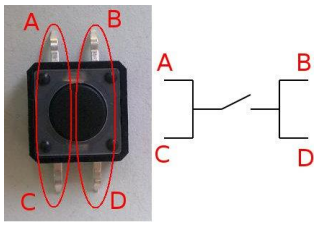
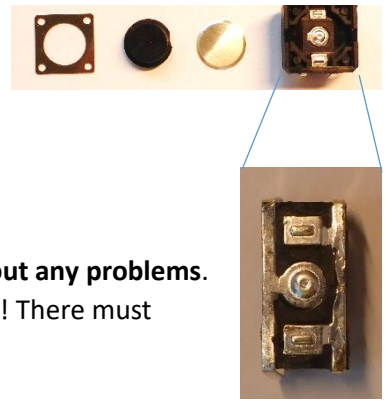
vvvvvvvv01.04 Digital input introduction



Above: Connections of a push button

Below: Internal operation of a push button



The aim of this assignment is to use a controller pin as a digital input and to investigate possible input characteristics.

We speak of a digital input if the pin can only detect high/low or 1/0.

We need the following for this assignment:

- 1 push button
- 1 resistor of 10KOhm (10K = 10000)
  [brown, black, orange, gold] Or
  [brown, black, black, red, brown]
- the breadboard
- 3 wires.

Attention! A pin configured as **input** can be connected to the power supply **without any problems**. A pin configured as **output** may **never** be connected **directly to the power supply**! There must always be a **device** (resistor, LED+resistor, ...) inbetween.

For the use of the breadboard you can look at "01.03 running light"
For the use of the colour code of the resistors you can also look at "01.03 running light".

01.04 Digital input exercise

Compile and load the corresponding code into the controller. This code will digitally read the value of pin 2 and place it on pin 13. Pin 13 is connected to the internal LED (see 01.01 blinky) This way you can see what the microcontroller sees.

Pin 2 is now connected to nothing.

What do you expect the LED to do? Lights up when I press the button

What does the LED do? Lights up when I press the button

Connect only one **wire to** Pin 2. The other side of the wire hangs **loose**.

What does the LED do? Behaves erratically

What does the LED do when your hand is near the wire? Lights up

With your right hand, hold the end of the wire (at the metal part) and tap the usb connector (ground connection) of the controller board with your finger.

What do you see now? The LED is switch off.....

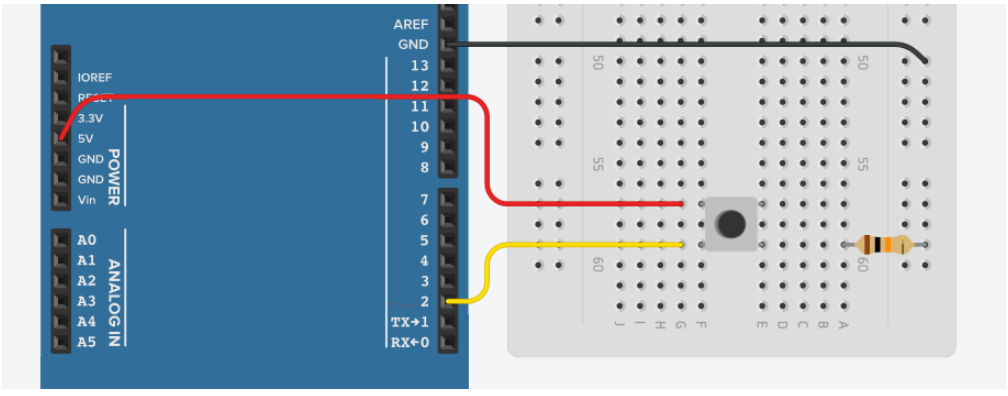
All the above tests indicate that you should **never** let a **pin configured as an input** hang **loose**. The level (high/low) is undetermined. The slightest (RF) disturbance can cause the level to flip.

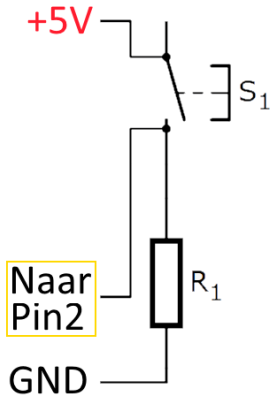
Does the LED light up if the end of the wire is in the +5V? YES

Does the LED light up if the end of the wire is in +3.3V? YES

Does the LED light up if the end of the wire is in the GND? NO

Build the following circuit:





You can straighten the pins of the button with a small pair of pliers (ask your teacher).
The button should be deep enough and you should leave it there. The breadboard may wear out quickly if these components are taken in and out a lot.

Press the button. What does the LED do now? It remains Light Up

Release the button. What is the LED doing now? Behaves erratically again

Can you still turn on the LED with your fingers? As Long as the button is not pressed, yes

Why or why not? Because one of the input pins is hanging loose and therefore effectively presents infinity resistance, therefore any change in the ambient (RF noisy) could sign a change of state to either zero (LOW) or one (HIGH)

Now adjust the code so that the LED changes state each time it is pressed.

Example: Led is off.

You press once (and also release) => the LED lights up.

You press one more time (and release) => the LED is off.

You press 1 more time (and also release) => the LED lights up.

You press one more time (and release) => the LED is off.

You press 1 more time (and also release) => the LED lights up.

…

If the LED does not remain in its state reliably (as described above), this can also be due to the input. A pushbutton can also suffer from contact bounce.

The figure below is what a microcontroller sees when it presses the push button:

The contact bounce period depends on the quality of the button and is typically in the large order of 20mS.

To write the code is important to clarify something which from the figure above can be seen:

First: There are two input states of the switch that are completely different!

- If the LED is OFF and you press the switch (closed circuit), then the signal sent to the PIN, after bounce-time, is HIGH.
- If the LED is ON and you press the switch (open circuit), then the signal sent to the PIN, after bounce-time, is LOW.

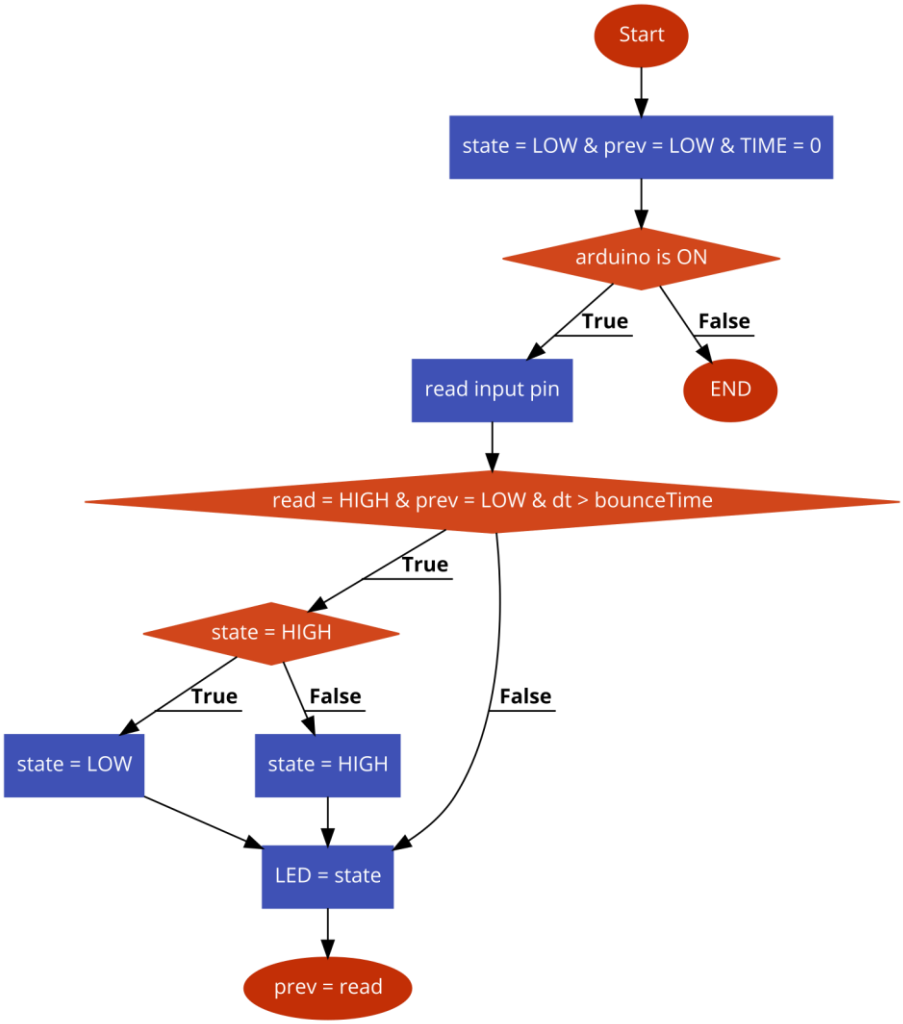
Second: The bounce-time can be disregarded if one assures that the measure time after reading is larger than 20 ms. This can be accomplished if one uses the built-in function in Arduino millis();

Code:

```
#define LED 13
#define BUTTON 2
#define BOUNCETIME 20

unsigned char reading, state = LOW, previous = LOW;
long int time = 0;

void setup() {
 pinMode(BUTTON, INPUT);
 pinMode(LED, OUTPUT);
}

void loop() {
 reading = digitalRead(BUTTON);
 // Condition that tests 3 things:
 // (I)  If someone has pressed the button
 // (II) If the if the previous reading was LOW (LED is OFF)
 // (III)If enough time has passed to discard the bouncing time of the switch
 if (reading == HIGH && previous == LOW && millis() - time > BOUNCETIME)
 {
   if (state == HIGH)
```

```
    state = LOW;
  else
    state = HIGH;
  time = millis();
 }
 digitalWrite(LED, state);
 previous = reading;
}
```

Flow-Chart Scheme:



Running manually some steps:

1. state is LOW and previous is LOW (INITIAL CONDITIONS)
2. READ from a LOW state gives you READ = HIGH. (STARTS FIRST LOOP)
3. CHECK if READ HIGH and previous is LOW and BOUNCE-TIME is bigger than delay (TRUE)
4. STATE = HIGH, then state changes to LOW
5. LED is switch OFF (remain the state LOW)
6. Previous = HIGH. (FINISHES FIRST LOOP)
7. READ from a LOW state gives you READ = HIGH. (STARTS SECOND LOOP)
8. CHECK if READ HIGH and previous is HIGH and BOUNCE-TIME is bigger than delay (FALSE)
9. LED is switch OFF (remain the state LOW)
10. Previous = HIGH (FINISHES SECOND LOOP)
11. READ from a LOW state gives you READ = HIGH. (STARTS THIRD LOOP)
12. CHECK if READ HIGH and previous is LOW and BOUNCE-TIME is bigger than delay (TRUE)
13. STATE = LOW, then state changes to HIGH
14. LED is switch ON (the state is now HIGH)
15. Previous = HIGH (FINISHES THIRD LOOP)
16. READ from a HIGH state gives you READ = LOW (STARTS FOURTH LOOP)
17. CHECK if READ HIGH and previous is LOW and BOUNCE-TIME is bigger than delay (FALSE)
18. LED is switch ON (the state remains HIGH)
19. Previous = LOW (FINISHES FOURTH LOOP)
20. …

So, the loop is built in such a way that it becomes a trap when no input is given!