Activity 1:

Parking Gate!

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

Duration: *45 minutes* **Difficulty:** *Intermediate*

Lesson objectives

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

- *Use event based coding*
- *Store data in variables*
- *Work with a different kind of Motor, the servo*
- *Build an Automation solution*

Overview

In cities, parking space for vehicles is rare. A parking area with restricted access increases safety and comfort as well. In this activity students will build an automatic gate system that lets vehicles pass in both directions. Starting simple, the gate opens by a press of a button and closes after some time. This is the starting point of a journey into STEAM… depending on the students' interests, this simple system can be extended:

- *Secured area, access only with temper-proof ID. Use Image recognition (TM, see Lesson 9)*
- *Remote management: Display the current number of parking vehicles on the internet.*
- *Use colour codes for special discounts on the parking fee*
- *Use multiple entry systems that communicate with each other.*
- *Challenge the other teams in the class to "hack" your system and open the barrier unauthorized. Can you develop temper-proof systems?*

Next to the fun-factor for students, there are quite some STEAM-related topics involved. Starting simple with event based coding, the level of complexity can be adjusted on the fly.
This activity does not include ready-made solutions but explains the most important concepts and code-fragments, so students can develop the entire algorithm and program code instead of just replicating a given computer program.

Focus

makeblock education

By the end of this lesson, students will know:

- *How to make use of event-based coding*
- *Use sensor data and variables to store data*
- *Use a different kind of motor, the servo.*
- *How to Realize their own projects: starting small, adding complexity step by step*

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*
- *mBot Bracket Pack Add-On*
- *9g Micro Servo Robot Pack*

Activity Plan

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

Duration	***Contents***
5 minutes	***1. Warming up*** • *Getting to know a different kind of motor* • *Discussing simple approach for opening & closing the gate*
10 minutes	***2. Hands-on*** • *Getting to know the coding blocks* • *Building the gate* • *Testing simple program*
25 minutes	***3. Test and Development*** • *Discuss additions your team wants to add* • *Test, Improve, Repeat – then challenge other teams*

5 minutes	***4. Wrap-up*** • *Showtime: Which improvements were realized by the teams?* • *Did anybody develop a temper-proof solution?* • *Reflection: what are you most proud of? What would you like to improve about your robot and code?*

Activities

1. Warming up
(5 min)

Step 1: Warming up

This step consists of two parts:

1. *Getting to know a different kind of motor*
2. *Discussing simple approach for opening & closing the gate*

1. Getting to know a different kind of motor

"Normal" motors start to spin when they are powered up. The Servo motor contains a motor like that (you can see it through the plastic casing), but gears and electronics as well. Its purpose is to rotate to a given position and keep this. It only rotates within a range of ca 180°. The third wire is used to provide the information of which angle to turn to in a coded format, the other are just powerline, like every ordinary motor.

Whenever small, precise movements/rotations are needs, these motors are widely used (in RC toys for steering, e.g.).

2. Discussing simple approach for opening & closing the gate

The scope of this project is to start small and then grow the complexity of the project: Implement a new feature, test, improve, repeat... following this circle, students can develop individual solutions with different levels of complexity. The first task is to open by pressing a button, wait a few moments, then close again.

2. Hands-on
(20 min)

Step 2: Hands-on:

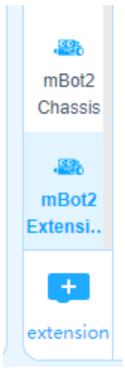
This step consists of 3 parts:

1. *Getting to know the coding blocks*
2. *Building the gate*
3. *Testing simple program*

1. Getting to know the coding blocks

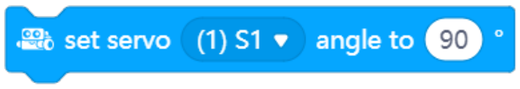
Before building the extension that opens and closes the gate, students need to be aware of the relevant coding blocks. The servomotor rotates in a range of 0-180°, but the current position is unknown until the first position command is executed.

To build the gate perfectly aligned and opening upwards (not "down"), students first need to connect the servomotor and set it to a starting position. The servo will be connected to the S1 port next to mBot2's power switch. The connector cannot be reversed, it only fits one way.



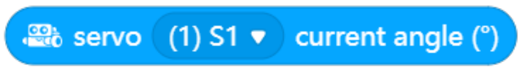
So far, the mBot2 extensions from the Chassis have been used, now we will cover coding blocks from the mBot2 Extension Part (scroll down the extensions list, if you don't see them):

Code block:



```
set servo (1) S1 ▾ angle to (90) °
```

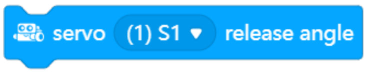
This block make the Servo connected to Part S1 to move to 90°, a middle-position of the entire range. By default, all Servo-Ports are selected; this should be changed to S1 only.



```
servo (1) S1 ▾ current angle (°)
```

This block reports the degrees the servo was instructed to turn to. It does not read its "real" position. In case the servo can't turn, because the load is too heavy or if the servo has been switched off and moved, this block will report a wrong position (the intended position, not the actual one). The servo has an internal control mechanism to turn to the destination, but there is no feedback to the computer if the position is reached.

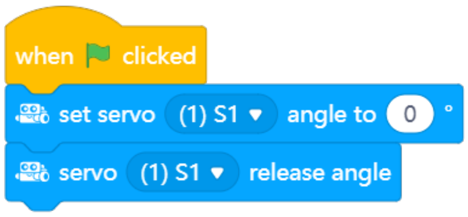
However, it is useful in finding out the position that the servo should turn to – you don't need to keep it in a separate variable.



```
servo (1) S1 ▾ release angle
```

This block will switch the power of the servo off. It then can be moved by hand or under a load big enough. If not switched on, the internal control electronic will constantly try to hold the desired position and keeps the little motor inside the servo switched on…

Before you actually start building the gate, connect the Servo to port S1, connect to the mBot2 in mBlock5 in live-mode and run the following program:

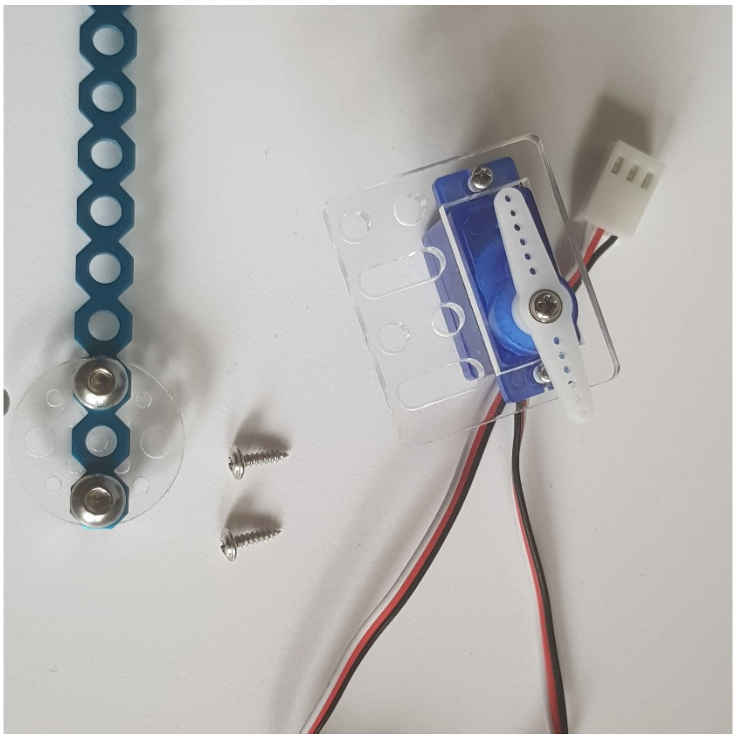
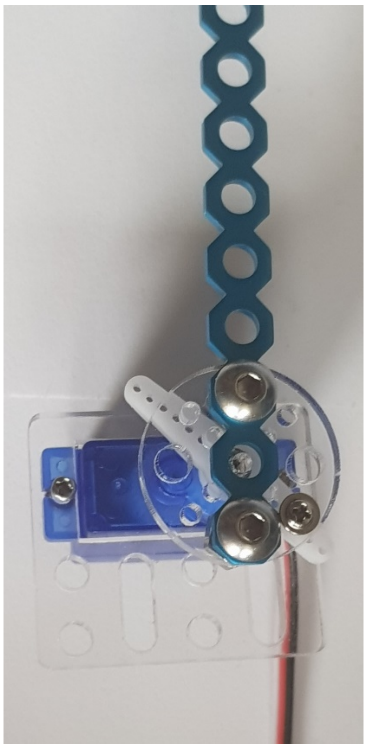


```
when 🏁 clicked
set servo (1) S1 ▾ angle to (0) °
servo (1) S1 ▾ release angle
```

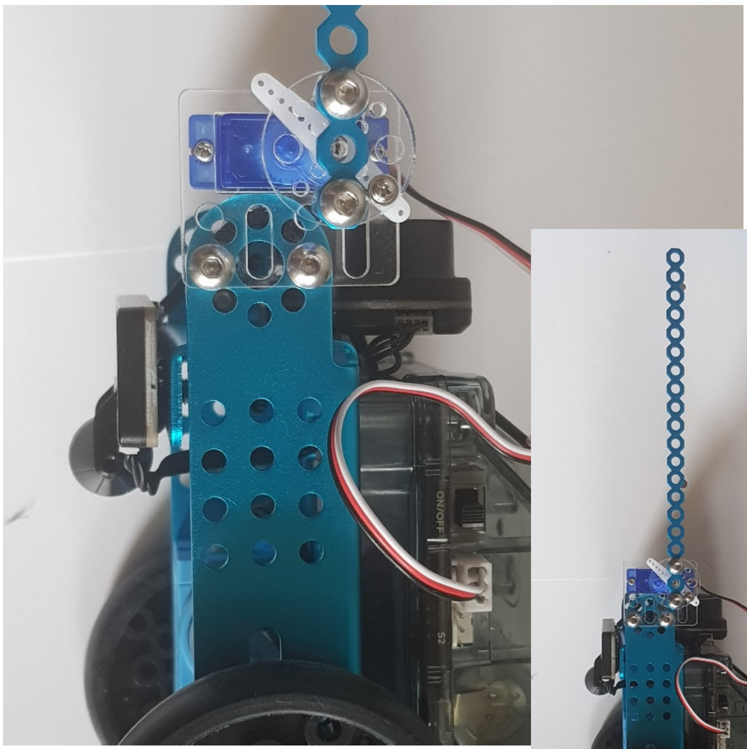
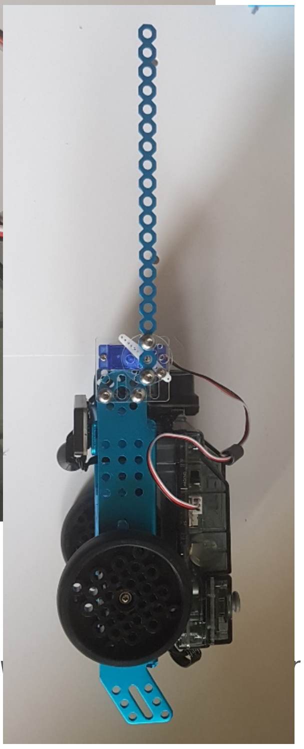
This will position the servo to the starting position. So as soon as you attach the gate, it will be in the correct position.

2. Building the gate

Look at the instructions on how to connect the servo with mBot2 and the beam to act as a gate. The servo axle is ribbed to allow a fine tuning of position while avoiding any slip. The laser-cut acryl disk is used to connect the servo axle to the beam; the other acryl part is for mounting the servo itself. Have a look at these pictures:

Preparation and assembly – check that the beam (gate) is at the 0-position (run the code before)

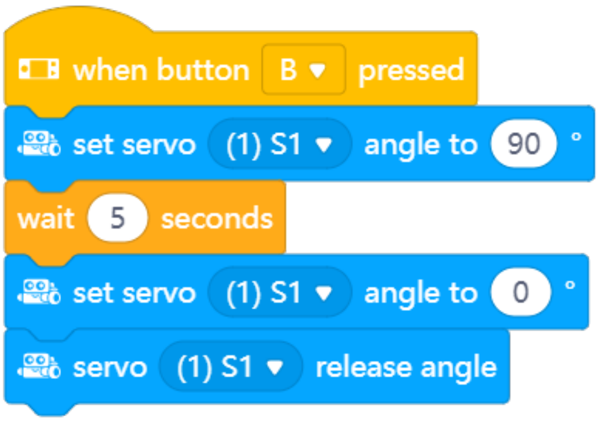



Assembled on mBot2… and complete view.

If you are finished, the servo motor should no … *r the small beam!*

3. Testing simple program

As a basis to start from, we will discuss a simple first program that opens on command and closes after a short moment again. For this, we will use event based coding – the code is executed if the liked event takes place (like a button press), there is no need to constantly check for a key being pressed in a loop:



```
when button B ▾ pressed
set servo (1) S1 ▾ angle to 90 °
wait 5 seconds
set servo (1) S1 ▾ angle to 0 °
servo (1) S1 ▾ release angle
```

By pressing button B on mBot2, the gate moves up (90°), and is lowered again after 5 seconds.

Make sure this works and everybody from the team has understood the concept and different way to control this servo motor.

In the next step you are going to improve this simple program.

3. Test and Development
[25 minutes]

Discuss within the team which limitations you see with this simple design and try to improve it. Consider safety of use over comfort!

You can have a short brainstorming on ideas, then link them with suggestions how to realize the ideas and check if you are confident realizing it in the timeframe of this step.

Suggestions for improvement:

- *What happens if the vehicle needs to stop under the gate or can't pass in time?*
- *Can you manually temper with the gate if it is locked?*
- *Think of colour codes you can use ... for a rebate on parking bill or as access restrictions*
- *Draw an image to serve as a "water mark" on a ticket system for high secure areas (banks, e.g.). Use the Teachable Machine feature to recognize this with a webcam (see getting started activities unit 9)*
- *Use different buttons for entering or leaving and count the number of vehicles in the parking zone. Make sure your code does not open the gate of there are no vehicles, or that the number of vehicles does not exec the available space (or drops negative)*
- *You can exchange data between multiple mBot2 – how about a networked control system that keeps track of entering and leaving vehicles?*

There is a huge variety of options students can choose from to improve this starting code, and for most of them multiple solutions should be possible. This helps increasing interest in STEAM by allowing a diversity of ideas, concepts and different path to a solution.

Now it is time for a challenge: Invite other teams to test your team's development... and maybe they find further suggestions how to improve? Can they "hack" your secure gate system? Can you circumvent their security? Discuss, why thinking this way is an important approach as well.

4. Wrap up
(5 min)

Step 4: Wrap up

At the end of the activity, the variety of solutions should be summarized for all teams. Which safety systems were put in place? What improvements did the teams think of, but decided not to implement for now?

Discuss a review of the activity by the following questions:

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

But mostly, have fun doing STEAM