<Creating a Remote Controlled Car>

ECU Makerspace Group

Written by Julia Szymanski

Getting started

For this workshop, we are going to use this website:

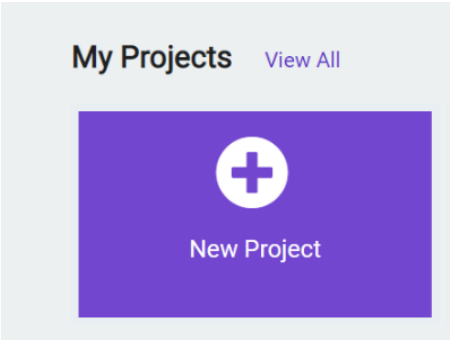
https://makecode.microbit.org/

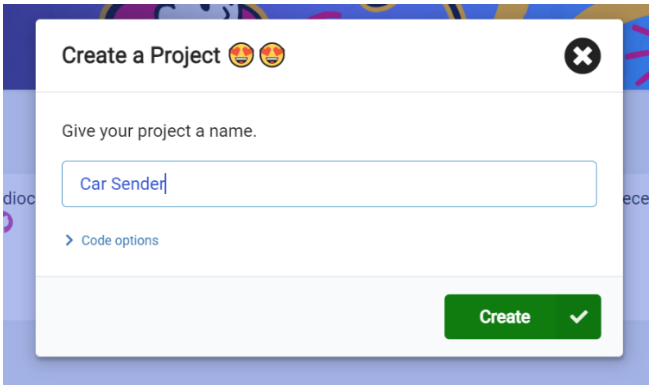
It works in the same way as the Makerspace website, but we use it for some additional functionality.

In this workshop, we are going to create a remote controlled car that can be controlled by using the accelerometer on another microbit. Depending on into which direction the sender microbit is tilted, it sends a radio signal to the microbit that controls our car, which then moves into different directions.

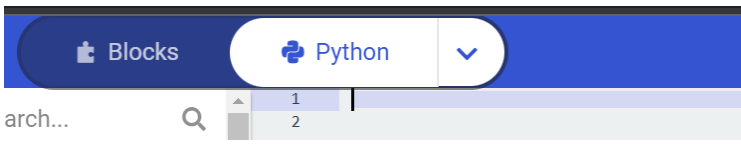
In the first part of the workshop, we are going to create the code for the sender microbit.

Once you have opened the website, click on **"New Project**" and give it a name, for example "**Car sender**"

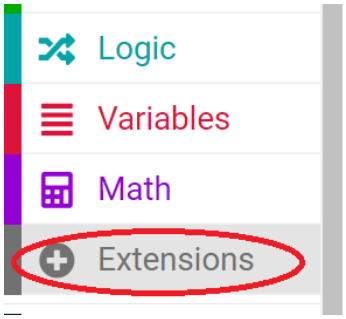




At the top of the page, click on "Python" and make sure you delete any code that might already be in there.



For our project, we need to import an extension- to do this, click on "**Extensions"** on the menu that is to the left of your code.



In the search bar, enter "**Kbit"** and select "**apprentice_Car"**.

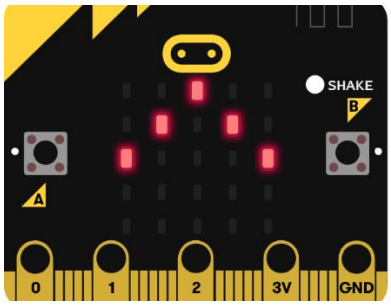
apprentice_Car

Makecode library for Apprentice car | Resolute

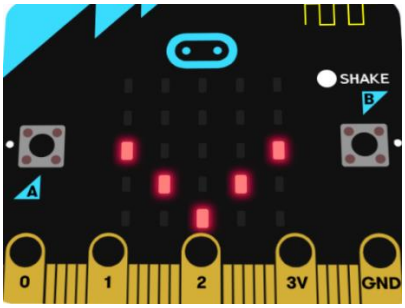
Part 1- Microbit Sender

We want the sender microbit to do the following:

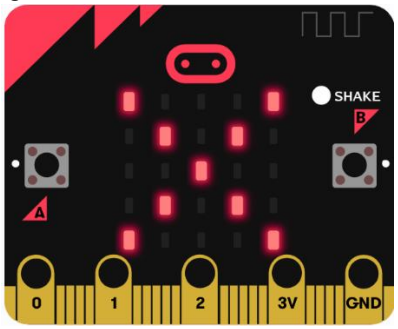
1. If the microbit is moved up, we want it to show this image and send a radio signal with the message “1”.



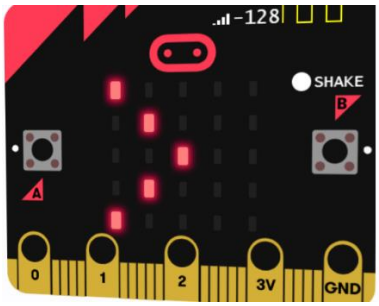
2. If the microbit is moved down, we want it to show this image and send a radio signal with the message “2”.



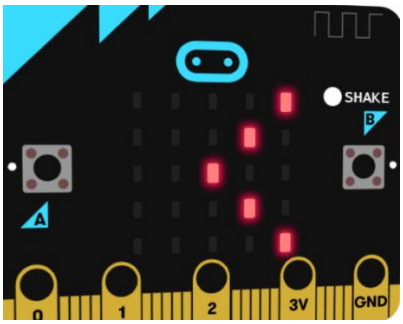
3. If we shake the microbit, we want it to show this image and send a radio signal with the message “3”.



4. If the microbit is tilted to the left, we want it to show this image and send a radio signal with the message “5”.



5. If the microbit is tilted to the right, we want it to show this image and send a radio signal with the message “4”.



Lets start writing some code…

We want our program to run indefinitely, so we need to define the function **on_forever()**
In the first line of the code, write **def on_forever():**

```
2
3   def on_forever():
```

Next, we create a **while loop**, that is set to **True**

```
2
3   def on_forever():
4       while True:
5
```

Inside the while loop, we place an **if statement**, so that when the microbit faces up, it sends a radio signal with "1" as the message and shows the LED's accordingly. The microbit has five LED's in each row and five in each column. For every LED that we want to light up, we replace the " . " in the code with " # ".

```
2
3   def on_forever():
4       while True:
5           if input.is_gesture(Gesture.LOGO_UP):
6                   radio.send_number(1)
7                   basic.show_leds("""
8                       . . # . .
9                       . # . # .
10                      # . . . #
11                      . . . . .
12                      . . . . .
13                  """)
```

This is the code for the first gesture "face up".
Can you write the code for the other ones as well?

Hint: Create if statements for each of them, and place them inside the while loop, under each if statement.

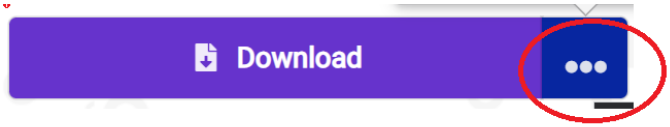
As a last step, we need to move out of the while loop and call the on_forever() function that we just created.
Just type **basic.forever(on_forever)** outside of the while loop.

```
41          if input.is_gesture(Gesture.TILT_RIGHT):
42              radio.send_number(4)
43              basic.show_leds("""
44                  . . . . #
45                  . . . # .
46                  . . # . .
47                  . . . # .
48                  . . . . #
49              """)
50  basic.forever(on_forever)
```

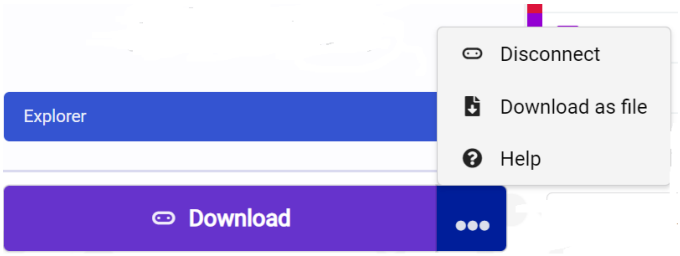
Well done! 😊

Now you can connect your microbit to your computer.

Once you have done this, click on the three dots next to **Download**.



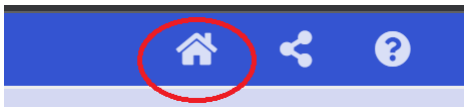
If yor microbit has connected automatically, you can just click on **Download** to load the code onto your microbit. If is not yet connected, just click on **"Connect"** and select your microbit.



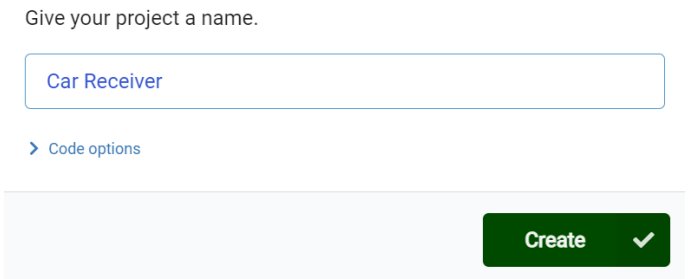
Part 2- Microbit Receiver

In part 2, we are going to write the code for the microbit that receives a signal from the sender microbit and makes the car go forwards, backwards, left right and stop.

Click on the house symbol to return to the main menu and



Then create another project named "**Car Receiver**", just as you did for Part 1. Remember to delete any code that might be already in there by default.



We want the robot car to **do the following**:

1. If a radio signal with a message "1" is received, set the forward speed to 40%.
2. If a radio signal with a message "2" is received, the backwards speed to 40%, so that the car runs backwards.
3. If a radio signal with a message "3" is received, get the car to stop.
4. If a radio signal with a message "4" is received, get the car to turn left at a speed of 40%.
5. If a radio signal with a message "5" is received, get the car to turn right at a speed of 40%.

You can try different speeds between 0 and 100%, however 100% is so fast that it becomes very difficult to control the car.

In the first step, we are going to create a function so that the microbit can receive radio signals and process the content of the messages sent to it.

In the first line, write

def on_received_number(receivedNumber):

```
1
2   def on_received_number(receivedNumber):
3
```

Next, we are going to set the forward speed to 40, if a message of "1" is received. kBit stands for the robot car and KBitRic is the direction the car goes into.

```
1
2   def on_received_number(receivedNumber):
3       if receivedNumber == 1:
4           kBit.run(KBitDir.RUN_FORWARD, 40)
5
```

Can you finish the code for moving the car backwards, to the left, right and make it stop?

Hint:

Write **if-statements** that go below the first one of message "1".

As a last step, we need to call the function that we just created by typing **radio.on_received_number(on_received_number)** below the code.

The code now should look like this

```
def on_received_number(receivedNumber):
    if receivedNumber == 1:
        kBit.run(KBitDir.RUN_FORWARD, 40)

    if receivedNumber == 2:
        kBit.run(KBitDir.RUN_BACK, 40)

    if receivedNumber == 3:
        kBit.car_stop()

    if receivedNumber == 4:
        kBit.run(KBitDir.TURN_LEFT, 40)

    if receivedNumber == 5:
        kBit.run(KBitDir.TURN_RIGHT, 40)

radio.on_received_number(on_received_number)
```

Now you can grab another microbit, **connect it** to your computer and **download** the code. Make sure both microbits are connected to a battery pack. You should now be able to control the robot car with the sender microbit.

Good job!

Part 3 - Create your own project

In this part of the workshop, you can either create **your own project** using the microbits, breadboard and different sensors

OR

You can **modify the robot car** and add some fancy add-ons to it.

Which add-ons can you think ok?

To give you some ideas:

a. Add **lighting** that changes depending on which direction the car robot moves into

b. If you feel confident, you can also include the built in **distance sensor**. Can you get the car to stop automatically once it moves too close to another object?

c. The receiver microbit in your car robot picks up *any* **radio signal** that is around- so if someone else also creates a sender microbit, they might be able to control YOUR robot car. Can you think of any ways to stop this from happening?

d. Pssst... this is a secret, so don't tell anyone. Did you know that you can get the robot car to **play some beats**, too?

e. You can change up the code and adjust the speed by pressing button A and button B to make the car go faster/slower

f. Can you get the car to follow a track and move along black lines on a piece or paper?

Hint:
If you would like some hints, check out the Cheat Sheet on the last page!

Cheat sheet

a. Add lighting

You can add lighting to the robot car by just adding one line of code.
By adding **kBit.led(KBitColor.COLOUR)** to the if- statements for each number that is received via a radio signal message, you can change the colour of the LED's of the robot car. Just replace COLOUR with a colour of your choice. Make sure you increase the brightness to a number between 0 and 100 (zero: dark, 100: very bright) to ensure the LED's light up.

```
6       if receivedNumber == 2:
7           kBit.run(KBitDir.RUN_BACK, 40)
8           kBit.led(KBitColor.BLUE)
9           kBit.led_brightness(100)
```

Here, the colour of the LED's changes to blue when the robot car moves backwards.

Can you change the colour of the LED's for when the car moves forward, backwards, to the left, right and when it stops?

Have you come across any colours that don't work?

b. Distance sensor

This one is up to you to try out 😊

c. Radio Communication

To stop your microbit from getting controlled by anyone else, you can set up a radio communication channel- this way only the mircobits in the same channel are able to communicate with each other.

You can do this by adding one line to your code- either on top or below the other code
radio.set_group(x)
Just replace **x** with an integer of your choice. Make sure you add this line of code to the sender microbit as well, so that both mircrobits are on the same communication channel.

In this example, x = 139.

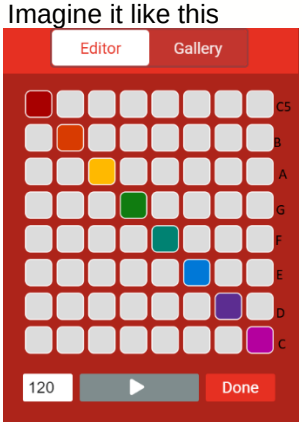
```
17
18      radio.set_group(139)
19
20
```

What happens if you change x on only one of the microbits?

d. Play some music

If you would like the robot car to play some beats while it moves around, you can add
music.play_melody("C5 A B G A F G E ", 120)
your code.

Imagine it like this



(This melody would be `("C D E F G A B C5 ", 120)`)

You can have multiple beats per row, but only one per column. Try some different ones, and if you like, you can change the speed, too. Create a melody, download it to your microbit and see what happens!

Challenge:
Can you get the robot car to play different melodies for the different directions it can go into (forward, backwards, left and right)?

e. Change speed

If you would like to adjust the speed of the robot car and make it go faster or slower by pressing button A or button B, we need if statements to check if a button has been pressed … What else do we need to include in our code?

f. Follow a track

You can get the car to follow a track of black lines on a piece of paper. The car seems to have sensors underneath…I wonder if they could be useful?