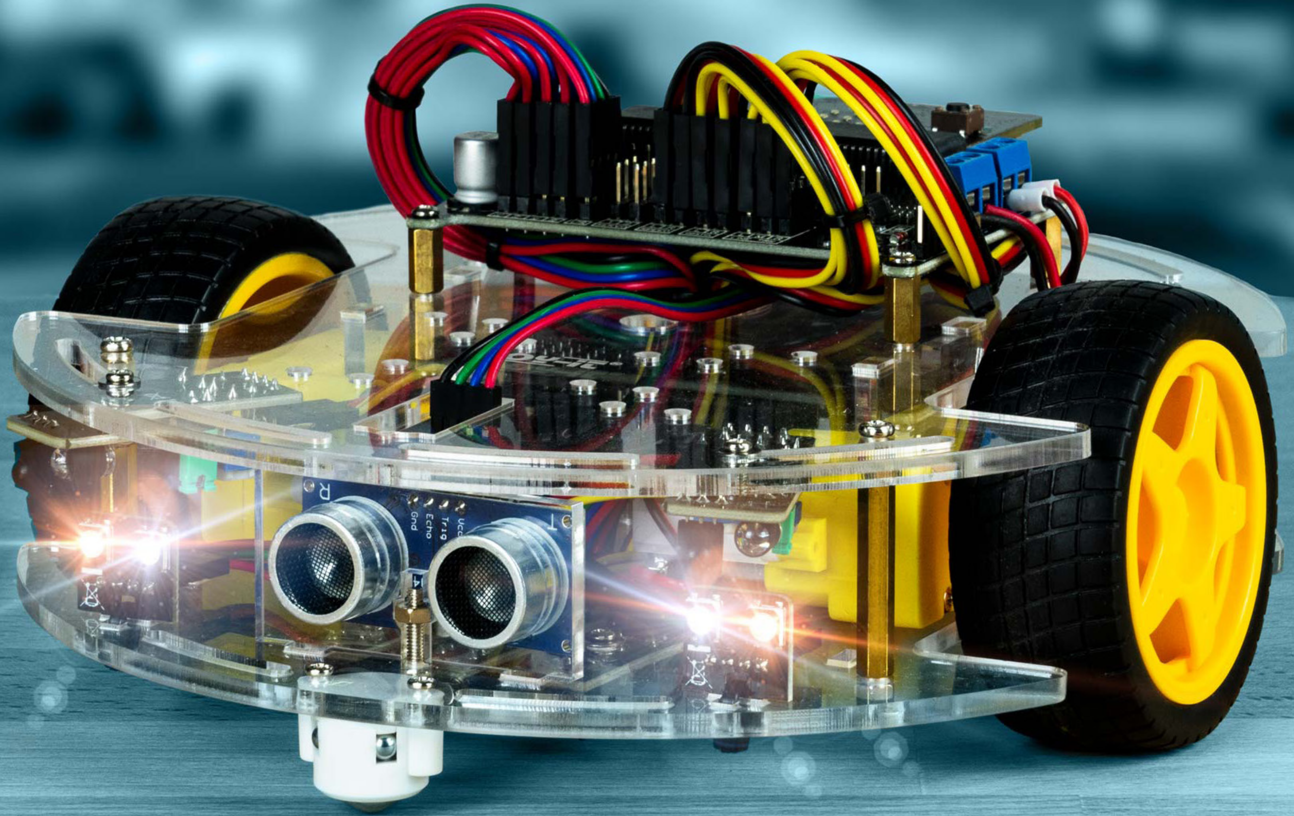
JOY-CAR

Education robot based on micro:bit



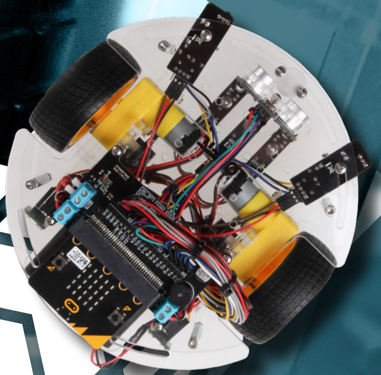
JOY-IT

WWW.JOY-IT.NET

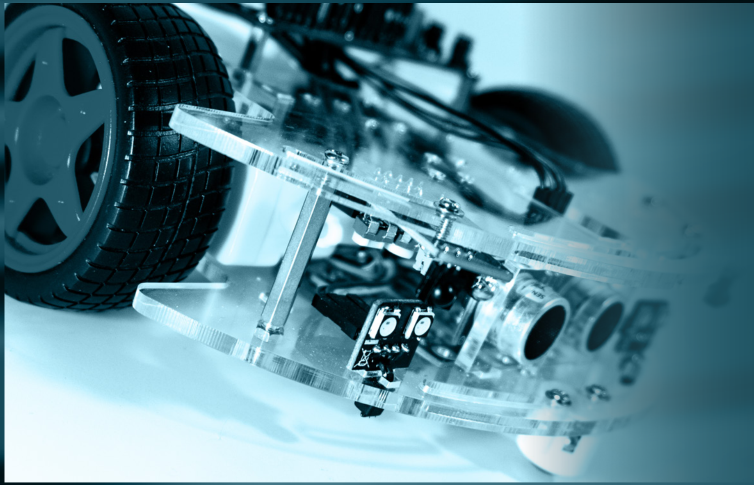
TABLE OF CONTENTS

JOY-CAR

The Joy-Car is a modular built education kit for Robotics. It serves for learning of building parts and their function inside a whole machine. Learning is especially easy because of the detailled manual and the programming. Joy-Car has sensors like the line-finder or the ultra sonic senor as well as a programmable RGB-lighting. You can even control the Joy-Car with the acceleration sensor via a wireless BT connection by using a second micro:bit



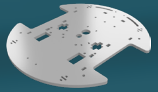
COMPONENTS

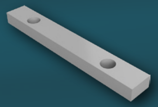


Welcome to your own Joy-Car Assembly kit! There is much to discover, but you won‘t enter this journey alone! In the following chapters, we show you step by step, how you create your own, ridable project out of this assembly kit.

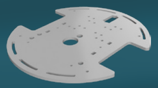
Now, it‘s your time to familiarise yourself with the components, and afterwards, we are ready to start!

CHASSIS

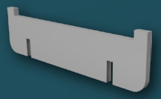

BASE
1x

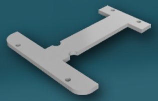

BATTER BRACKET C
1x

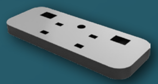
PARKING BRACKETS A
4x

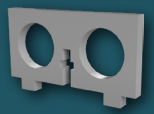

CHASSIS
1x

MOTOR BRACKETS
2x

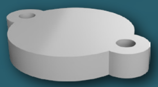

PARKING BRACKETS B
2x


BATTERY BRACKET A
1x

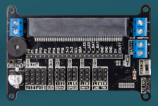

SERVO BRACKET
1x

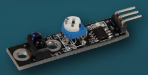

ULTRASONIC BRACKET
1x

BATTERY BRACKET B
1x

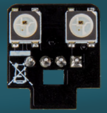

SPACER
1x

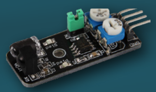
ELECTRONICS


MAINBOARD
1x

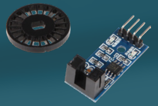

LINETRACKING-SENSOR
3x

BATTERY CASE
1x


LED BOARD
4x

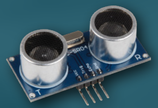

OBSTACLE-SENSOR
2x

3-PIN DUPONT-CABLE SET
7x

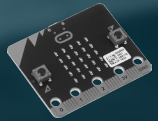

SPEEDSENSOR
2x

SERVOMOTOR
2x

4-PIN DUPONT-CABLE SET
5x


ULTRASONIC-SENSOR
1x

MOTOR
2x


MICRO:BIT
1x
NOT INCLUDED IN EVERY JOY-CAR SET!

ASSEMBLY MATERIAL

SCREW
4x M2,5 x 5 mm

SCREW
5x M3 x 14 mm

SPACER
4x M2,5 x 10 mm

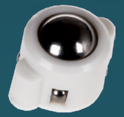
SCREW
12x M2,5 x 10 mm

WASHER
30x M2,5

SPACER
4x M3 x 30 mm

SCREW
4x M2,5 x 22 mm

NUT
20x M2,5

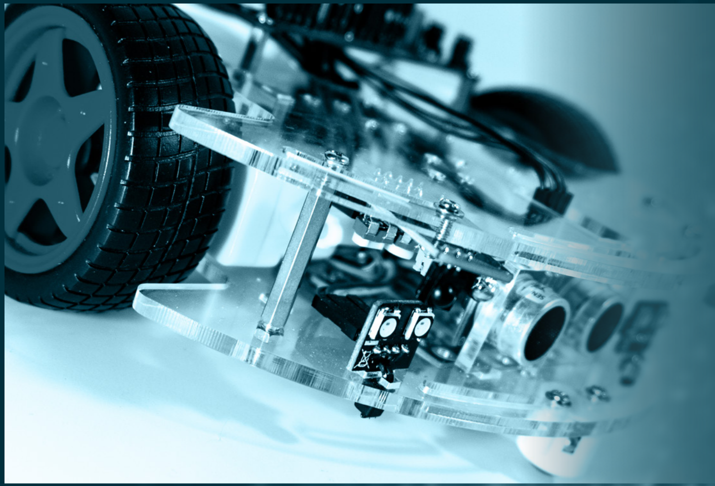

BALL CASTER
2x

SCREW
6x M3 x 8 mm

NUT
15x M3

CABLE TIE
15x

ASSEMBLY



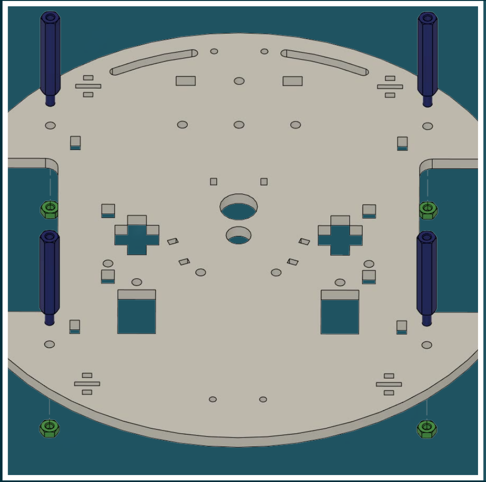
Before we start with your Joy-Car, it has to be assembled first. But no worries! Even though there are many components, modules and cables, we won‘t let you down. In this chapter, we show you step by step how you will assemble your Joy-Car.

You will see, it is easier than you might think and you will learn also about every single part of the Joy-Car!

BASE ASSEMBLY

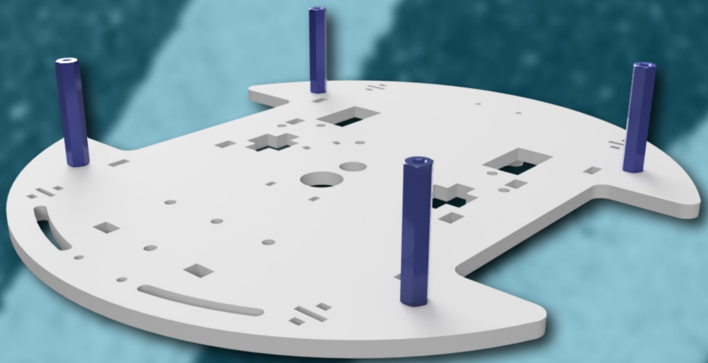
First, we assemble the base. Any brackets and spacers, that are necessary for the following steps, will be mounted now.

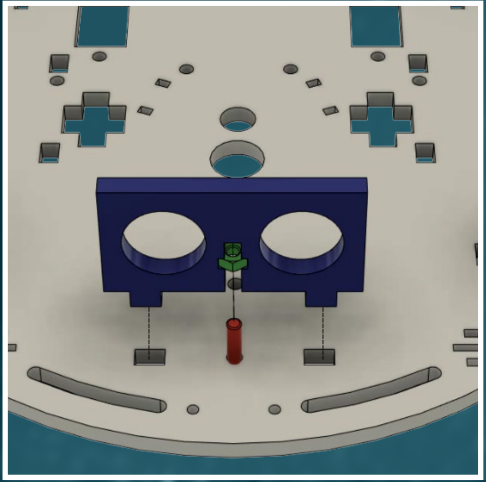
ATTENTION! All acrylic parts are laminated with a protection foil. This foil must be removed before any assembly works.



1. MOUNT SPACERS

First, mount the 4 spacers (M3 x 30mm) to the base and secure them from below with the matching nuts (M3).

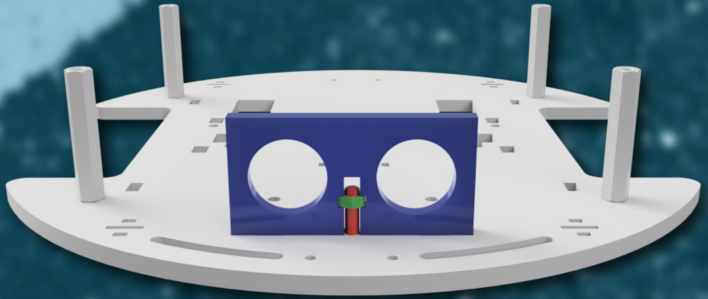


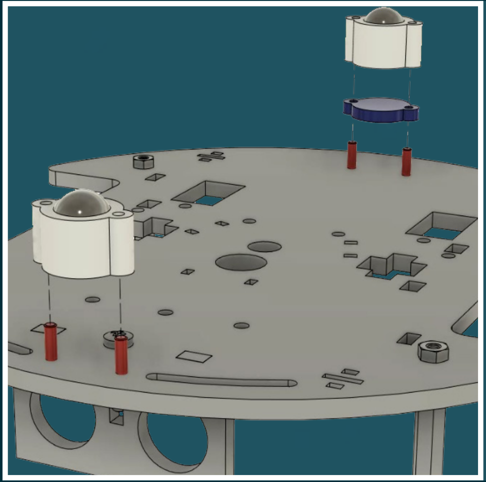


2. MOUNT ULTRASONIC BRACKET

Mount the ultrasonic bracket to the base. Secure them from below with the matching screw (M3 x14 mm) and the matching nut (M3).

ATTENTION! Advanced users can also mount the ultrasonic sensor moveable together with the servor motor onto the chassis top part (see for alternative ultrasonic assembly). For beginners, the following assembly suits the most.

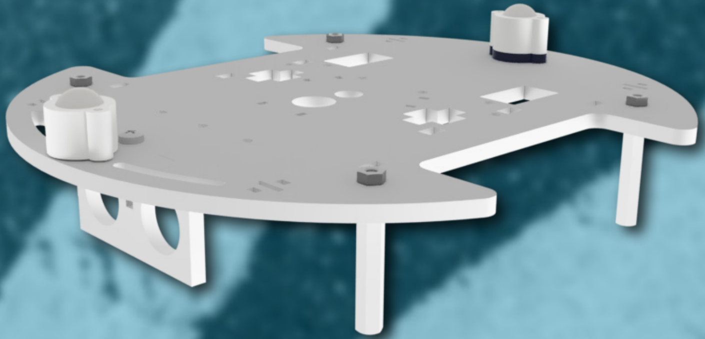


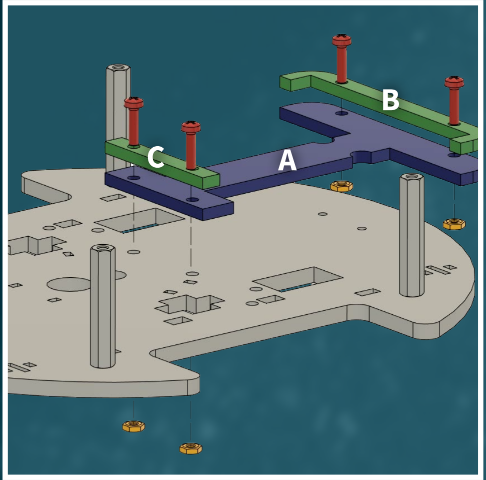


3. MOUNT BALL CASTERS

Mount the ball casters to the base and lock them from above with the matching screws (M2,5 x10mm)

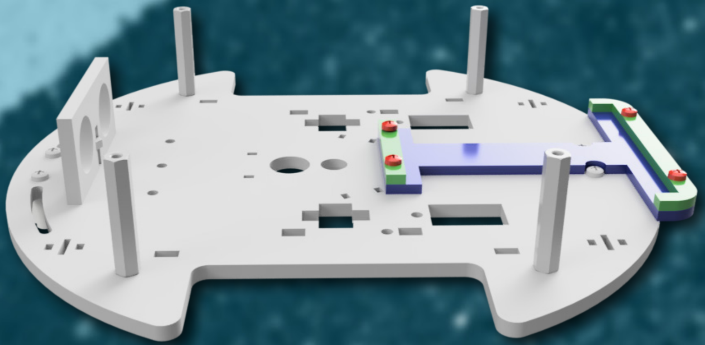
ATTENTION! Use the spacer for the mounting of the ball caster at the rear axle.





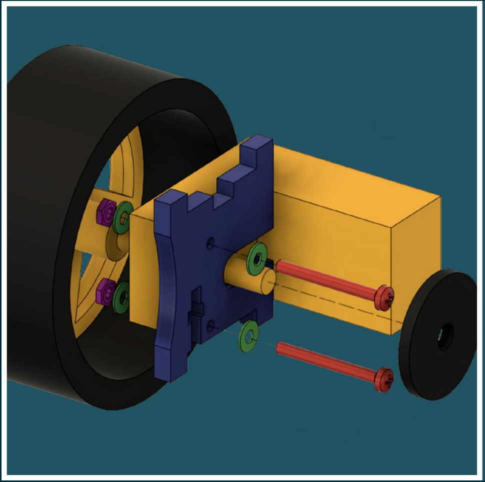
4. MOUNT THE BATTERY CASE

Take the battery bracket A and put it on the upper side of the base. The short end of the battery bracket should point towards the middle of the base. Place battery bracket B onto the wide side of the battery bracket A and secure them with the matching screws (M2.5 x10mm) and nuts (M2.5). The battery bracket C is placed on the narrow side of the battery bracket A. This also needs to be secured with the matching screws (M3x 14mm) and nuts (M3).



DRIVE TRAIN ASSEMBLY

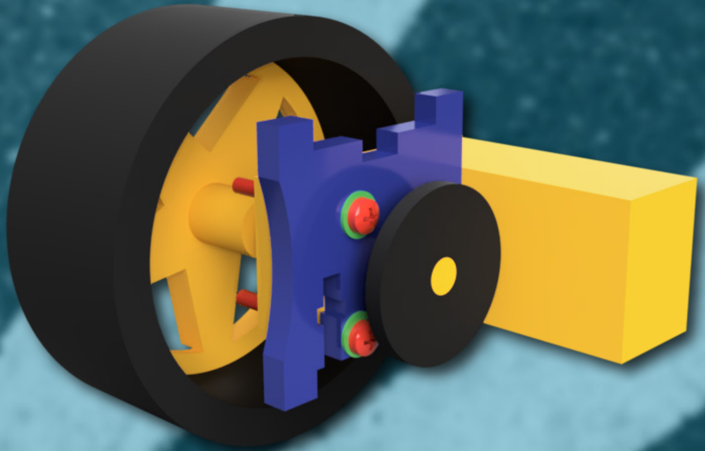
As next step, we assemble the drive train and mount it on the Joy-Car. The drive train contains the motors and is therefore responsible for the propulsion of the Joy-Car.

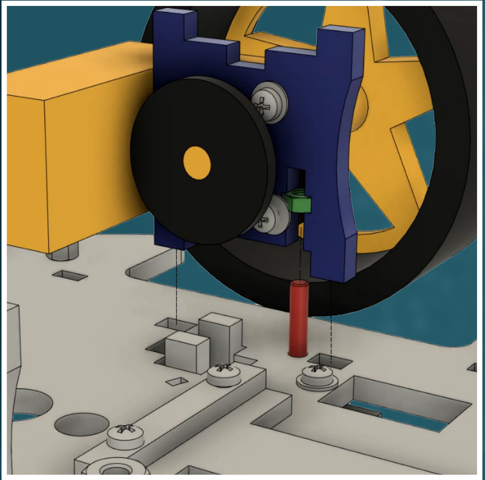


1. PRE-ASSEMBLE THE MOTORS

Put both motors into the ducts of the motor brackets and secure them with the matching screws (M2.5 x22mm) and nuts (M2.5). Additionally use two washers (M2.5) per screw, one on the screw head and one on the screw end.

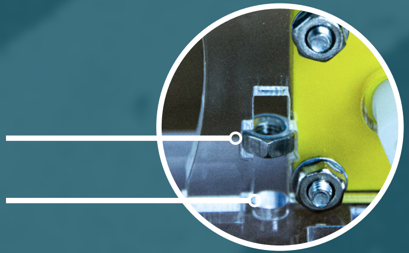
Afterwards, put the black perforated disk on the inner side of the motor.

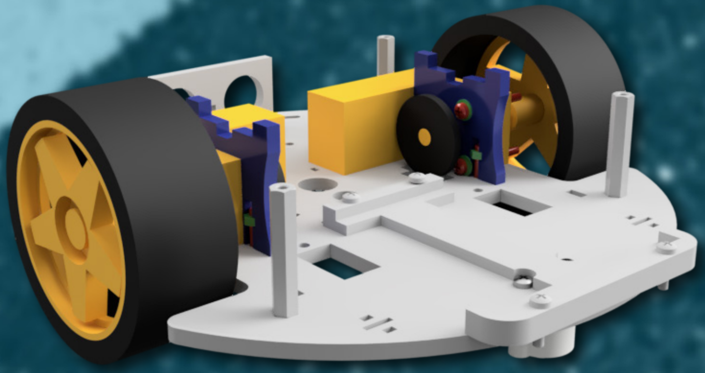




2. MOUNT THE BRACKET TO THE CHASSIS

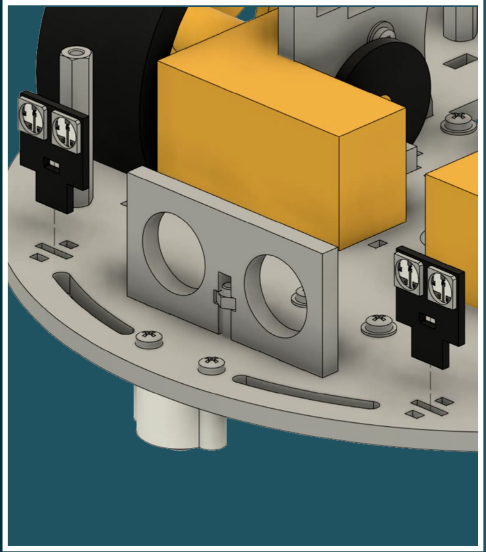
Now mount the motor brackets onto the base. Secure them from below with the matching screw (M3 x14mm) and nut (M3).





ELECTRONICS ASSEMBLY

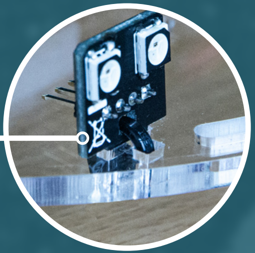
Next, we insert all electronic components into the base. This includes not only the individual sensors, but also the LED modules which are used as headlights.

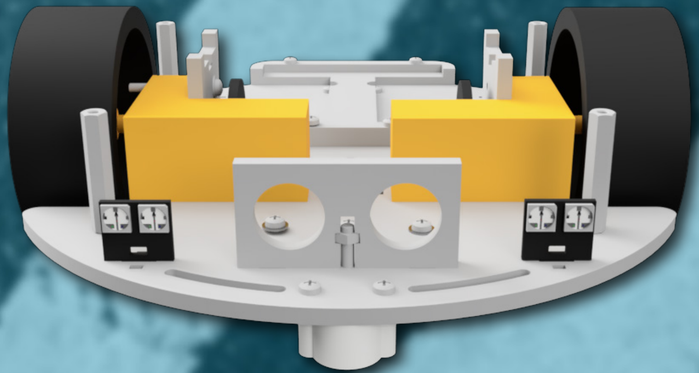


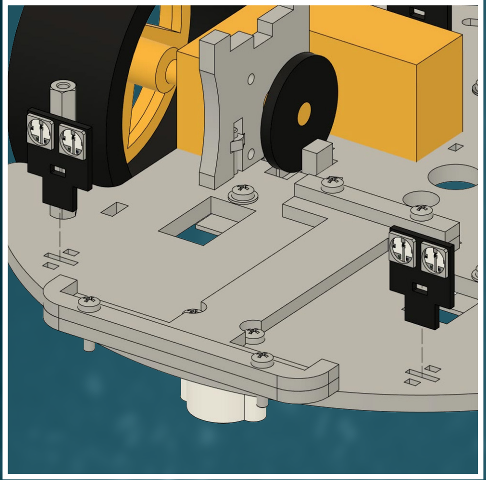
1. FRONT HEADLIGHTS

Insert the WS2812B LED headlight modules into the front of the base. Secure them with the cable ties. You can simply insert the cable tie through the hole of the headlights and the hole of the base and set it tight.

INSERT CABLE TIE TROUGH HOLE OF HEADLIGHT AND HOLE OF BASE AND SET IT TIGHT.

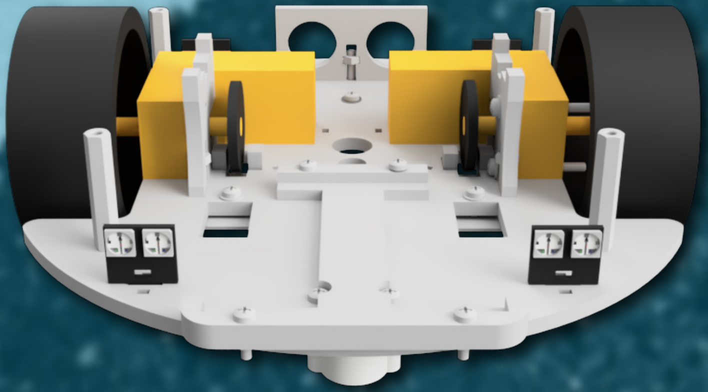


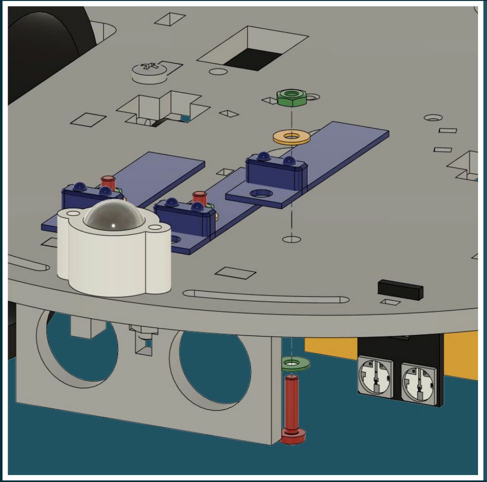




2. REAR HEADLIGHTS

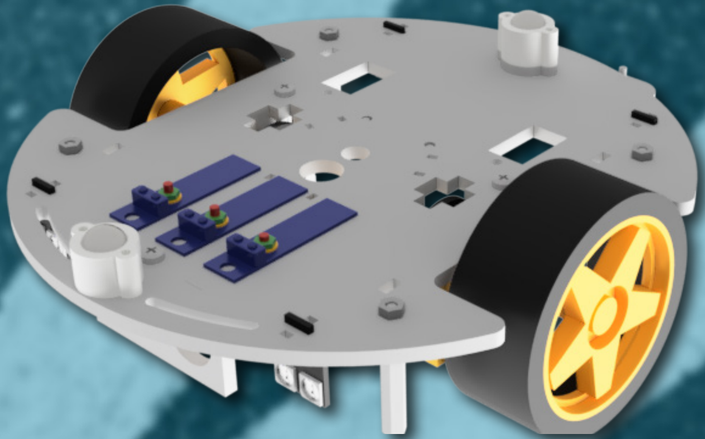
Mount the headlight modules on the rear of the base with the cable ties.

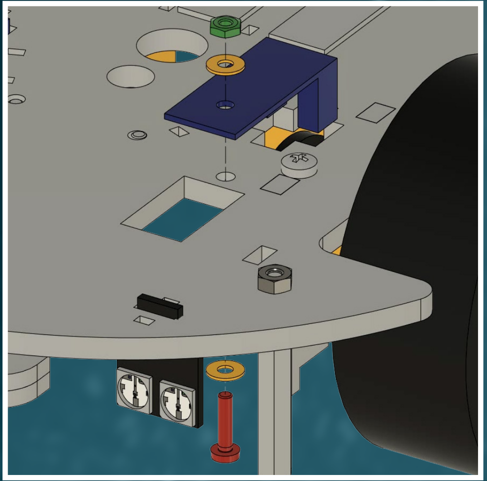




3. LINETRACKING-SENSORS

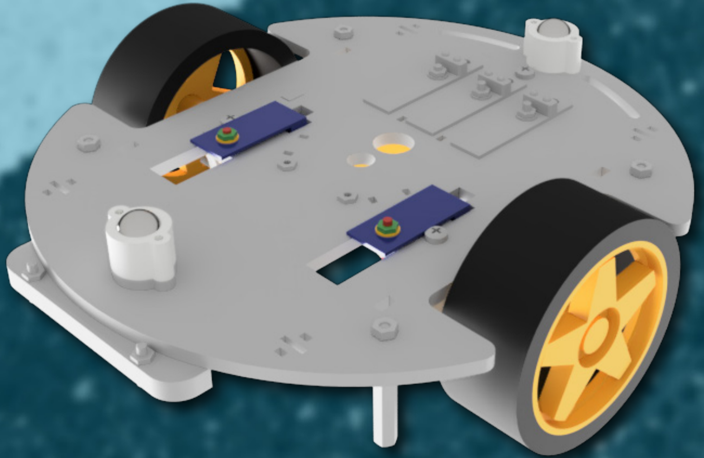
Mount the linetracking-sensors at the bottom side with the matching screws (M2.5 x10mm) and nuts (M2.5). Use 2 washers per screw (M2.5).

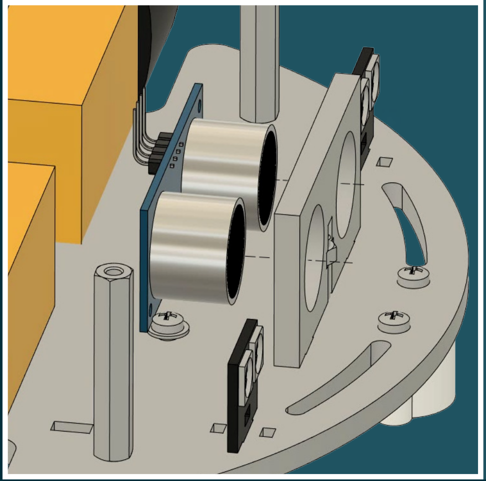




4. SPEED-SENSORS

Insert the speed-sensors into the bottom side of the base. Secure them with the matching screws (M2.5 x 10mm) and nuts (M2.5). Again, use 2 flat washers (M2.5) per screw.

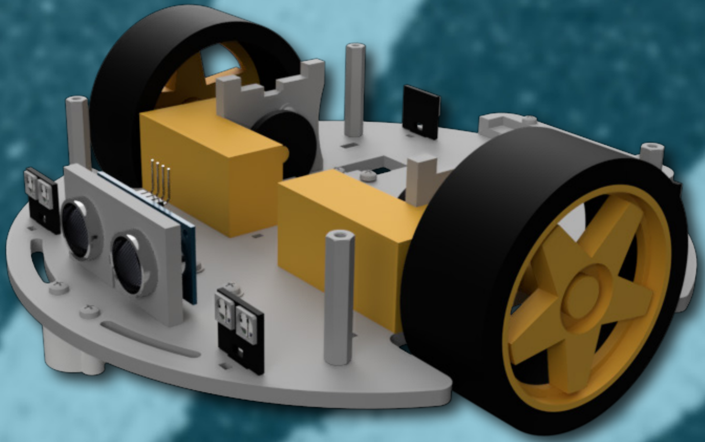


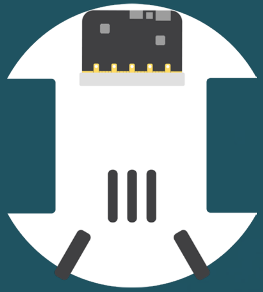


5. ULTRASONIC-SENSOR

Insert the ultrasonic-sensor into the bracket. If needed, you can additionally secure it with some glue. The 4 connection pins need to point upwards.

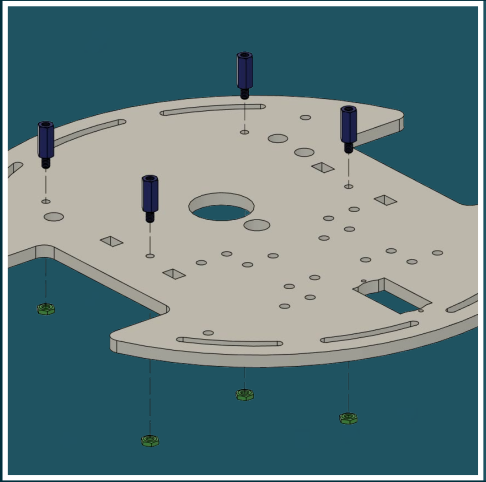
ATTENTION! If you should have decided to assemble the ultrasonic-sensor on a moveable servo motor at the second step of the base assembly procedure, then please skip this step.





CHASSIS ASSEMBLY

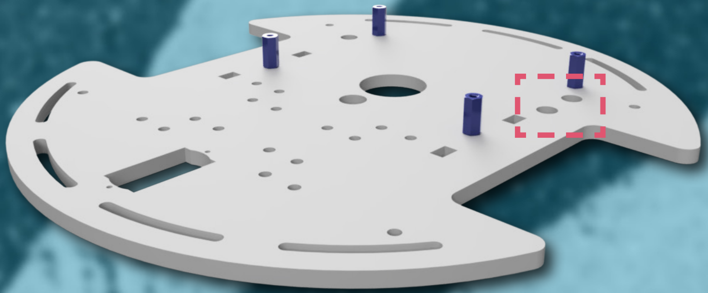
The base is now ready. We will now start with the chassis. This includes, besides the mainboard bracket, the obstacle sensors.

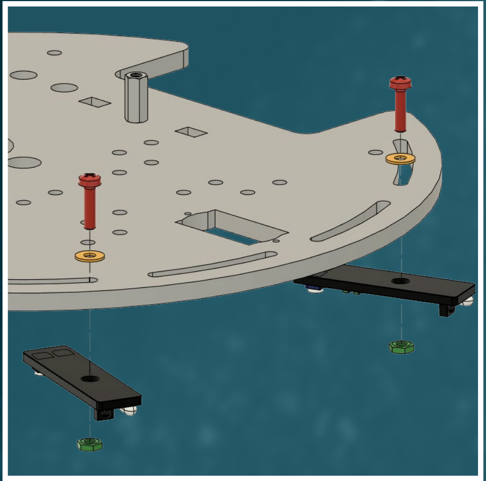


1. ATTACH SPACERS

Attach the 4 spacers (M2.5 x10mm) onto the chassis and secure them from below with the matching nuts (M2.5).

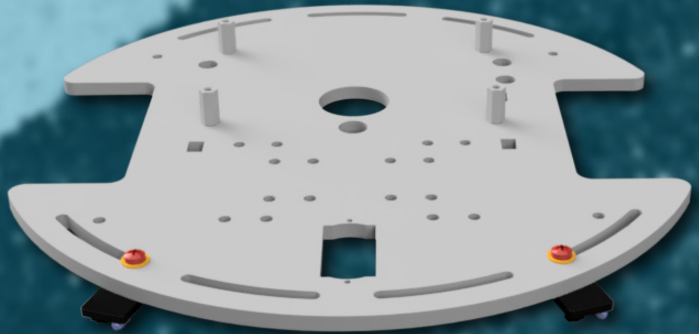
ATTENTION! Please mind that the two highlighted holes are located on the left side of the driving direction.

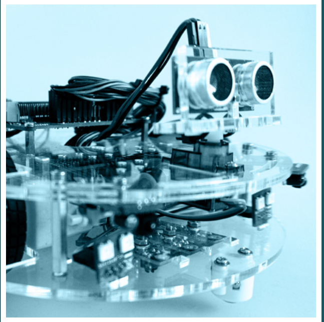




2. OBSTACLE-SENSORS

Mount the obstacle-sensors on the bottom side of the chassis with the matching screws (M2.5 x10mm), washers and nuts (M2.5).



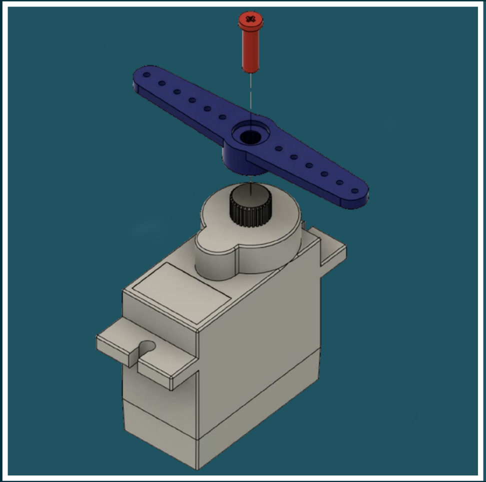


⚠ ULTRASONIC-ALTERNATIVE-ASSEMBLY ⚠

The ultrasonic-sensor can alternatively be mounted on the chassis. Here, it will be installed with a servo Motor and thereby offers a bigger measurement area.

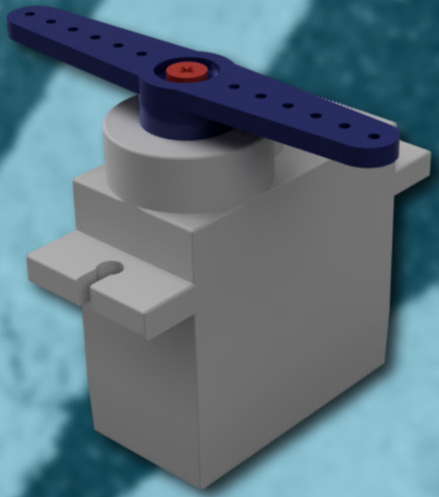
If you prefer this variant and have skipped the mount on the base, you can continue with this chapter.

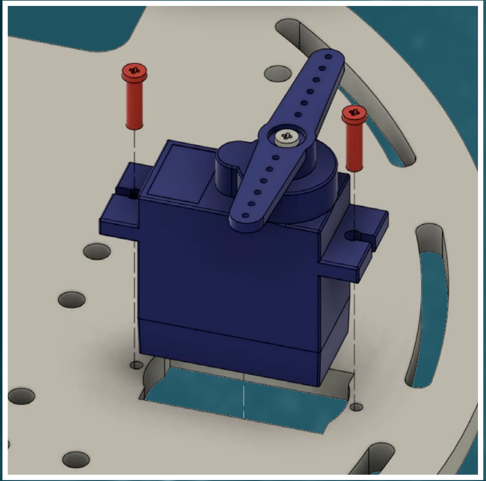
Otherwise, please continue with the next chapter.



1. ATTACH SERVO ARM

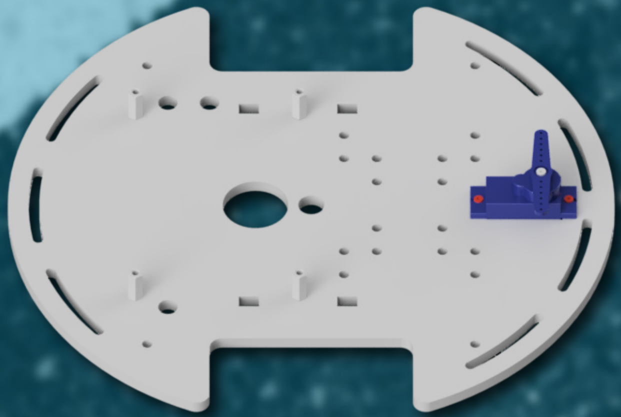
Attach the servo arm onto the gear head of the motor. Secure it with the screw delivered with the motor.

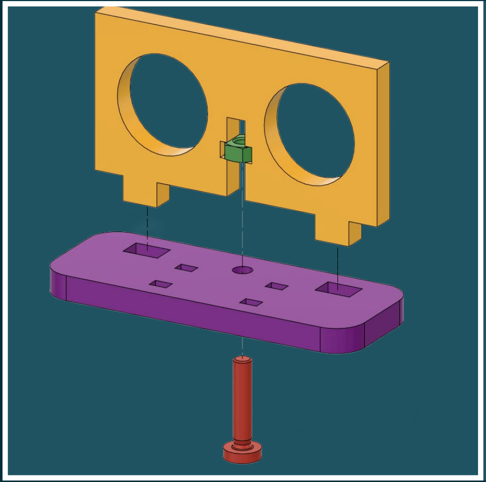




2. INSERT SERVO MOTOR

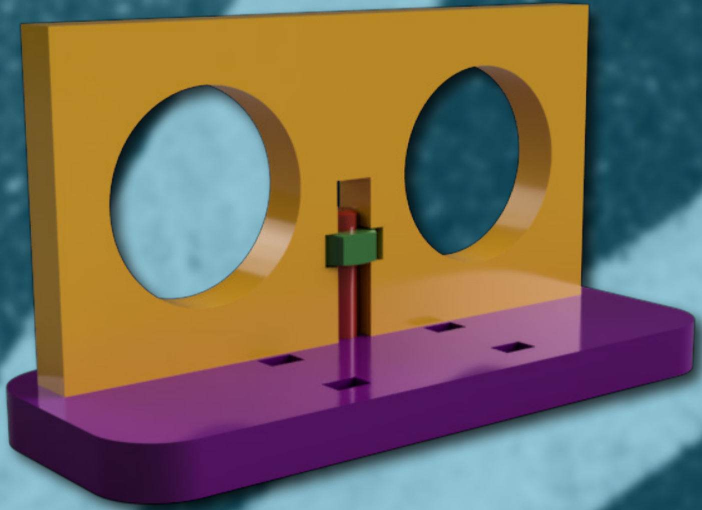
Now insert the servo motor into the slot on the chassis. But the cable of the motor first, then the motor.

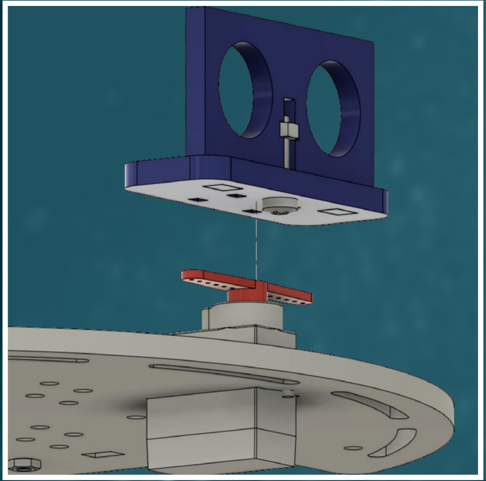




3. ASSEMBLE THE ULTRASONIC-BRACKET

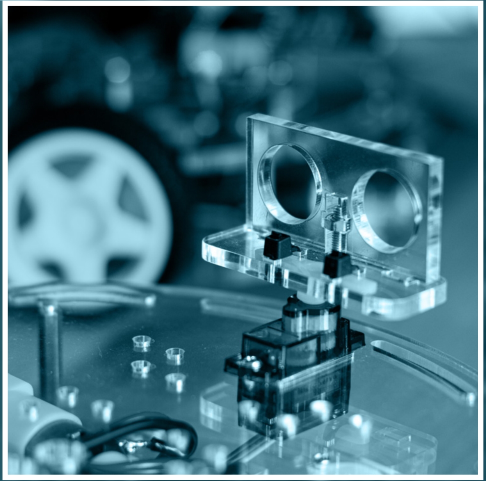
Assemble the bracket onto the bracket of the servo motor and screw them with the matching screw (M3 x 14mm) and nut (M3).





4. MOUNT THE ULTRASONIC BRACKET

Place the ultrasonic-bracket onto the servo motor arm and secure it with the delivered cable ties.



WIRING

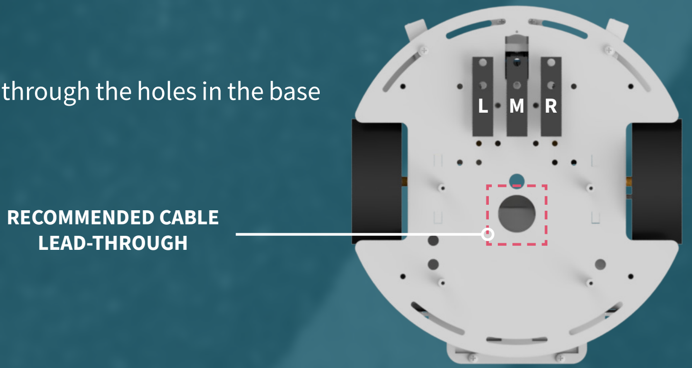
Now it is time to wire the electronic components to the mainboard of the Joy-Car.

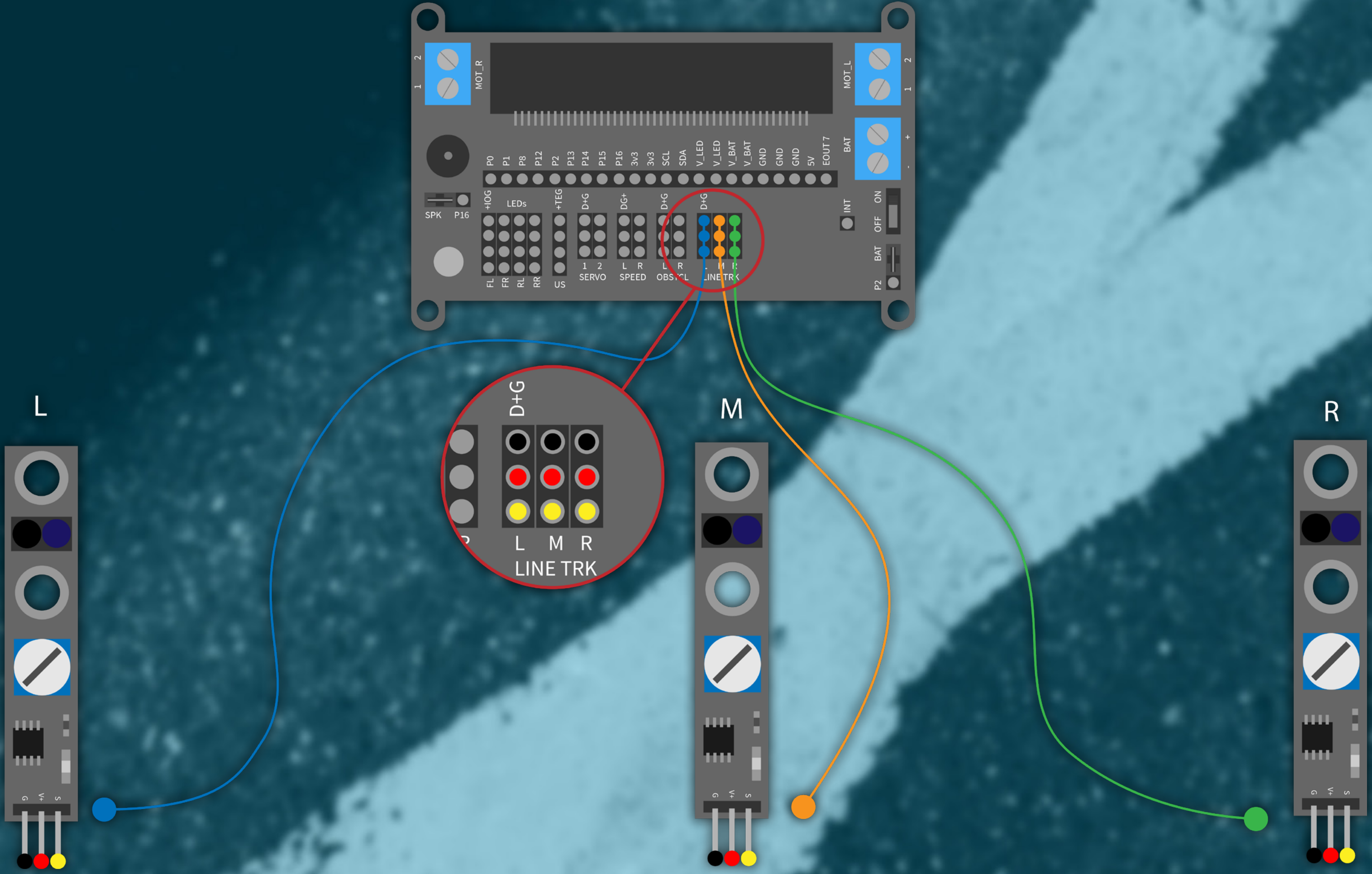
ATTENTION! The mainboard, chassis and base are not screwed together yet. But now is the best time to connect and lay the cables easily before connecting the three units together.

1. WIRE THE LINETRACKING-SENSORS

The 3 linetracking-sensors are each connected with a 3 pin cable. The other cable end can be lead through the holes in the base and the chassis and can be connected to the board of the Joy-Car.

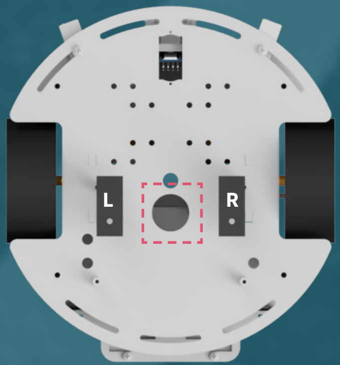
ATTENTION! The sensor's sensitivity can be adjusted additionally. You can find the necessary details **here**.

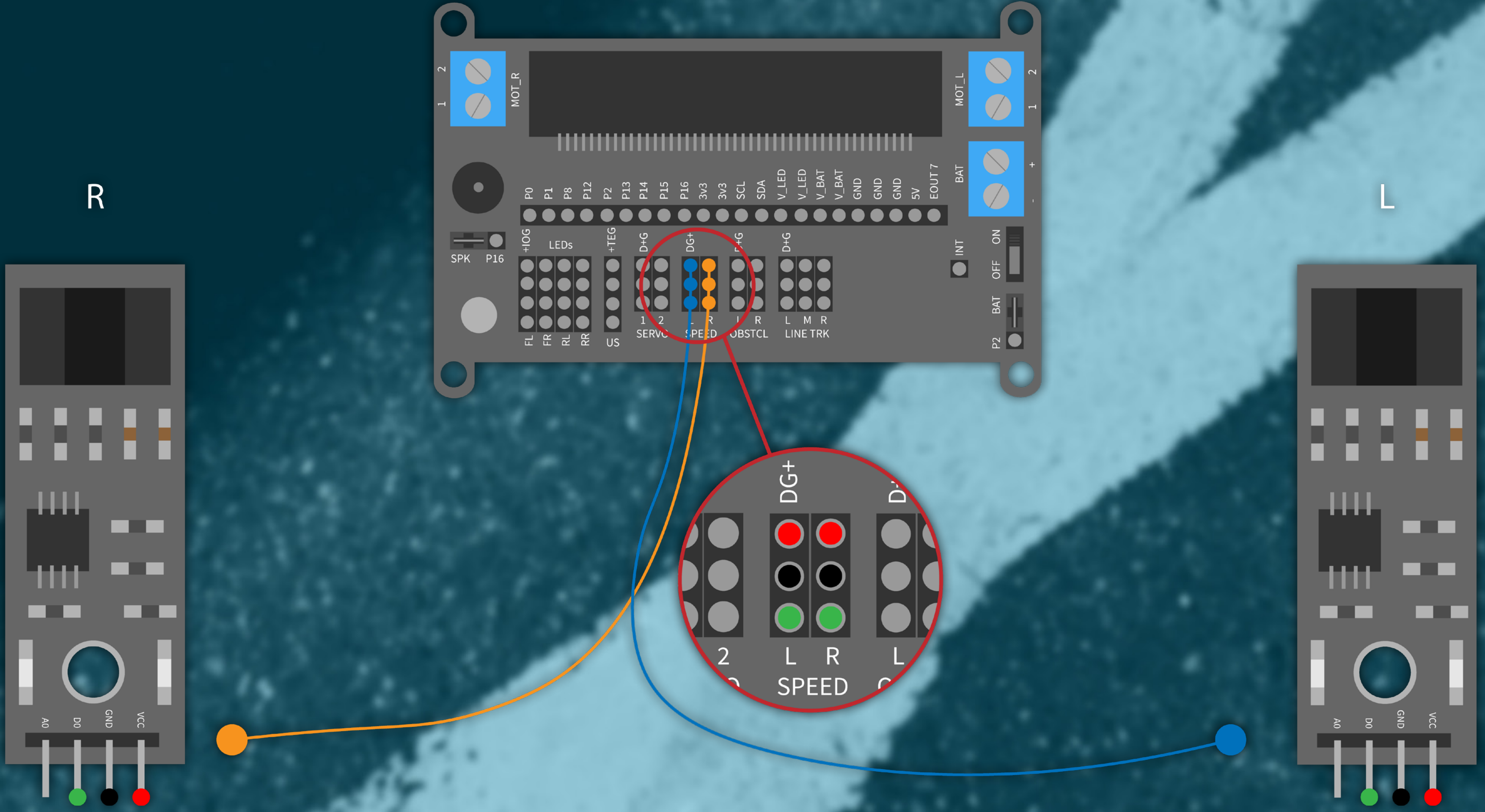




2. WIRE THE SPEED-SENSORS

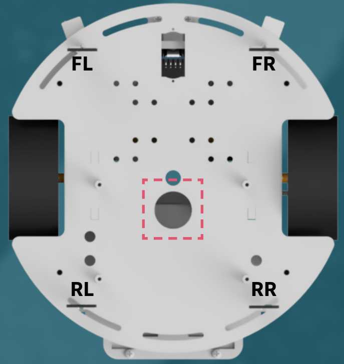
The 2 speed sensors are also connected with a 3-pin cable each and connected to the board of the Joy-Car.

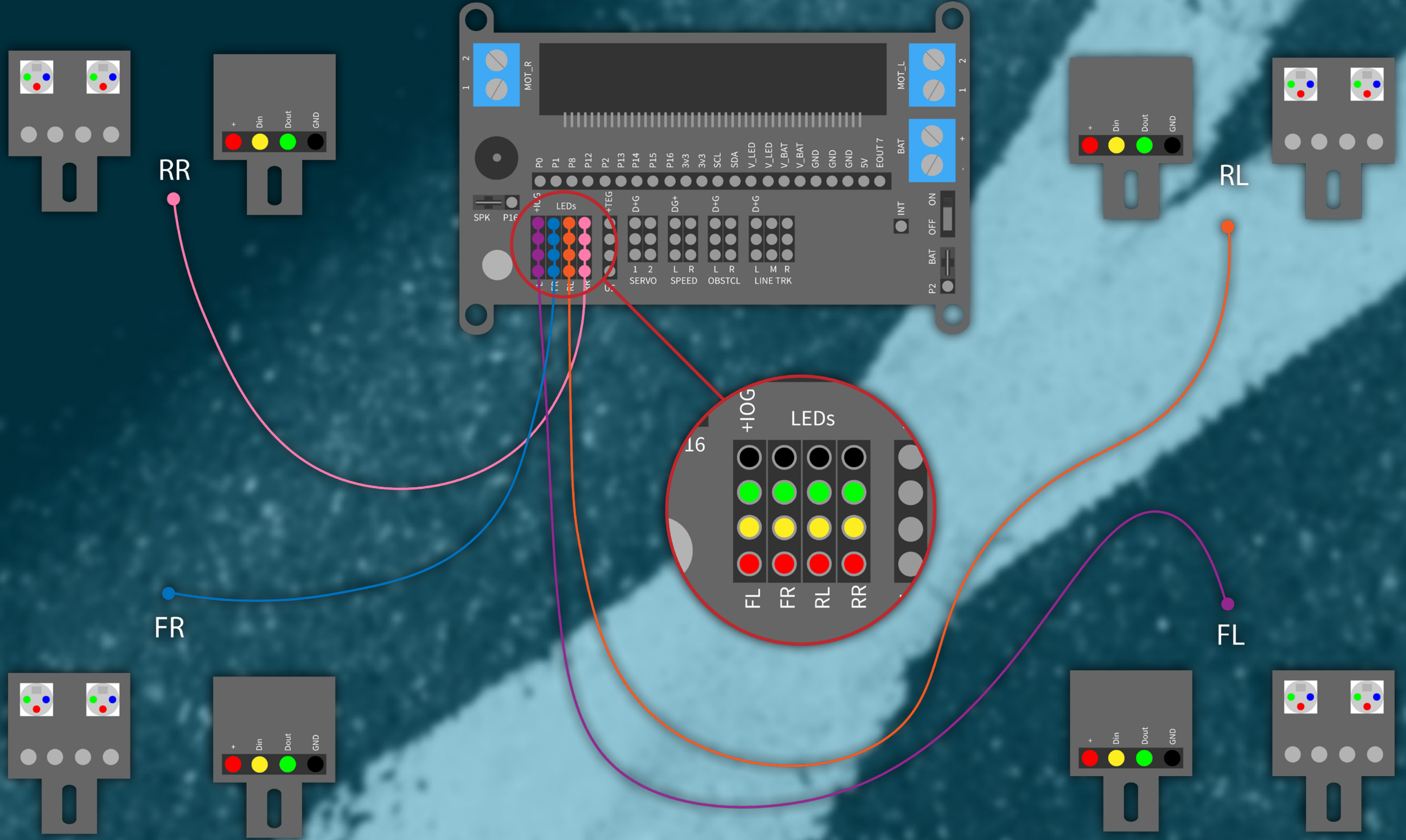




3. WIRE THE HEADLIGHTS

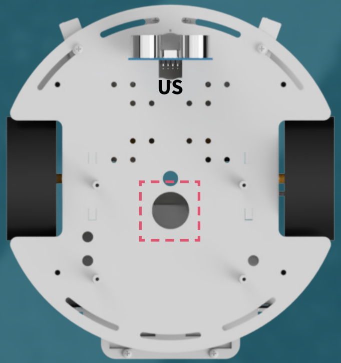
The 4 headlight modules are each connected with a 4-pin cable and conducted to the mainboard of the Joy-Car.

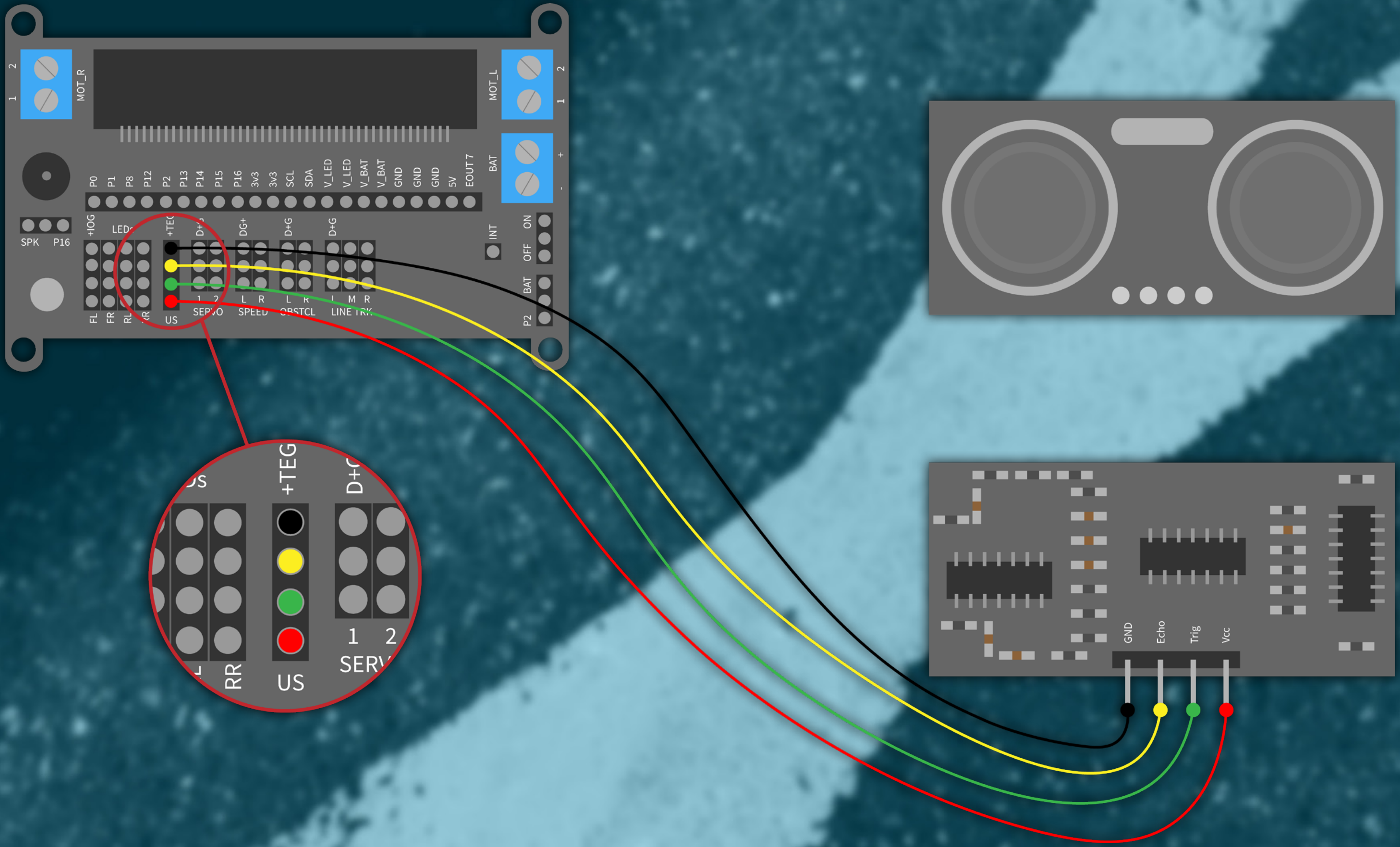




4. WIRE THE ULTRASONIC-SENSOR

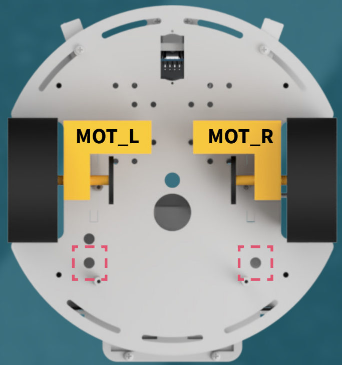
The ultrasonic-sensor is connected with the mainboard by a 4-pin cable as well.

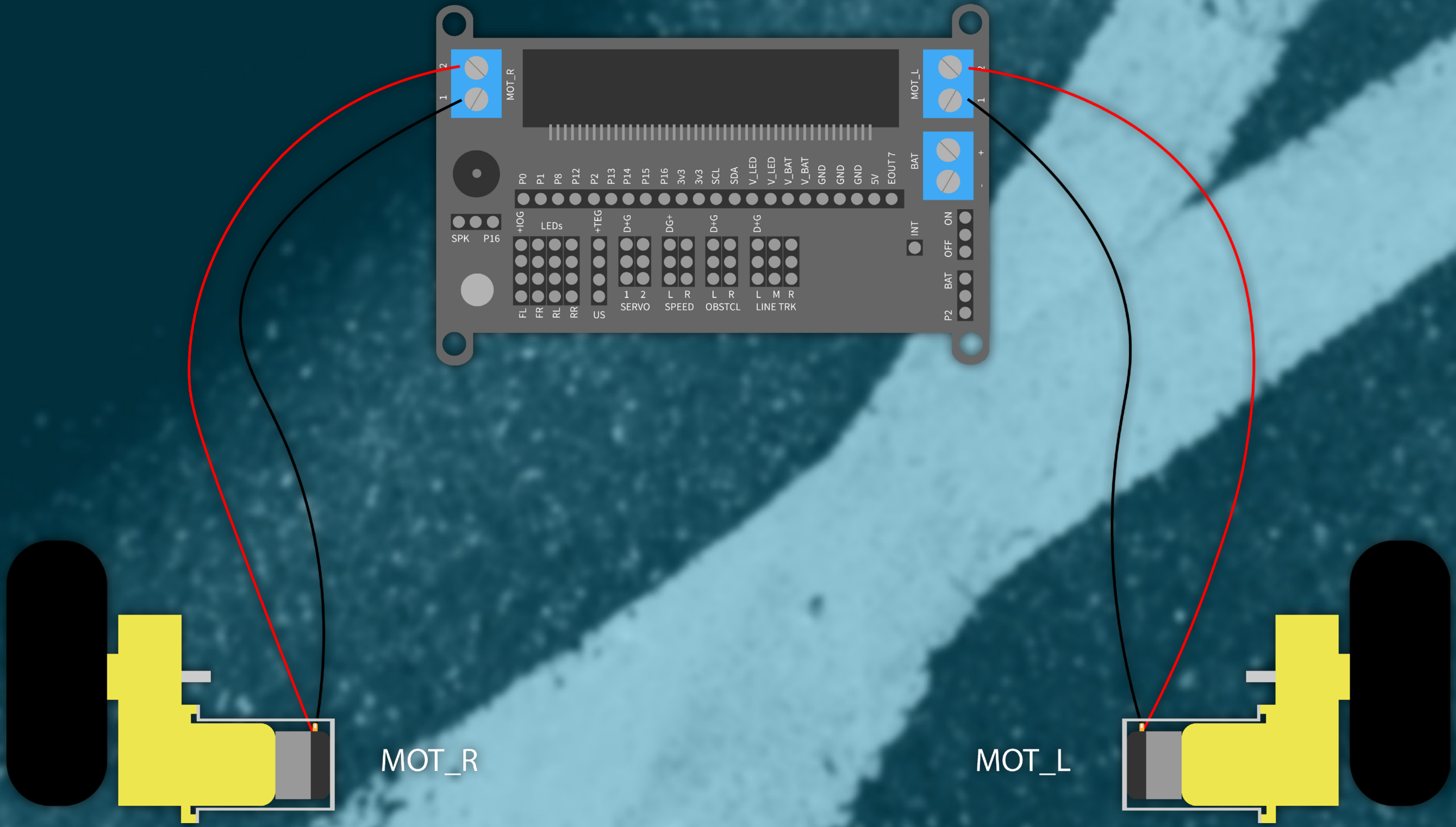




5. WIRE THE MOTORS

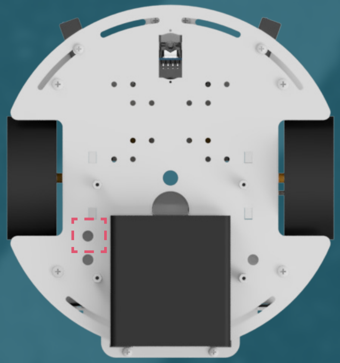
Both motors are already pre-wired with two cables. These are conducted to the terminal screws on the side of the board of the Joy-Car. Here you will need a screwdriver to loosen and secure the clamps again, after you have inserted the cables.

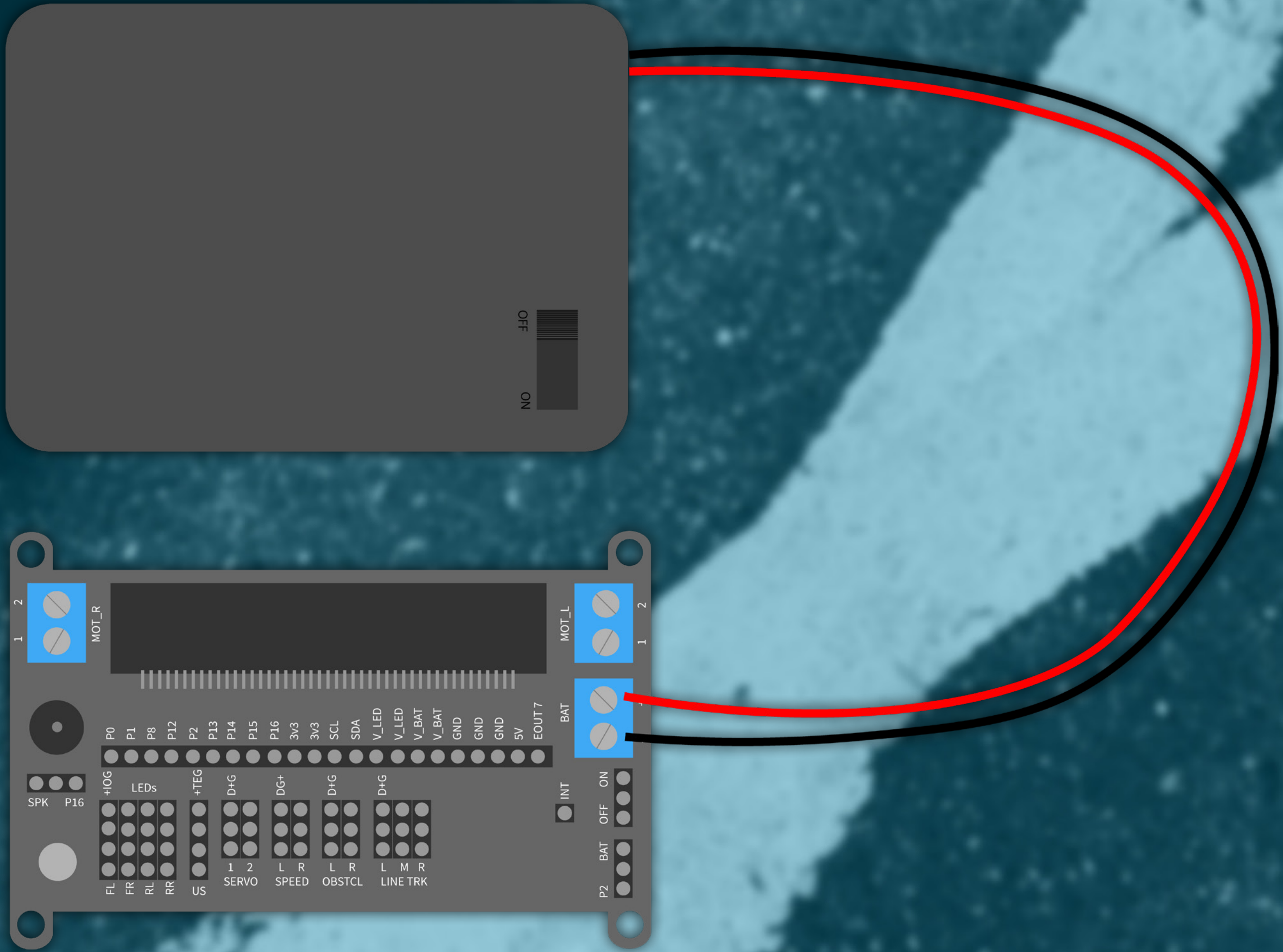




6. WIRE THE BATTERY CASES

The battery cases are also pre-wired with 2 cables, just like the motors before. These are secured at the corresponding clamps on the board of the Joy-Car, as well. Thereby, the red cable is destined for the ‚+' clamp and the black cable is destined for the ‚-' clamp. Afterwards, you can simply place the battery case into the already mounted battery bracket. It is safe there and cannot slip.

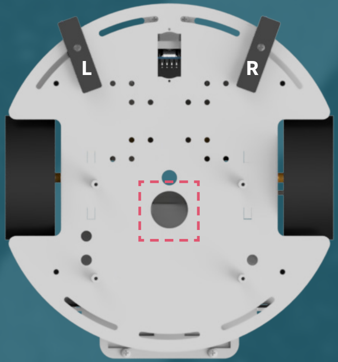


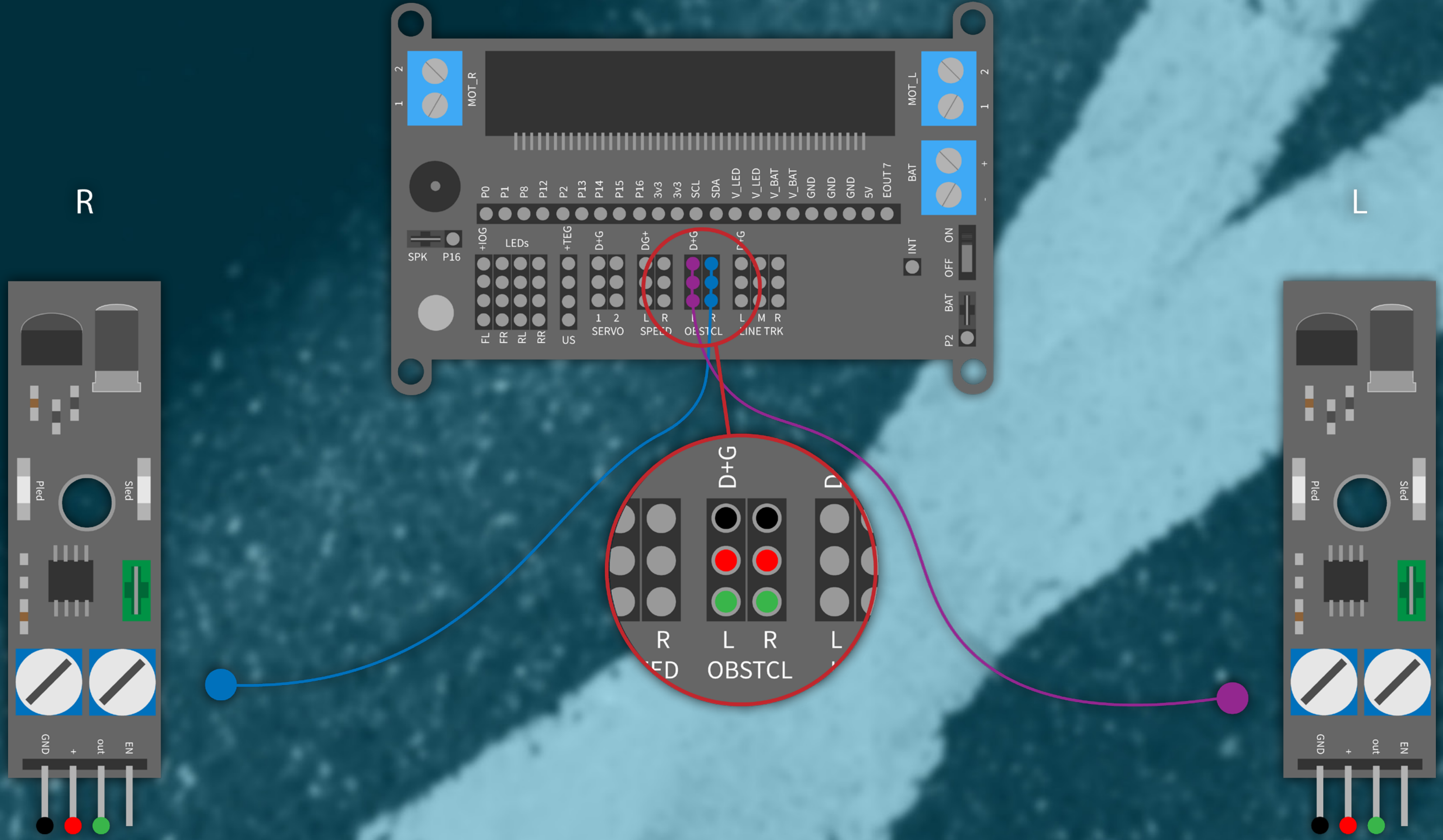


7. WIRE THE OBSTACLE SENSORS

The 2 obstacle sensors have 4 pin headers, but only require 3 cables. They are therefore connected with a 3-pin cable only and conducted to the board.

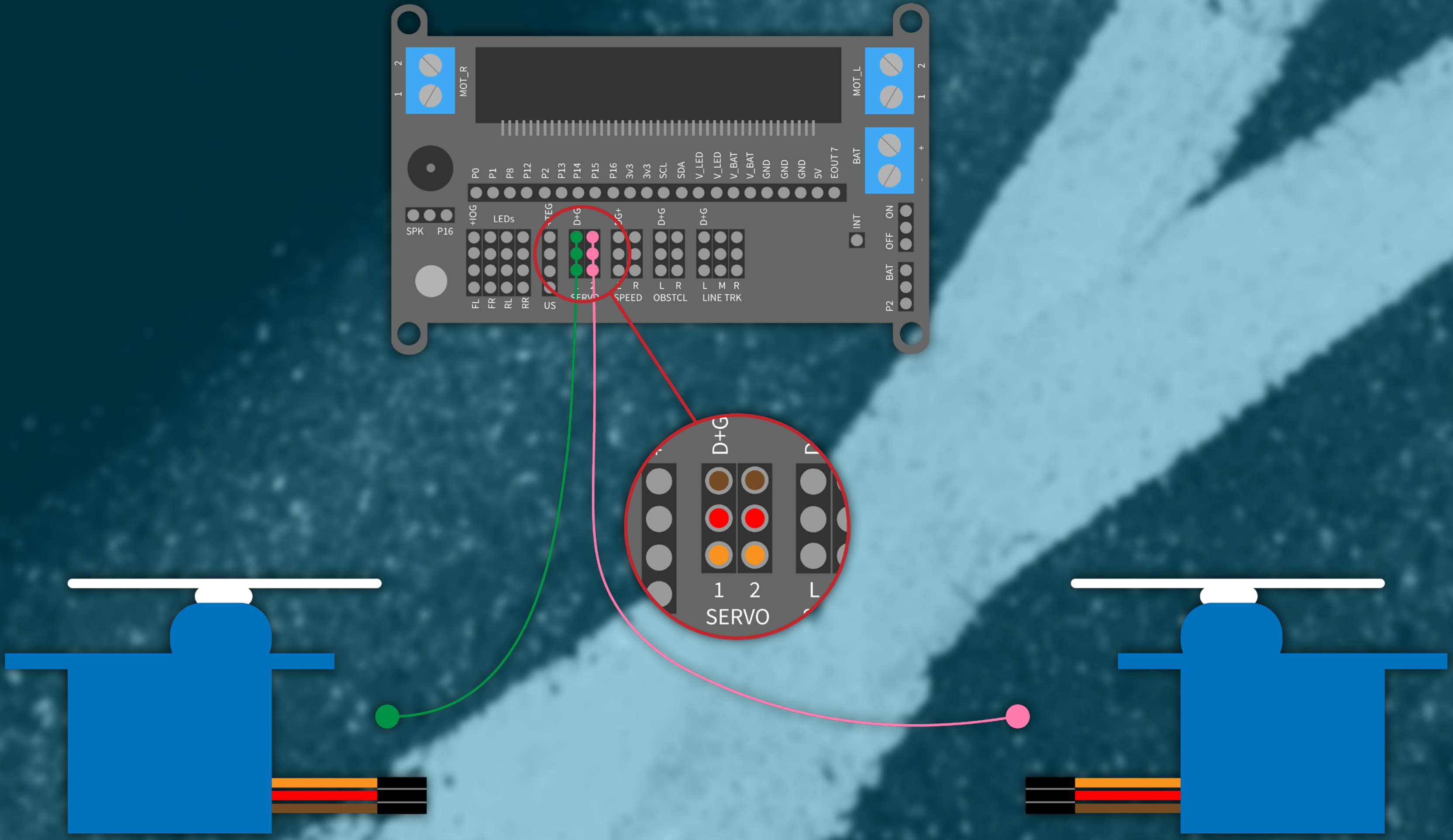
ATTENTION! The sensor‘s sensitivity can additionally be adjusted. You can find more details **here**.





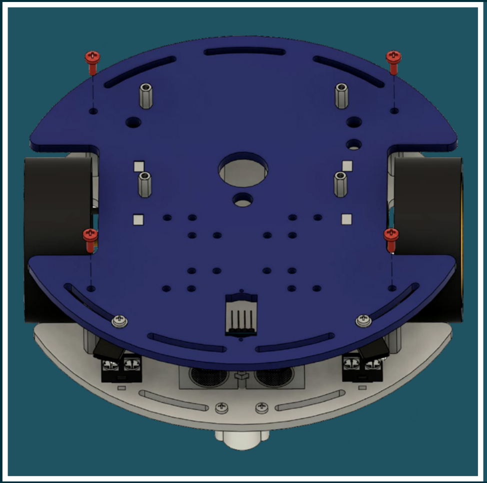
8. OPTIONAL: WIRE THE SERVO MOTORS

If you have mounted the ultrasonic-sensor with a servo motor in the **alternative ultrasonic assembly**, then the servo motor will be connected with a 3-pin cable to the first servo motor connector. You can also optionally connect a second servo motor, which you can use for individual programming. In case you have neither used the alternative ultrasonic assembly nor wish to install an optional servo, please skip this step.



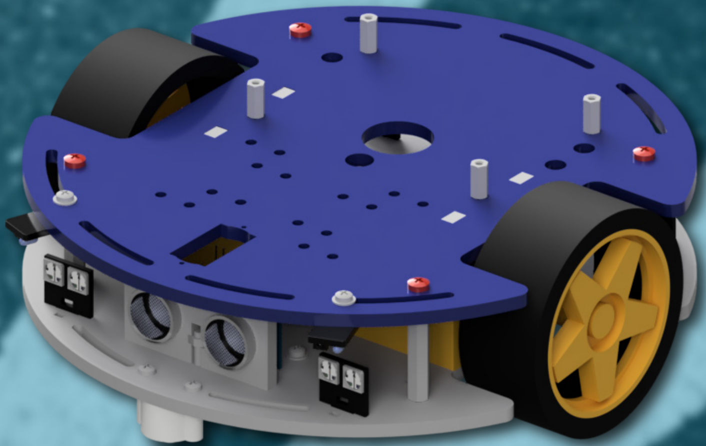
FINISH

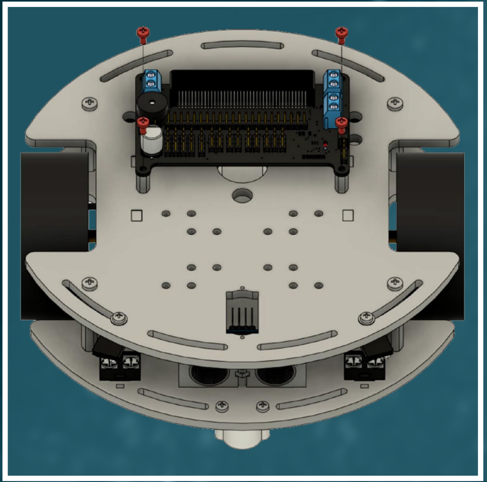
You are almost done! Since now everything is monted and wired, we will now attach the chassis to the base, secure the mainboard and insert the Micro:Bit.



1. WEDDING

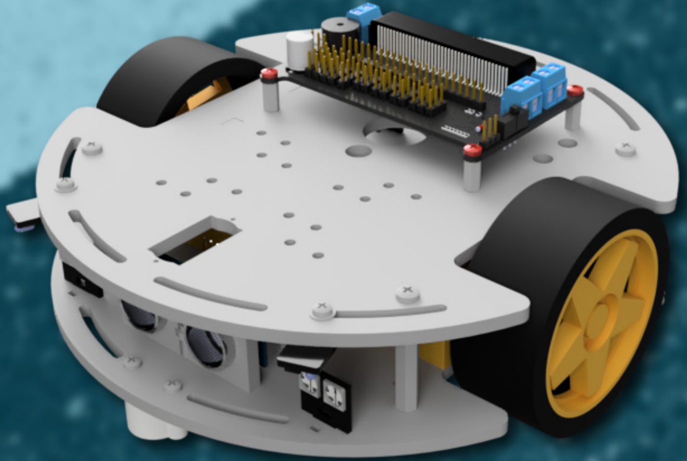
Since all of the sensors and cables are installed, the chassis can finally be attached. Put the chassis on the top of the base and secure it with the 4 spacers using the matching screws(M3 x 8mm).

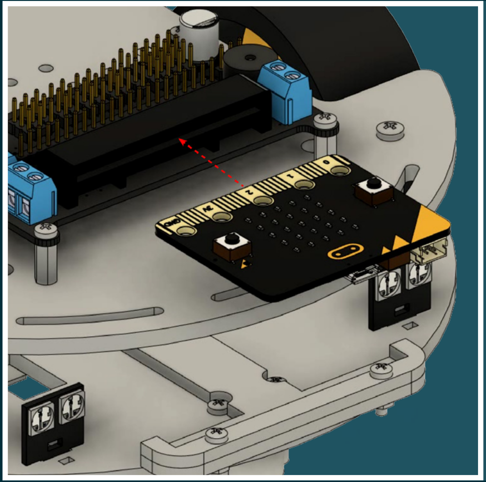




2. MAINBOARD

Now put the mainboard of the Joy-Car onto the already mounted spacers on the chassis and secure it with the 4 matching screws (M2.5 x 5mm).

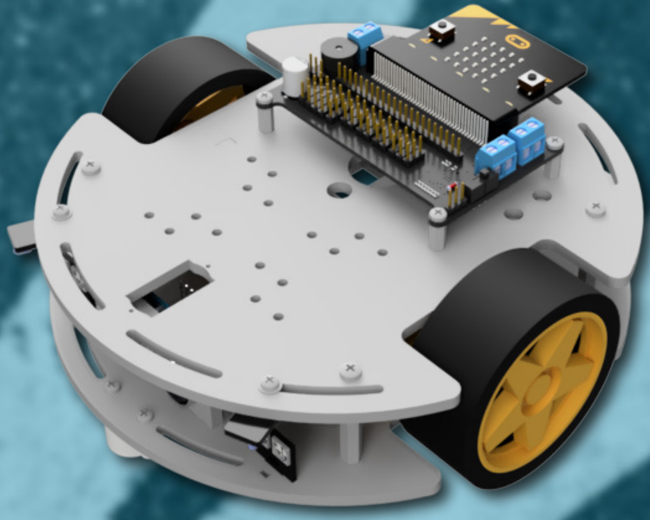


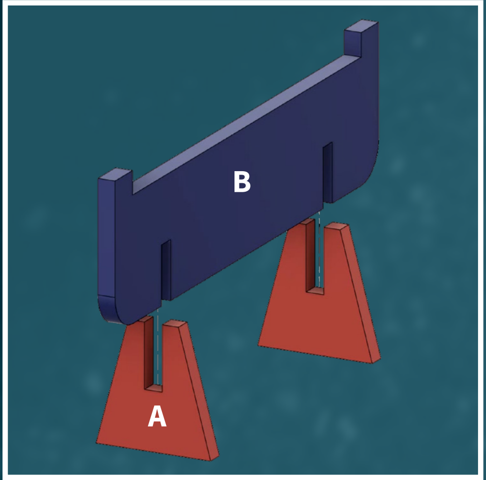


3. INSERT THE MICRO:BIT

Now insert your micro:bit into the brackets of the mainboard and make sure the two buttons point upward.

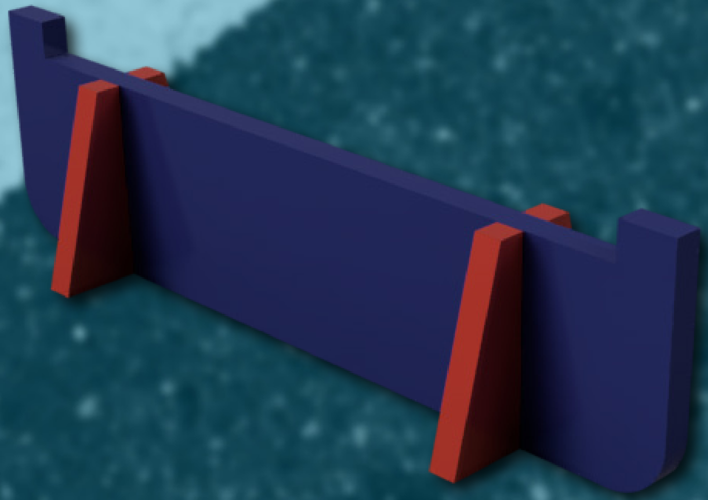
ATTENTION! Depending on the model, the micro:bit might not be included and has to be purchased separately.

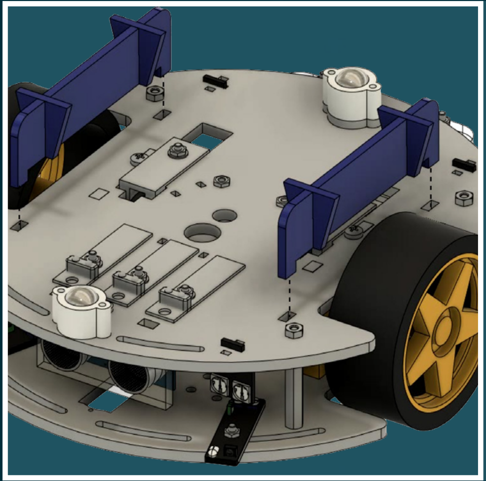




4. PARKING BRACKETS

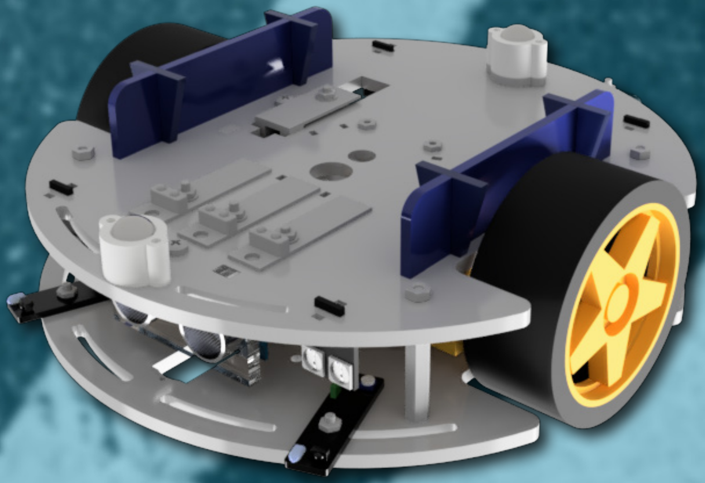
Take the two big parking brackets B and insert two of the parking bracket holders A in each. You can secure the construct additionally with some glue.





4. SET UP THE JOY-CAR

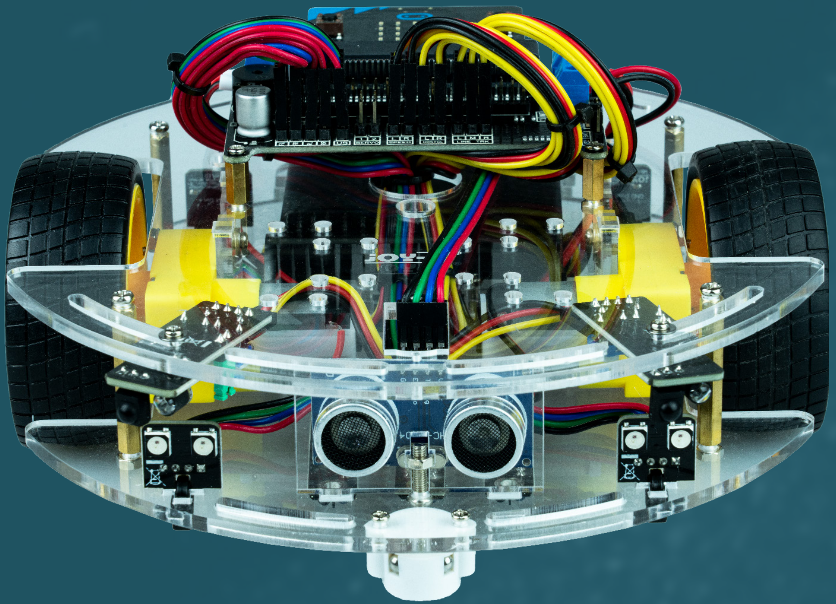
You can now set up the Joy-Car onto the parking brackets when you f. i. programm the motors, so the Joy-Car won‘t drive away when you try out your code.

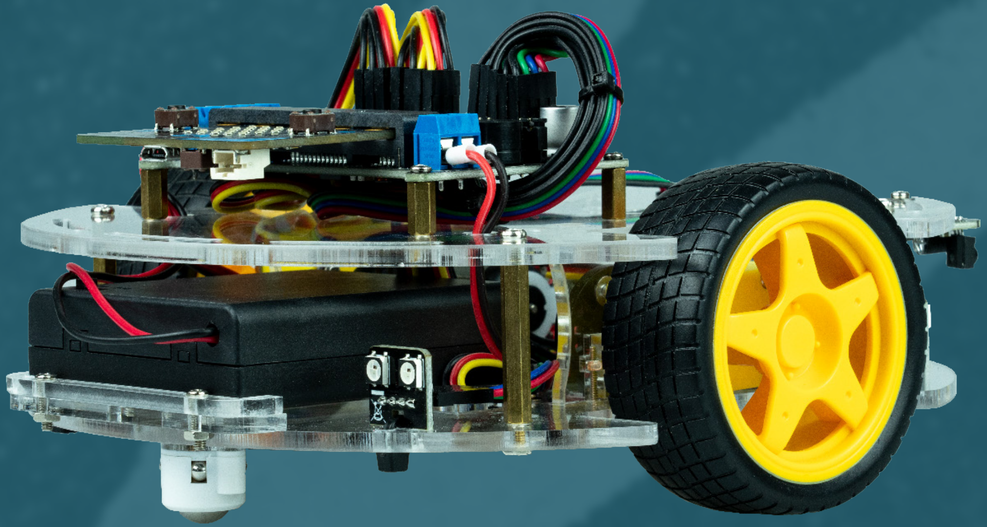


The assembly of your Joy-Car is now completed. You can now either proceed with the next chapter, where we explain the function of the single sensors step by step and how these can be utilised, or you spring ahead to the programming. Also for the programing, we have prepared the corresponing introductions and explanations for you in the next chapters.

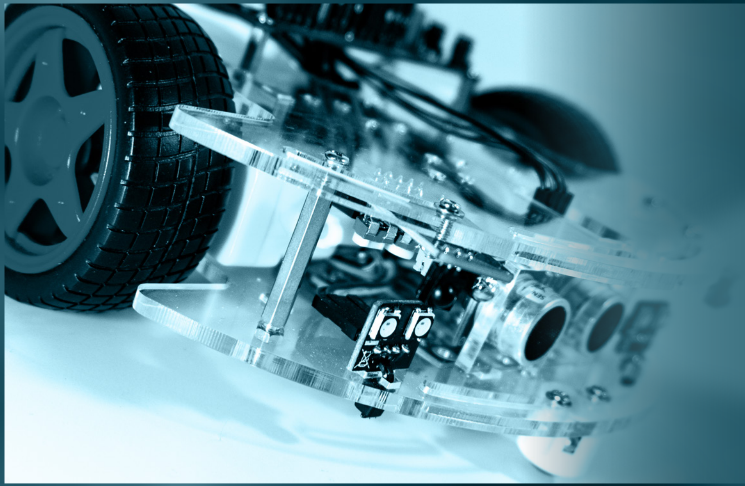
You are still unsure about the wiring of your joy car? You still don't know where to run the cables and somehow everything doesn't look right?

Have a look at our example wiring, how you can lead the cables best and at which places you can connect them with cable ties, so that the wiring of your joy car also makes a good impression.





TRAINING

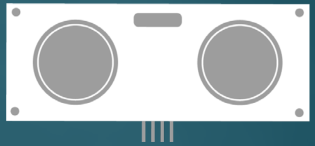


Your Joy-Car is assembled and freshly polished? Great! But you cannot start driving yet. In this chapter, we will go into detail and explain the modules, how they work and how they communicate with the Joy-Car.

This knowledge will help you in your own projects.

You don‘t want to wait and like to start up instantly? Simply skip this chapter and go ahead with the next one, where things get started!

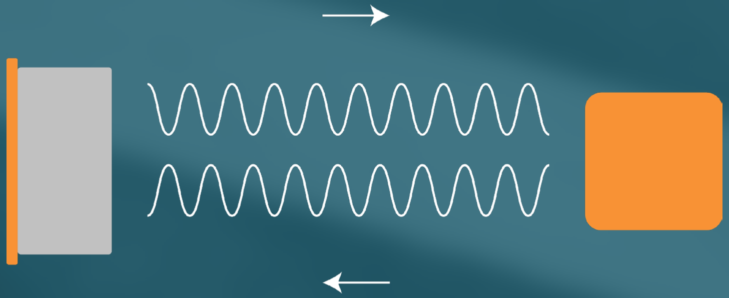
SENSORS



ULTRASONIC-SENSOR

The ultrasonic-sensor can be used with the Joy-Car for detection of objects and obstacles in a range between 2 and 300 cm. Therefore, it can avoid obstacles with a bigger distance or even can ride towards the objects.

The ultrasonic-sensor is capable of measuring distances precisely, as it sends out highly frequent acoustic pulses. When the pulse hits an object, the sound is reflected. The reflected echo is detected by the sensor. The distance to the object can be calculated by the time span between sending and receiving the ultrasonic pulse.



INFRARED-SENSORS

The obstacle-sensor, the linetracking-sensor and the speed-sensor all work with the same principle: infrared. The sensors use an infrared LED and an infrared receiver to detect the LED light.

OBSTACLE-SENSOR

The obstacle sensor can detect objects near the Joy-Car. For this purpose, the LED infrared light is radiated to the front. If an object / obstacle enters this light beam, the light is reflected and can be detected by the infrared receiver. The range of this sensor can be adjusted with the potentiometers. However, this sensor can only detect the distance preset by the potentiometers and not actively measure the distance to the nearest object, like the ultrasonic-sensor.

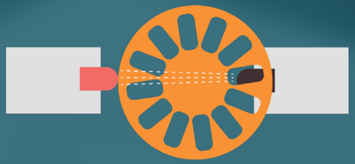
LINETRACKING-SENSOR

The line tracking sensor emits the infrared light downwards. If there is a light surface under the line-tracking sensor, the light is reflected and detected by the infrared receiver. However, if the infrared light is radiated at a black, non-reflecting surface, there is no infrared light reflected that the infrared receiver could detect. To follow a line, at least 2 sensors, but better 3 sensors, are needed. With the help of 3 sensors, you are then able to determine where the black line is and in which direction you have to steer to follow it.

SPEED-SENSOR

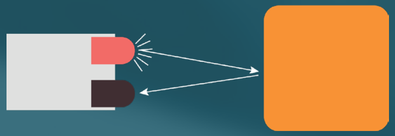
With the speed sensor, the infrared receiver and infrared LED are placed directly opposite to each other. Between receiver and LED is a perforated disc, which is mounted on the shaft of the motor. If the motor shaft rotates, the perforated disc also rotates. This repeatedly interrupts the infrared light of the LED. If you count the number of holes in the perforated disc (in this case 20), you know that the wheel has turned 1/20 per interruption. With this information, it is possible to determine the distance travelled and if you take the past time from interruption to interruption, you can determine the speed.

SPEED-SENSOR



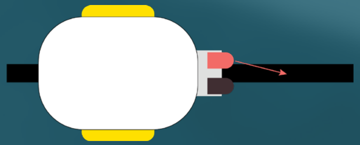
IR LIGHT REACHES THE SENSOR THROUGH THE HOLES OF THE PERFORATED DISK

OBSTACLE-SENSOR

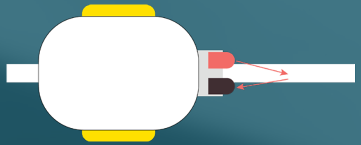


IR LIGHT IS REFLECTED BY A NEARBY OBJECT

LINETRACKING-SENSOR



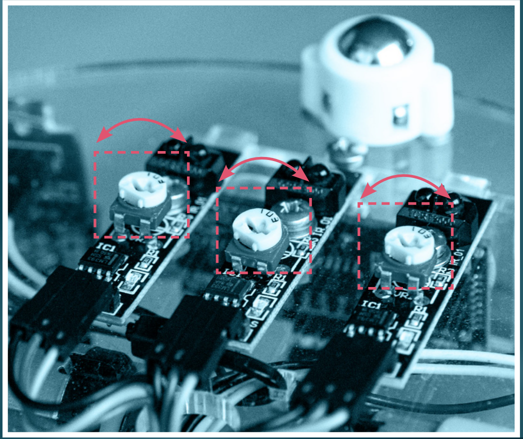
IR LIGHT IS NOT REFLECTED BY A DARK SURFACE, THE JOY-CAR CAN RIDE STRAIGHT AHEAD TO FOLLOW.



IR LIGHT IS REFLECTED BY A WHITE SURFACE AND IS DETECTED BY THE SENSOR. THE DIRECTION HAS TO BE CORRECTED TO FOLLOW.

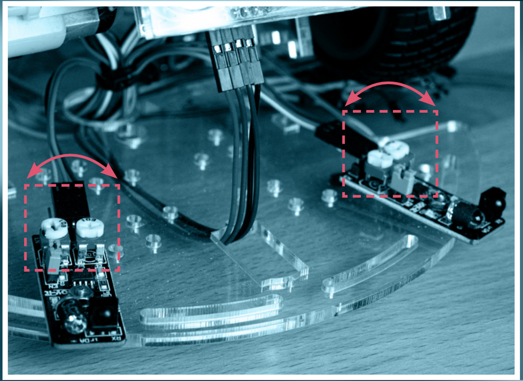
SENSITIVITY (LINETRACKER- & OBSTACLE-SENSORS)

The sensitivity of the sensors can be adjusted if they do not operate reliable. The line tracking sensors and the obstacle sensors are equipped with potentiometers that can be adjusted with a screwdriver to set the sensitivity. Each sensor has an additional LED that only lights up when the sensor is detecting something. It helps you to check the function of your sensor and to adjust it as well as possible.



LINETRACKING-SENSORS

Take a white sheet of paper and stick a strip of black adhesive tape on it. Now place your Joy-Car alternatingly on the sheet of paper and on the black tape. Your sensors are correctly adjusted when the LED on each sensor lights up, when the Joy-Car is on the sheet of paper and when the LEDs turn out again when you put the Joy-Car on the tape.



OBSTACLE-SENSORS

First remove the chassis from the base so that you can reach the potentiometers of the obstacle-sensors more easily. The obstacle sensors, unlike the linetracking-sensors, have two potentiometers each. Here you can adjust both the strength of the infrared LED and the sensitivity of the sensor. Hold an object in front of the sensors. The LED should light up and go out again when you remove the object.

ADVICE: You can increase the sensitivity by turning the potentiometer clockwise. Turning it counterclockwise reduces the sensitivity.

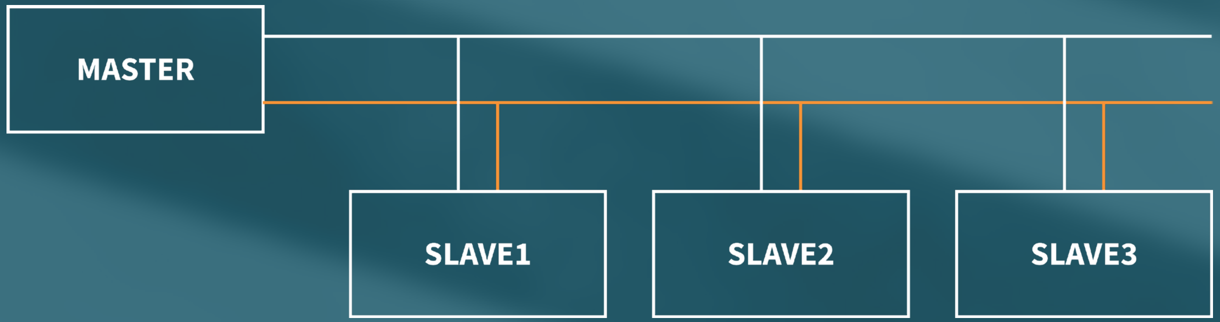
COMMUNICATION

I2C

The term I2C stands for Inter-Integrated Circuit, represents a serial data bus and describes how devices communicate with each other and exchange data. The data is transmitted via two lines, the SDA (Serial Data) and the SCL (Serial Clock) line. The SDA line is used to transmit the actual data. The SCL line only specifies the clock frequency and signals when a bit is present on the data line. On the I2C bus, all devices communicate via the so-called master/slave principle. Hereby, the entire communication is controlled by a single device, the micro:bit (master), and all other devices only wait for permission to transmit and are therefore called slaves.

I2C is used in the Joy-Car for communication and control of the infrared sensors (IO Expander) and the motor control unit (PWM controller).

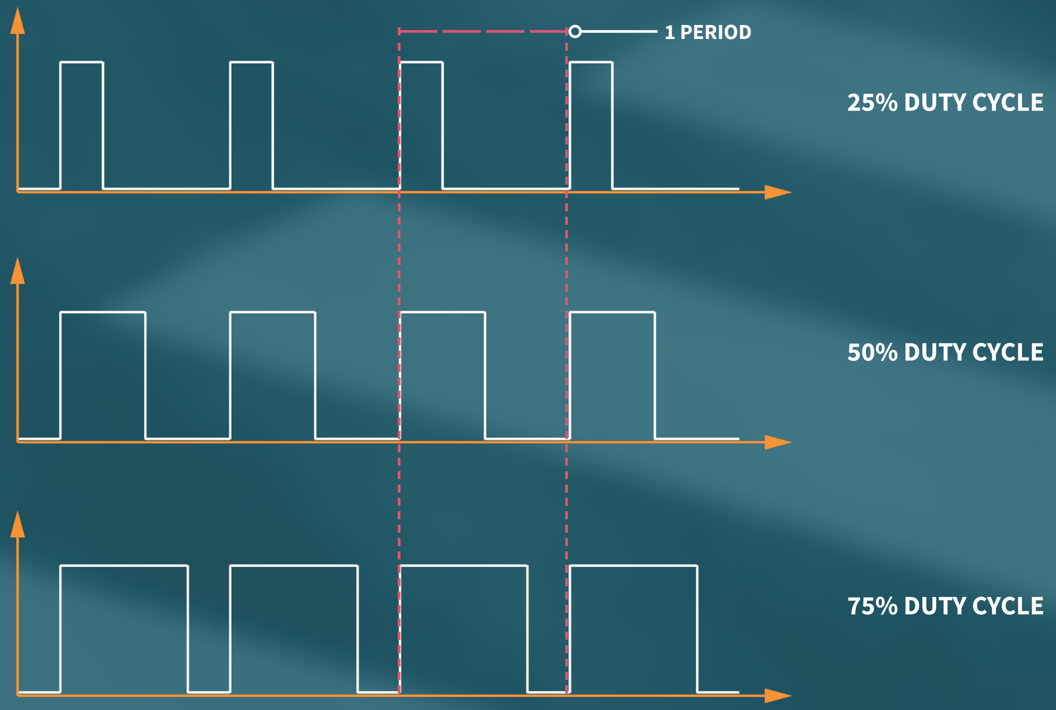
FOR EXPERTS: The addresses 0x70 (PWM controller) and 0x38 (IO expander) are used to control the systems via I2C.



PWM

PWM stands for „Pulse Width Modulation“. With this method, the ratio of the switch-on time to the defined period duration is varied. Pulse width modulation is used to control the speed or brightness of loads such as motors or LEDs.

The duration of a period is usually a few milliseconds or less. In practice, this means that the corresponding consumer is switched on and off several hundred times per second. The longer the switch-on time in a period, the more energy can be transferred to the consumer. In other words, the longer the duty cycle, the faster the motor turns or the brighter the LED lights up. In the following 3 examples you can see how PWM signals with 25%, 50% and 75% duty cycle differ from each other. At 0% duty cycle the consumer is off. Whereas at 100% duty cycle the motor runs at full speed and the LED is as bright as possible.



MOTORS

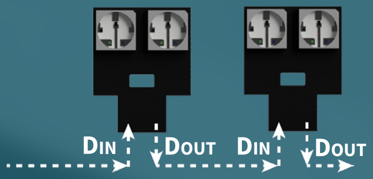
The motors are controlled by the built-in PWM controller. The speed can be regulated via the PWM signal (0-255), as well as the direction of rotation (forward & reverse) and the braking mode (strong & soft braking).

FOR EXPERTS: The PWM controller can be controlled by I2C via address 0x70. A total of 4 channels (2, 3, 4 & 5) are available for the two motors, which can be used as follows:

MOTOR RIGHT		MOTOR LEFT		
KANAL 2	**KANAL 3**	**KANAL 4**	**KANAL 5**	**FUNCTION**
0	PWM	0	PWM	Forward
0	0	0	PWM	Left
0	PWM	0	0	Right
PWM	0	PWM	0	Reverse
255	255	255	255	Soft Braking
0	0	0	0	Strong Braking

HEADLIGHTS

Addressable WS2812B RGB LEDs are used for the headlights. A controller is built into each of these LEDs, which makes it possible to define the colour and brightness of each LED individually. The data is transferred to the first LED via a bus line connected to a pin from the micro:bit. This bus line is then continued from the first LED to the second LED, from the second LED to the third LED and so on. The data is then transferred from LED to LED via this bus line. In other words, the LEDs form a kind of light chain, where each light can be controlled separately.

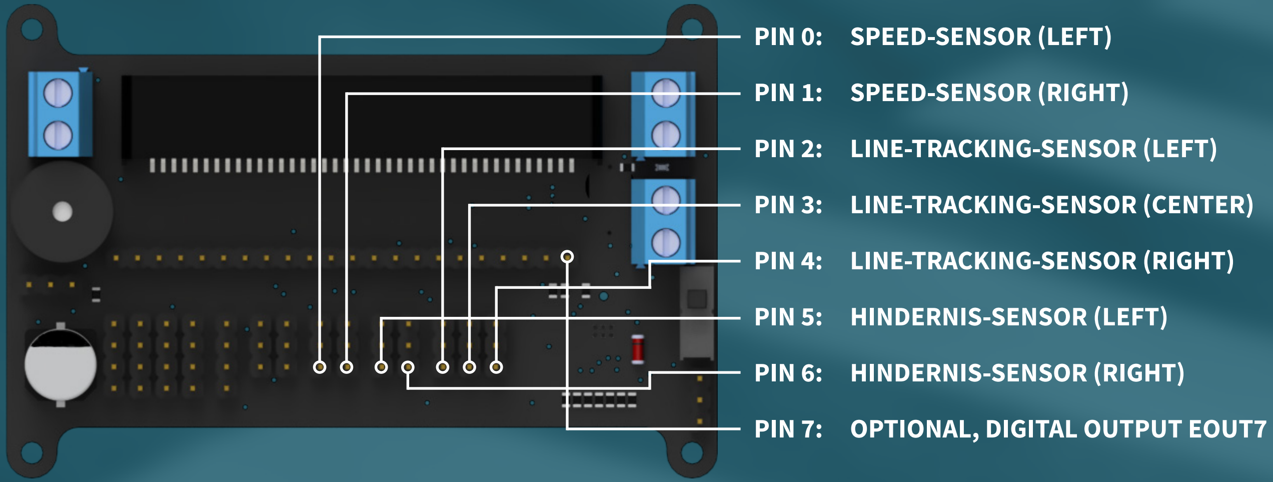


In the case of the Joy-Car, this LED chain has been solved in a way, that 2 WS2812B LEDs are mounted on one headlight module. These two LEDs are already connected to the bus line on the circuit board. The connection pins of the board have a „Din“ (data in) and a „Dout“ (data out) pin. These pins are used to connect the boards together. To keep the wiring clear, the LED boards are not directly connected to the bus line. The bus line is led back to the mainboard of the Joy-Car and is routed from the „Dout“ pin to the „Din“ pin of the following LED module.

ATTENTION! If a LED module is not connected, the modules further back in the chain no longer work, because the data connection is interrupted.

IO-EXPANDER

The IO-Expander is a central unit on your Joy-Car, to which most of the sensors are connected. Since the micro:bit does not have enough inputs for all sensors, these are connected to the IO expander. This then communicates with the micro:bit via the I2C interface. This way only two ports of the micro:bit are used. There is even an unused digital output on the IO expander available, which you can use for your own projects and sensors. The IO expander is structured as follows:


PIN 0: SPEED-SENSOR (LEFT)
PIN 1: SPEED-SENSOR (RIGHT)
PIN 2: LINE-TRACKING-SENSOR (LEFT)
PIN 3: LINE-TRACKING-SENSOR (CENTER)
PIN 4: LINE-TRACKING-SENSOR (RIGHT)
PIN 5: HINDERNIS-SENSOR (LEFT)
PIN 6: HINDERNIS-SENSOR (RIGHT)
PIN 7: OPTIONAL, DIGITAL OUTPUT EOUT7

FOR EXPERTS: The IO expander is addressed via the I2C address 0x38. It checks the sensors connected to it and returns the results summarized as byte. Each bit stands for one pin of the expander. In case of a corresponding detection the bit of the sensor is set to 1 (True).

EXAMPLE BYTE: 1 0 1 1 0 1 1 1

BIT 0 - Speed-Sensor L - True
BIT 1 - Speed-Sensor R - True
BIT 2 - Line-Tracking-Sensor L - True
BIT 3 - Line-Tracking-Sensor C - False
BIT 4 - Line-Tracking-Sensor R - True
BIT 5 - Hindernis-Sensor L - True
BIT 6 - Hindernis-Sensor R - False
BIT 7 - EOUT7 - True

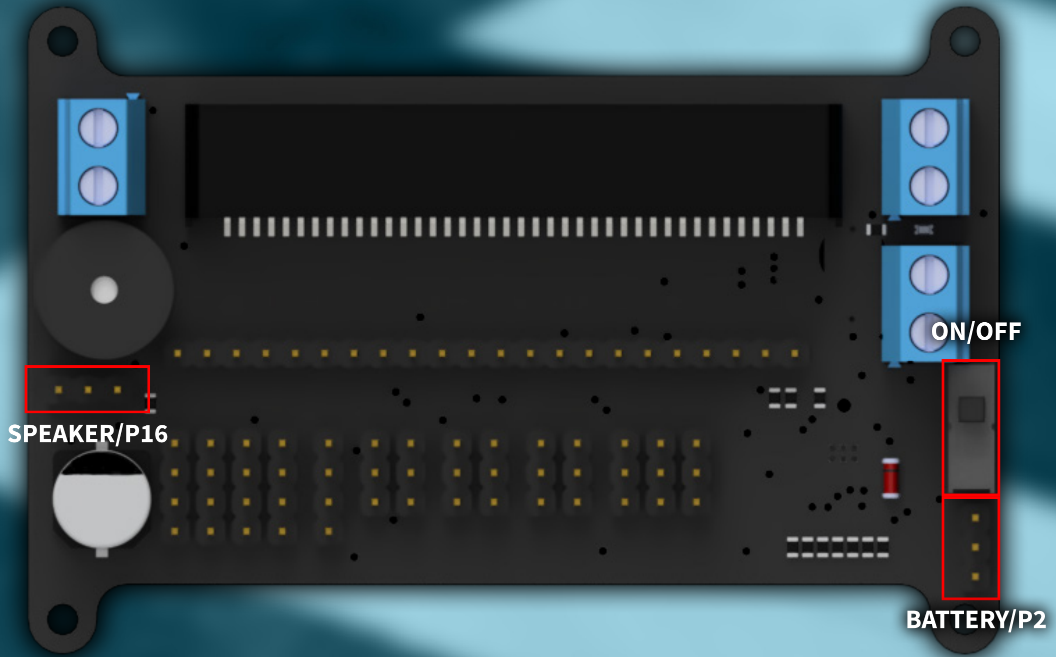
DETAILS

JUMPER

On the mainboard of your Joy-Car you can make further configurations with the jumpers. Here you can deactivate the **Speaker [SPK]** and the **battery voltage measurement [BAT]** and thus activate pins P16 and P2 on the pin header of the mainboard. This way you can activate two more pins for your own development, if you need them.

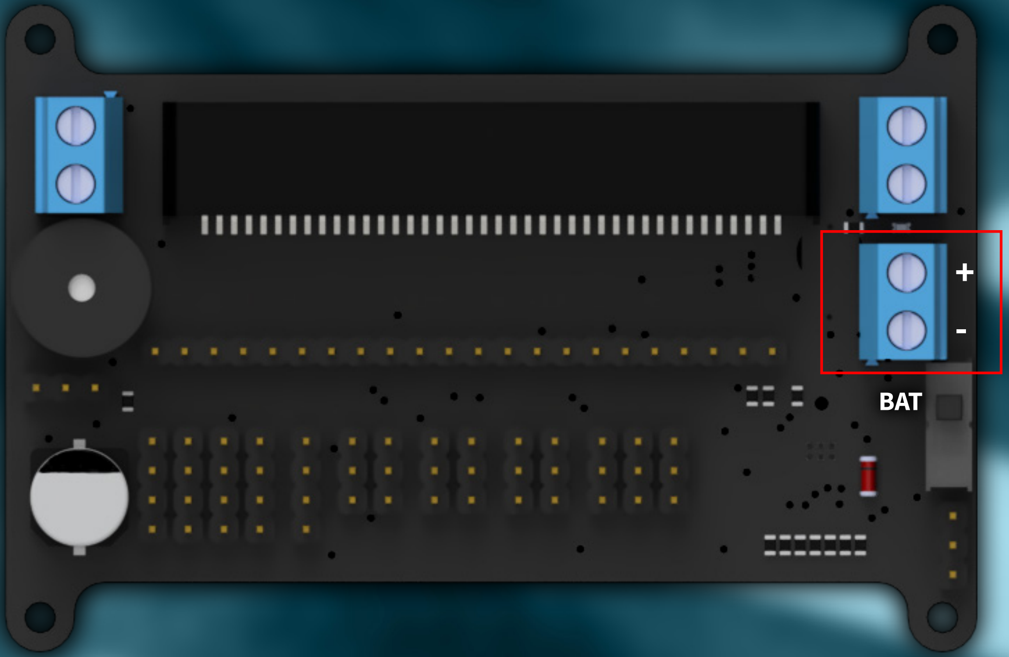
ON/OFF

To switch your Joy-Car on or off, you don‘t have to insert or remove the batteries again and again. You can easily disconnect the power supply via the on/off switch.



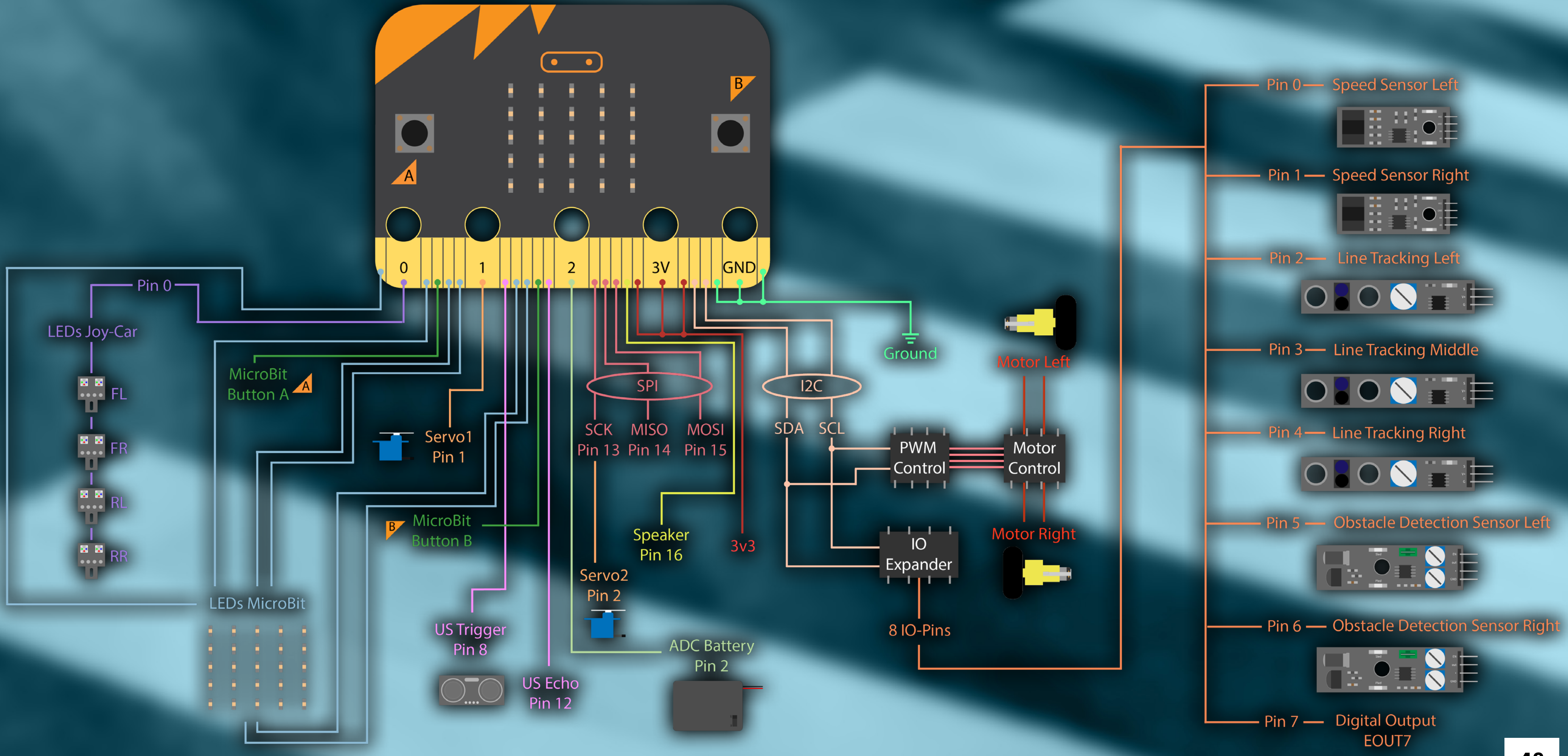
POWER SUPPLY

In the assembly instructions, you have already learned that you can connect the battery holder to the BAT terminal. However, if you should make your own modifications, you are not bound to the battery holder. Here it is good to know: You can connect any voltage source between 4.5-9V to the BAT terminal.



EVERYTHING IN THE RIGHT PLACE

The mainboard of the Joy-Car is of course only the connecting unit between the individual sensors and modules and the micro:bit. You want to know where and how the individual units are connected to the micro:bit? Or maybe you want to make changes yourself? On our schematic diagram we have summarized all units and show you how they are controlled by the micro:bit.



AGONY OF CHOICE

MAKECODE OR MICROPYTHON?

MAKECODE OR MICROPHYTON? OH WELL, WHAT THEN ...?

You can program your Joy-Car with MakeCode as well as with MicroPython. You ask yourself, which of these is now right? We'll help you!

MakeCode and „Mu for MicroPython" are both development environments. In principle, there is no right or wrong here. With both variants you can use all functions of your Joy-Car and also create your own applications on the Joy- Program Car. The two variants differ rather in their target group:

MakeCode is mainly aimed at beginners. You have little or no programming experience? Then you are at MakeCode right. Here, all functions can be pushed together via graphic blocks, which are then called by the development environment for the micro:bit. At the same time more advanced users can use the environment to JavaScript or Python and continue programming from there. But in principle: **You do not have to write a single line of code in MakeCode if you don't want to.**

MicroPython is rather aimed at advanced users. You already know how to program and can realize your own projects? Then MicroPython is your best choice. MicroPython is an implementation of the Python programming language and is optimized for microcontrollers. Even though the Mu development environment is also designed for beginners, it is still recommended to already have some basic knowledge here.

AND THAT DRIVES?!

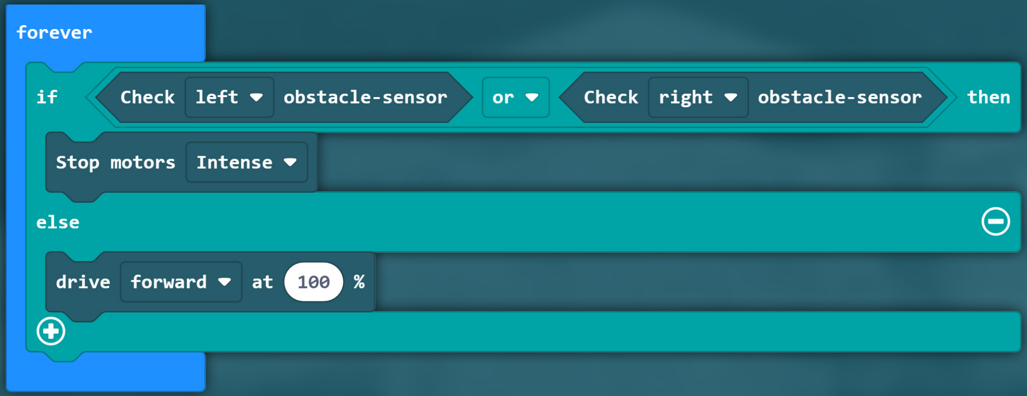
Now comes the part where your Joy-Car is brought to life, the programming. We designed the Joy-Car to replicate some of the functions of a real car. Aside from driving, you can reproduce various lighting elements of a car (e.g. low beam, high beam, brake lights, indicators, etc.). Furthermore the Joy-Car with the buzzer has a (multi-tone) horn on board. But also sensors, such as the ultrasonic-sensor in today's cars. You don't want to copy a car but program a colorful flashing fun-mobile? Don't worry, this is also possible. On the following pages you will find explanations and examples how to program your Joy-Car.

MAKECODE

MAKECODE?

You have only little programming experience so far? Then MakeCode is the perfect start to get familiar with the Joy-Car. With MakeCode you can assemble colored blocks that represent the functionalities of the Joy-Car without writing a single line of code. And yet, this system introduces you to programming and prepares you for practical programming.

With MakeCode, for example, it can be child's play to make the Joy-Car drive and stop it as soon as an obstacle is detected:

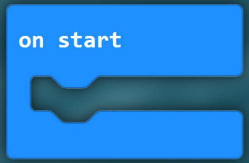


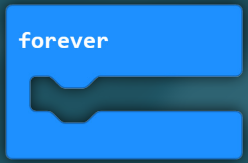
```
forever
  if Check left obstacle-sensor or Check right obstacle-sensor then
    Stop motors Intense
  else
    drive forward at 100 %
```

You already know your way around? You are not in the mood for explanations? You prefer to start driving straight away instead of developing your Joy-Car yourself? **HERE** you can directly find out how you can start immediately with our programming developed for you.

A NEW START

At the beginning, each project consists of two basic blocks: the **„on start“-block** and the **„forever“-block**. All instructions within these two blocks are executed by the program. However, the **„on start“-block** is only executed once when you start your program. The **„forever“-block**, on the other hand, is executed over and over again. As soon as all commands and instructions, from top to bottom, have been processed in this block, the execution starts again from the beginning. Here you can for example permanently check the sensors and react to new events.

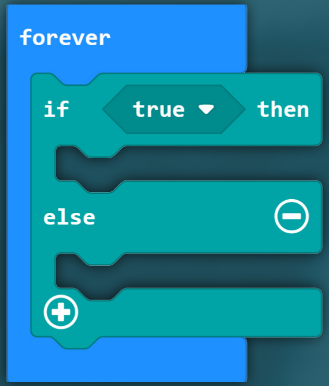




WITH FUSS OR QUIBBLE...

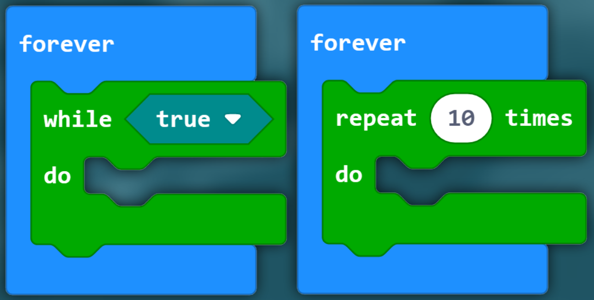
In principle, the JavaScript programming language is represented visually by the blocks. This allows you to use queries and dependencies that you may have already gotten to know. For example, you can check if a certain condition is fulfilled. There you can also place additional statements that are only executed if the previous condition was fulfilled: **if <condition> then...**

Optionally, you can also add an alternative to this block, which will be executed if the previous condition has **not** been fulfilled. This is the **„else“-statement**.



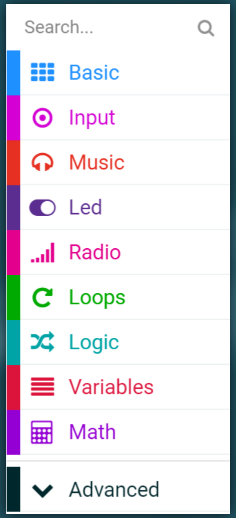
OVER AND OVER AGAIN...

Even loops can be easily mapped with the blocks. Here you can execute something until the condition is no longer met (**„during“-block**) or repeat something based on a certain number (**„repeat x times“-block**).



```
forever
  while true
  do

forever
  repeat 10 times
  do
```

ON THE LOOKOUT!

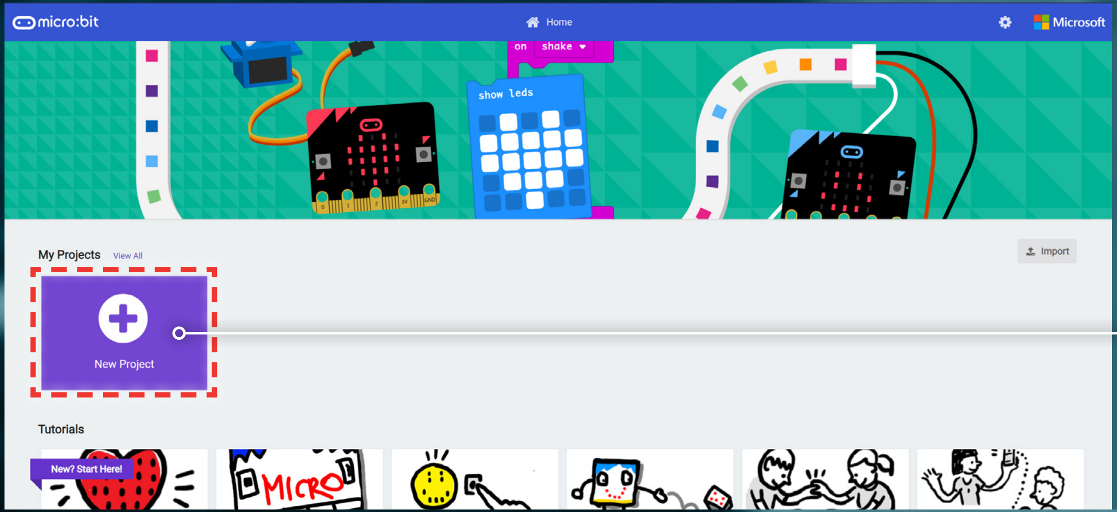


In the left area of your project window you will find the **block overview**, with all blocks that are available to you. They are arranged according to categories and can be opened. They give you a great overview of all your options. Just click through the categories and find out for yourself which paths you can take.

THE FIRST START

A NEW CHAPTER

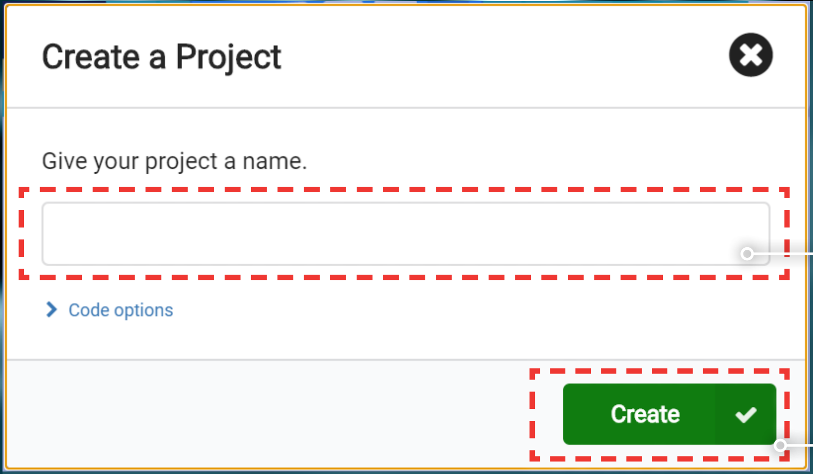
You have no experience with the MakeCode development environment yet, but would like to start with a small example? We'll get you the lay of the land! On **HTTPS://MAKECODE.MICROBIT.ORG/** you will get to the development environment.



Here we go, as soon as you enter the site. In the middle area of the page you will find a list of all your projects. You can also create a new project here.

Try it out and create your first project!

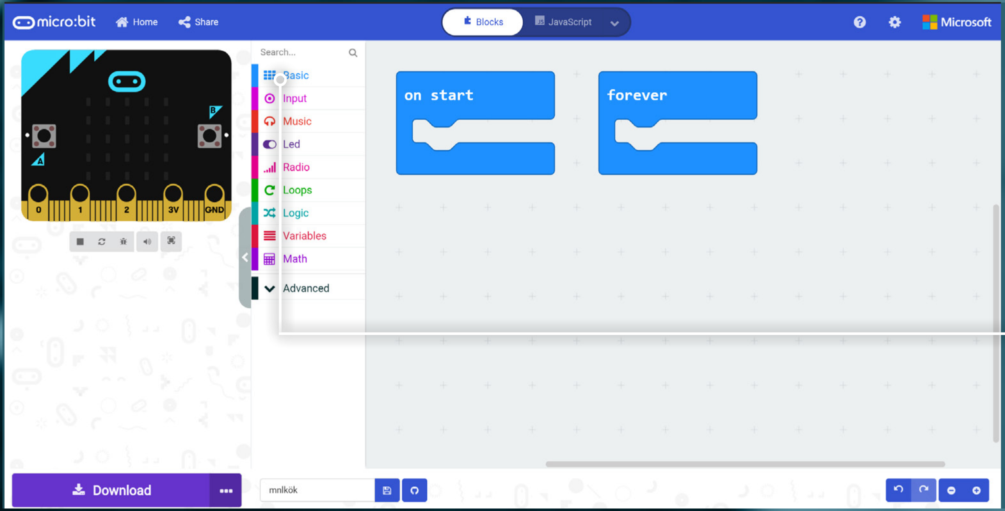
CREATE YOUR FIRST PROJECT HERE



All you have to do is give your new project a name and you‘re good to go.

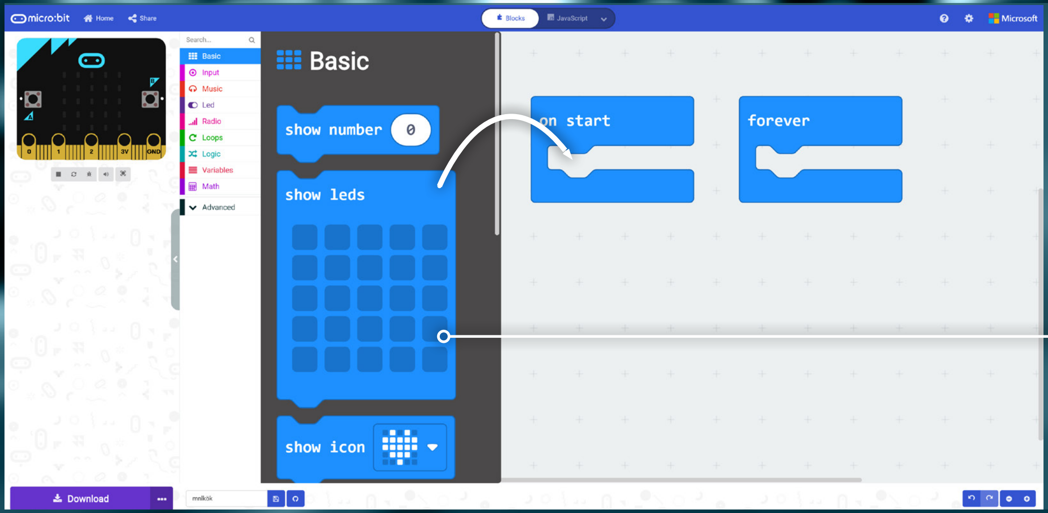
GIVE A NAME TO YOUR PROJECT HERE

THEN CONFIRM YOUR ENTRY



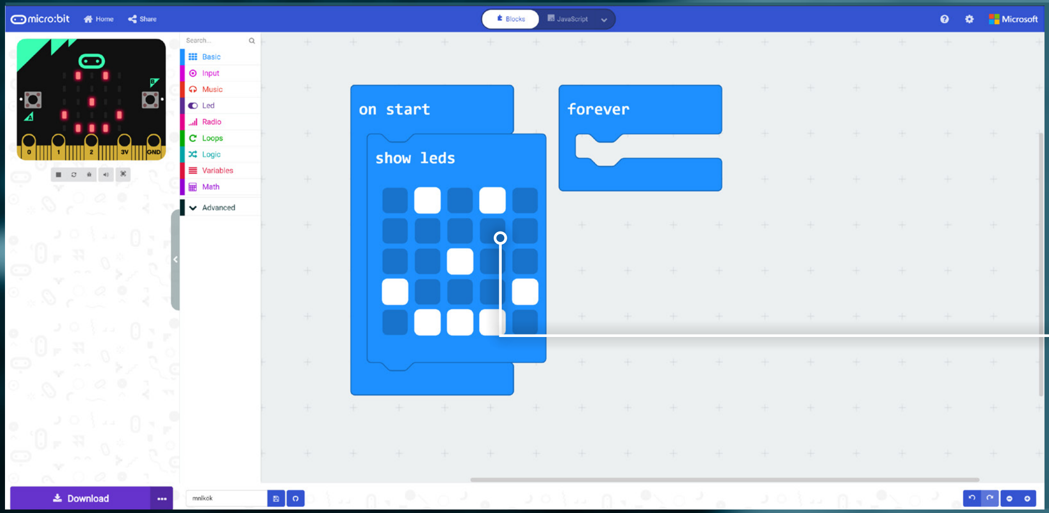
Next you start directly in the development environment and you can put together your first blocks. For your first project we will now create a small sample project together.

OPEN THE BASIC AREA IN THE BLOCK OVERVIEW



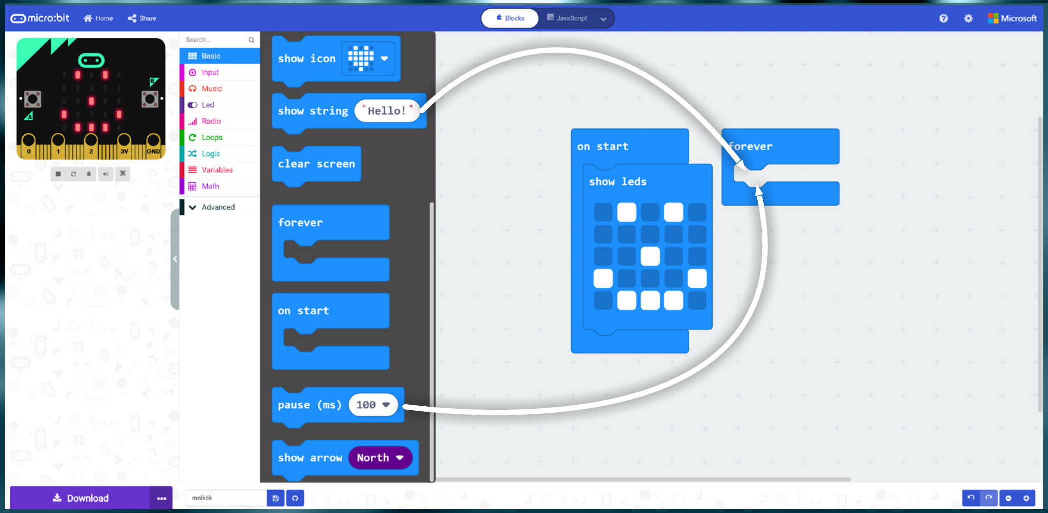
You will see that the block overview has enlarged and the category you just clicked on has opened.

TAKE THE „SHOW LEDS"-BLOCK AND SIMPLY DRAG IT WITH YOUR MOUSE INTO THE „AT START"-EXECUTION-BLOCK



You can click the individual boxes and set which of the LEDs you want to be activated.

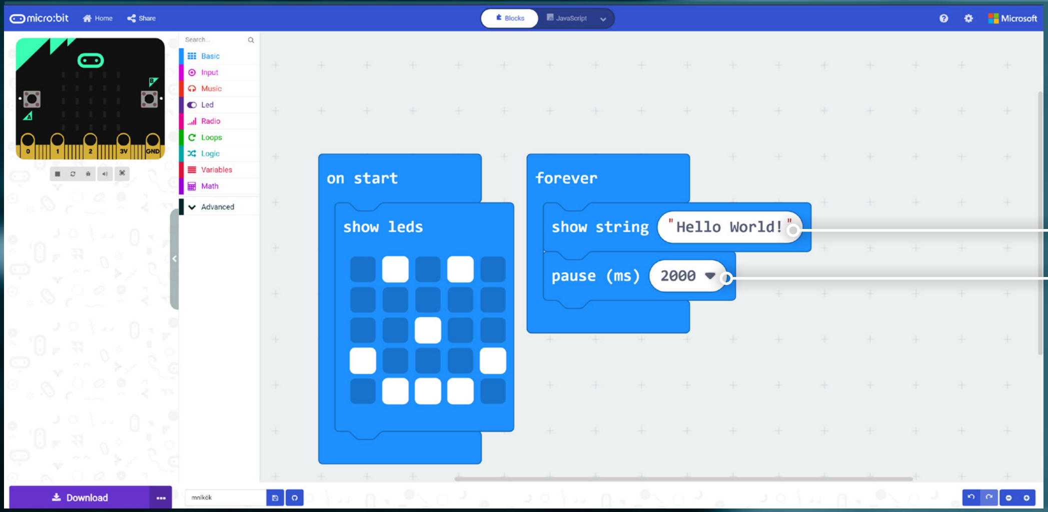
CLICK ON THE LEDS YOU WANT TO ACTIVATE



```
show icon
show string "Hello!"
clear screen
forever
on start
pause (ms) 100
show arrow North

on start
  show leds
forever
```

But that was only the part that is executed once at startup. Now drag the „show text“ block and a „pause“ block from the Basics category into your „permanent“ execution block.

TAKE THE „SHOW LEDS“-BLOCK AND DRAG IT WITH YOUR MOUSE INTO YOUR „AT START“-EXECUTION-BLOCK

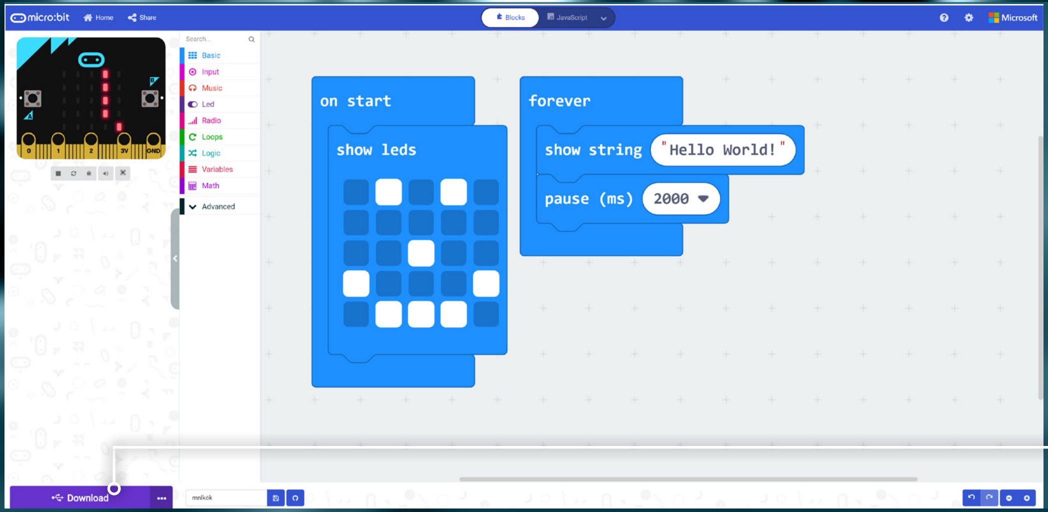


```
on start
  show leds

forever
  show string "Hello World!"
  pause (ms) 2000
```

You can also change the text and also the duration of the pause by clicking on the white fields.

ENTER YOUR INDIVIDUAL TEXT HERE

SET THE DURATION OF THE PAUSE HERE

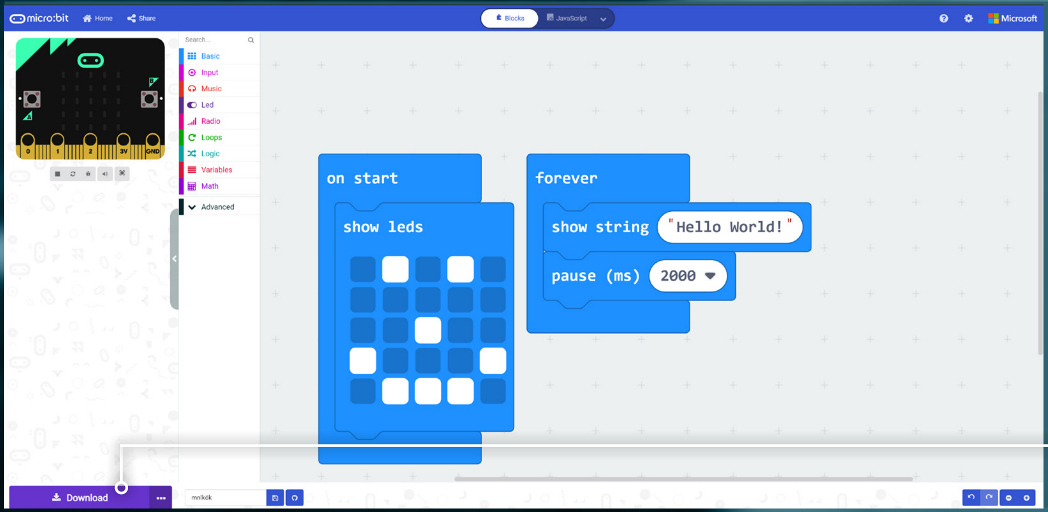


```
on start
    show leds
forever
    show string "Hello World!"
    pause (ms) 2000
```

Your first sample code is ready and can now be transferred to your micro:bit.

First connect your micro:bit to your computer. In most cases, your micro:bit should be automatically recognized and paired, so your code can be conveniently downloaded directly to your device.

CLICK HERE FOR AUTOMATIC TRANSFER. THE USB-SYMBOL NEXT TO „DOWNLOAD“ MEANS THAT YOUR MICRO:BIT HAS BEEN RECOGNIZED AND PAIRED AUTOMATICALLY.

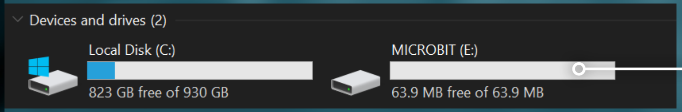


```
on start
    show leds
forever
    show string "Hello World!"
    pause (ms) 2000
```

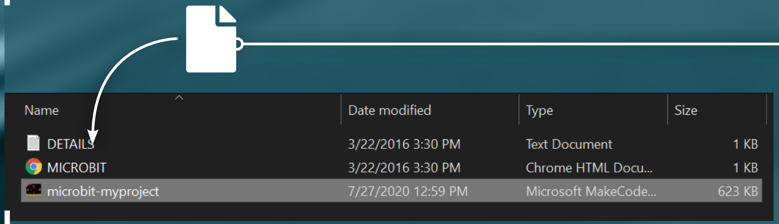
Unfortunately it can occur that the browser does not recognizes the micro:bit, so the automatic coupling fails. In this case you can download your programming as a file.

CLICK HERE FOR MANUAL DOWNLOAD. THE DOWNLOAD SYMBOL NEXT TO "DOWNLOAD" MEANS THAT THIS BUTTON ACTIVATES THE DOWNLOAD.

Even though the browser may not have recognized your micro:bit, it should still have been recognized as a drive in your Windows Explorer.



HERE YOU CAN OPEN YOUR MICRO:BIT AS DEVICE



Name	Date modified	Type	Size
DETAILS	3/22/2016 3:30 PM	Text Document	1 KB
MICROBIT	3/22/2016 3:30 PM	Chrome HTML Docu...	1 KB
microbit-myproject	7/27/2020 12:59 PM	Microsoft MakeCode...	623 KB

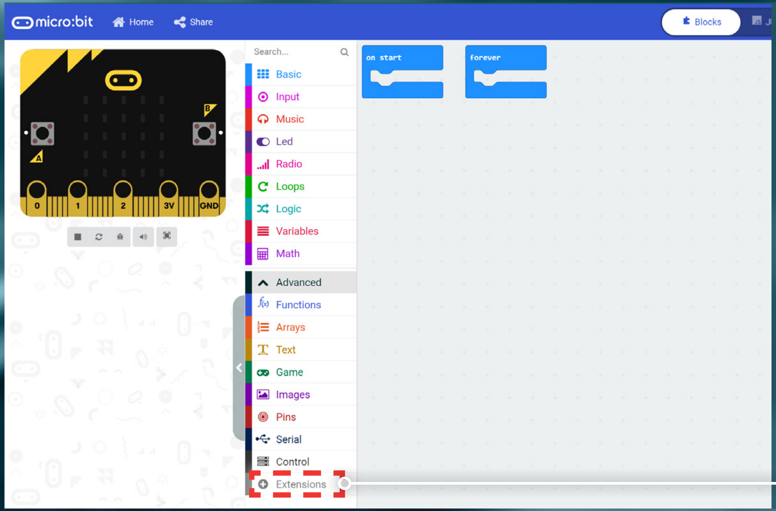
Then copy your programming that you have downloaded before into the disk folder of your micro:bit. Your code will then install itself automatically.

COPY THE FILE TO YOUR MICRO:BIT

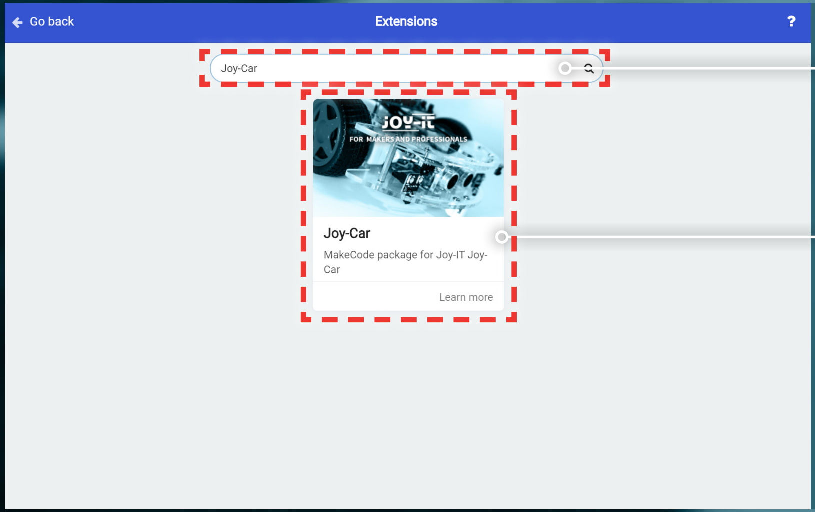
The automatic installation takes only a few seconds. After that the code will always be executed automatically whenever you power up your micro:bit.

THE JOY-CAR EXTENSION

We put together all functions of the Joy-Car in a separate extension for you. To use it for your project, first open the tab **Advanced** in your block overview and click on **Extensions**. Now search for Joy-Car and click on our extension. It will then automatically be added to your project.

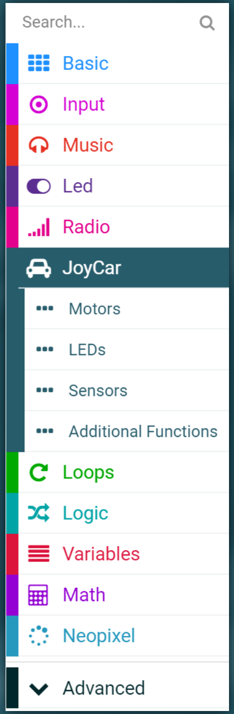


OPEN EXTENSION MENU



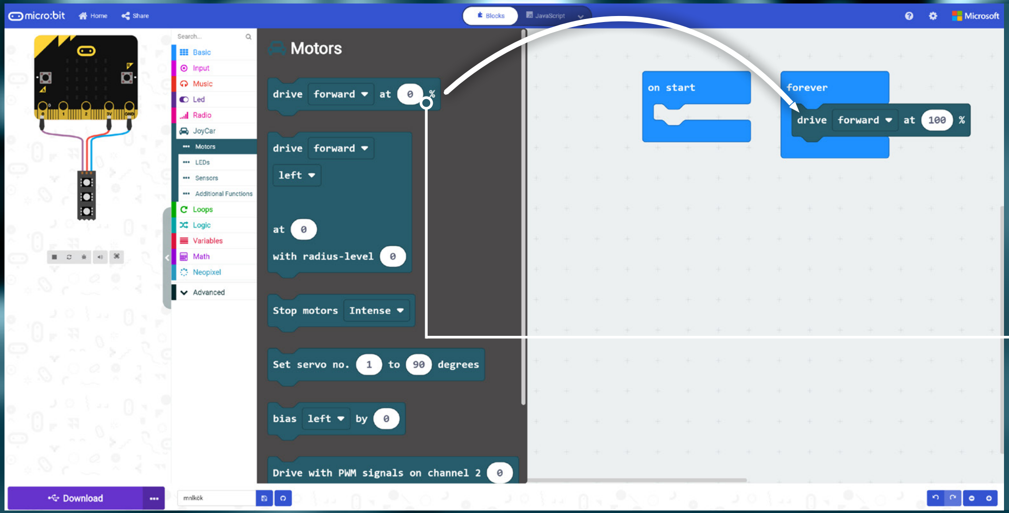
ENTER "JOY-CAR" IN THE SEARCH FIELD, CONFIRM WITH ENTER

CLICK ON THE EXTENSION, THE INSTALLATION IS DONE AUTOMATICALLY



ALL IN ONE PLACE

After you have added the Joy-Car extension to your project via the extension menu, you will find the Joy-Car tab in your block overview. Here all the functions of the Joy-Car are combined, so you can start right away. The functions are categorized by **motors, lighting, sensors and other functions**. The individual functions should be easy to understand. Nevertheless, we will go into more detail for you on the following pages.



Motors

```
drive forward ▾ at 0 %

drive forward ▾
left ▾
at 0
with radius-level 0

Stop motors Intense ▾

Set servo no. 1 to 90 degrees

bias left ▾ by 0

Drive with PWM signals on channel 2 0
```

```
on start

forever
    drive forward ▾ at 100 %
```

After installing the extension, you can simply drag any block from the overview into your program.

For example, you can let your Joy-Car move forward with only one single block.

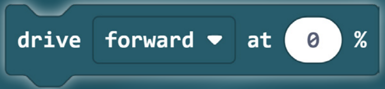
SIMPLY DRAG YOUR DESIRED BLOCK INTO YOUR APPLICATION

MY BLOCK

The blocks of the Joy-Car extension give you access to all functions of your Joy-Car. So you can customize your blocks and your entire programming. On the following pages we have provided you with every single block and explains its function.

MOTORS

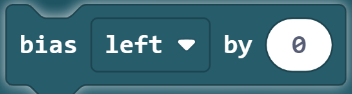
The motors are the drive of the Joy-Car. You can move the Joy-Car back and forth, drive at different speeds, turn and brake. The two servo motors can also be controlled in this category.



```
drive  forward ▾  at  0  %
```

DRIVE

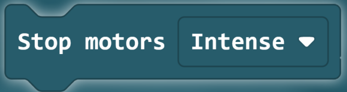
Drive forwards or backwards. You can also choose the speed in percent between 0 (no drive) and 100 (maximum drive).



```
bias  left ▾  by  0
```

MOTOR DELAY

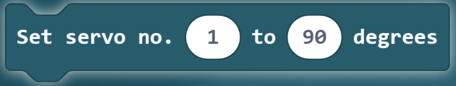
Due to manufacturing tolerances, it can happen that the two motors don't turn at exactly the same speed. Hereby you can set a permanent deceleration of an engine in percent.



```
Stop motors  Intense ▾
```

BRAKE

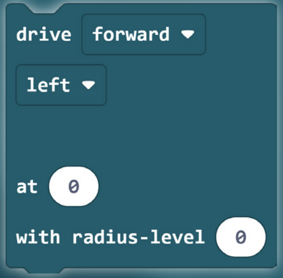
Brake the Joy-Car to a standstill. You can additionally choose between a hard emergency stop or a soft brake, where the Joy-Car is slowly rolling out.



```
Set servo no.  1  to  90  degrees
```

SERVO MOTORS

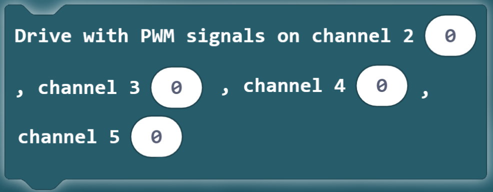
The two optional servomotors can be controlled and adjusted in an angle between 0 and 180 degrees.



```
drive  forward ▾
left ▾

at  0
with radius-level  0
```

CURVES

Take a left or right turn. Here you can also set the speed in percent. Additionally you can set the radius level of the curve from 0 - 5 (0 = tight curve, 5 = wide curve).



```
Drive with PWM signals on channel 2  0
, channel 3  0  , channel 4  0  ,
channel 5  0
```

PWM-SIGNAL

The motors can also be controlled directly via PWM signals. For this purpose a PWM value between 0 and 255 can be sent to each channel.

HEADLIGHTS

The four LED modules, the headlights of the Joy-Car, can be controlled in this category. So you can control the front head-lights, activate turn signals and brake lights and find even more functions here.

```
Toggle light  on ▾
```

HEADLIGHTS

The headlights are controlled here. A white light is activated in the front and a red light in the back.

```
Turn  on ▾  left ▾  indicator
```

INDICATOR

Set the indicator for one side (left/right) of the Joy-Car. The front and rear headlights on the selected side will start flashing yellow.

```
Turn  on ▾  hazard lights
```

HAZARDLIGHTS

This is to control the hazardlights. All the head-lights will start flashing yellow.

```
Toggle brakelight  on ▾
```

BRAKELIGHTS

With this, you control a brighter red light on the rear headlights of your Joy-Car.

```
Toggle reverse light  on ▾
```

BACKLIGHT

Use this to control the reversing light. This acti-vates a white light on the rear headlights.

SENSORS

The sensors on your Joy-Car allow you to react to certain events. Obstacles, lines, markings and speed? The sensors on your Joy-Car can detect that.

```
Check left ▾ linefinder-sensor
```

LINEFINDER-SENSOR

Checks the left/center/right linefinder sensor whether a line on the floor could be detected. The function returns **True** or **False** in response.

```
Check left ▾ speed-sensor
```

SPEED-SENSORS

Checks the left/right speed-sensor whether the signal has been interrupted by the perforated disc. The function returns **True** or **False** in response.

```
Check left ▾ obstacle-sensor
```

OBSTACLE-SENSORS

Checks the left/right obstacle-sensor to see if an obstacle has been detected. The function returns **True** or **False** in response.

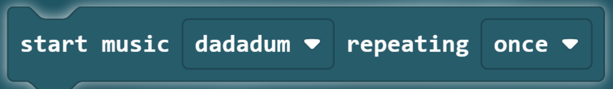
```
Check ultrasonic-sensor
```

ULTRASONIC-SENSOR

Checks the ultrasonic-sensor for the distance to the nearest detected object. The function returns the distance in response.

ADDITIONAL FUNCTIONS

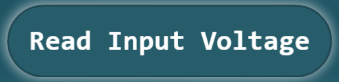
Here you can find more functions of the Joy-Car, which go beyond the previous motor funtions, sensor queries and light settings.



```
start music dadadum ▾ repeating once ▾
```

BUZZER

Play a predefined melody with the Buzzer. You can additionally choose, whether the melody should be played only once or be repeated permanently.



```
Read Input Voltage
```

BATTERY VOLTAGE

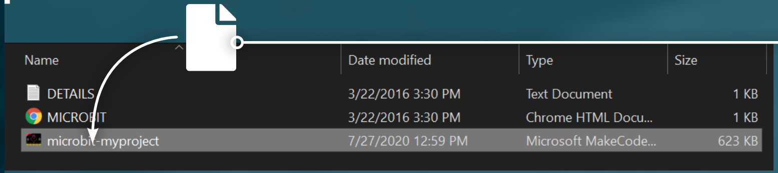
The battery voltage can be queried over the analog to digital converter pin of the micro:bit. This way, for example, the battery capacity can be measured.

ALL THE WAY

MAXIMUM FUN, MINIMUM EFFORT?

You prefer to drive right away? You can also, instead of developing your Joy-Car yourself, use our prepared script on your micro:bit. Here the most important functions are already arranged in an application with three different modes.

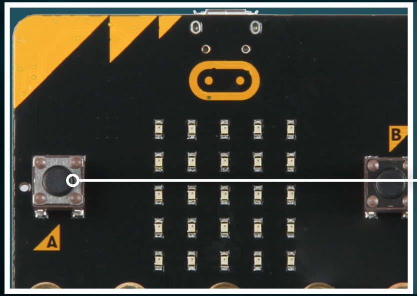
Just download the MakeCode sample file from our **Joy-Car-Website**.



Name	Date modified	Type	Size
DETAILS	3/22/2016 3:30 PM	Text Document	1 KB
MICROBIT	3/22/2016 3:30 PM	Chrome HTML Docu...	1 KB
microbit-myproject	7/27/2020 12:59 PM	Microsoft MakeCode...	623 KB

Then copy your programming, which you downloaded before, into the folder of your micro:bit. Your code will be installed automatically.

COPY THE FILE TO YOUR MICRO:BIT



The different modes can then simply be switched with the **button A** on your micro:bit.

PRESS BUTTON A TO CHANGE THE MODE

MICROPYTHON

MICROPYTHON?

MicroPython is an implementation based on the Python 3 language. It was written in the C programming language and is optimized for use on microcontrollers such as the micro:bit. Users who are already familiar with the basics of software programming can start directly with this variant.

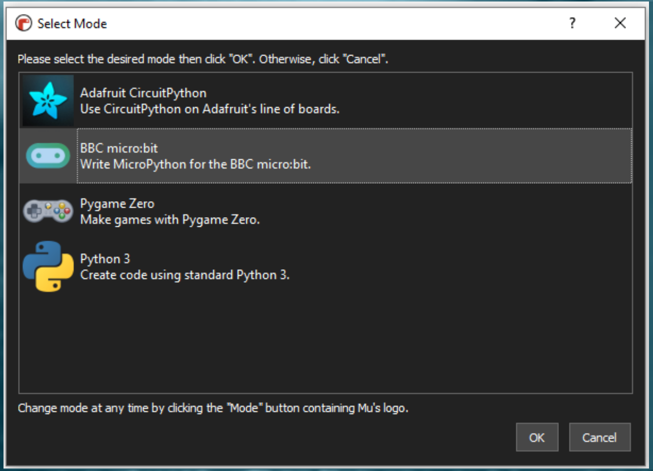
WITHOUT PRIOR EXPERIENCE IN PROGRAMMING, IT IS RECOMMENDED THAT YOU FIRST USE THE MAKE:CODE VARIANT FROM THE PREVIOUS CHAPTER.

ENTWICKLUNGSUMGEBUNG

MicroPython can be compiled with the Mu Editor. This editor is mainly aimed at beginners and is therefore especially easy to use.

The editor can be downloaded here: **https://codewith.mu/en/downloads**

INTERFACE


Select Mode

Please select the desired mode then click "OK". Otherwise, click "Cancel".

Adafruit CircuitPython
Use CircuitPython on Adafruit's line of boards.

BBC micro:bit
Write MicroPython for the BBC micro:bit.

Pygame Zero
Make games with Pygame Zero.

Python 3
Create code using standard Python 3.

Change mode at any time by clicking the "Mode" button containing Mu's logo.

SETUP

When starting the Mu Editor for the first time, it is first necessary to select the desired mode. Select BBC MICRO:BIT here and confirm the choice with OK.

The detailed English MicroPython documentation can offer additional help and can be found **here**.

The most important options and possibilities of the editor interface are explained in the next steps.

REPL

REPL stands for READ-EVAL-PRINT-LOOP and represents the console in the Mu-Editor. Here, outputs can be displayed during code execution and custom commands can be sent to the micro:bit.

In addition, key combinations can be used for control within the REPL:

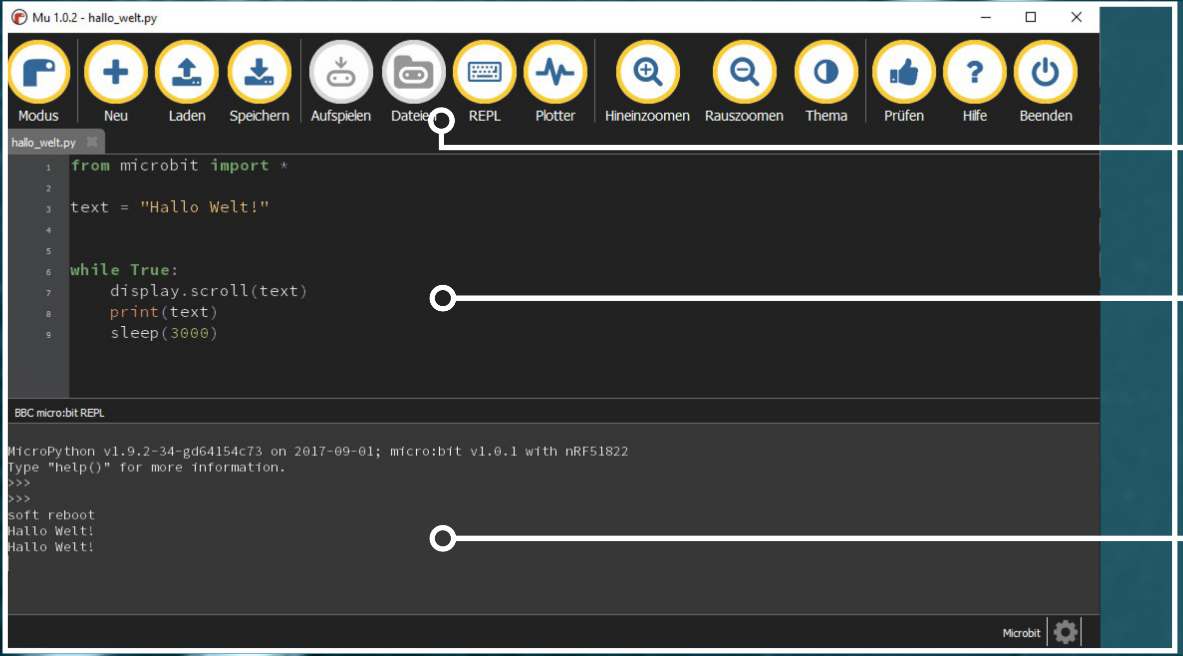
STRG + D:	Soft reboot
STRG + C:	Stop code execution

CHECK

Hereby, the written source code can be checked. Mistakes are automatically detected and displayed accordingly.

FLASH

The written source code is checked, compiled for the micro:bit and then transferred to the device.



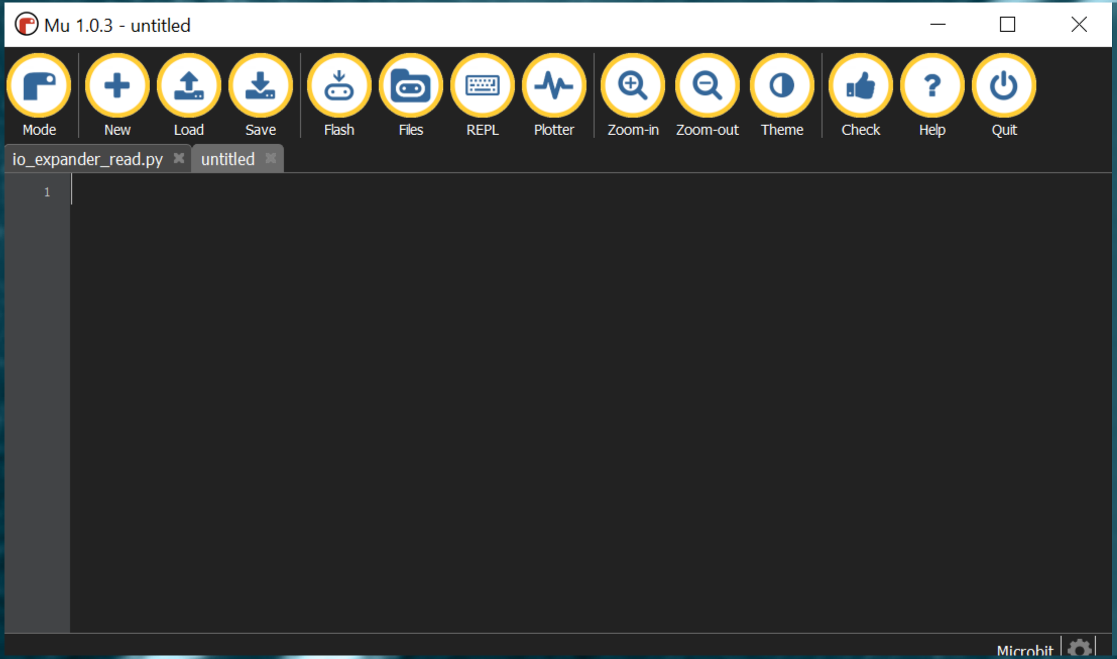
```
from microbit import *

text = "Hallo Welt!"


while True:
    display.scroll(text)
    print(text)
    sleep(3000)
```

```
MicroPython v1.9.2-34-gd64154c73 on 2017-09-01; micro:bit v1.0.1 with nRF51822
Type "help()" for more information.
>>>
>>>
soft reboot
Hallo Welt!
Hallo Welt!
```

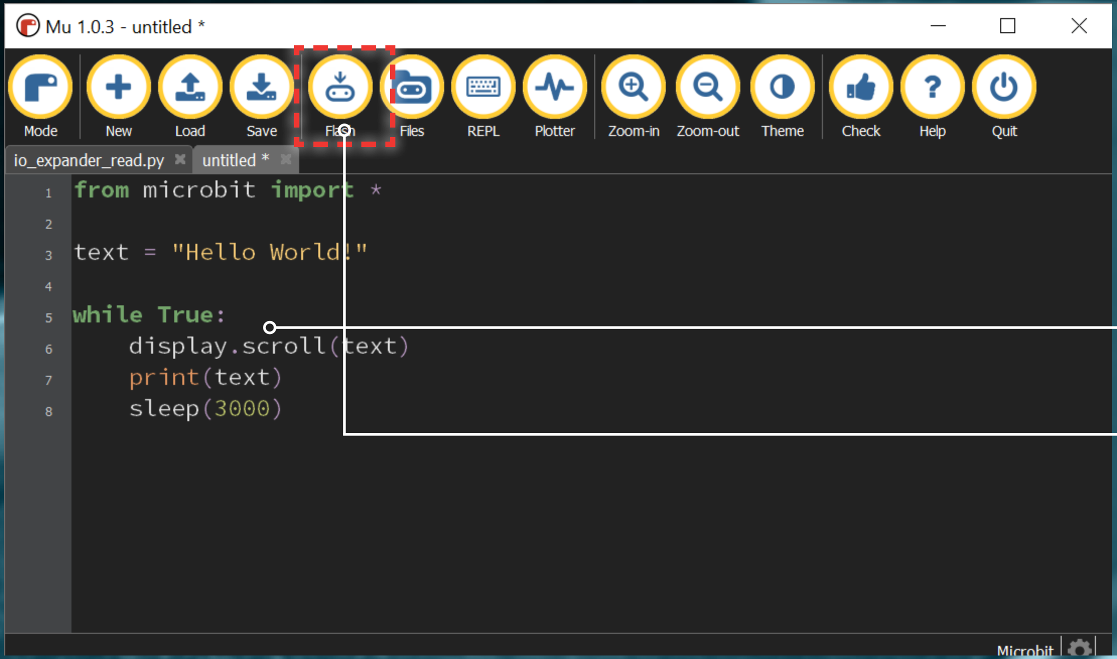
THE FIRST START

A NEW CHAPTER

You have no experience yet with the MU development environment but you prefer to start with a small example? We are showing you the environment!



After the start of the development environment, you begin with an empty Project. You can start right away from there!

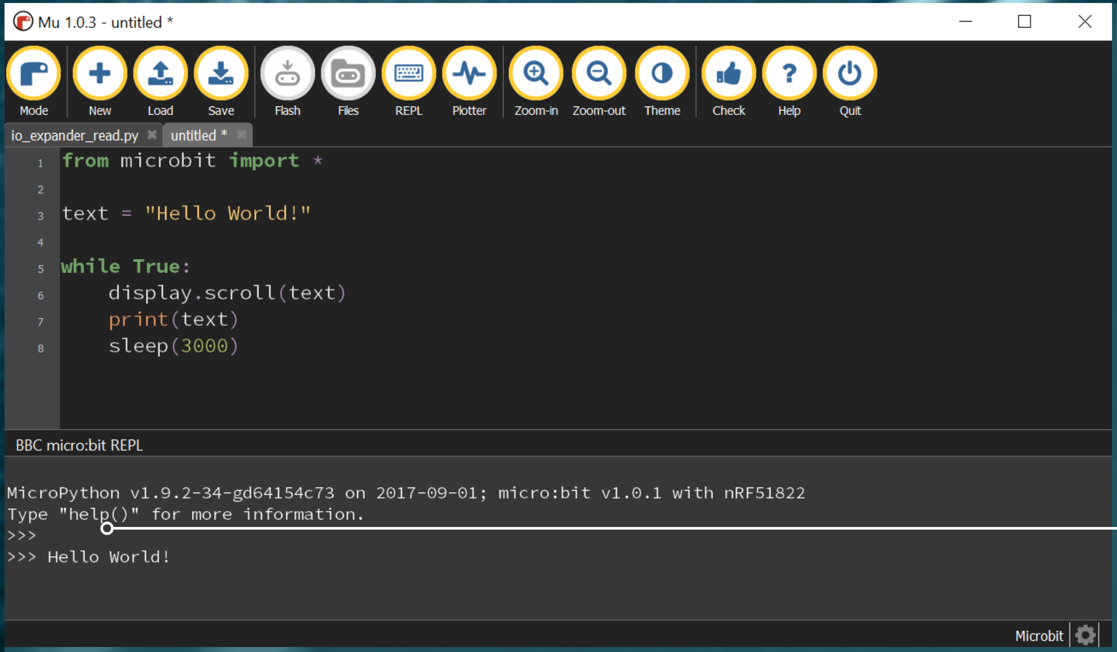


```python
from microbit import *

text = "Hello World!"

while True:
    display.scroll(text)
    print(text)
    sleep(3000)
```

It is best to start with a simple test. Feel free to take over the left example. The text "Hello world!" will be displayed on the LED matrix of your micro:bit and on the console.

INSERT YOUR CODE HERE

WHEN YOU ARE FINISHED, CLICK ON FLASH TO DOWNLOAD THE CODE TO YOUR CONNECTED MICRO:BIT.



```python
from microbit import *

text = "Hello World!"

while True:
    display.scroll(text)
    print(text)
    sleep(3000)
```

```
MicroPython v1.9.2-34-gd64154c73 on 2017-09-01; micro:bit v1.0.1 with nRF51822
Type "help()" for more information.
>>>
>>> Hello World!
```

After you click on flash, the code is transferred immediately to your micro:bit and is automatically launched. The output of the text should then be in the console, as well as on the LED matrix of your micro:bit.

From here all possibilities are open to you! Take a look at our sample files, which we have compiled in the next subchapter for you.

CONSOLE OUTPUT

CODE

MODIFICATIONS

You want to learn how the Joy-Car works? Want to add optimizations? Add your own developments and ideas? No problem!

In the following subchapter we have broken our code into its parts and split it into individual files for you. We explain the relevant functions and what they do. This way you can learn all functions step by step and can put together the elements that are important for you and develop your own Joy-Car programming.

KEEP THE OVERVIEW

We have divided the programming of the Joy-Car for you into 12 files. This way you will get to know every function, understand and can put together what you need for your own development. We have put together the following files for you:

IO_EXPANDER_READ.PY	Reads the sensor data from the IO expander
IO_EXPANDER_CONTROL.PY	Control of the optional sensor at the 7th output of the IO expander
MOTOR_CONTROL.PY	Control of the two motors by the PWM controller
ADC_READ.PY	Reading the supply voltage by the internal ADC
RC_JOYCAR.PY	Simple remote control of the Joy-Car (Joy-Car file)
RC_REMOTE.PY	Simple remote control of the Joy-Car (remote control file)
RC_PLUS_JOYCAR.PY	Extended remote control of the Joy-Car (Joy-Car programming)
RC_PLUS_REMOTE.PY	Extended remote control of the Joy-Car (remote control file)
LIGHTS.PY	Control of the headlight modules
DEMO.PY	Example code
SONAR.PY	Control of the ultrasonic sensor
SERVO.PY	Control of the servo channels

IO_EXPANDER_READ.PY

All sensor information is read by the IOExpander via I2C here. The readout is done by the function **fetchSensorData()**. No pass parameters are needed here. The return value is a dictionary which contains the following information:

0:	Speed-Sensor Left
1:	Speed-Sensor Right
2:	LineTracking-Sensor Left
3:	LineTracking-Sensor Center
4:	LineTracking-Sensor Right
5:	Obstacle-Sensor Left
6:	Obstacle-Sensor Right
7:	Optional Sensor

The individual parameters specify the sensor status as a boolean variable. **True** stands for a HIGH detection and **False** for a LOW detection.

ADDITIONAL INFORMATION: The function **zfill(s, width)** is used in this file. It is actually part of Python, but was omitted during the port to micro:bit for memory reasons. However, the function is relevant for reading sensor data and was therefore added by us. If you create your own files that read out the sensor data, this function is necessary and should also be used.

IO_EXPANDER_CONTROL.PY

A device connected to output 7 of the IOExpander can also be controlled by the IOExpander or set to HIGH/LOW. This is demonstrated in this example by the functions **out7off()** and **out7on()**.

MOTOR_CONTROL.PY

The motors can be controlled via the PWM controller. The drive(PWM0, PWM1, PWM2, PWM3) function takes four values, each between **0** and **255**. This defines the motor speed. **PWM0** and **PWM1** define the left motor and control the forward and reverse movement. **PWM2** and **PWM3** define the right motor accordingly.

The additional function **stop()** sets all motor movements to 0 and thus performs a braking maneuver.

ADC_READ.PY

The supply voltage can be read out via the internal analog-digital converter of the micro:bit. This can be used, for example, to integrate a voltage monitor or a battery warning. The function **supplyVoltage()** performs the reading from the ADC converter. It does not expect any transfer parameters and returns a single variable as return value. This contains the measured voltage.

RC_JOYCAR.PY & RC_REMOTE.PY

With the help of these two files you can turn your Joy-Car and another micro:bit into a remote controlled car. Load the rc_joycar.py on the micro:bit of your Joy-Car and the rc_remote.py on the micro:bit you want to use as remote control. Tilt the micro:bit with the remote program and your Joy-Car will drive in that direction. On button A you use the horn. Button B turns on the light.

RC_PLUS_JOYCAR.PY & RC_PLUS_REMOTE.PY

These scripts work like the normal remote control scripts. However, the driving functions have been fundamentally revised. Here it is now possible to adjust the speed in all directions. The more you tilt the micro:bit in one direction the faster your Joy-Car will drive or the tighter it will turn.

LIGHTS.PY

The control of the WS2812B RGB LED headlight modules is performed with this script. Thereby, the functions **lights(x)** stands for the low beam, **highBeam(x)** stands for the high beam, **breakLight(x)** stands for the brake light, **reverseLight(x)** for the reverse light and **indicator(y)** for the turn signal. Thereby, X can take over the values 0 and 1 (Off & On). Y, the function of the indicators, can take over the values 0, 1, 2 and 3 (Off, Left, Right, hazard lights).

DEMO.PY

This script serves as an example application and consists of three modes that can be switched with the button A:

Mode 0: Standby, **Mode 1:** Linefollowing, **Mode 2:** Obstacle detection and avoidance

SONAR.PY

This script shows you how to use the sonar of the Joy-Car. The function **sonar()** does not expect arguments and returns the measured distance to the next object in cm.

SERVO.PY

This script shows you how to use the two servo channels. The **servo(x, y)** script expects two arguments. **x(1 - 2)** is the servo channel to be controlled and **y(0 - 180)** is the position the servo should move to.

GET STARTED

LET'S RIDE!

Your Joy-Car is now ready for action. You can either develop your own programming or try out our sample codes and get to know the functions of your Joy-Car. All files and examples can be found for download on our **Joy-Car-Website**.

ADVICE: IF YOU WANT TO START IMMEDIATELY, TRY THE DEMO.PY - FILE. HERE YOU CAN START RIGHT AWAY!

SUPPORT

We also support you after your purchase! If you have any questions left or if you encounter any problems please feel free to contact us by mail, phone or via ticket-supportsystem on our website.

E-Mail: service@joy-it.net
Ticket-System: http://support.joy-it.net
Phone: +49 (0)2845 9360 – 66 (10 - 17 Uhr)

Please visit our website for more information:

www.joy-it.net

You can also find more guides, instructions and support on:

JOYCAR.JOY-IT.NET