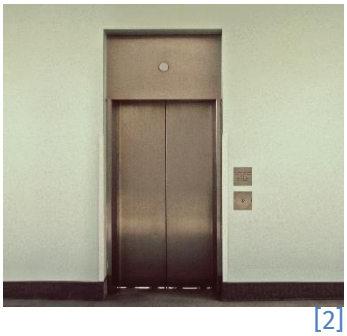
Station 3 – Speed Measurement

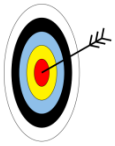

[1]

Invisible Light and Speed

You will encounter light barriers everywhere in everyday life, for example in elevators or at automatic doorbells in stores. Even though, you run into these barriers frequently, you often do not notice them, because they operate with invisible light: **infrared light (short: IR light).**

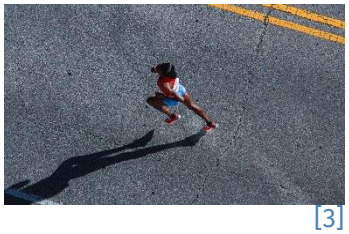

[2]

Light barriers work similarly to push-buttons, and they have a multitude of applications.



In this worksheet, you…
- will build IR light barriers to measure speed.
- learn how to measure speed.

These constructions are used, for example, for time measurement in sports.


[3]

The Construction of the Circuit

Every IR light barrier includes two elements: an **emitter** and a **receiver**. The emitter, in this case an IR LED, sends out light, and the receiver, in this case an IR photodiode, "sees" this light. If the IR photodiode registers IR light, it conducts electricity. If the line of sight is interrupted, no electricity is conducted.
You will set up two light barriers. Thus, you will need the following components (besides the breadboards and the Arduino):

- 2 220 Ω resistors (figure 4)
- 2 100 kΩ resistors (figure 5)
- 2 IR photodiodes (figure 6, black)
- 2 IR LEDs (figure 6, blue)
- 2 yellow, 3 blue, 3 red long wires


[4]


[5]


[6]

Station 3 – Speed Measurement

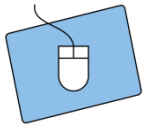
IR LEDs and Photodiodes

IR LEDs basically work like regular LEDs, which you already know from the introduction. However, there is a little disadvantage: With the naked eye you cannot see whether they are on or off. Here, a little trick will help you: To check, if your circuit is correct and closed, you can replace your IR LED and photodiode with normal LEDs. If the normal LEDs light up, then your circuit is working so far.

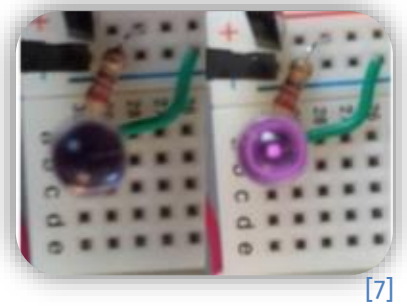
An IR photodiode only conducts electricity in one direction, if it is illuminated with the right kind of light. Thus, an IR photodiode only conducts electricity in one direction, if it is illuminated with an IR LED, for example.

The IR photodiode only conducts in one direction, if it is illuminated by an IR LED, for example. In the other direction, it always conducts electricity.

Now, you can start to build the circuit. At first, only one light barrier is built. On the next page you can see how the circuit should finally look like.



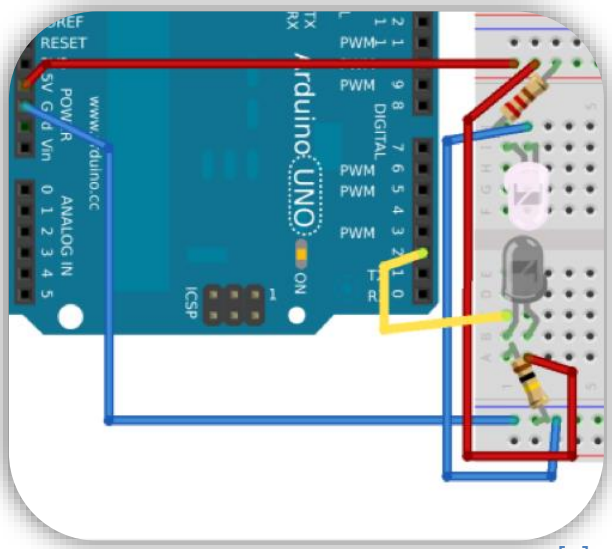
1. First, you only work with the lower part of your breadboard. Connect the plus bar to the **5V** pin, and the minus bar to a **GND** pin.
2. Plug the IR LED, and the IR photodiode into the far left of your breadboard.
3. For the light barrier you do not need to switch the IR LED on and off. Just connect it to the plus bar using a **220 Ω resistor**, and take a wire to connect it to the minus bar. Figure 7 gives you an orientation:



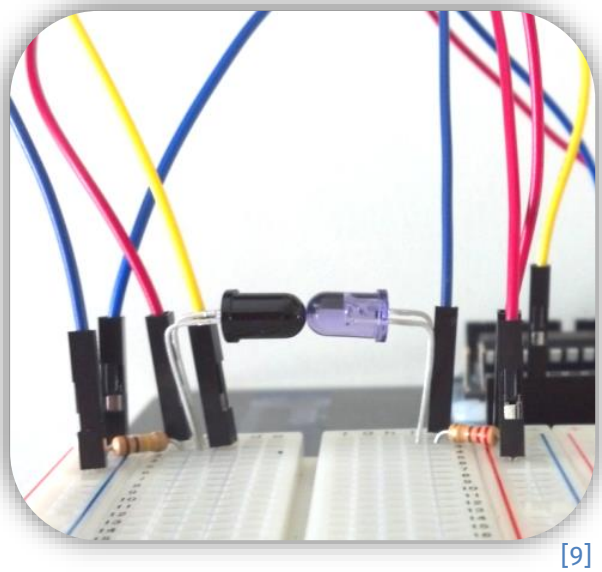
[7]

4. Test the IR LED (as described in the text at the beginning of this page).
5. Both, the IR LED and the IR photodiode, can only send and receive light in a small radius. Bend both components, so that they "look at each other". The best way is to put the top of their heads directly in front of each other.
6. The same way, you can connect the IR photodiode. Connect the short leg to the plus bar and the long leg with a wire to the **digital pin 2,** which you should use as an input. In addition, you connect the long leg to the minus bar using a **100 Ω resistor.**
   Hint: If you set up the IR photodiode in the wrong way, it will always conduct electricity!

Station 3 – Speed Measurement



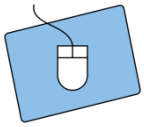
[8]


[9]

The First Test

Fine, your circuit is ready. Let's write the program. Until now, you are not able to output anything. Fortunately, the developer of Arduino came up with something. **Pin 13** is internally connected to a small LED on the Arduino (see figure 10). If pin 13 is switched on (set high), the LED lights up without having to plug in any cables. Now, your task is to switch on the LED exactly when the light barrier is interrupted.

[10]



1. Open a new sketch, and save it under a reasonable name.
2. Declare all necessary variables. Here, you should also choose reasonable names:
   a. one `int` variable for the **Arduino LED** with the value 13
   b. one `int` variable for the **IR photodiode** with the value 2
3. Now to `setup()`: First, you have to define the type of the pins using the following expression:
   `pinMode(pin-name, pin-type);`
   From the introduction you already know the pin type of the output pin. For an input pin you have to use `INPUT`.
   a. Use the `pinMode` command to define the Arduino LED as an output.
   b. Use the `pinMode` command a second time to define the IR LED as an input.
4. First, you will make the Arduino LED to blink in `loop()`. You already know how to do this from the introduction. You need `delay()` and `digitalWrite()`.
5. Test your program. Does the LED blink? Fine! If the LED does not blink, take a look at the worksheet of the introduction. This will help you to program the `loop ()`.

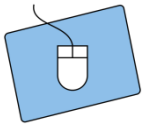
Station 3 – Speed Measurement

Reading Input Pins

By using `digitalRead(pin-name);` you can read an input pin. To be able to continue working with the measured value later you can store it in a variable. This is how it works:

```
variable-name = digitalRead(pin-name);
```

Now, you will expand the `loop()` to be able to test the light barrier. Of course, you do not want to switch on the LED all the time, but only when the light barrier is interrupted. For this, you need an **`if else`** construction, which is nothing new for you. Have a look at the introductory sheet, if you do not remember exactly how it works.



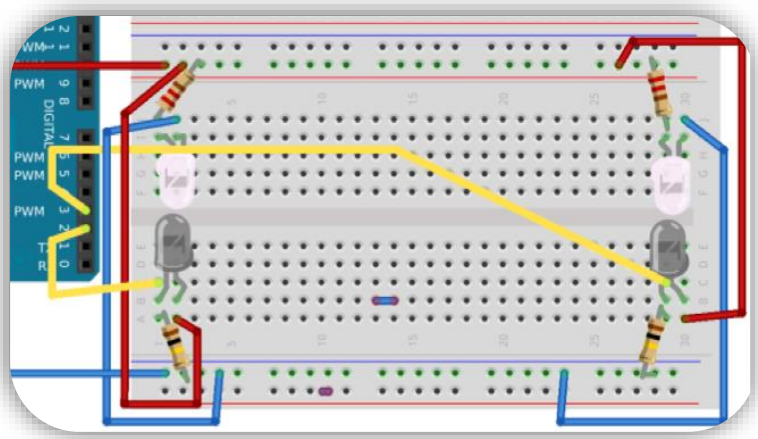
1. You can use the **`digitalRead`** command in the condition part, if you want to check whether there is electricity (high) or not (low).
   ```
   if (digitalRead(pin-name) == high/low)
   { … }
   ```
2. If the digital input is low, the LED is switched on. You can use the `else` to define what happens, if the input is low. To do this, you need to move your commands, that make the Arduino LED blink, to the right place.
3. Delete all **`delays`**, which are left. You do not need them anymore.
4. Now, it is time to test your sketch. Use the official light barrier interrupting automobile (or any other thin object) to interrupt the light barrier. Check, if the LED on the Arduino is doing the right thing.


[11]

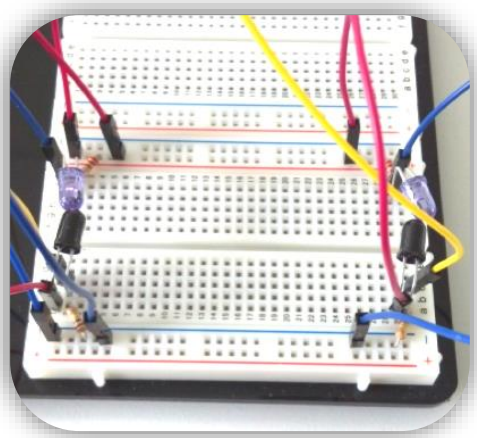
5. If everything is working fine, you can skip this exercise. But it is totally normal, if it is not working from the very beginning. That happens to the best programmers. The only important thing is to keep calm and look for the error. Make sure, that your `if` command is correct, that both pins are defined as in- or output in the `setup()`, and that you have used `digitalWrite()` and `digitalRead()` correctly. Also, test your LED again. Perhaps, your photodiode is connected the wrong way around, so it always conducts electricity.
6. Save your sketch, because you will need it for the next excises.

The Second Light Barrier

The circuit is the same as for the first light barrier. But this time, you use the lines 29 and 30 of the same breadboard. As an input for the IR photodiode you can use pin 3. Your circuit should now roughly look like this:



[12]


[13]

How can you quickly test the functions?

1. Take the same sketch as before.
2. Just change the input pin to 3.

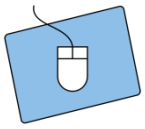
Does the LED now light up when the second light barrier has been interrupted?

Everything is working? Wonderful! Now, you got everything to measure speed. However, a few preparations still need to be made. If the first light barrier is interrupted, the measurement should start; if the second light barrier is interrupted, the measurement should end. The following excises explain, how to add the necessary `if` statements to your sketch.

1. Change the value of your IR pin back to **2**. Now, save the sketch under a new name, so you can use it without deleting your previous results.
2. First, you have to declare a **variable** for the **second IR photodiode**. Also, you have to define the pin type in the `setup()`.
3. Then, you have to save in your sketch whether the measurement has already started or not. Therefore, you can use **another** `int` **variable**. 0 means, that the measurement has not yet been started. The other way around, 1 means, that the measurement is already running. So, in the **settings**, you have to assign the value 0 to this variable.

Continue on the next page...

Station 3 – Speed Measurement

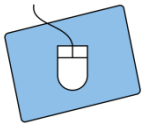


4. Now, you need an **`if`** statement to recognize, that the measurement should start. To do this, adjust your `if` statement, so it sets your measurement variable to 1, and outputs a text via the serial monitor. This text should tell, that the measurement has started. This should exactly happen, when the first light barrier has been interrupted, and the measurement is not already running. Take a look at the figure below.
   Hint: You do not need the `else` for this exercise.

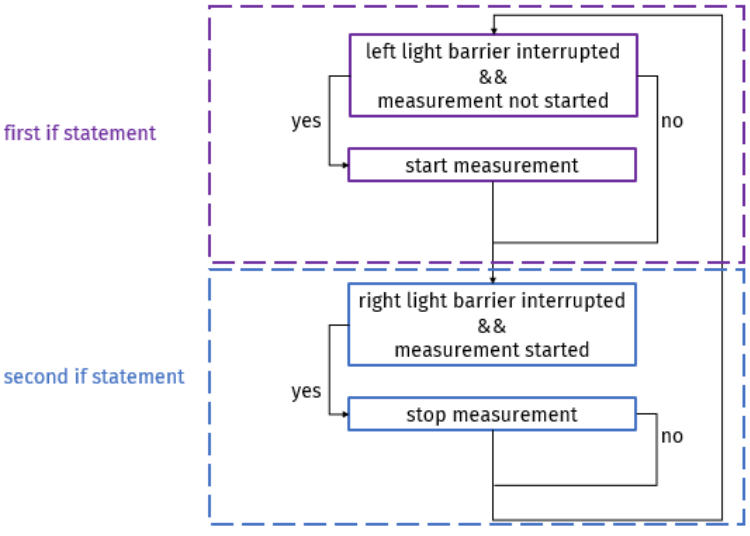
If Statements with More than One Condition

Using an `if` statement, you can query more than one condition. To do so, you have to link these conditions with &&. However, you have to put every single condition in parentheses. This way, the following expressions will only be executed, if both conditions are true.

```
if ( (conditionA) && (coditionB) ){expression}
```



5. Test your program. Start the serial monitor, and interrupt the light barriers. Your text should only be displayed once, because the measurement is started then. If the text is displayed again and again, you have to think about the condition again.
6. Now, create a second `if` statement (without else). This one should work the other way around. It should set the measurement-started variable to 0, and output via the serial monitor, that the measurement has been finished (later the speed should be output here as well). Again, have a look at the figure below.
7. Test your sketch again. Now, the texts “Measurement starts” and “Measurement is finished” should be displayed alternately on the serial monitor, when you interrupt the light barriers alternately.
   Hint: Be careful not to accidentally move your IR LEDs and IR photodiodes. The wires can disturb the measurement. Simply tape them to the side, if they disturb.



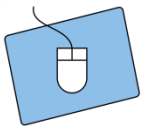
[14]

Station 3 – Speed Measurement

Now, you have prepared everything you need to measure speed.

Measurement of Speed

The function **`millis()`** gives you the number of milliseconds, that have passed since you have uploaded your program to the Arduino. This time should be saved. However, an **`int` variable** can only store numbers up to a certain size. Since the number of milliseconds can become very large, you need a new type of variable: **`unsigned long`**. This type is also for whole numbers, but larger numbers can be stored.

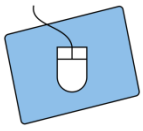


1. Declare two new **`unsigned long`** variables in the settings. These should store the star,t and the end time of the measurement. Give them reasonable names.
   ```
   unsigned long variable-name;
   ```
2. Think about the following question: When should time be measured? Store the times in your two variables using the **`millis()`** function at the right places in your program.
3. Extend the **`if`** statements in your program to output the start and end time via the serial monitor.
4. Test your program. Do not be surprised: The numbers become very quickly very large. That is what happens, when you display the time in milliseconds.
5. Is it working? Fine! Now, you just have to calculate, how long the measurement took. Therefore, you need a third **`unsigned long` variable** to store the duration.

Calculations

In your program you can easily do calculations by connecting two variables or numbers with +, -, *, or / and then store the result with = in a third variable.
For example, a simple subtraction looks like this:

```
resultvariable = number1/variable1 - number2/variable2;
```



6. Use your variables, and this simple subtraction to calculate the measurement duration, and save it in your measurement duration variable.
7. Display the result via the serial monitor once the measurement has ended.

Great! Now the serial monitor can display how long the car needed to cover the distance between the two light barriers. Wait a second: Time is not speed, right? The rest is physics. If you want to master this challenge as well, ask for the bonus sheet.

List of references:

Fig. 1 – *Source: pxhere.com (https://pxhere.com/de/photo/981592), CC0 1.0 Universal (CC0 1.) Public Domain Dedication (https://creativecommons.org/publicdomain/zero/1.0/), 2022-07-06*
Fig. 2 – *Source: pxhere.com (https://pxhere.com/de/photo/927145), CC0 1.0 Universal (CC0 1.) Public Domain Dedication (https://creativecommons.org/publicdomain/zero/1.0/), 2022-07-06*

Fig. 3 – *Source: pxhere.com (https://pxhere.com/de/photo/53576), CC0 1.0 Universal (CC0 1.) Public Domain Dedication (https://creativecommons.org/publicdomain/zero/1.0/), 2022-07-06*
Fig. 4, 5, 8, 12 – *Source: Screenshot of the Fritzing Software (http://fritzing.org), CC BY-SA 3.0 Attribution-ShareAlike 3.0 Unported (https://creativecommons.org/licenses/by-sa/3.0/), 2022-06-14*
Fig. 6, 7, 9-11, 13, 14 – *Source: InfoSphere*

– *Source: InfoSphere*