GCSE Computer Science

Arduino programming exercise: LED operated by a button

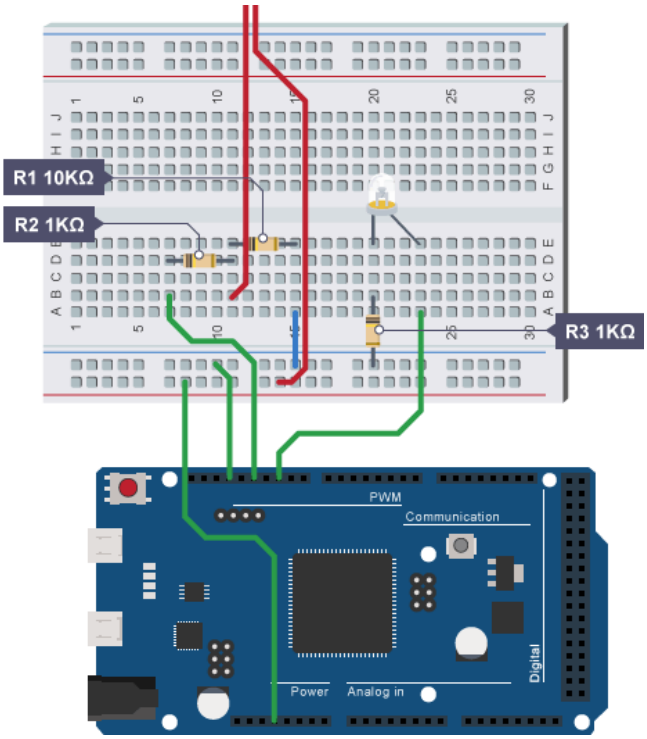
This guide explains how to set up a circuit using an Arduino, a switch and an LED. It will be programmed so that each time the button is pressed it toggles the light on or off.

Equipment

- a computer such as a PC
- an Arduino Due (the same circuit should work with other Arduino boards but make a careful note of which connectors are being used and alter your program as necessary)
- one breadboard – the breadboard should have numbers, letters and positive and minus signs above the rows and columns. These will be referenced as grid positions in the following instructions
- five male-to-male breadboard connector wires (the turquoise and blue lines in the diagram)
    - one of length approximately 10 cm
    - three of length approximately 5 cm
    - one of length approximately 1 cm
- two 1 K ohm resistors
- one 10 K ohm resistor
- an LED
- a push button switch (you can just touch the ends of the two wires together if a suitable switch is not available – these are the red lines in the diagram)
- a USB lead

If this is the first time you have connected an Arduino to a computer, you may need to install a driver. You will also need to install the Arduino IDE which you can download from the Arduino website.

The following diagram shows how the boards should be setup:



Method

1. Connect the breadboard to the Arduino.
    a. Connect the Arduino to the breadboard using the wires, as follows:

Wire	**Arduino pin**	**Breadboard point (column, row)**
10 cm jumper	3 V (Power)	Positive, 16
5 cm jumper	Digital pin 12 (PWM)	B,15
5 cm jumper	Digital pin 10 (PWM)	A,31
5 cm jumper	GND (digital GND, not the power GND)	Negative, 18

b. Add the resistors to the breadboard, as follows:

Resistor	Breadboard point A (column, row)	Breadboard point B (column, row)
10k ohm	E,19	E,22
1k ohm	D,15	D,19
1k ohm	B,28	Negative, 28

c. Connect the red switch wires to the breadboard:

   i. one in the positive column, row 22.

   ii. one in B19.

d. Connect the LED to the breadboard:

   i. the shortest straight leg in E28.

   ii. the longer leg in E31.

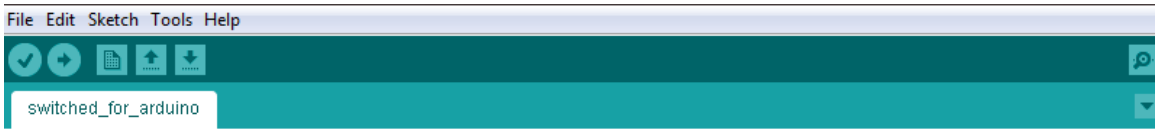
2. Connect the Arduino to the computer using a USB cable.

   a. Start up the Arduino IDE on the computer.

3. Write the program code.

   a. Enter the code, exactly as shown below.

   NOTE: This set of code includes lines which start with //. These markers tell us that this line is a comment to communicate messages to people using the code. They are not actual lines of code that will be executed by the compiler. They therefore do not need to be typed in.




```
//Set up some global variables which will be accessible from any part of the program
//The button is connected to pin 12
int pushButton = 12;
//The LED is on pin 10
int LED = 10;
//The variable we are going to read the button pressed state into is initially set to 0
//meaning not pushed
int button = 0;
//To begin with we want the LEDto be off so we set lightOn to false
bool lightOn = false;

void setup() {
  // put your setup code here, to run once:
  // The LED is for output (sets pin 10 from above to be an Output pin)
  pinMode(LED, OUTPUT);
  // The switch is an input (sets pin 12 from above to be an Input pin
  pinMode(pushButton, INPUT);
}


void loop() {
  // put your main code here, to run repeatedly:
  //Read whether the button has been pushed (has pin 12 got a HIGH signal, otherwise
  // known as a 1 or true)
  button = digitalRead(pushButton);
  //If the button has been pushed execute the code in the curly brackets otherwise ignore it
  if (button == true)
  {
    //The button has been pushed so change the state of the lightOn variable ! means change
    //from false to true or vice versa
    // in technical terms it is known as a logical not
    lightOn = !lightOn;
    // Wait 200 milliseconds (1/5 of a second) to give a chance for the user to let the button
    // go otherwise the button gets multiple times and toggles more than once. Experiment with
    // this value try 1 or 10 and see what happens, try 100, then try 1000. In technical terms
    //this is a kludge to avoid having to "debounce" the button, debouncing buttons can get very
    // technical but is essential knowledge when designing hardware.
    delay(200);
  }


  //This code is execute every time either making sure the light is on or it is off
  //If the lightOn value is true turn on the light
  if (lightOn)
  {
    // Send power through pin 10 turning on the light
    digitalWrite(LED, HIGH);
  }
  else
  {
    // Stop power from passing through pin 10 turning off the light
    digitalWrite(LED, LOW);
  }
  //Go back to the start of the loop and do it all over again.
}
```

4. Save this set of code with a sensible name such as **switched_for_arduino**. This will automatically save the file with a **.ino** ending in a folder of the same name as the file.

5. Click on the circle with the arrow pointing to the right in the Tool Bar, this will compile your code, upload it to the Arduino and start it running.

6. If you press the button briefly it should toggle the LED, if you hold the button it will flash the LED slowly. Experiment with the delay timing in the code.