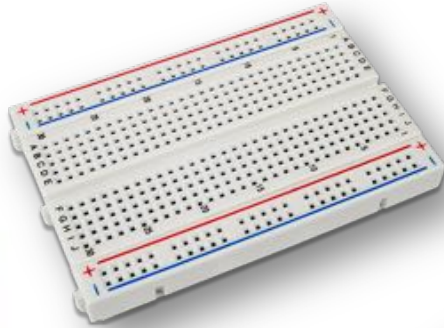
Ora apriamo S4A



S4A, alla partenza, imposta la porta verso Arduino e si vedono i sensori Analog cambiare velocemente

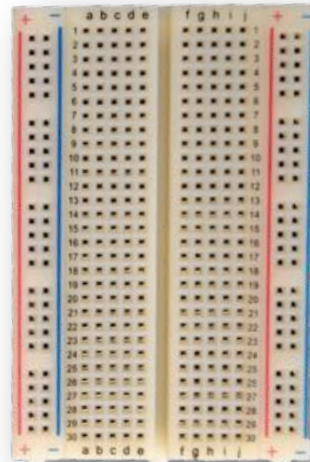
Primo progetto S4A

Cosa ci serve:



Impariamo ad usare la breadbord

Impariamo ad usare i collegamenti della breadboard

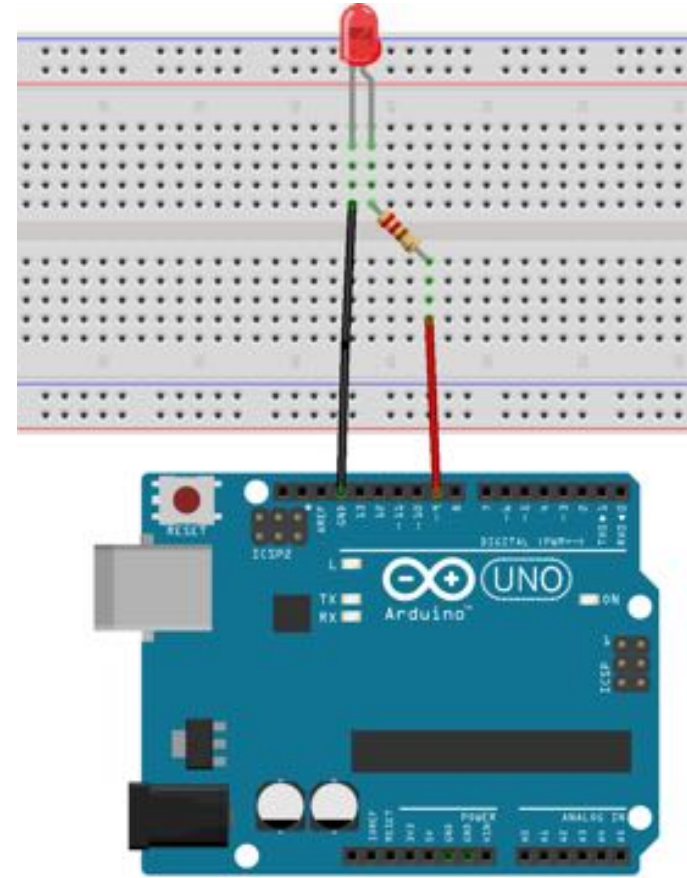


Due binari collegati di lungo

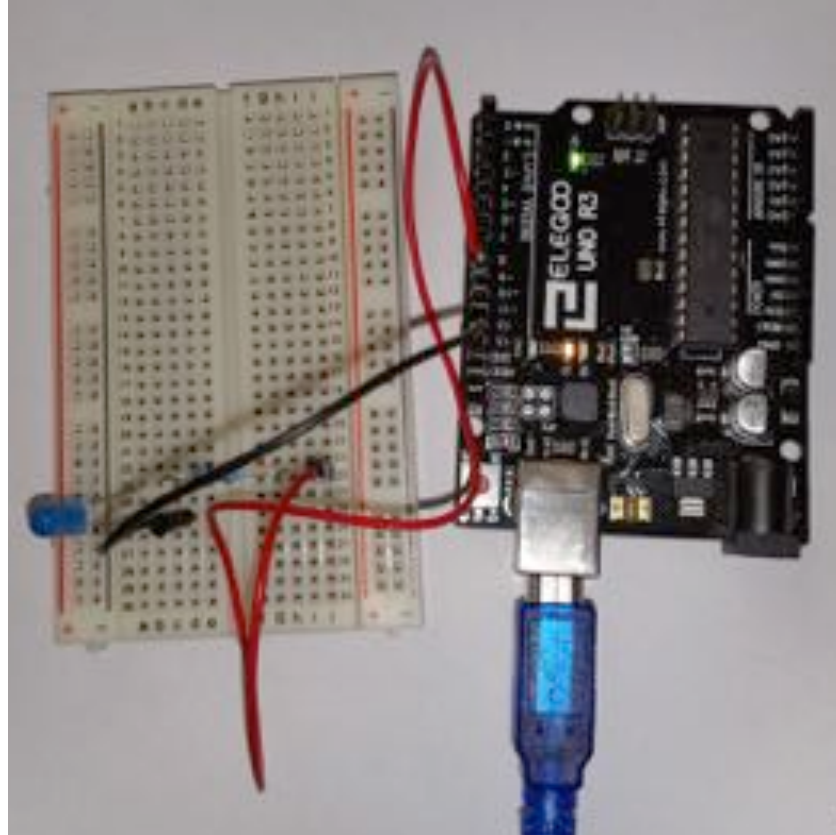
Tanti collegamenti in orizzontale

Collegamenti

Colleghiamo il led con un filo che va nel pin 9 e un cavo nero che va a ground (terra).

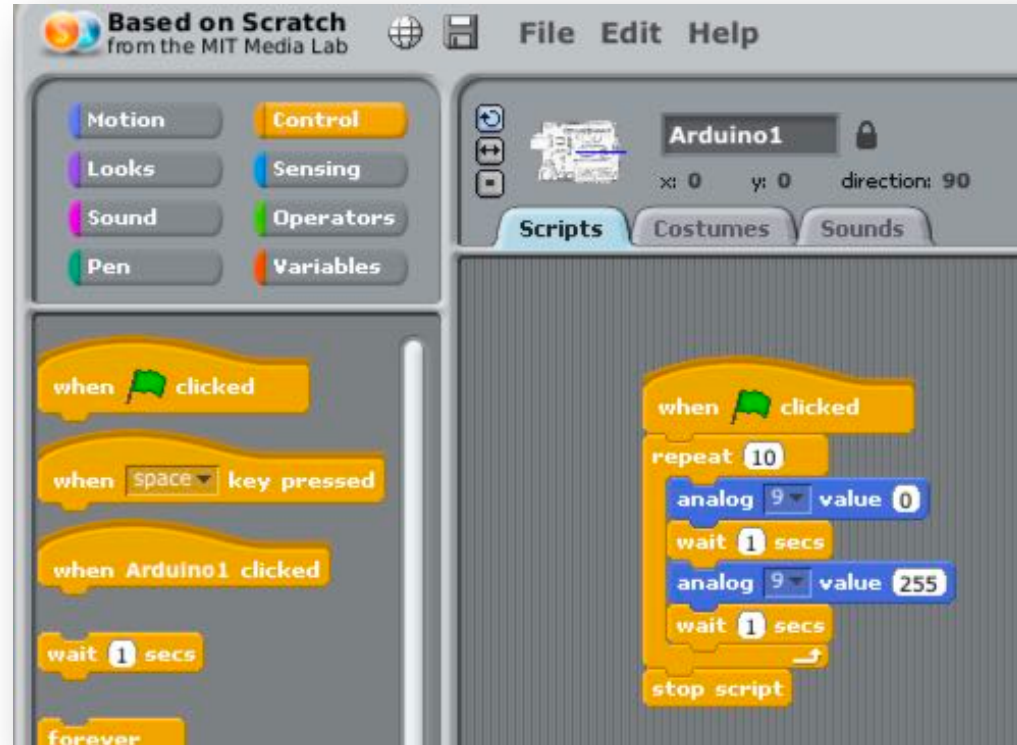


Collegamenti reali

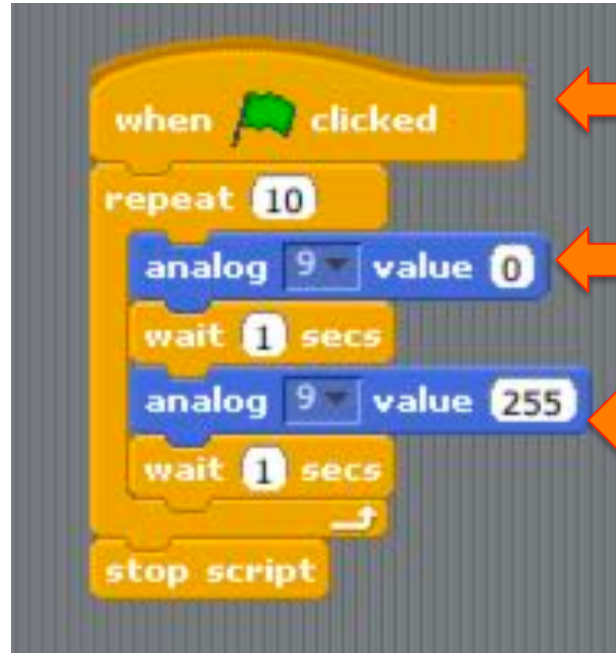


Codice S4A

Per accendere e spegnere il led



Codice S4A



Quando si clicca la bandierina verde il programma parte

Sul pin 9 si invia 0, il led si spegne

Sul pin 9 si invia 255, il led si accende

Modifichiamo i valori



Sul pin 9 si invia 0,
il led si spegne

Sul pin 9 si invia
255, il led si spegne

Sul pin 9 si invia
100, il led è fiasco