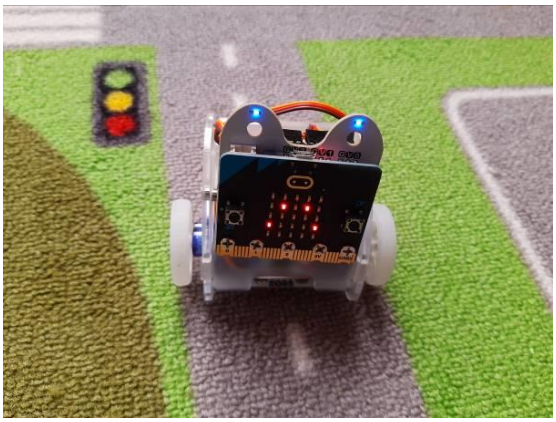
The Ringbit Car



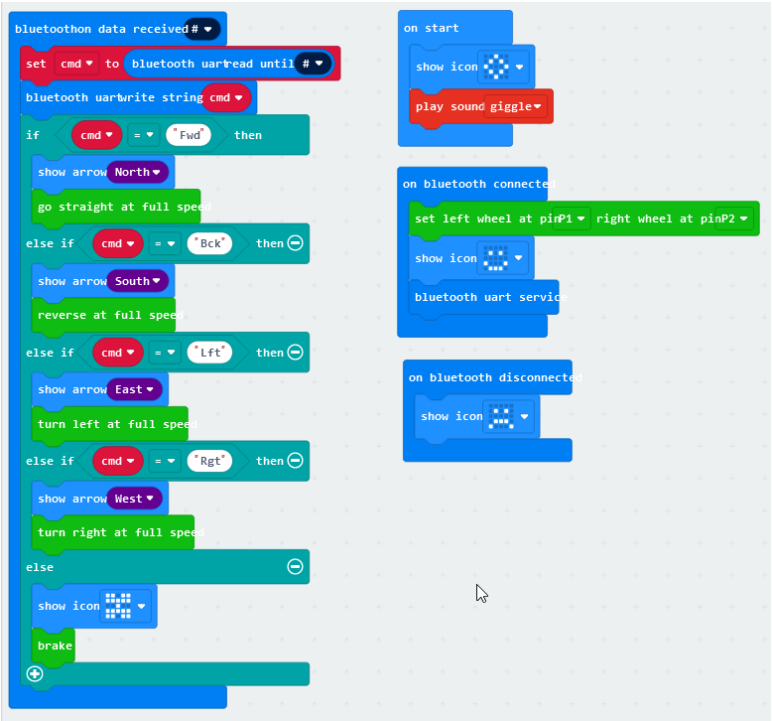
If you followed our tutorial on communicating with a micro:bit using UART and BLE, it will be rather simple now to build an app controlling a Ringbit car. In this first version we will just have commands to make the car go straight, reverse, or left and right.

Obviously, you should have a Ringbit car. It costs about $25 + a micro:bit. It is fairly easy to assemble, but be careful not to lose the tine screws! Also, my car did not want to go anywhere, until I discovered, by looking at a YouTube tutorial, that the two connectors that connect to the wheels, should be inserted upside down. If you have one of the many other toy cars with micro:bits you can buy, you can probably have very similar code to make your car drive around.

Having the car assembled, try some of the examples that you can find here:
https://www.elecfreaks.com/learn-en/microbitKit/ring_bit_v2

When convinced that the car operates properly,

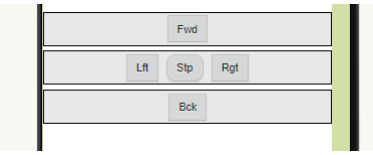
- Open the microbit-UART_RW.hex on the MakeCode site, that you made for the first part of this tutorial.



- Click on *Advanced* and *+ Extensions.*
- Search for *Ringbit*
- Click on the extension found
- You will see that *RingbitCar* appeared in the left-hand side menu.
- Adapt the code you had to look like the picture above.
- Rename the project to *UART_Ringbit1*
- Save and flash the .hex file to the micro:bit, by connecting the micro:bit via an USB cable to your PC and dragging the .hex file to the drive that represents the micro:bit.

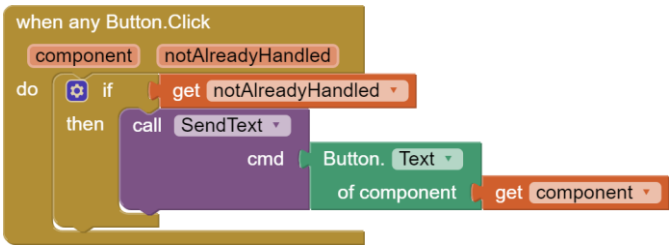
Next, we turn our attention to the App Inventor app.

Start from the *BTReadWrite.aia* that we created in the first part, or from a similar app you created. Add three horizontal arrangements and position five buttons on it as in this picture:



Hide the text field if you wish.

Go to the blocks tab. The blocks you need to add are surprisingly simple. This is all you need:



How does it work?

We carefully made the text on the buttons the same as the commands we handle on the micro:bit. When a button is clicked, App Inventor will look whether there is an event handles for that button, when *btnConnect.Click* for example. Next it will look whether there is *when any Button.Click* block. We specify in this block to *call SendText* only for those buttons for which there was no specific block. These are exactly the 5 buttons we have on our screen as direction buttons: *Fwd, Bck, Flt, Rgt* and *Stp*.

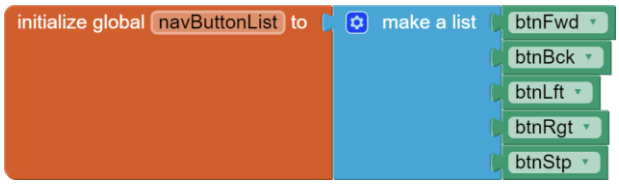
When you connect to the Ringbit car now, it will drive around like crazy and it is very hard to control. Therefore we make a slight modification: We would like the specific command to be sent when the button is pressed, and a Stp (stop) command when the button is released.

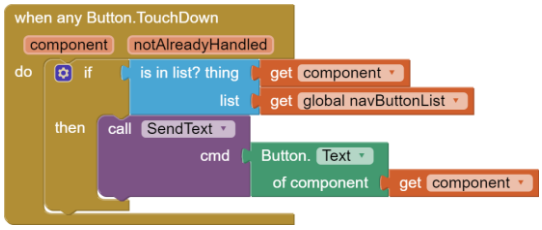
There are *when any Button.Touchdown* and *when any Button.Touchup blocks*, but there is a problem. Our *Scan, Stop Scanning* etc. buttons do not handle these events, therefore we cannot use the *if notAlreadyHandled* logic.

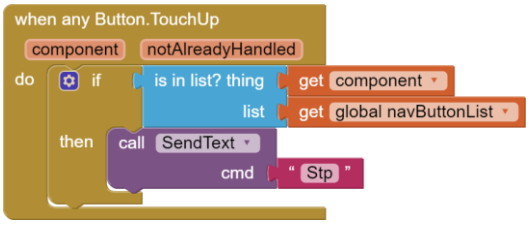
We create a list of all navigation buttons, like the list below.

When you touch-down any button, a check will be made whether it is a navigation button. If yes, the appropriate command will be sent to the micro:bit.

When a touch-up occurs, the same check will be done, and for all navigation buttons a Stp (stop) command will be sent.







Disable or delete the *when any Button.click* block.

When you reconnect to your micro:bit and try to move your Ringbit car, you will see that it behaves much better.

This concludes our example of controlling the Ringbit Car. You can find the ready-made .aia's and .hex files with the tutorial.

Possible enhancements

We did not try to use the LED lights on the car or the sound possibilities of the micro:bit v2.

It would be possible to control the car by using the Accelerometer. Speed and direction would be determined by tilting the phone. See here:
https://www.elecfreaks.com/learn-en/microbitKit/ring_bit_v2/ring_bit_car_v2_accelerometer_arithmetic.html

for an example how that can be done. In the example a second micro:bit is used that communicates via radio instead of Bleutooth, but the principle is the same.

Another possibility would be to send sequences of commands to the car, for example to drive in a square, or circle, or hexagon.

And also, you could buy enhancement boards that would enable the car to do obstacle avoidance or line following.