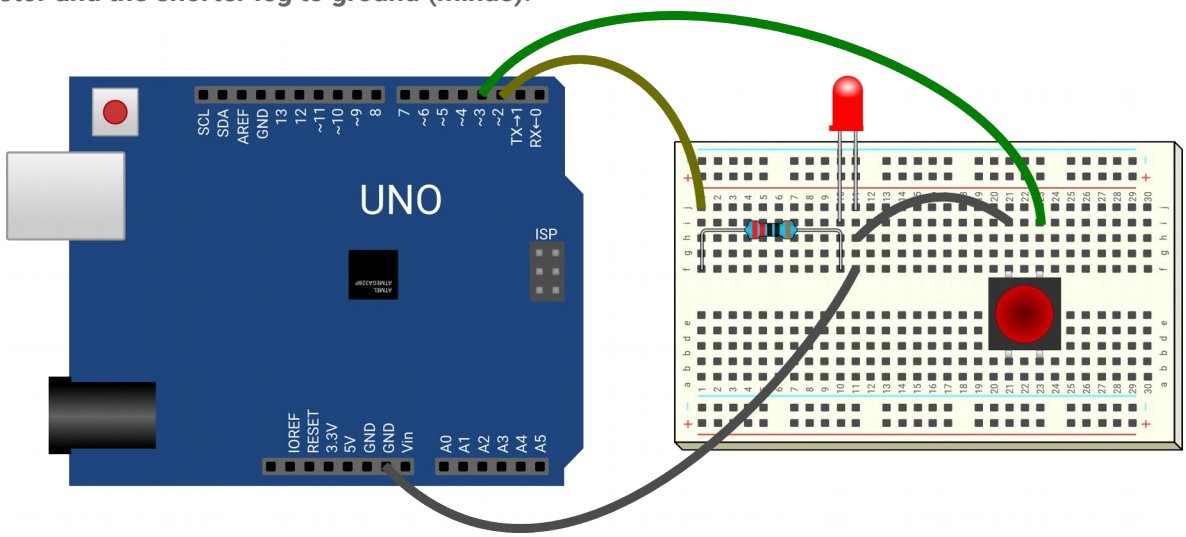
Arduino-project 003 – Toggling LED

This project shows the basics of how to get your Arduino to react to external events. In this case we use a push button to toggle a LED on and off. The circuit is similar to that for projekct 001, but we've added a push button.

The circuit

Just as in project 001 the LED has to be connected in the right direction, i.e. the long leg to the resistor and the shorter leg to ground (minus).



At startup

We begin our program by setting the in- and outputs. The input is set to "INPUT_PULLUP" which means that the internal so called pull-up-resistor is activated for this input. This is simply a resistor of a few kiloohms connected between the input and plus (Vcc) guaranteeing that the input will always have a high value (logic one) if nothing else is connected. The inputs of Arduinons are very sensitive and if nothing is connected to them they will pick up interference from light fixtures, radio transmitters and other electrical devices leading to the input toggling between high and low on its own randomly. The pull-up-resistor prevents this by forcing the input to always be a one if nothing else is connected. In our circuit the input is connected to a push button, but when the button isn't pressed there is no voltage on the input and without the pull-up-resistor the input would, as mentioned, catch random interference. When the button actually is pushed it will pull the input to ground and become a logic zero. You might think that this would cause a short circuit as the input is already connected to plus, but since the connection to plus is through a rather large resistor very little current will actually flow through the resistor and the chip will not be damaged and there will be no short cicuit.

The main program

The rest of the program consists of Arduinos usual loop()-function that runs its code over and over again as fast as possible. The loop reads the state of the push button and if the button is pressed the input will become low. That the input becomes low may seem a bit weird but it is a consequence of the pull-up-resistor always forcing the input to plus (high) and the push button therefore has to be connected to ground. If the input is low the code checks if the variable `led_state` is 1 or 0.

`led_state` is then changed to the opposite value and the output for the LED is set to either high or low. The last thing that happens is that the code waits 250ms before continuing. This short delay is required because the micro controller is incredibly fast and can run the code in loop() thousands of times per second and if the button is pressed the LED will toggle between on and off so quickly it is impossible to see. The delay also protects us from another phenomena that is a common trap for beginners: contact bounce. When a push button is pressed or a switch is toggled it doesn't happen cleanly but the contact elements will bounce against each other a little. To the human eye it happens so fast that we never notices it, but to a micro controller that can read the button many thousand times a second it will look as if you're pressing and releasing the button many times very fast. This can be solved using electronics, but if you're using a micro controller it's easier and cheaper to fix it using code instead. There are many ways to solve it and a delay as in our example is one of the simplest, but it works well for our application.