makeblock education

Lesson 6: Careful drive

Subject: STEAM **Grade(s):** 5th and up

Duration: 45 minutes **Difficulty:** Beginner

Lesson objectives

By the end of this lesson, students will be able to:

- Recognize and use the code blocks for the gyroscope accelerometer in CyberPi
- Build their own computer program in mBlock to measure the changes on the inclination of mBot2 and adjust its movements accordingly

Overview

A gyroscope measures tipping movements, more precisely the speed of turning/tipping movements. An accelerometer measures the change in velocity. Both sensors provide different information about the position in space.

Combining both types of data, the position and the movement of vehicles, robots or even space probes can be calculated. On bumpy roads, for example, robots might drive slower to prevent tipping over.

Focus

By the end of this lesson, students will know:

- How gyroscopes and accelerometers work and how their measurements can be combined
- How to make a robot react to sudden changes and detect crashed or bumpy roads

Pre-lesson Checklist

What do you need?

- PC or laptop (with USB output) with the mBlock software installed, the web version (also for Chromebook), or a tablet with the mBlock app installed
- The mBot2 with a CyberPi
- A USB-C cable or Makeblock Bluetooth dongle

Lesson plan

This lesson consists of four steps and takes a total of 45 minutes.

Duration	Contents
5 minutes	1. Warming up • Gyroscopes and accelerometers in everyday life • Which tilt movements can the mBot2 detect?
10 minutes	2. Hands-on • Getting acquainted with the different programming blocks of the gyroscope and accelerometer • Recreating and testing some programming examples of these sensors.
25 minutes	3. Trying out • Writing your own program for the robot
5 minutes	4. Wrap-up • What did you learn in this lesson? • Showtime: show what you did with your robot in a fun, short movie for later discussion • If your teacher allows, share the end result on social media with the hashtag #mBot2 • Reflection: What are you most proud of? What would you like to improve about your robot?

Activities

1. Warming up (5 min)

Step 1: Warming up

This step consists of two parts:

1. Gyroscopes and accelerometers in everyday life
2. Which tilt movements can the mBot2 detect?

1. Gyroscopes and accelerometers in everyday life

How do devices like smartphones identify their position (hold upright, display facing up/down)? How to determine if a crash is severe enough to ignite airbags? Estimating the current position without an external precise reference has been a challenge at sea with cloudy skies for centuries – and it is still a very important task for household items, robots and even space probes. They all use at least two different sensors, gyroscopes, and accelerometers. A gyroscope works with a crystal that is under electric tension. This converts the direction of the force of the tipping motion into an electrical signal. Gyroscopes measure the rotational speed in degrees per second and with the help of math, they can estimate the rotated angle as well – but the latter is not that accurate: gyroscopes "drift" over time. Accelerometers report the change of position (like in- or decrease of speed) – and regarding the earth's gravity, they report the gravitational acceleration (as compared to free fall). Both types of sensors can measure on each spatial axis, reporting either rotational speed or acceleration on X-, Y- and Z-axis.

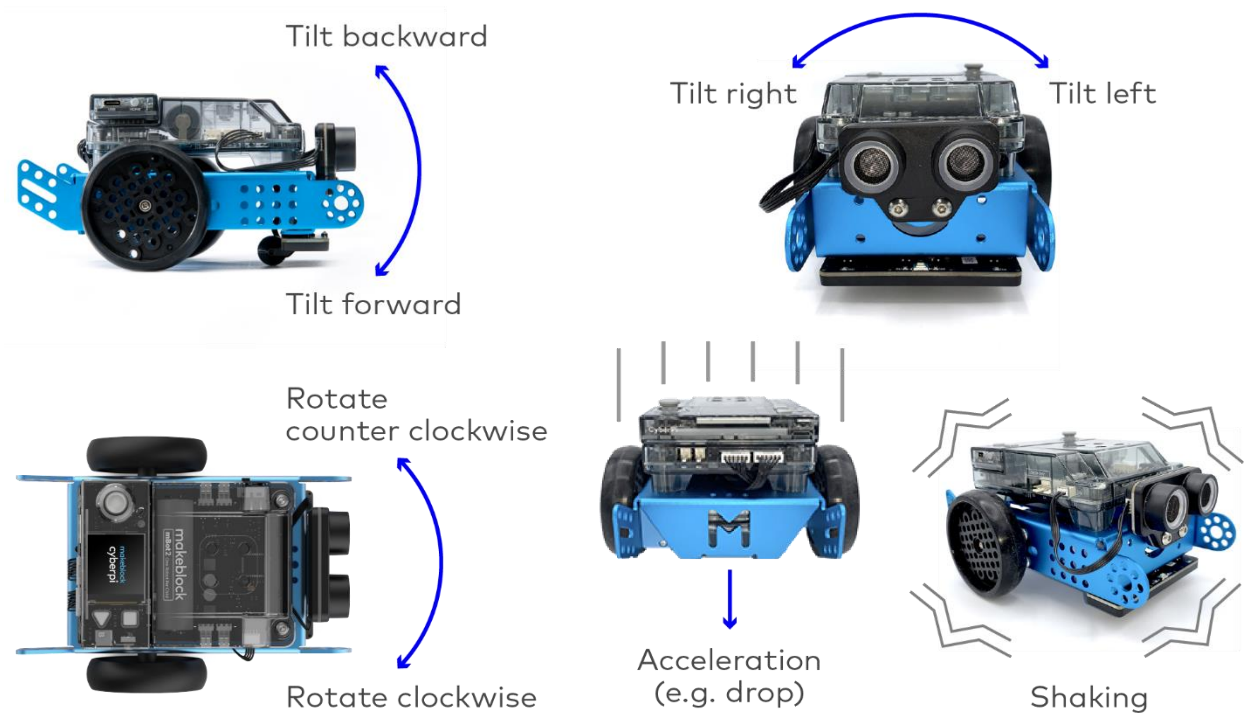
Combining data from both gyroscope and accelerometer, the direction and strength of movements can be calculated (in reality this is not super precise, so it is more a very good estimation). In Robotics these sensors are used to stabilize vehicles in uneven ground (or in flight) and help them maintain their programmed path. For the mBot2, these sensors are integrated into a small chip on the CyberPi.

Therefore, we will address the combined sensor as "motion sensor".

2. Which tilt movements can the mBot2 detect?

Before you learn to program in mBlock5 with the motion sensor, it is useful to know which tilt motions the mBot2 can measure. You can show the measurements of the tilting movements on the CyberPi display. In lesson 2 you have already learned about how to do this.

The mBot2 can detect the following tilt movements:



2. Hands-on (10 min)

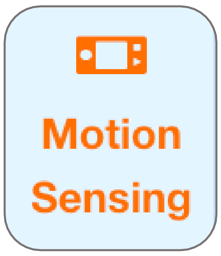
Step 2: Hands-on

This step consists of two parts:

1. Getting acquainted with the different programming blocks of the motion sensor
2. Recreating and testing some programming examples of the motion sensor

1. Getting acquainted with the different programming blocks of the motion sensor

In mBlock there are several code blocks for the motion sensor you can use in your own programs. These code blocks can be found in the 'Motion Sensing' category of the block field in mBlock. These code blocks are orange.



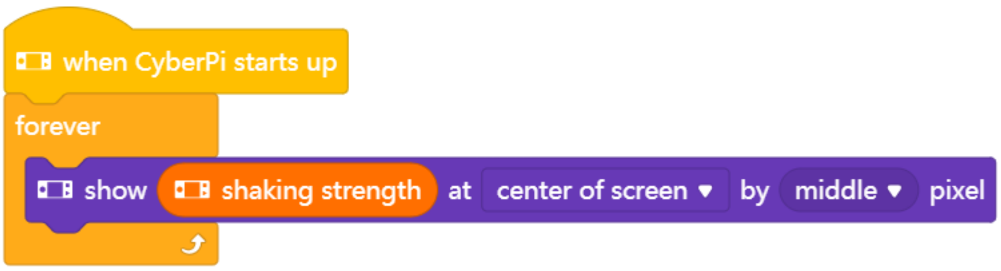
makeblock education

The table below gives a detailed explanation on the code blocks together with some sample program code to test it.

Code block:

shaking strength

This code block reports how hard the mBot2 is being shaken. At a value of '0' the mBot2 is not shaken at all. With a value of '100' a very hard shaking movement is detected. In the programming example below, the strength of the mBot2's shaking is shown on the display. If the mBot2 stops shaking, the display is blank.

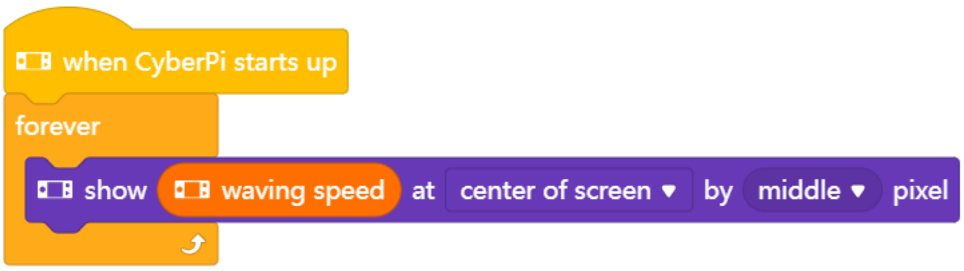


```
when CyberPi starts up
forever
    show (shaking strength) at center of screen ▾ by middle ▾ pixel
```

Code block:

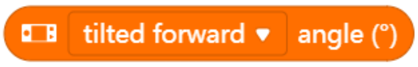
waving speed

This code block is used to record the waving speed. The waving speed ranges from 0 to 100. These units do not represent speed in m/s or in degrees/s, they are just a simple reference. In the programming example below, the waving speed of the mBot2 is shown in the display. If the mBot2 stops waving, the display is blank.

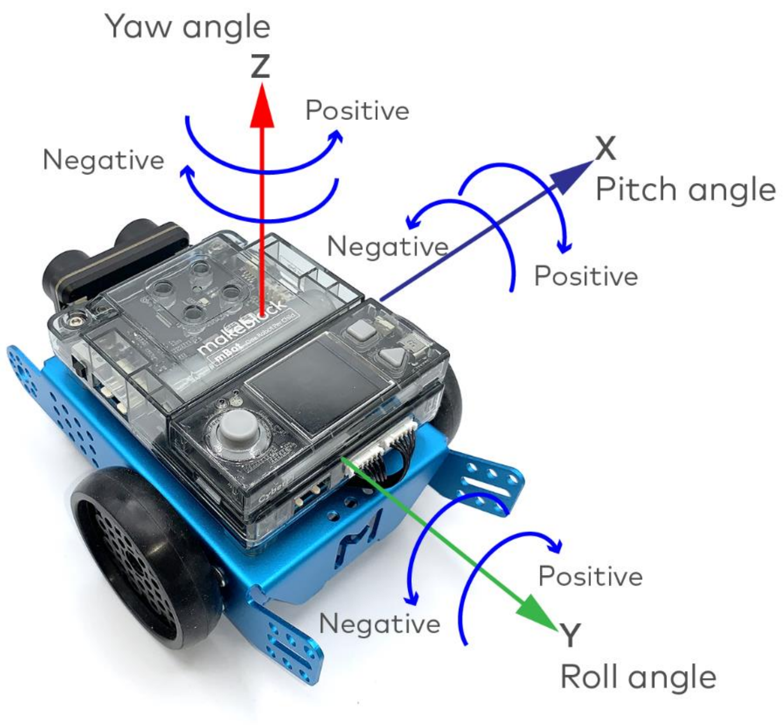


```
when CyberPi starts up
forever
    show (waving speed) at center of screen ▾ by middle ▾ pixel
```

Code block:


tilted forward ▾ angle (°)

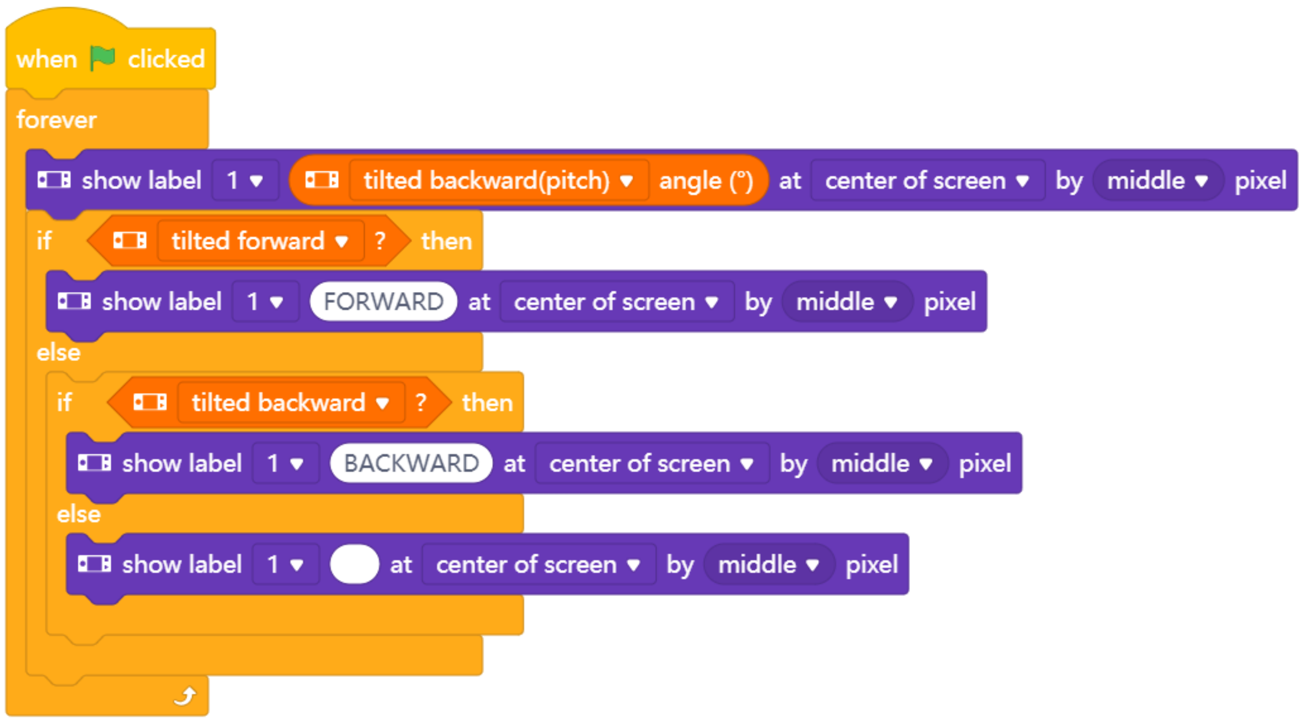
Each tilt makes a movement at a certain angle in a certain direction. When specifying an angle, the X, Y and Z axes are used to refer to the direction. Instead of these mathematical terms, they are addressed as roll, pitch and yaw as well. These terms originate from navigation of planes. Roll is the axis front-back, pitch refers to up/down turns (axis from left to right) and yaw is the top-down axis responsible for tuning left/right.
Take a look at the image below.



Angles are given in degrees:

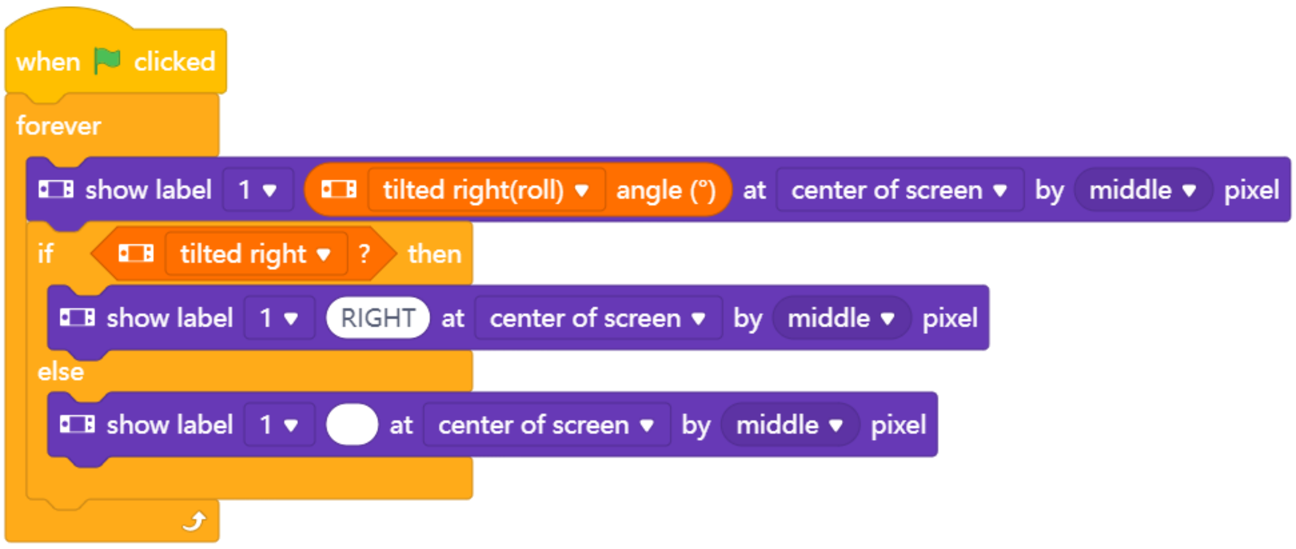
- Roll angle: -180° to +180°
- Yaw angle: -180° to +180°
- Pitch angle: -90° to +90°

In the program below, the pitch angle will be displayed in the top row and if a tilting forward or backward movement is detected, the words “forward” or “backward” appear on the last line. Can you identify the threshold for detection?



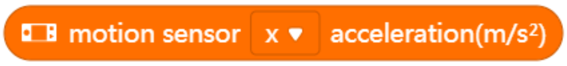
```
when 🏳 clicked
forever
    show label 1 ▾ (tilted backward(pitch) ▾ angle (°)) at center of screen ▾ by middle ▾ pixel
    if <tilted forward ▾ ?> then
        show label 1 ▾ FORWARD at center of screen ▾ by middle ▾ pixel
    else
        if <tilted backward ▾ ?> then
            show label 1 ▾ BACKWARD at center of screen ▾ by middle ▾ pixel
        else
            show label 1 ▾ ( ) at center of screen ▾ by middle ▾ pixel
```

Try to experiment with the different settings from the dropdown-menu. Now you have analyzed the tilting movement on the pitch/x-Axis. If you switch to tilt_left/right, remember to switch the Boolean reported blocks as well:



```
when 🏳 clicked
forever
    show label 1 ▾ (tilted right(roll) ▾ angle (°)) at center of screen ▾ by middle ▾ pixel
    if <tilted right ▾ ?> then
        show label 1 ▾ RIGHT at center of screen ▾ by middle ▾ pixel
    else
        show label 1 ▾ ( ) at center of screen ▾ by middle ▾ pixel
```

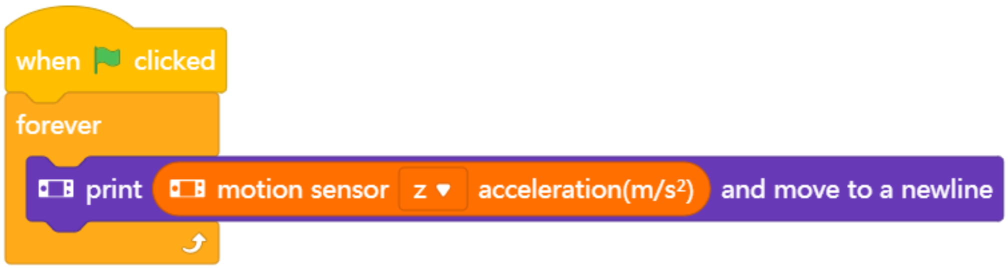
Code block:



```
motion sensor [x ▾] acceleration(m/s²)
```

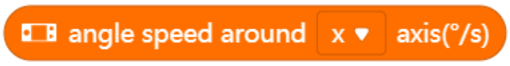
So far, we have looked at tilting movements, but the motion sensors incorporate another sensor, the accelerometer. It measures changes in the velocity of the mBot2 over time. This can be slow increases in driving speed as well as a sudden stop or even a side-impact from another vehicle. Since these measures are based on the x, y and z axes of the robot, you can detect a crash if you look for a rapid change in acceleration. The accelerometer indicates the acceleration in m/s2.

In the programming example below, the acceleration of the mBot2 on the z-axis is shown in the display. Move the mBot2 by hand and check the display. What readings does it show if the robot "crashes" into your hand or a book? Can you change the program and simulate a side-impact?



```
when 🏁 clicked
forever
    print (motion sensor [z ▾] acceleration(m/s²)) and move to a newline
```

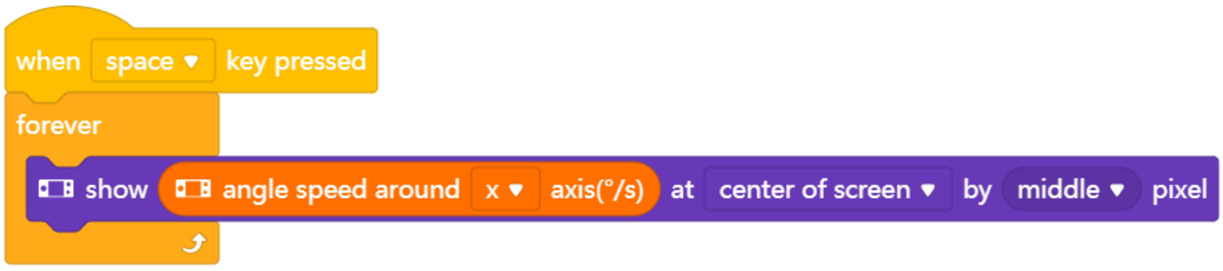
Code block:



```
angle speed around [x ▾] axis(°/s)
```

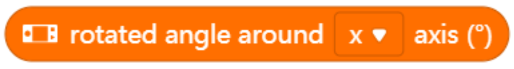
This code block shows the speed at which the mBot2 rotates around a certain axis. This is also called the angular velocity. The angular velocity is shown in degrees per second (°/s).

In the programming example below, the angular velocity of the mBot2 around the x-axis is shown on the display.



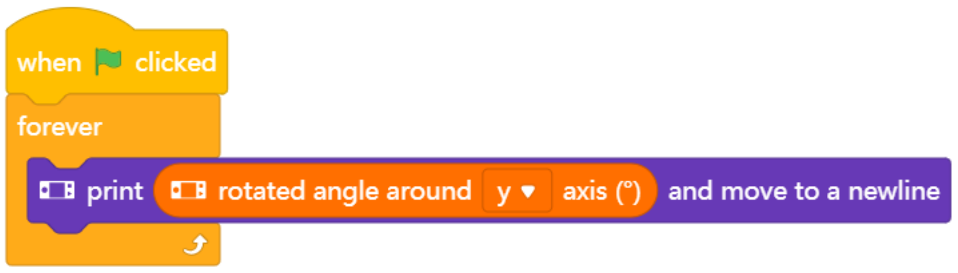
```
when [space ▾] key pressed
forever
    show (angle speed around [x ▾] axis(°/s)) at [center of screen ▾] by [middle ▾] pixel
```

Code block:



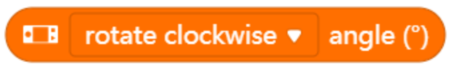
```
rotated angle around x ▾ axis (°)
```

This code block indicates the angle at which the mBot2 rotates around a given axis. The angle is displayed in degrees (°) and is measured relatively. The initial position is set when the mBot2 is turned on. This data is calculated from the rotational speed the gyroscope reports. It can drift over time and it is not that stable – try tilting the mBot2 sideways with the program below and check the values. Do they go back to their initial state?
In the programming example below, the angle at which the mBot2 rotates around the y-axis is shown on the display.



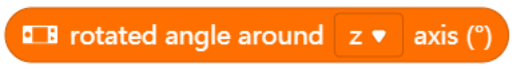
```
when 🏳 clicked
forever
    print (rotated angle around y ▾ axis (°)) and move to a newline
```

There are two important aspects: this block will keep track of rotational movements that are greater than one revolution. Use this block to count how many times the robot has turned. Looking at the z-Axis, a clockwise rotation will increase the values reported by this block:



```
rotate clockwise ▾ angle (°)
```

While the rotated angle around the z-axis decreases:



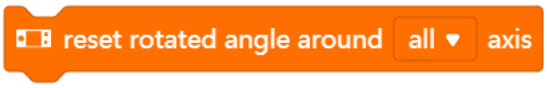
```
rotated angle around z ▾ axis (°)
```

After 3 Rotations of 360° the last block will report "-1080°", but the previous block will just show "0" (back to initial position, ignoring multiple rotations). If then rotated counterclockwise 4 times (one more time previously), the "rotate_clockwise" angle is at 0° again, while "rotated_angle_around_z axis" now shows 360° (all figures without gyroscope drift).

If you just need to take small rotational movements into consideration, for the x- and y-axis, the "tilted (___) angle" blocks discussed beforehand provide a more robust measurement.

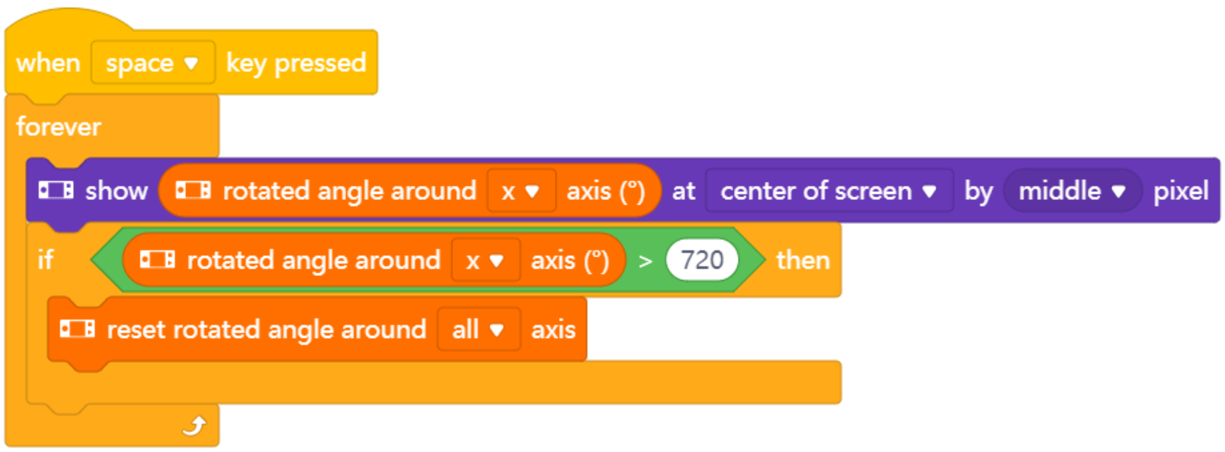
There is one exception: rotations around the yaw/Z-Axis can only be measured with the help of the gyroscope. If you want to measure the degrees, the mBot2 is turning left/right, you need to work with this data.

Code block:



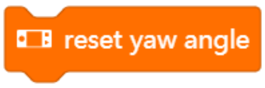
```
reset rotated angle around [all ▾] axis
```

If you want to determine the angle that the mBot2 is making, it is easier to measure the change from the value 0. This code block resets the measured value of an angle to the starting value 0, defining a new starting position for future turns.
In the programming example below, the angle at which the mBot2 turns around the z-axis is shown on the display. If the angle is greater than 720°, then the measured value of the angle is set to 0 and it is measured again.



```
when [space ▾] key pressed
forever
    show (rotated angle around [x ▾] axis (°)) at [center of screen ▾] by [middle ▾] pixel
    if <(rotated angle around [x ▾] axis (°)) > (720)> then
        reset rotated angle around [all ▾] axis
```

Remember, there are two blocks to measure the rotation: one only cares for full rotations, the other keeps counting. This reset-block sets all (or selected) "rotated_around_axis" blocks to zero. It does not have an effect on the "rotate clockwise"/ "rotate counterclockwise" block. This one has its own reset block:



```
reset yaw angle
```

Any setting-to-zero only applies to rotations reported by the gyro sensor alone. The tilted ___ angle reports always from a perfectly leveled plane perpendicular to the earth center and cannot be set to zero at will.

2. Recreating and testing some programming examples of the motion sensor

In the table above each code block of the motion sensor has a programming example. You are going to recreate these programming examples in mBlock and test them.

The motion sensor on the CyberPi registers the tilting movements of the mBot2. When testing the above programming examples; therefore, you actually only need the CyberPi. If it is more convenient, remove the CyberPi from the mBot2 and make the tilting movements with the CyberPi yourself during testing. You can test the programming examples in live mode. Since the battery is in the mBot2 shield, you will need to connect the CyberPi to the computer via the USB-C cable to run the programs.

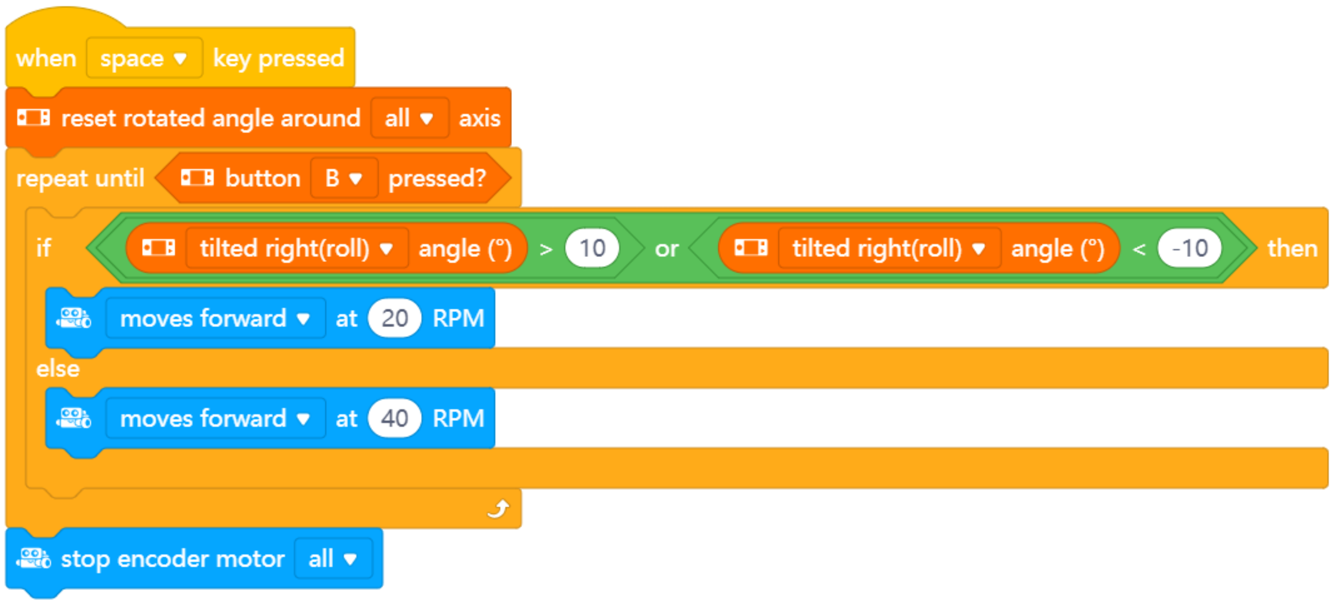
3. Trying out (25 min)

Step 3: Trying out

You have already learned a lot about the motion sensor on the mBot2. You are now going to create your own computer program in mBlock using this sensor.

In this assignment you are going to create a bumpy track for the mBot2. You can make this track from, for example, (small) stones or plasticine on cardboard. The idea is for the mBot2 to drive over the course. If the mBot2 is shaken (hard), it must drive slower. You can also put an object on the mBot2 that must not fall while driving.

Tricky task, isn't it? Fortunately, you don't have to invent everything yourself! Below you can see a programming example that you can extend. In this programming example, the mBot2 will slow down if it makes too large tipping movements.



```
when space ▾ key pressed
reset rotated angle around all ▾ axis
repeat until button B ▾ pressed?
    if tilted right(roll) ▾ angle (°) > 10 or tilted right(roll) ▾ angle (°) < -10 then
        moves forward ▾ at 20 RPM
    else
        moves forward ▾ at 40 RPM
stop encoder motor all ▾
```

Use the knowledge you gained in 'Step 2' of this lesson. Of course, you can do plenty of experimenting yourself with the different programming possibilities in mBlock.
When thinking about this assignment, it is helpful to use the following step-by-step plan.

	Explanation
Step 1: What do you want to do?	• What route do you want the mBot2 to drive? • When should the mBot2 slow down? • Do you want the mBot2's sensors to also show data on the display?
Step 2: What do you need?	• What do you need in addition to the mBot2?
Step 3: What code blocks do you need to make the mBot2 drive?	• How are you going to make the mBot2 drive? • What code blocks will you use? • Make a brief description on how your program works (pseudocode/natural language, flowchart or UML) • If you need further explanation, you can discuss with your fellow students, the teacher, or do a research on the topic. There is help available for every code block in mBlock as well
Step 4: How do you want to show the data from the sensors on the display?	• How do you want to show the data on the display? • What code blocks will you use? • Make a brief description on how your program works (pseudocode/natural language, flowchart or UML) • If you need further explanation, you can discuss with your fellow students, the teacher, or do a research on the topic. There is help available for every coding block in mBlock as well
Step 5: Testing and implementation	• Is the first version ready? Test it! During the testing round, write down areas of improvement • Work on the improvement points until your mBot2 does exactly what you had in mind • Successful? Film the end result and consult with your teacher if you can post it on social media with the hashtag #mBot2

4. Wrap up
(5 min)

Step 4: Wrap up

How did this task go? Did the object stay neatly on the mBot2 while driving?

In this lesson, you learned about sensors for measuring movement around axes and acceleration, the gyroscope and the accelerometer. They are integrated into a single little component placed on the CyberPi. These combined sensors are used to determine the position of a robot in 3D-space. They are called inertial measurement units, IMU. You have learned where to encounter them in everyday life. You know what tilt-movements and movement-changes the mBot2 can measure and how you can use different code blocks to take advantage of these measurements in your programs with mBot2.

It is now time for a brief reflection. Think on your own and discuss with the group:

- What do you think turned out well?
- What could be even better?
- Which parts of the lesson did you find easy and which did you find more difficult?
- What would you like more explanation about?
- Who could help you with that?