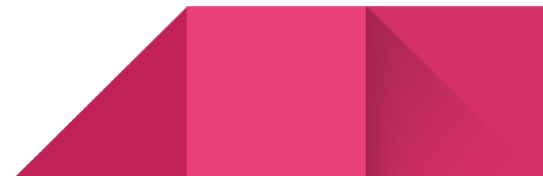




Extended micro: bit kit

Jørgen Larsen
Jacob Nielsen



Extended micro: bit kit

Jørgen Larsen and Jacob Nielsen

Issue 1, revision 2

Published by Teknologiskolen, Odense

Publication year: 2020

This book is published under Creative Commons (CC BY-NC-ND 4.0).



<https://creativecommons.org/licenses/by-nc-nd/4.0/deed.da>

Credit - You must provide appropriate credit, provide a link to the license, and indicate if any changes have been made. You may do so in any reasonable manner, but not in any way that indicates that the licensor approves you or your use.

Noncommercial - Do not use the material for commercial purposes.

No diversions - If you remix, rework, or build upon the material, you may not distribute the modified material.

No Additional Restrictions - You may not add legal terms or technological measures that legally restrict others from doing what the license allows.

Extended micro: bit kit	2
1 Overview	4
Objectives	4
Specifications	4
2 Basic setup	5
Micro: bit	5
Adafruit DragonTail for micro: bit	6
Breadboard	6
3. Flashing with LEDs	8
4. Measuring light	11
5. Adjustment with potentiometer	14
6 Power Supply	17
7 Hall effect sensor	19
8 NeoPixels	21
9 H-Bridge and DC motor	24
10 Exercises	28
10.1 Turn on light in the dark:	28
10.2 Control the power of the light:	28
10.3 Different lights with NeoPixels	28
10.4 Position of the motor	28
Program overview	29

1 Overview

To give an overview of the box contents, this guide will provide examples of circuits for each component in the box, as well as a functional description of each.

Objectives

1. **To become familiar with the contents of the box:** In order to use the contents in a practical context, it is first necessary to become familiar with the functionality of each component and discover what it can be used for.
2. **To gain a general understanding:** Once the functionality of the individual components of the box has been reviewed, it should be possible for the user to do his/her own inventions that use the components in new ways, and combinations as well as other contexts.

Specifications

To get started with this guide it would be preferable if the user already has basic knowledge of micro: bit as well as programming it using Microsoft MakeCode.

In addition, it will be advantageous to have access to a small screwdriver (straight notch), as well as 2 AAA batteries - preferably rechargeable to protect the environment. Throughout this guide, various supplies will be used and information will be provided on an ongoing basis.

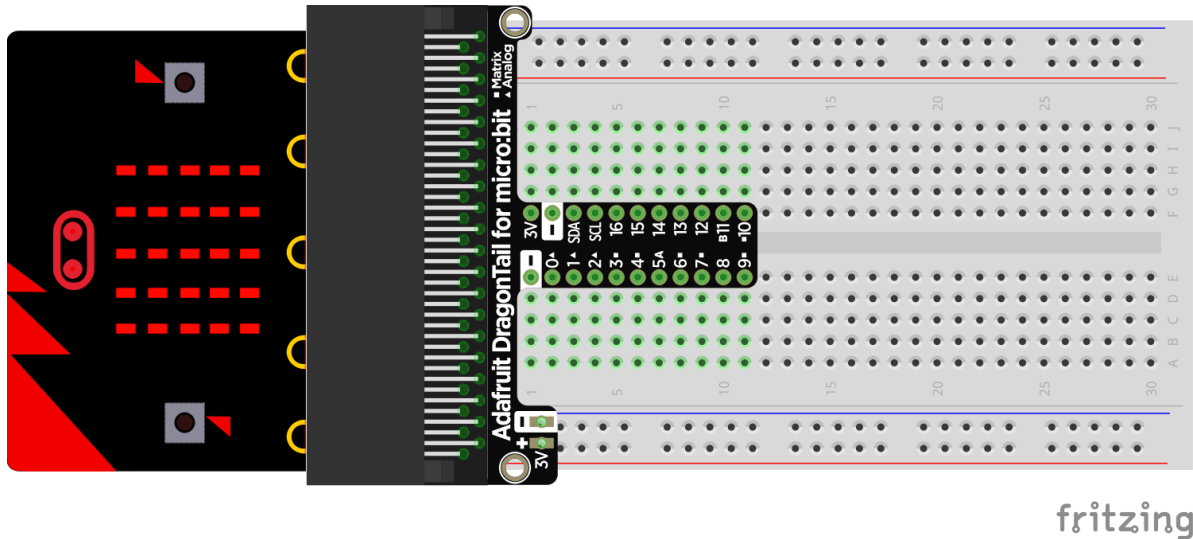
For each component, both a diagram and a picture of how the circuit is built on a breadboard will be used. These are made in Fritzing. The code to be created on the microbit is also displayed, in both MakeCode from Microsoft and JavaScript.

This book is not intended as a basic book in neither electronics nor programming, but merely as a review of the contents of the box.

At the back of the book you will find an overview of links to all MakeCode programs used in this book.

2 Basic setup

Throughout the book, the following will be the basic setup used. The various examples will describe the workflow using this as a starting point.



*Figure 1. Basic setup consisting of a micro: bit, an Adafruit DragonTail
for micro: bit and a breadboard.*

The basic setup consists of a micro: bit, an Adafruit DragonTail that connects the micro: bit with a breadboard. We will then go through the individual parts of this basic setup.

Micro: bit

This micro:bit is exactly the same as known from BBC or DR Ultra. As you know, it can be programmed either via MakeCode from Microsoft or JavaScript.

Adafruit DragonTail for micro: bit

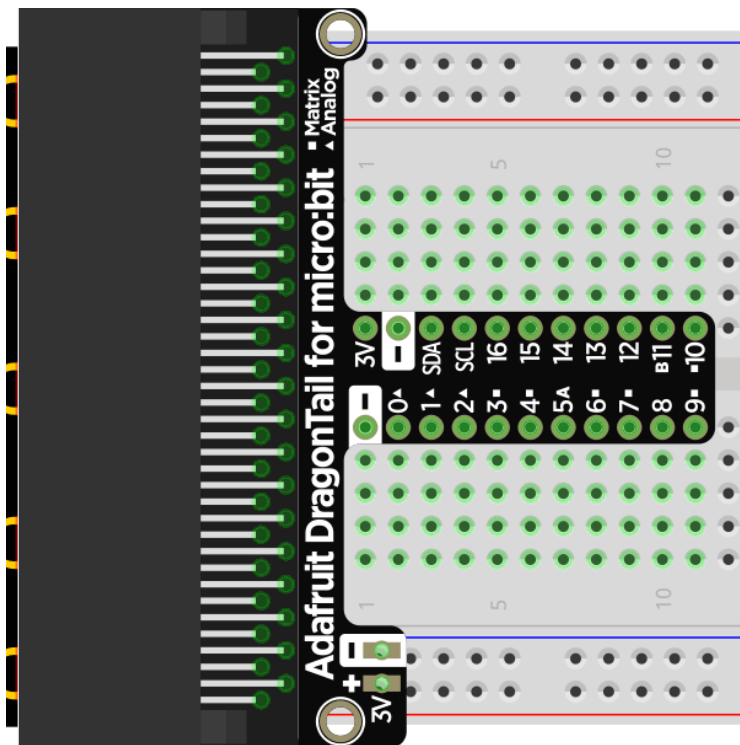
This is a connector that connects all the connections at the bottom of a micro:bit to rows on the breadboard. In this way, it becomes possible to use all the extra I / O (Input / Output) pins that are otherwise unavailable.

If we look more closely at the picture on the right, we see that several of the legs are marked with either a "square", a "triangle" or "A / B".

A mark with a "triangle" means that the leg can be used for analog signals.

A "square" mark indicates that the leg can only be used for digital signals, just as the leg is also part of the matrix display that is on a micro: bit.

A letter "A" or "B" indicates that the leg is connected to either button "A" or "B" on the micro: bit.



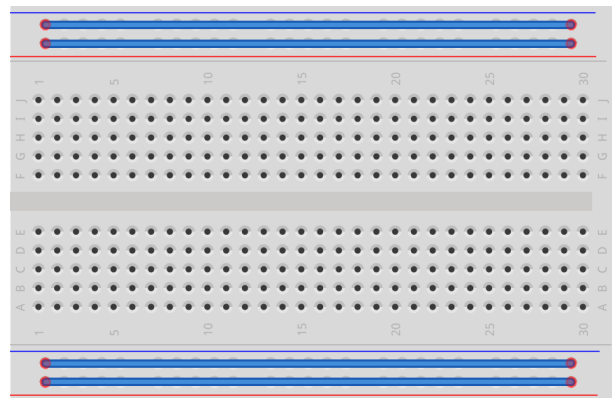
Breadboard

A breadboard makes it easy to build small circuits without having to solder. In addition, in most cases the components can be reused after having been mounted in the board.

Most breadboards are divided into two main areas. A building area and a supply area.

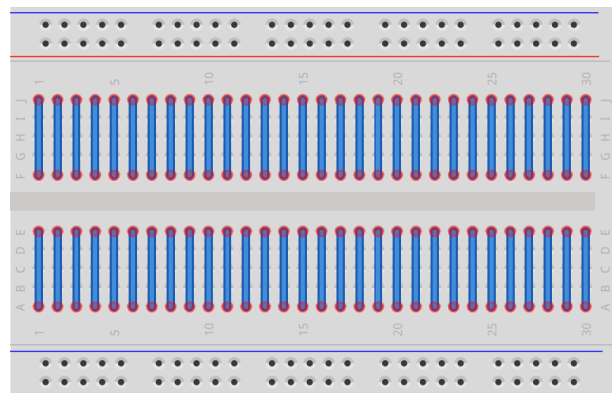
At the top and bottom of the breadboard are two parallel rows of holes that run throughout the length of the board. These four rows (2x2) are most often used for the supply voltage. When an Adafruit DragonTail is used, it connects 3V from the micro: bit on the two lower rows on the

board. Remember, when mounting an Adafruit DragonTail in a breadboard, it's good practice to flip the breadboard so that the row marked with "red" becomes "+" and the row marked with "blue" becomes "-". The picture below shows how the holes are connected into rows within the supply area.



fritzing

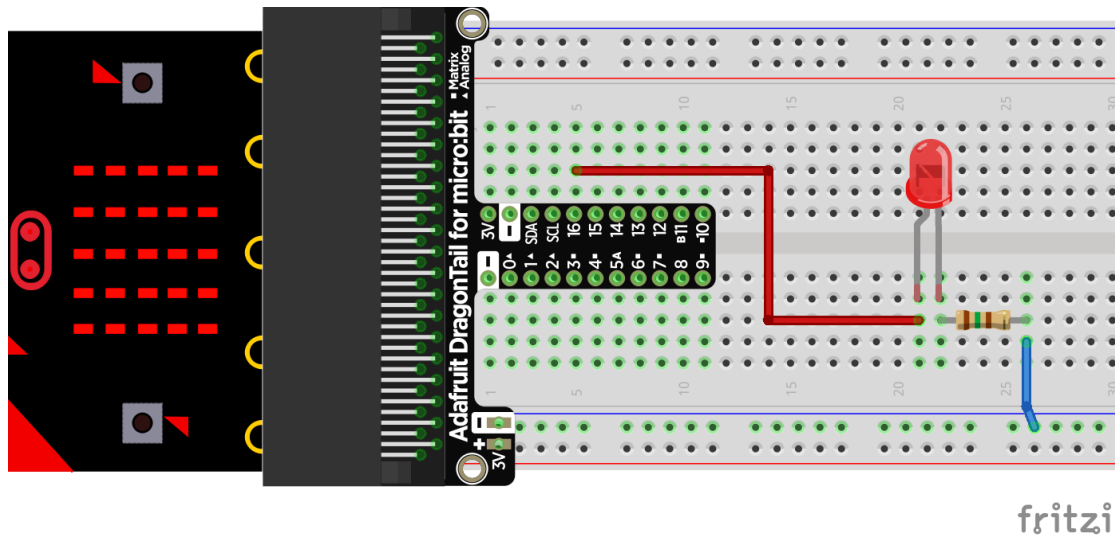
In the middle of the breadboard is the construction area. This is divided into two areas. The drawing below shows how the holes are connected into vertical rows within the construction area. The connections do not cross the center area.



fritzing

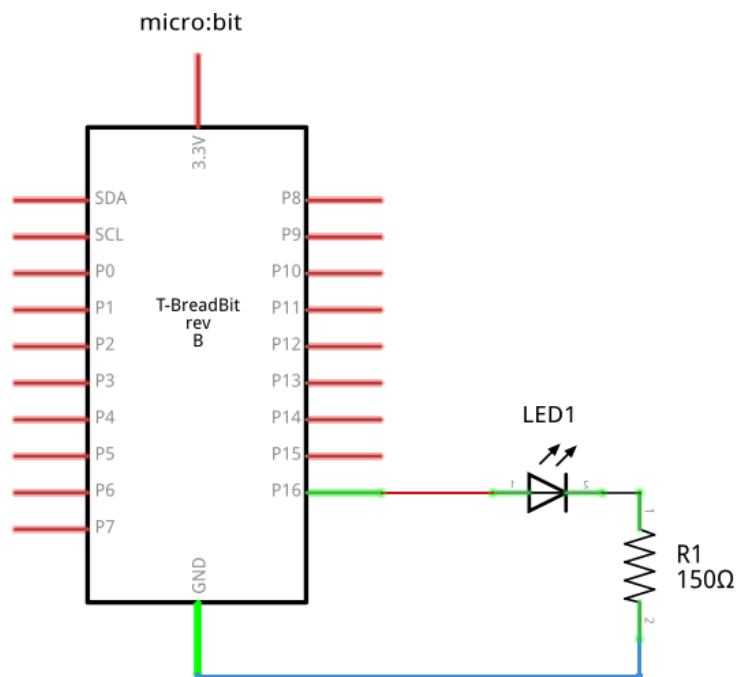
3. Flashing with LEDs

The box has 5 LEDs in the colors red, blue, green, yellow and white. The picture below shows how the LEDs must be connected to a micro:bit so that we can flash them.



fritzi

The diagram for the above circuit is shown below.



A 150Ω resistor is used in the circuit. This resistor is used as the red, yellow and green LEDs cannot withstand the 3.3V micro: bit supply. Therefore, a current limiting resistor is needed.

In order to determine the necessary resistance, we need to know specific details about the LEDs. The table below shows the basic data that you would normally work on.

Color	Voltage (voltage drop)	Current
Red	2.5V	5mA
Yellow	2.5V	5mA
Blue	4V	5mA
Green	2.5V	5mA
White	4V	5mA

The size of the LED resistor can then be determined. This is done because we know the voltage a micro: bit can supply on its I / O pins as well as the current and voltage (voltage drop) of the LEDs as shown in the table above.

First, the Ohm's law is isolated with regards to "R1":

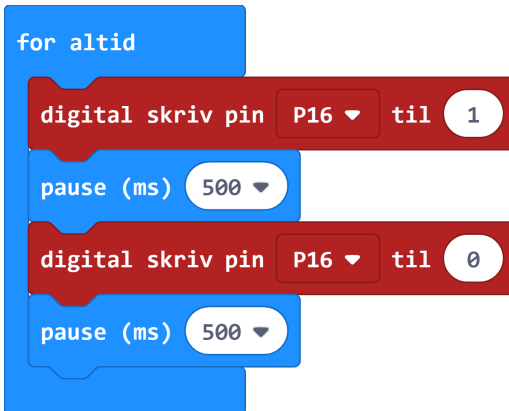
$$U = R1 * I \Leftrightarrow R1 = \frac{U}{I}$$

Then we insert values to determine "R1": $R1 = \frac{3.3V - 2.5V}{5mA} = 160\Omega \approx 150\Omega$

Here, we find a value of 160Ω, which cannot be found in our kit. Therefore, the value closest to the one in the kit is chosen, namely 150Ω.

With the colors white and blue, this resistance is not necessary as these can hold a greater voltage than what a micro: bit delivers. Therefore, the resistance can be removed and the LED connected directly to minus, "-".

The code needed to make the LED blink is quite simple and is shown below.



MakeCode - Blockly

```
basic.forever(function () {  
  pins.digitalWritePin(DigitalPin.P16, 1)  
  basic.pause(500)  
  pins.digitalWritePin(DigitalPin.P16, 0)  
  basic.pause(500)  
})
```

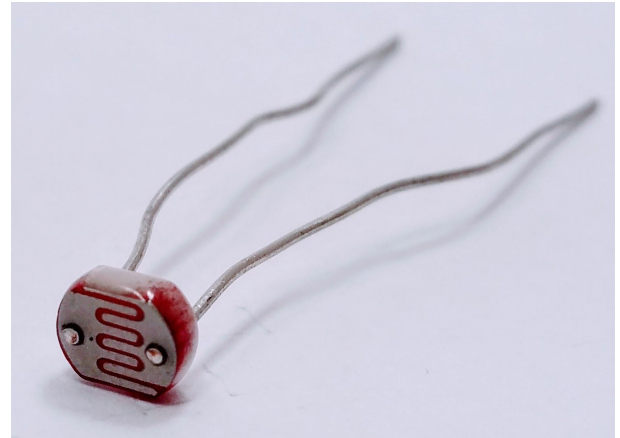
MakeCode - Javascript

The above code causes the LED to flash at 1Hz, i.e. it either turns on or off every half second.

The project is shared here: https://makecode.microbit.org/_CU4Lh7EJsDMV

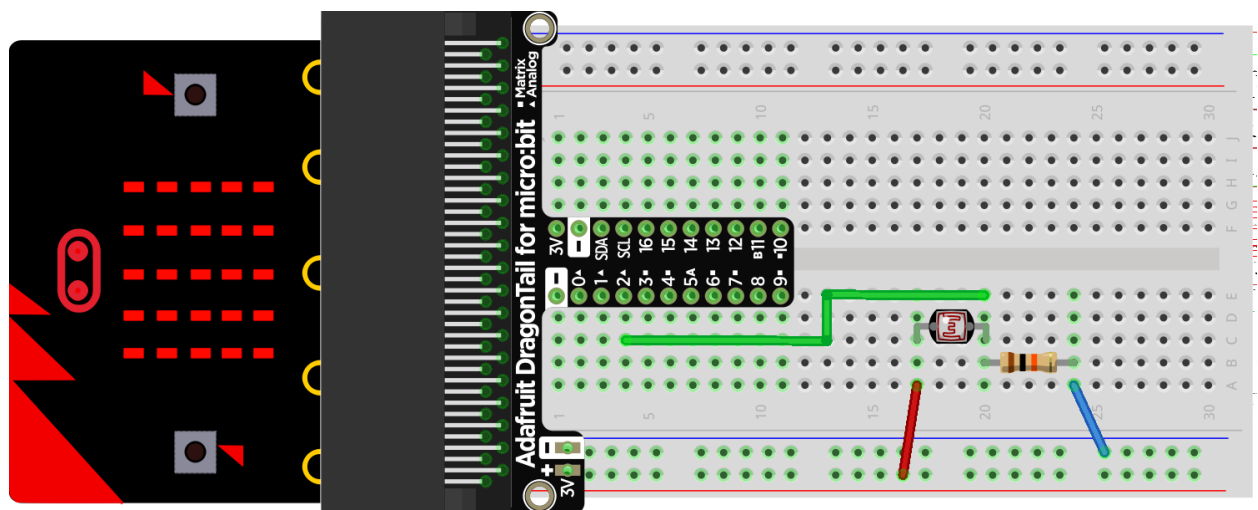
4. Measuring light

We will look at how to measure light brightness in the next section. For this purpose, a so-called LDR resistor is used. This is a type of resistor that changes its resistance as a function of the light intensity of the component. This means that the resistance decreases at high light intensity and increases at low light intensity. The picture to the right shows an LDR resistor.



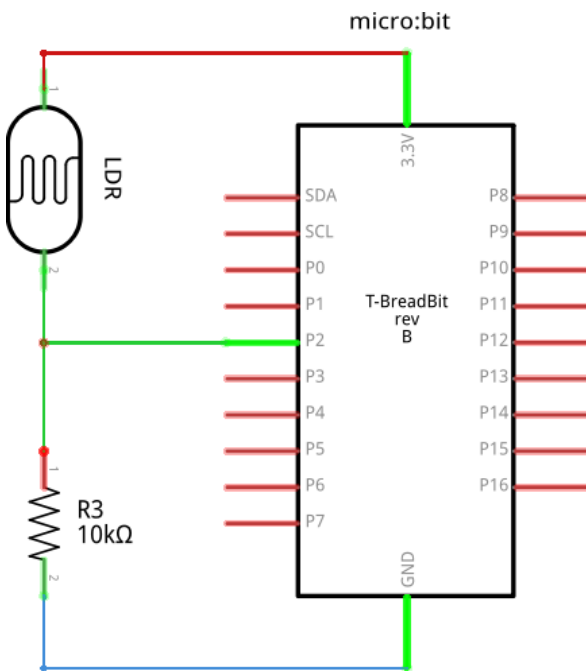
The LDR in these boxes is very similar to a PDV-P8001. Optionally search for datasheets on google.

In order to use an LDR sensor with a micro: bit, it must be connected in a small circuit with another fixed resistor called a voltage divider. This is done to convert the resistance change from the LDR resistance to a voltage change that our micro: bit can measure. The circuit is shown in the picture below.



fritzing

Below is the diagram.



The diagram on the left shows an LDR resistor connected in series with a fixed resistance between 3.3V and GND. The connection between the LDR resistor and the fixed resistor is further connected to P2 on our micro: bit. This coupling between two resistors is called a voltage divider and in this case one resistor is an LDR resistor whose resistance changes as a function of light intensity.

To retrieve data into a micro: bit we use an analog input, in this case pin 2 on the dragontail connector in the breadboard as shown in the picture on the previous page as well as on the diagram next to it.



MakeCode - Blockly

```
let light = 0
basic.forever(function () {
  lys = pins.analogReadPin(AnalogPin.P2)
  basic.showNumber(lys)
})
```

MakeCode - JavaScript

The above project can be found here: https://makecode.microbit.org/_ipTC0bqgYm6

The above program prints a number between 0 and 1023 representing the amount of light measured by the LDR sensor. It is not precisely LUX that is measured, but a relative light intensity amount.

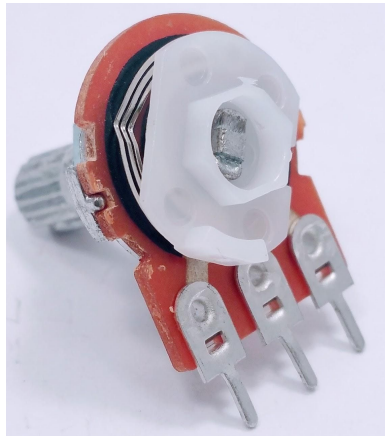
5. Adjustment with potentiometer

In the last task we worked with a variable resistor called an LDR resistor. In this assignment we will work with another type of variable resistor, namely a potentiometer.

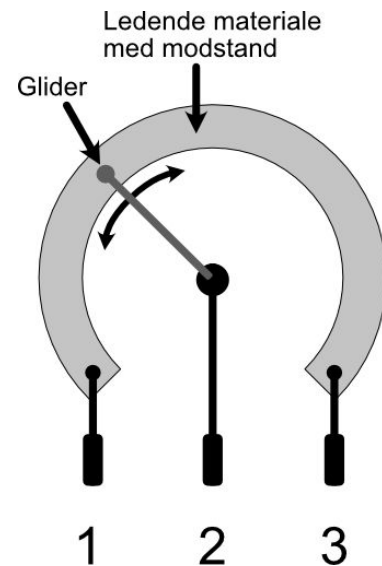
Unlike an LDR resistor that changes resistance as a function of light intensity, we can more specifically control the resistance of a potentiometer. Below, to the left rotary, we see a potentiometer, at the center the potentiometer is open and to the right we have a schematic drawing of how it works.



Overall



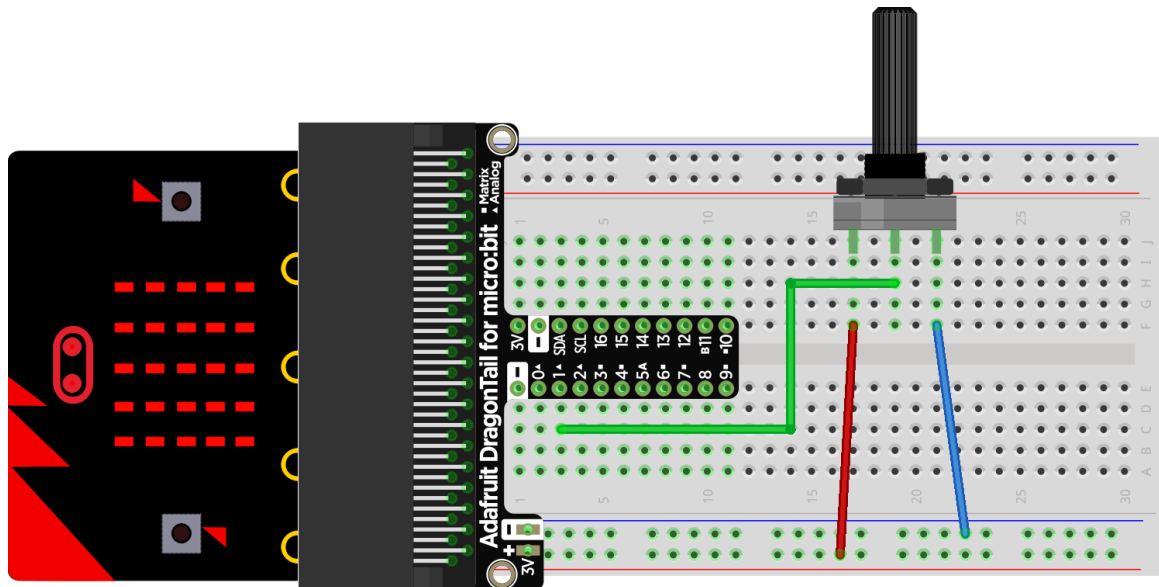
Open



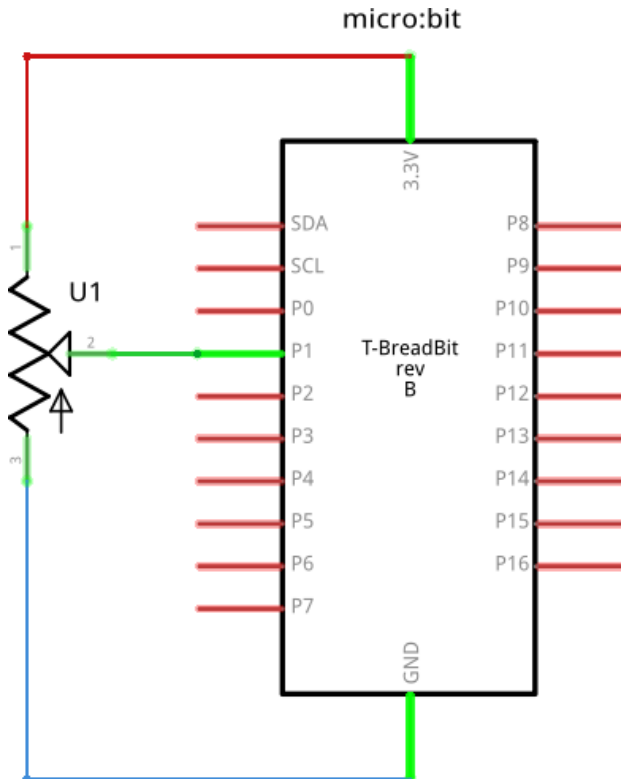
Functional drawing

As can be seen from the functional drawing above, pins "1" and "2" are connected by a conductive material with resistance. This means that in a potentiometer with a resistance of e.g. 10K Ω , there will be a fixed resistance between pins "1" and "3" of 10K Ω . The variable part of the component is measured between pins "1" and "2" or pins "2" and "3" as the slider, whose position can be adjusted by turning the knob in the middle, connects e.g. a circuit from pin "1", through the conductive material to the slider position and out through pin "2".

In this small setup we want to be able to measure the change in resistance of a potentiometer from a micro: bit. To do this, the potentiometer must be connected as shown in the picture and diagram below.



fritzing



The diagram below shows how the potentiometer is connected between 3.3V from a micro: bit and GND. By adjusting the slider - when turning on the potentiometer - a voltage of between 0V and 3.3V on leg 2 is then obtained, depending on the setting.

For this example, the code from the last example can be used almost directly. This is because this also involves measuring a varying analogue voltage. If the program from the last example with light is used, 0-1023 will be read out on the display on one's micro: bit. Simply change the input of

one's micro: bit (*Analog pin*) from "P2" to "P1". It will look like this.



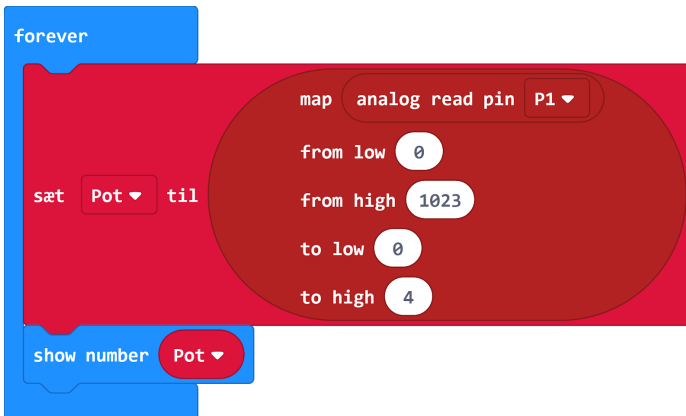
MakeCode - Blockly

```
let Pot = 0
basic.forever(function () {
  Pot = pins.analogReadPin(AnalogPin.P1)
  basic.showNumber(Pot)
})
```

MakeCode - JavaScript

In the example above, the input (*analog pin*) is changed to "P1" and the variable is renamed to "Pot".

Should you instead want to get the real voltage shown in the display, a little code must be added. The reason for this is that the value otherwise displayed on the display is the binary value of the conversion from analogue to digital. You could say it is a value that resizes relative to the change in (*in this case*) analog pin "P1". The code for this is shown below.



MakeCode - Blockly

```
let Pot = 0
basic.forever(function () {
  Pot = pins.map(
    pins.analogReadPin(AnalogPin.P1),
    0,
    1023,
    0,
    4
  )
  basic.showNumber(Pot)
})
```

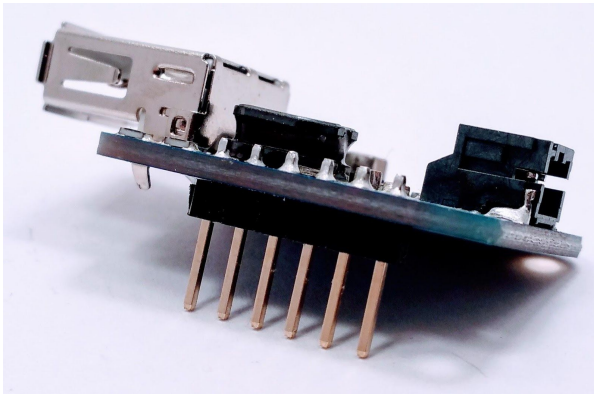
MakeCode - JavaScript

The above project can be downloaded here: https://makecode.microbit.org/_5zR3wyapA1f5

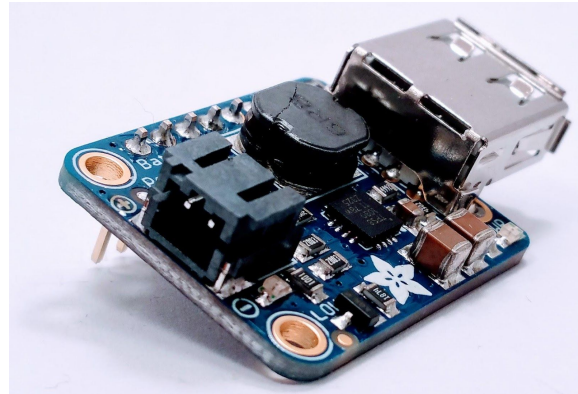
6 Power Supply

In this section we will look at what to do if you have equipment that requires more power than what a micro: bit can deliver. In fact, the vast majority of things that fall into the actuators category, such as DC Motors or Servo Motors will require more power. It can be said broadly that since actuators affect the world (something happens) and it requires more energy (power) than what a micro: bit can deliver.

To meet this challenge, a separate power supply module is included in the box.



Power supply - front view



Power supply - rear view

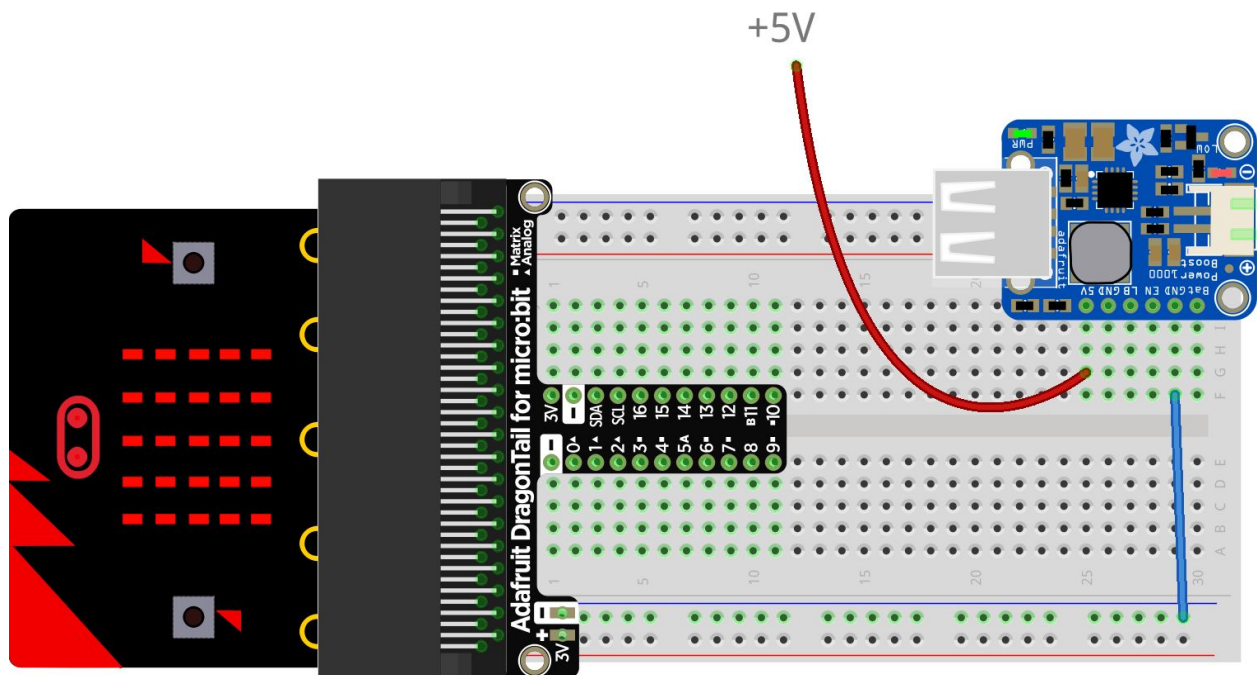
This power supply module, together with the battery pack that is also in the box, can supply a stable 5V supply of up to 1A. This is far more than the about 0.07A (absolute max) that a micro: bit can deliver.

When using an external power supply, there are a few things to be aware of in order for your circuit to work properly. The first is common GND. This is very important as otherwise you will not have a common reference.

Imagine a classic physics experiment where the speed of an object must be determined. This is also done on the basis of a common reference, namely the speed "0" where both the object and

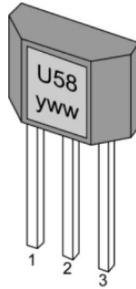
the meter are stationary. Now imagine that both object and meter are moving at different and unknown speeds. Then it becomes difficult to determine the speed of the object.

For the same reason, the first thing to do is to secure joint GND. After that it is possible to use the external power supply with a micro: bit without problems. A USB plug is also mounted on the power supply so that the power supply can supply a micro: bit from the battery if it is to be used without being connected to a computer. Below is shown how to connect the external power supply.



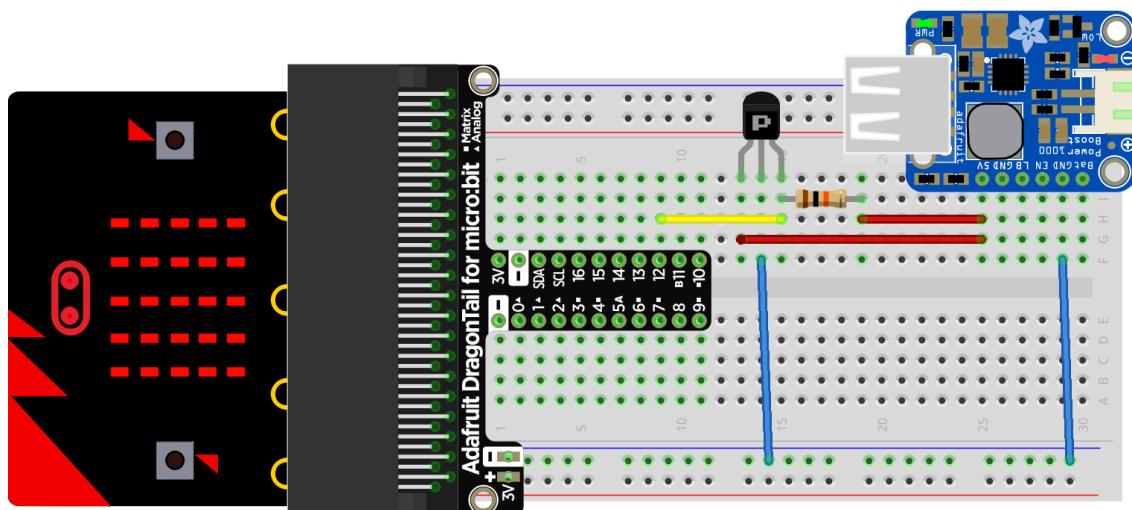
7 Hall effect sensor

In the following we will look at how a hall effect sensor works. This type of sensor acts as a switch that can be activated by a magnet. This specific sensor is activated only by bringing the "south pole" of a magnet near the sensor. A "north pole" will have no effect.



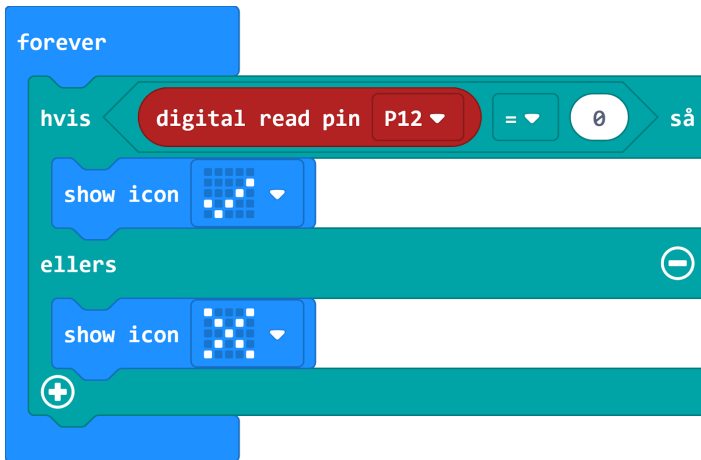
Hall effect sensor - US5881 - (Image from datasheet)

To use this sensor, the last section of the power supply from the last section must be used, as the sensor must have a 5V supply voltage. The connection should be as follows: Pin (1) 5V, Pin (2) GND, Pin (3) connected through a 10KΩ resistor to 5V supply as well as to Pin (12) on our micro: bit. See layout below.



fritzing

In this example, we just want to show on the display whether or not the sensor is activated by a magnet. The code for this can be seen below.



MakeCode - Blockly

```
basic.forever(function () {
  if (pins.digitalReadPin(DigitalPin.P12) == 0) {
    basic.showIcon(IconNames.Yes)
  } else {
    basic.showIcon(IconNames.No)
  }
})
```

MakeCode - JavaScript

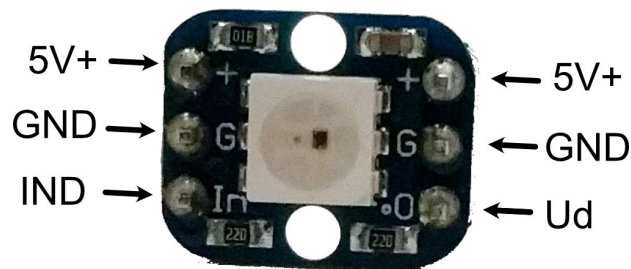
The program works so that a check mark appears on the display when a magnet's south pole is held near the sensor. Otherwise, an X will appear. The program can be downloaded here:

https://makecode.microbit.org/_H9fUzHU2gRvs

8 NeoPixels

To get started with code for this example, an extension to MakeCode called "NeoPixel" must be downloaded. Then a new item will appear in the block menus called "NeoPixel".

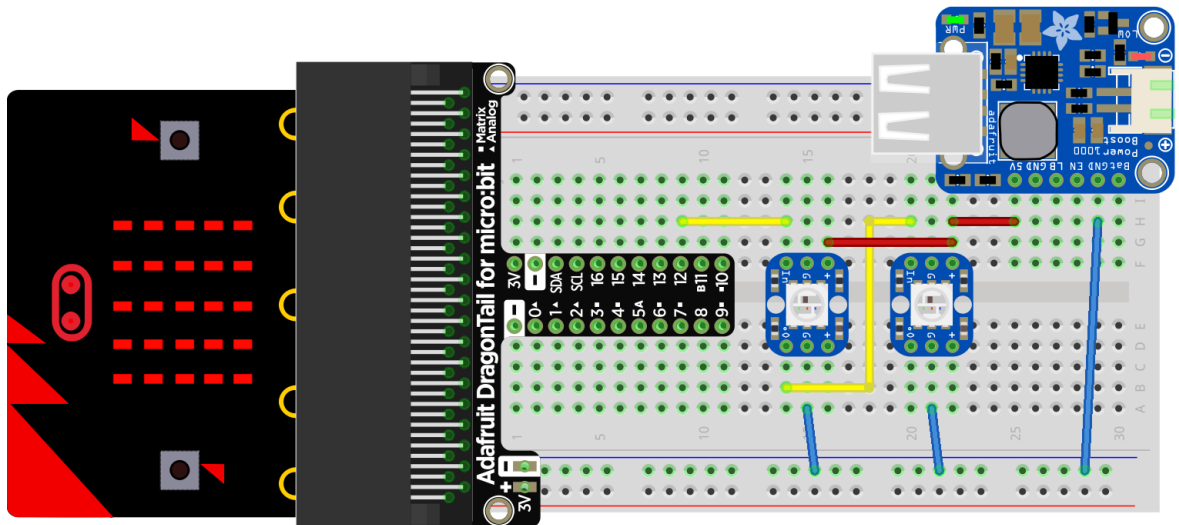
But what is a Neopixel? It is a component that contains 3 LEDs and a microcontroller. This means that a neopixel can display RGB colors and they can be chained in a series so that you can control multiple neopixels from a single I / O leg on one's micro: bit.



Neopixel

The picture above shows a Neopixel. A neopixel has pin connectors on the left and right sides, respectively. You have to think of it as the signals running from left to right - just like the reading direction. Supply voltage, respectively. 5V and GND are connected on the left side, and if there is one more neopixel in the row, the supply to the next can be taken from 5V and GND on the right side. The lower pin is the control signal. As with the supply voltage, it comes in from the left and passes on to the right side and can be connected to the next neopixel in a row.

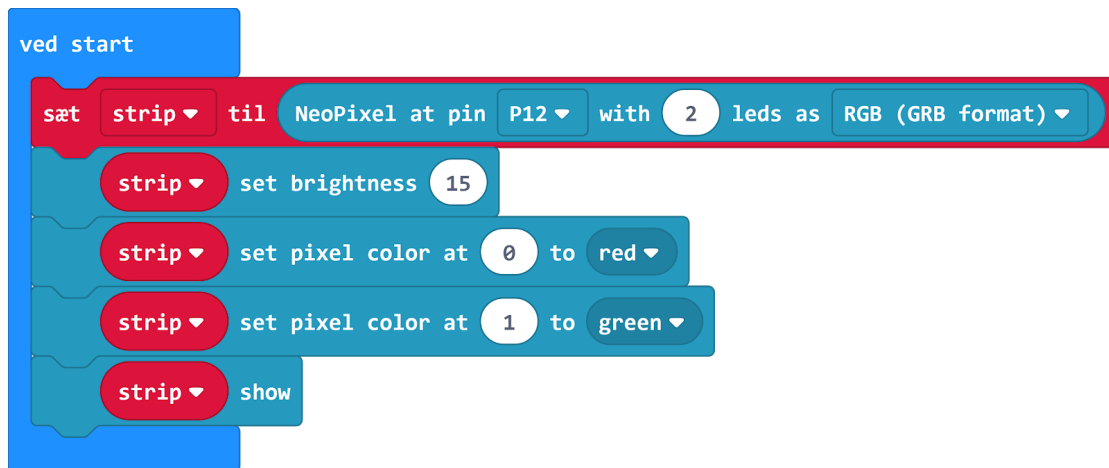
The circuit diagram below shows the setup of two neopixels in series. As before, the external power supply module is also used here.



fritzing

Construction Drawing. Here, "Red" is used for 5V, "Blue" for GND and "Yellow" for digital signal

When the circuit is assembled the code must be built. In this example, just make the code to turn on the two neopixels respectively. "Red" and "Green".



MakeCode - Blockly

We start by creating our series of 2 neopixels in a variable we call "strip". Then brightness is set. The color of respectively neopixels 1 and 2 are set to "Red" and "Green" and finally it is all brought into action and display with the "show" command.

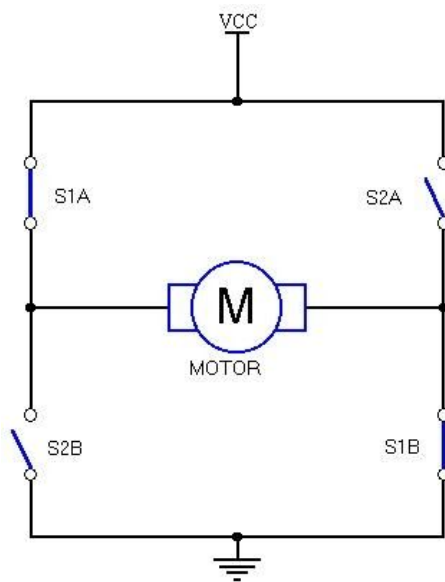
```
let strip = neopixel.create(DigitalPin.P12, 2, NeoPixelMode.RGB)
strip.setBrightness(15)
strip.setPixelColor(0, neopixel.colors(NeoPixelColors.Red))
strip.setPixelColor(1, neopixel.colors(NeoPixelColors.Green))
strip.show()
```

MakeCode - JavaScript

If you want a different color, some presets can be selected, and if you wish, you can also mix your own colors. Code can be downloaded here: https://makecode.microbit.org/_V2ubcyap1cYt

9 H-Bridge and DC motor

If you want to be able to control the direction of rotation of a motor you need an H-Bridge. The name covers a coupling of components that together allow you to control the direction of rotation of a DC motor. Overall, the coupling is shown as shown below:

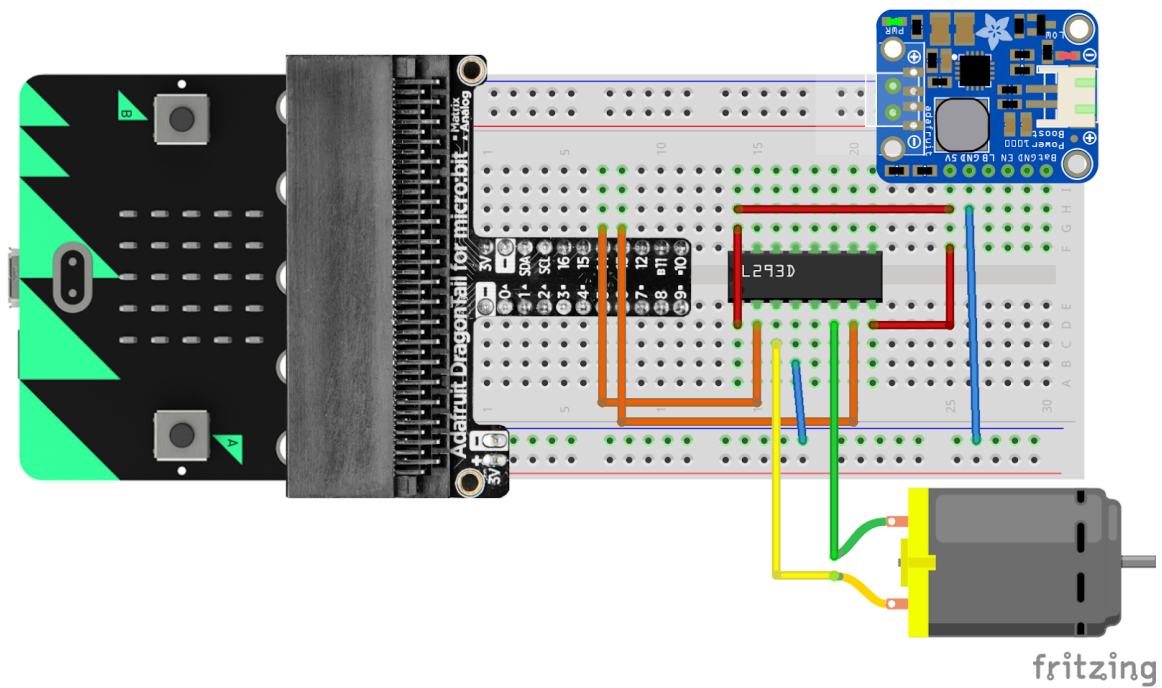


The picture on the left shows how an h-bridge works in a very basic way. We see two contact pairs "S1A and S1B" and "S2A and S2B" respectively. If we imagine that as shown in the picture, the contacts "S1A and S2A" are closed and the "S2A and S2B" are opened, then the current will flow from VCC to GND one way through the motor. If you switch the contacts "S2A and S2B" and open "S1A and S1B" instead, the current will flow the other way through the motor. By transmitting current differently through the motor, the direction of rotation of the motor is changed.

The contacts are of course only theoretical and in practice these will be digital so that they can be controlled from e.g. a micro: bit, but the basic theory is the same.

Fortunately, all of this can be obtained as a chip (or Integrated Circuit - IC), and in this example we will use the L293D for the purpose. This is a so-called "dual H bridge" which means it can control two DC motors.

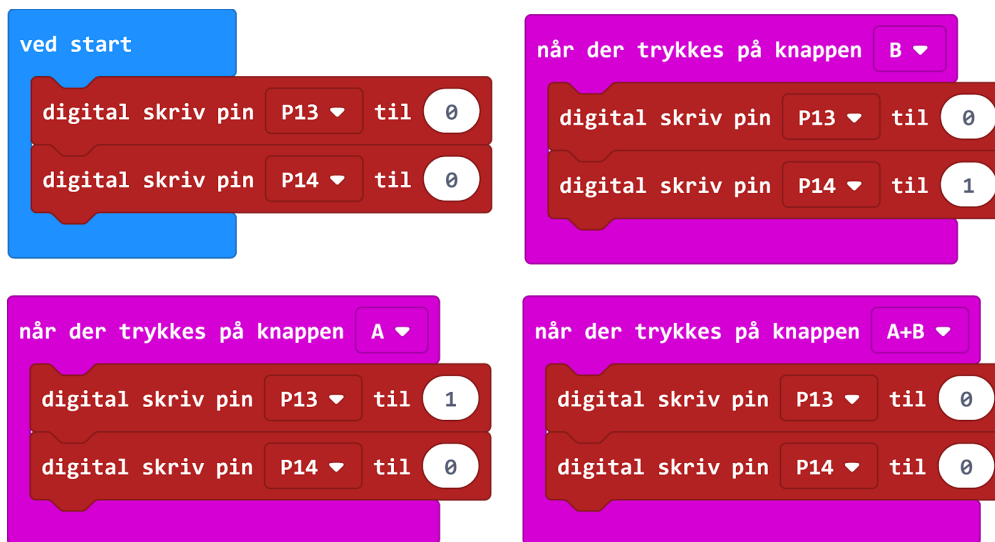
The drawing below shows how to connect this h-bridge to control an engine.



Construction drawing of H-bridge L293D with 1 DC motor

The code for this circuit is very simple, as all the difficult parts are handled within the chip L293D.

The motor is controlled by the buttons "A" and "B" being pressed individually or simultaneously. Button "A" causes the motor to rotate one way and button "B" in the opposite direction. Simultaneously pressing "A + B" causes the motor to stop.



MakeCode - Blockly

```
input.onButtonPressed(Button.B, function () {
  pins.digitalWritePin(DigitalPin.P13, 0)
  pins.digitalWritePin(DigitalPin.P14, 1)
})

input.onButtonPressed(Button.A, function () {
  pins.digitalWritePin(DigitalPin.P13, 1)
  pins.digitalWritePin(DigitalPin.P14, 0)
})

input.onButtonPressed(Button.AB, function () {
  pins.digitalWritePin(DigitalPin.P13, 0)
  pins.digitalWritePin(DigitalPin.P14, 0)
})

pins.digitalWritePin(DigitalPin.P13, 0)
pins.digitalWritePin(DigitalPin.P14, 0)
```

MakeCode - JavaScript

It can be seen from the code that the motor's direction of rotation is controlled by pins P13 and P14 respectively. You have to think of P13 as controlling "S1A and S1B" from the figure above and P14 for "S2A and S2B". The other wires are "Red" supply 5V and "Blue" is GND.

The code can be downloaded here: https://makecode.microbit.org/_aewPtMPYkaeW

10 Exercises

Below are a number of tasks that are based on the individual components and the combination of the components and examples described earlier in this book.

10.1 Turn on light in the dark:

Starting from sections 3 and 4, try to change the functionality so that the LED does not blink, but instead turns on when the measured light falls below a specific value.

10.2 Control the power of the light:

Use the tasks from sections 3 and 5 and change them so that the value read from the potentiometer controls how bright the light is.

10.3 Different lights with NeoPixels


Based on section 8 try to create a light effect with NeoPixels. Try letting them alternate between a variety of colors over and over.

10.4 Position of the motor

When controlling a DC motor, it is often useful to know where the motor is - that is, how it is rotated, what is its current angle here and now. This can be done very accurately with the right sensors, but less can also be of use. Based on sections 7 and 9, try to make a system that stops the DC motor after it has completed one full rotation. At the push of a button the motor must then run another full rotation etc. It will probably be necessary to mount the magnet on a disc of cardboard at the end of the geared DC motor axle.

Program overview

<p>Flash LED:</p> <p>https://makecode.microbit.org/_CU4Lh7EJsDMV</p>	
<p>Measuring light:</p> <p>https://makecode.microbit.org/_ipPTC0bqgYm6</p>	
<p>Potentiometer - voltage:</p> <p>https://makecode.microbit.org/_5zR3wyapA1f5</p>	
<p>Hall effect</p> <p>https://makecode.microbit.org/_H9fUzHU2gRvs</p>	

<p>NeoPixel:</p> <p>https://makecode.microbit.org/_V2ubcyap1cYt</p>	
<p>H-Bridge and DC motor:</p> <p>https://makecode.microbit.org/_aewPtMPYkaeW</p>	