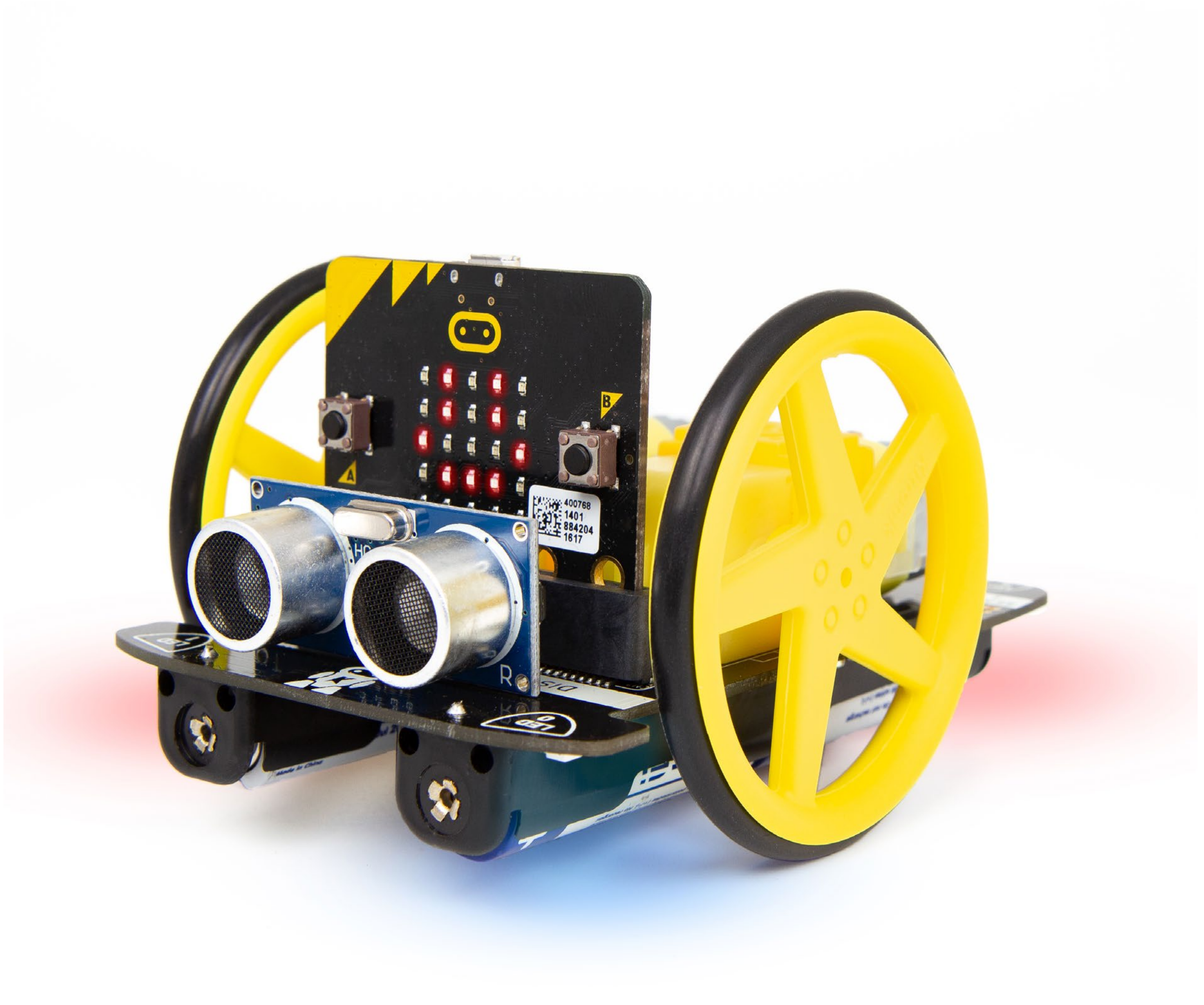
:MOVE MOTOR LIGHTS AND SOUND TUTORIAL

KITRONIK **RESOURCES**



INTRODUCTION

Learn how to use the :MOVE Motor's lights to make headlights and indicators, and then combine with the buzzer to create a police car.

SETTING UP

EQUIPMENT REQUIRED:

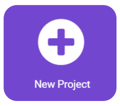
- 1 x BBC micro:bit (www.kitronik.co.uk/5613),
- 1 x :MOVE Motor Buggy (www.kitronik.co.uk/5683)

ADDING IN CUSTOM MAKECODE BLOCKS:

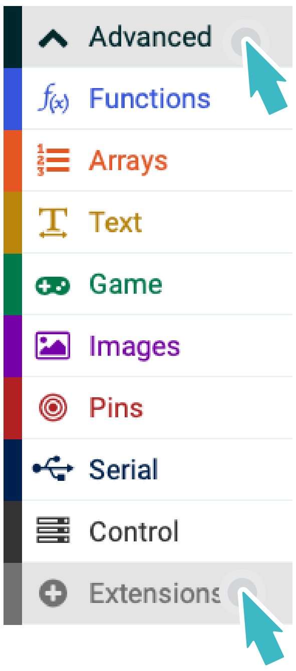
We have made custom coding blocks especially for the :MOVE Motor, which helps to make coding super simple within Microsoft MakeCode.
To add these blocks, follow the steps below:

STEP 1: Bring up the MakeCode Block Editor - (makecode.microbit.org).

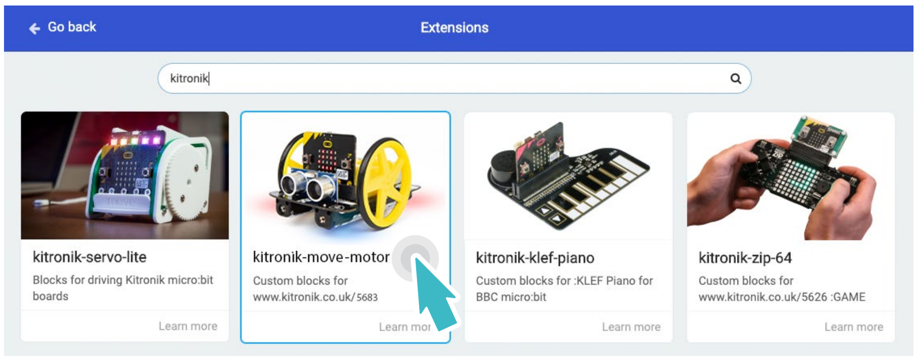
STEP 2: Click 'New Project'.



STEP 3: In the toolbox towards the left of the screen, select the **'Advanced'** section. Additional block categories will appear below.



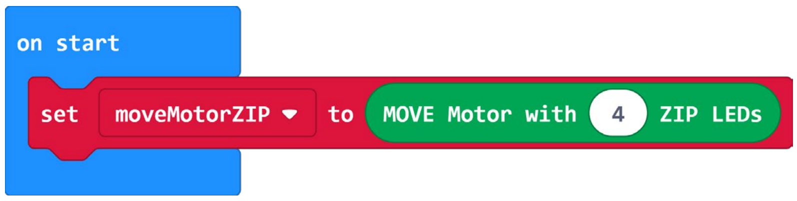
STEP 4: Select **'Extensions'.**
STEP 5: In the pop up's search bar type **'Kitronik'.**
STEP 6: Locate & select the **'kitronik-move-motor'** box.


Go back
Extensions
kitronik
kitronik-servo-lite
Blocks for driving Kitronik micro:bit boards
Learn more
kitronik-move-motor
Custom blocks for www.kitronik.co.uk/5683
Learn mor
kitronik-klef-piano
Custom blocks for :KLEF Piano for BBC micro:bit
Learn more
kitronik-zip-64
Custom blocks for www.kitronik.co.uk/5626 :GAME
Learn more

THE TUTORIAL

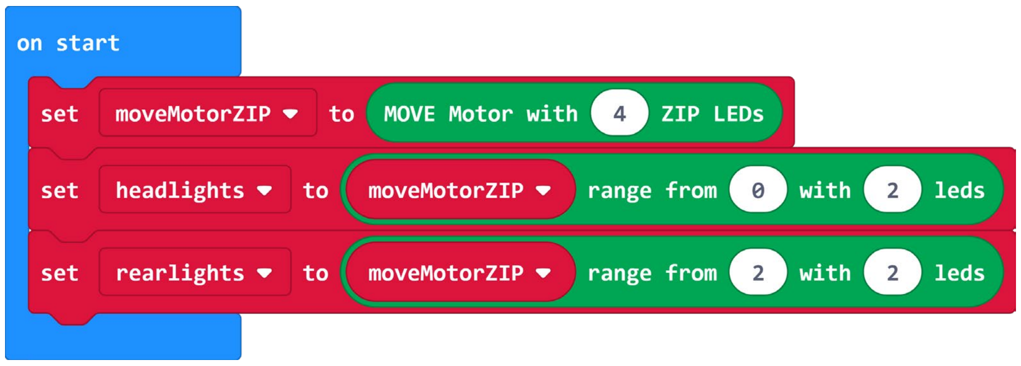
HEADLIGHTS AND REAR LIGHTS

STEP 1: To start with, let's make some headlights and rear lights for :MOVE Motor.
First, place the "set moveMotorZIP to MOVE Motor with 4 ZIP LEDs" block from the "Lights" section of the "MOVE Motor" category into the "on start" block. This sets up the ZIP LEDs on :MOVE Motor, ready to be used.



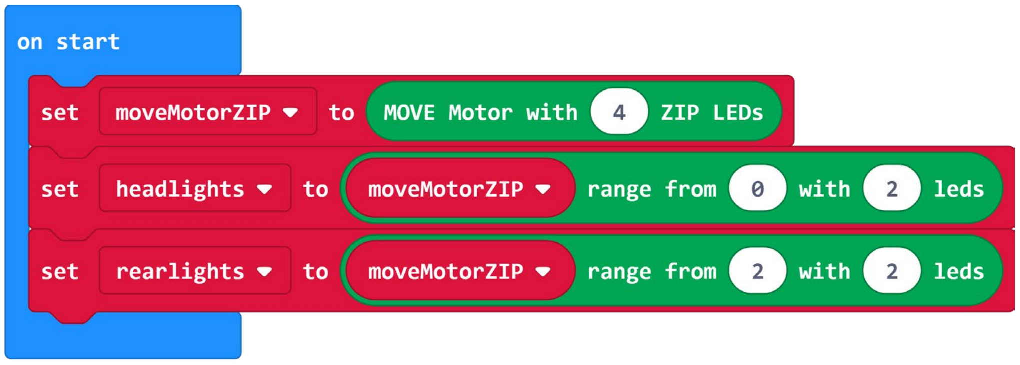
```
on start
    set moveMotorZIP ▾ to MOVE Motor with 4 ZIP LEDs
```

STEP 2: As we're going to have front lights and back lights, we also need to create two variables, "headlights" and "rearlights".
Using the "set variable to" block, make "headlights" equal to a "range from 0 with 2 leds" and "rearlights" equal to a "range from 2 with 2 leds". The seperate ranges mean the two sets of lights can be controlled individually.

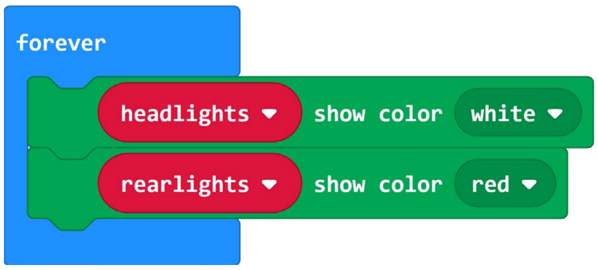


```
on start
    set moveMotorZIP ▾ to MOVE Motor with 4 ZIP LEDs
    set headlights ▾ to moveMotorZIP ▾ range from 0 with 2 leds
    set rearlights ▾ to moveMotorZIP ▾ range from 2 with 2 leds
```

STEP 3: Now we need to set the lights to be their correct colours: white for headlights, red for rear lights. Use the “show colour” block from the “Lights” section of the “MOVE Motor” category, changing the drop-down to select the “headlights” and “rearlights” variables in the different blocks. Place these in the “forever” loop.



```
on start
    set moveMotorZIP ▾ to MOVE Motor with 4 ZIP LEDs
    set headlights ▾ to moveMotorZIP ▾ range from 0 with 2 leds
    set rearlights ▾ to moveMotorZIP ▾ range from 2 with 2 leds
```



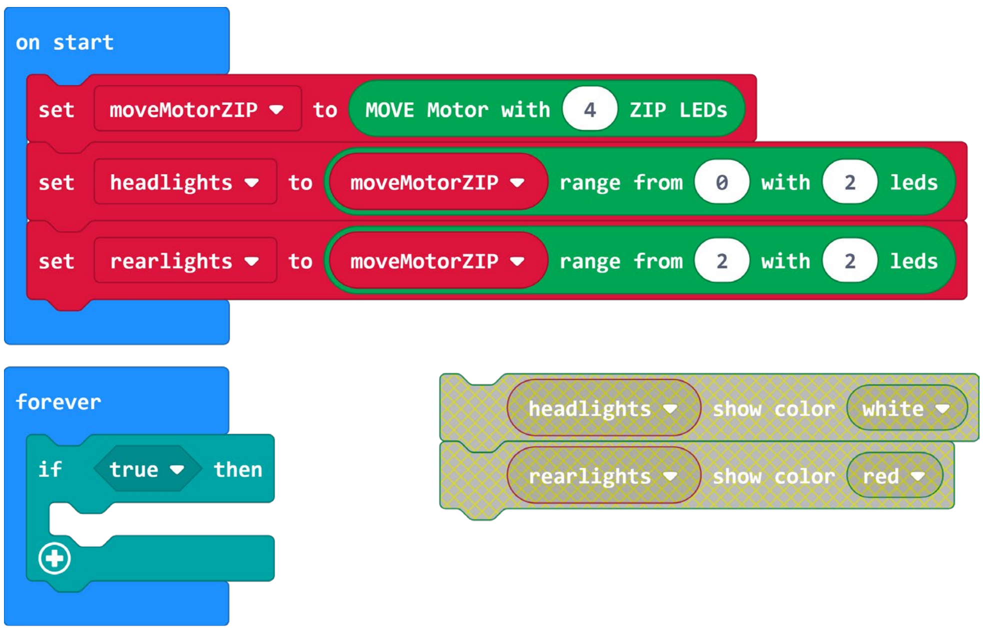
```
forever
    headlights ▾ show color white ▾
    rearlights ▾ show color red ▾
```

STEP 4: If you have a micro:bit connected, click “Download” to transfer your code. Then switch on :MOVE Motor and see the lights turn on!

AUTOMATIC LIGHTS

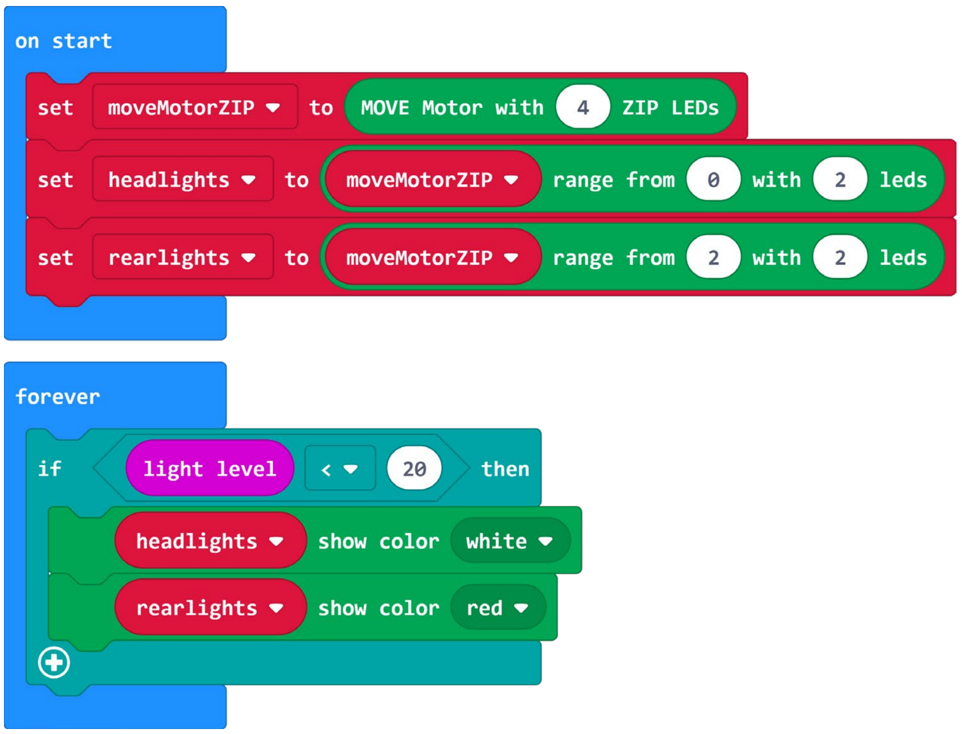
So, :MOVE Motor now has nice bright lights, but on real vehicles, the lights aren’t on all the time - they only need to be used when it’s dark. Lots of modern cars have lights which turn on automatically when a certain light level is reached, and we can do the same with :MOVE Motor...

STEP 5: The micro:bit LED display can also function as a light sensor, which we can then use like a switch to turn :MOVE Motor's lights on and off. Drag the "show colour" blocks out of the "forever" loop, leaving them to one side, and add an "if" block from the "Logic" category.



```
on start
    set moveMotorZIP ▾ to MOVE Motor with 4 ZIP LEDs
    set headlights ▾ to moveMotorZIP ▾ range from 0 with 2 leds
    set rearlights ▾ to moveMotorZIP ▾ range from 2 with 2 leds

forever
    if true ▾ then
    ⊕

headlights ▾ show color white ▾
rearlights ▾ show color red ▾
```

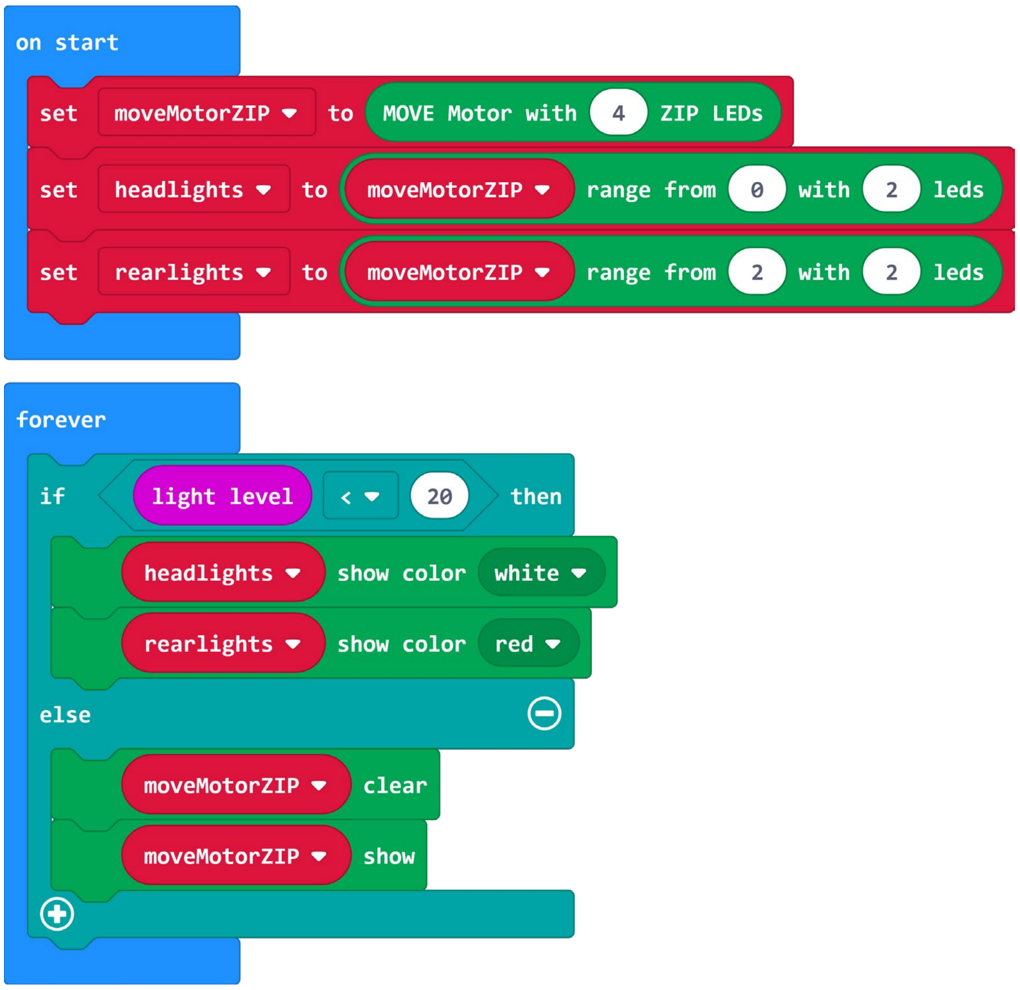
STEP 6: Add a test condition to the "if" statement to check whether "light level" (found in the "Input" category) is "< 20". **Note:** The actual number might need to be varied for your particular light conditions. 20 worked well during testing.
Finally, drag those "show colour" blocks inside the "if" block.



```
on start
    set moveMotorZIP ▾ to MOVE Motor with 4 ZIP LEDs
    set headlights ▾ to moveMotorZIP ▾ range from 0 with 2 leds
    set rearlights ▾ to moveMotorZIP ▾ range from 2 with 2 leds

forever
    if light level < ▾ 20 then
        headlights ▾ show color white ▾
        rearlights ▾ show color red ▾
    ⊕
```

STEP 7: If you have a micro:bit connected, click “Download” to transfer your code.
Try varying the light shining on the micro:bit display and see the :MOVE Motor lights turn on all by themselves.

STEP 8: The lights now turn on when it gets dark enough, but at the moment, they stay on forever. We need to make them turn off again when the light levels are high enough.
Click the “+” icon on the “if” block to add an “else” section, then add a “clear” block followed by a “show” block from the “Lights” section of the “MOVE Motor” category.



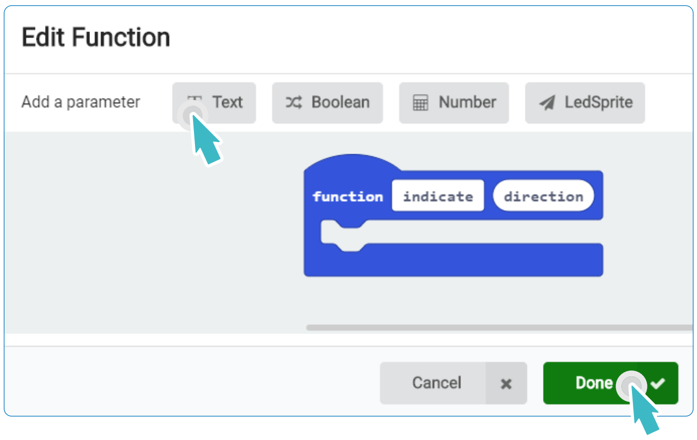
```
on start
    set moveMotorZIP ▾ to MOVE Motor with 4 ZIP LEDs
    set headlights ▾ to moveMotorZIP ▾ range from 0 with 2 leds
    set rearlights ▾ to moveMotorZIP ▾ range from 2 with 2 leds

forever
    if light level < ▾ 20 then
        headlights ▾ show color white ▾
        rearlights ▾ show color red ▾
    else
        moveMotorZIP ▾ clear
        moveMotorZIP ▾ show
```

STEP 9: If you have a micro:bit connected, click “Download” to transfer your code.
Now as you vary the light levels, the :MOVE Motor lights will turn on and off.

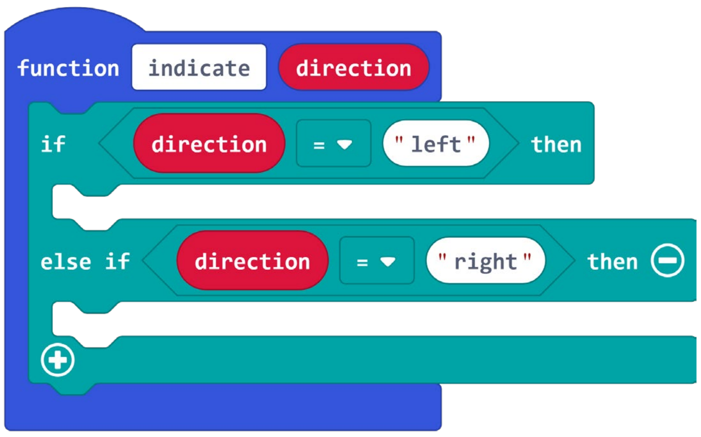
INDICATORS

Now that :MOVE Motor's headlights and rear lights are in place, let's move on to indicating when we turn.

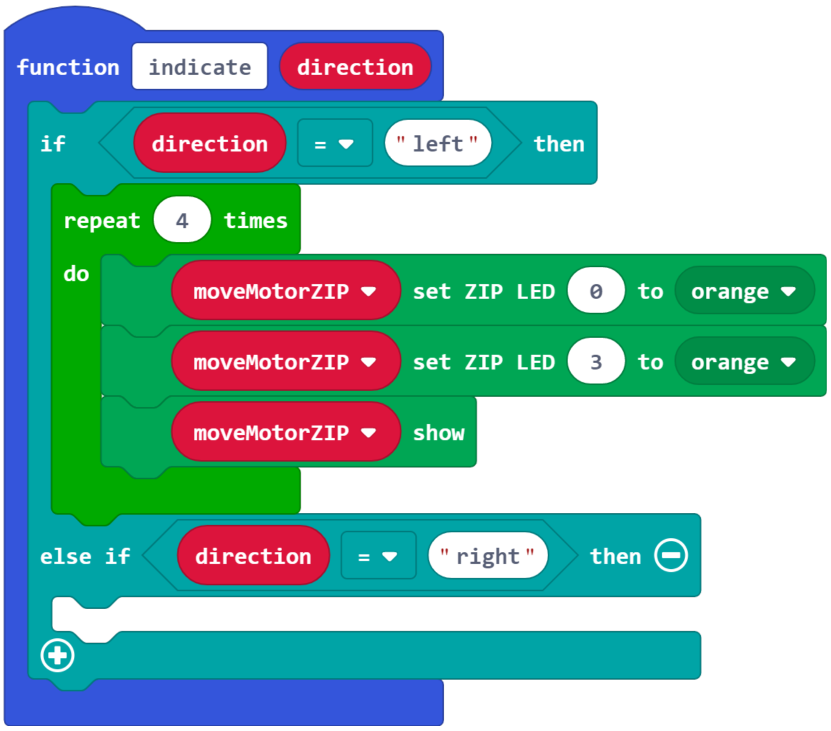
STEP 1: When indicating, we're going to want to turn either left or right. This means we're going to need to use different ZIP LEDs for each action. However, because nearly everything else is the same, we can use a "function" to make things easier.
Click on "Advanced" to reveal more block categories, and then in the "Functions" category, click "Make a Function...".
Add a "Text" paramenter and call it "direction", name the function "indicate" and click "Done".



STEP 2: Add an "if" block to the function, then click the "+" icon twice. This will add an extra "else" and "else if" section, but don't need the "else", so click the "-" icon to remove it.
Now we need some test conditions. In the "if" statement, check if "direction = left" (drag "direction" in from the "function" block). Use the same test block in the "else if" statement, but instead check for "right". **Note:** Make sure to use the text comparison block.

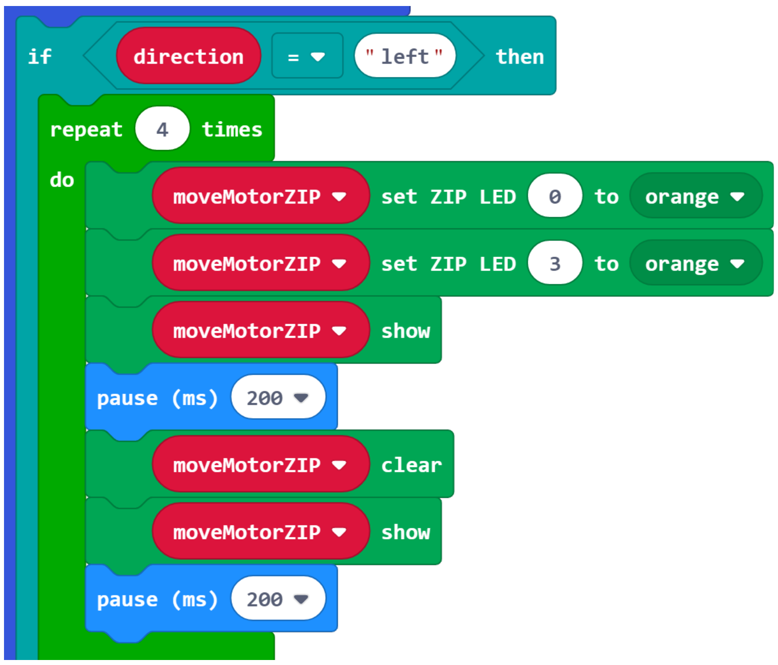

```
function indicate direction
    if direction = "left" then

    else if direction = "right" then

```

STEP 3: Next, we need to set up the left indicator. Put a "repeat 4 times" loop into the "if" statement, then inside, "set ZIP LED" 0 and 3 to be orange. Follow this with a "show" block.



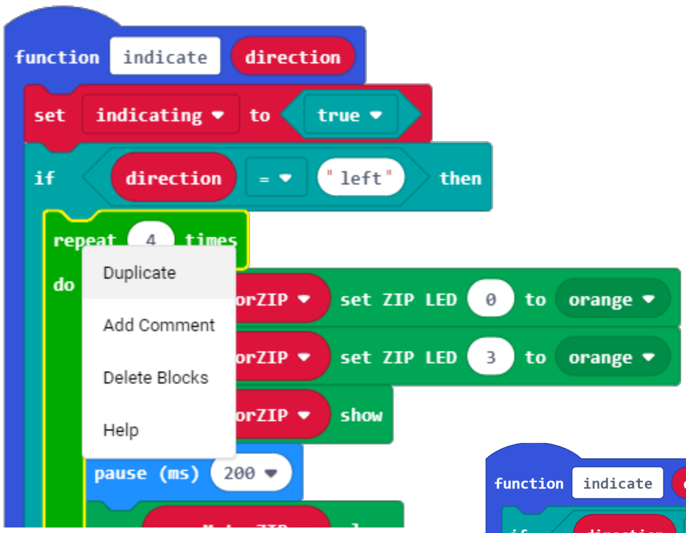
```
function indicate direction
  if direction = "left" then
    repeat 4 times
    do
      moveMotorZIP set ZIP LED 0 to orange
      moveMotorZIP set ZIP LED 3 to orange
      moveMotorZIP show
  else if direction = "right" then
```

STEP 4: If the function is called, the ZIP LEDs on the left side will now turn on. But indicators flash on and off, so we need to add some pauses and turn the LEDs off.
After the "show" add a 200ms "pause", followed by a "clear" and "show" block, and finally another 200ms "pause". The "repeat" loop will make the indicators turn on and off 4 times.

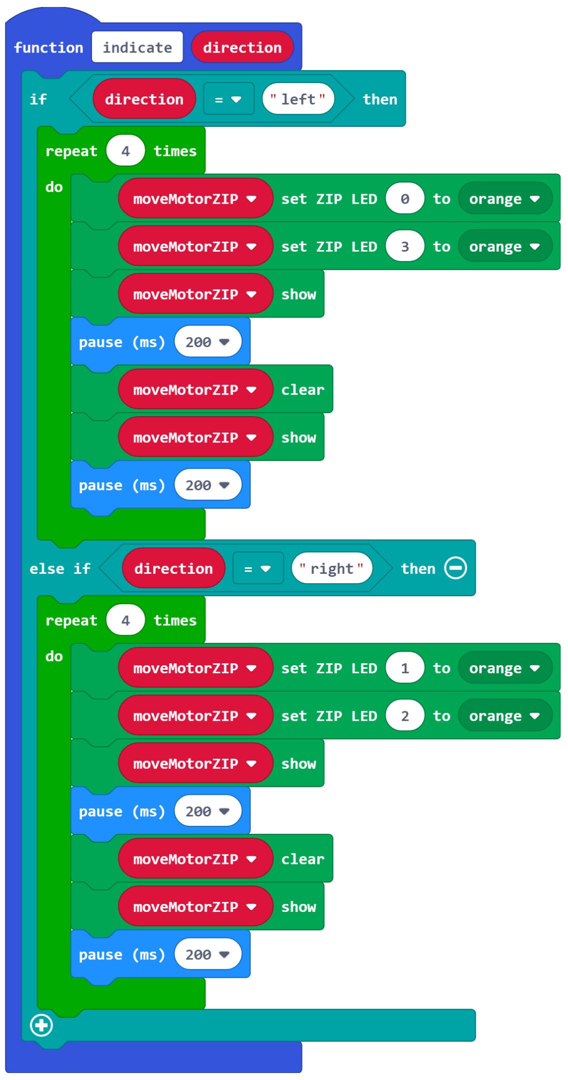


```
if direction = "left" then
  repeat 4 times
  do
    moveMotorZIP set ZIP LED 0 to orange
    moveMotorZIP set ZIP LED 3 to orange
    moveMotorZIP show
    pause (ms) 200
    moveMotorZIP clear
    moveMotorZIP show
    pause (ms) 200
```

STEP 5: The code for the right indicator is almost identical to the left, so right click and duplicate the “repeat” loop and everything inside, then put the new code in the “else if” section. The only thing left to change is the LEDs. To use the right side lights, “set ZIP LED” 1 and 2 to be orange.

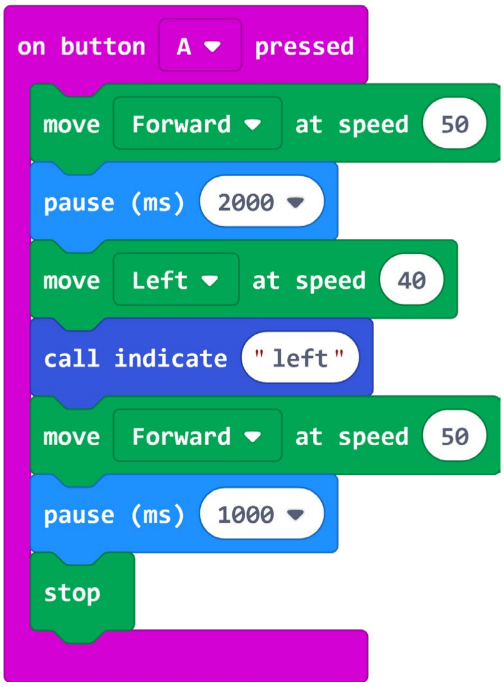


```
function indicate direction
  set indicating to true
  if direction = "left" then
    repeat 4 times
    do
      orZIP set ZIP LED 0 to orange
      orZIP set ZIP LED 3 to orange
      orZIP show
      pause (ms) 200
```

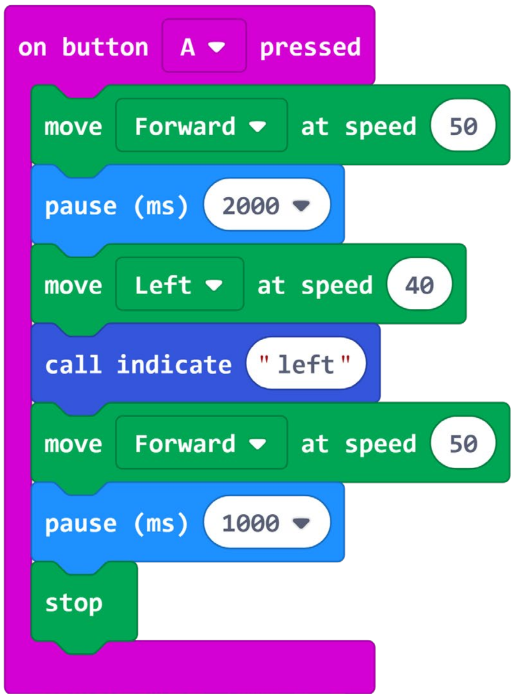


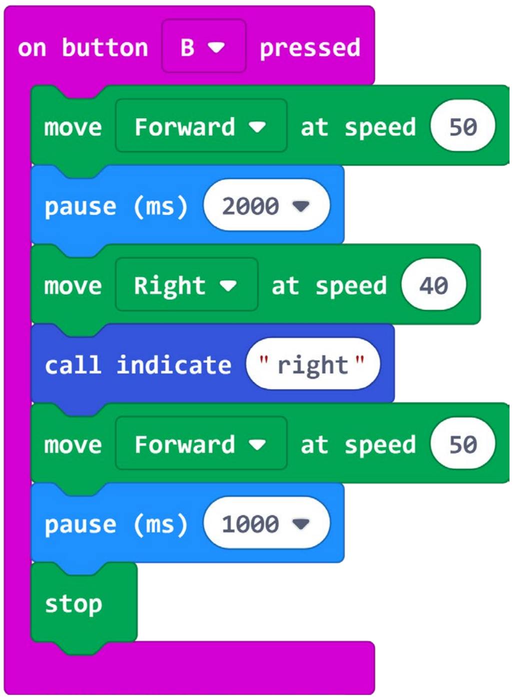
```
function indicate direction
  if direction = "left" then
    repeat 4 times
    do
      moveMotorZIP set ZIP LED 0 to orange
      moveMotorZIP set ZIP LED 3 to orange
      moveMotorZIP show
      pause (ms) 200
      moveMotorZIP clear
      moveMotorZIP show
      pause (ms) 200
  else if direction = "right" then
    repeat 4 times
    do
      moveMotorZIP set ZIP LED 1 to orange
      moveMotorZIP set ZIP LED 2 to orange
      moveMotorZIP show
      pause (ms) 200
      moveMotorZIP clear
      moveMotorZIP show
      pause (ms) 200
```

STEP 6: Now that we've completed the function, it's time to use it. Bring in an "on button A pressed" block and use the "MOVE Motor" blocks to make :MOVE Motor drive forward, turn left and then stop. Immediately after the "move Left at speed" block, add the "call indicate" block from the "Function" category. Type "left" into the function call block.


```
on button A ▾ pressed
    move Forward ▾ at speed 50
    pause (ms) 2000 ▾
    move Left ▾ at speed 40
    call indicate "left"
    move Forward ▾ at speed 50
    pause (ms) 1000 ▾
    stop
```

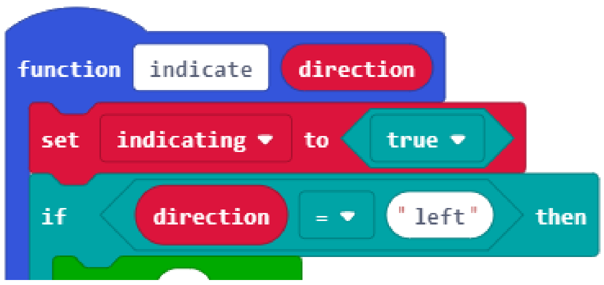
STEP 7: Duplicate the "button A" block. Then, change the drop-down to be "button B", change the "move Left" to "move Right", and the function call to "right".


```
on button A ▾ pressed
    move Forward ▾ at speed 50
    pause (ms) 2000 ▾
    move Left ▾ at speed 40
    call indicate "left"
    move Forward ▾ at speed 50
    pause (ms) 1000 ▾
    stop
```

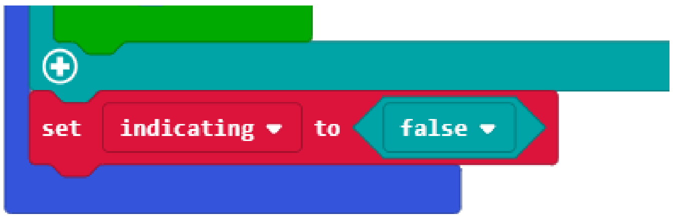

```
on button B ▾ pressed
    move Forward ▾ at speed 50
    pause (ms) 2000 ▾
    move Right ▾ at speed 40
    call indicate "right"
    move Forward ▾ at speed 50
    pause (ms) 1000 ▾
    stop
```

STEP 8: If you have a micro:bit connected, click “Download” to transfer your code.
Press “button A” or “button B” and see :MOVE Motor turn and indicate. However, there might be some issues with our automatic headlights, so there’s a couple more things we need to do…

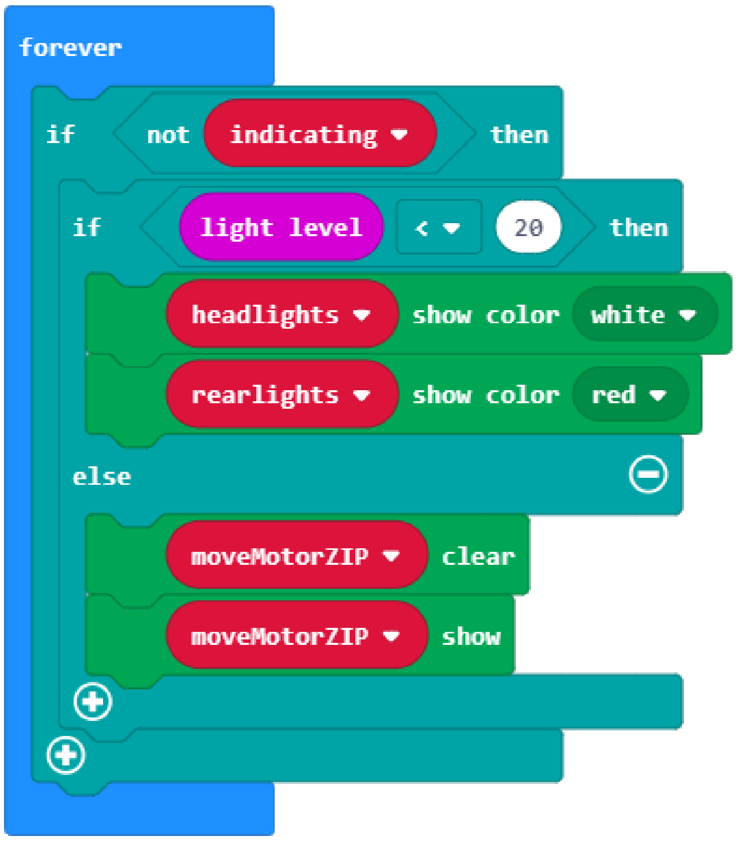
STEP 9: To stop interference from the headlights, we need to temporarily stop the headlights functioning. Create a new variable called “indicating”, set it to be “true” at the start of the “indicate” function, and “false” at the end.
Finally, put everything in the “forever” loop inside another “if” statement checking “if not” “indicating”. This will make sure that when :MOVE Motor is indicating, the headlights won’t try and turn on at the same time.



```
function indicate direction
    set indicating ▾ to true ▾
    if direction = ▾ " left " then
```



```
    ⊕
    set indicating ▾ to false ▾
```



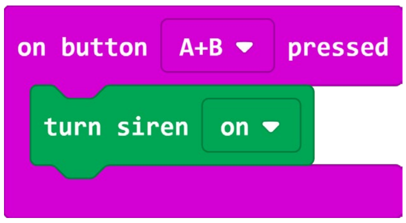
```
forever
    if not indicating ▾ then
        if light level < ▾ 20 then
            headlights ▾ show color white ▾
            rearlights ▾ show color red ▾
        else
            moveMotorZIP ▾ clear
            moveMotorZIP ▾ show
        ⊕
    ⊕
```

STEP 10: If you have a micro:bit connected, click “Download” to transfer your code.
Try it out. Now there should be no problem with the headlights or the indicators.

POLICE CAR

We're now able to control :MOVE Motors lights to be headlights, rear lights and indicators. In this last section, we're going to combine the lights with :MOVE Motor's buzzer to make a police car.

STEP 1: Just like with the indicators, the police car "mode" is going to be activated with button presses. Pull in an "on button A+B pressed" block, and then add the "turn siren on" block from the "Sounds" section of the "MOVE Motor" category. This will make the buzzer emit a continuous siren sound until it is switched off.

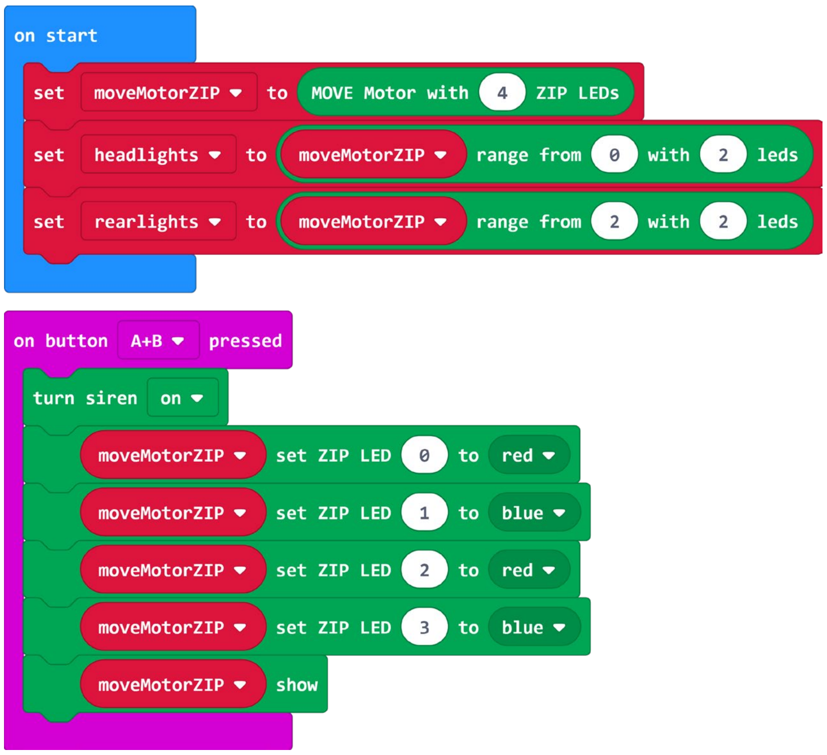


```
on button A+B ▾ pressed
    turn siren on ▾
```

STEP 2: That's the sound sorted, now for some police style lights.
Use 4 of the "set ZIP LED # to colour" blocks to set:

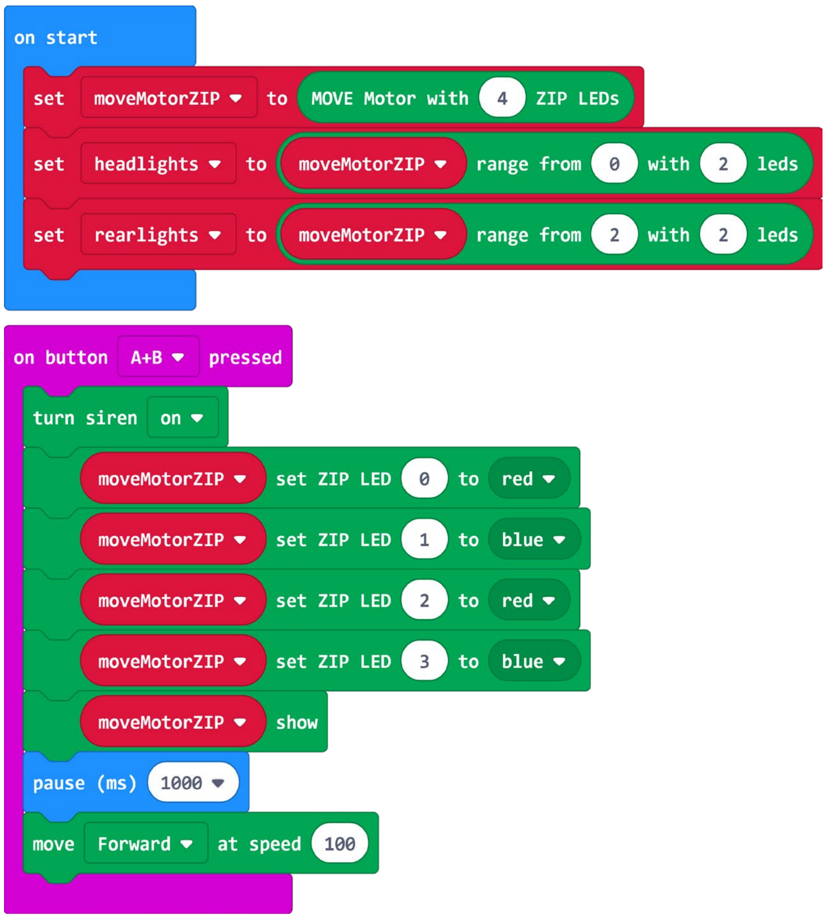
- LED 0 to "red"
- LED 1 to "blue"
- LED 2 to "red"
- LED 3 to "blue"

Follow these with a "show" block to make them display.



```
on start
    set moveMotorZIP ▾ to MOVE Motor with 4 ZIP LEDs
    set headlights ▾ to moveMotorZIP ▾ range from 0 with 2 leds
    set rearlights ▾ to moveMotorZIP ▾ range from 2 with 2 leds

on button A+B ▾ pressed
    turn siren on ▾
    moveMotorZIP ▾ set ZIP LED 0 to red ▾
    moveMotorZIP ▾ set ZIP LED 1 to blue ▾
    moveMotorZIP ▾ set ZIP LED 2 to red ▾
    moveMotorZIP ▾ set ZIP LED 3 to blue ▾
    moveMotorZIP ▾ show
```

STEP 3: Next, put in a 1 second "pause", and then start :MOVE Motor moving forward at 100% speed. If you have a micro:bit connected, click "Download" to transfer your code.
Press "buttons A+B" together and see :MOVE Motor drive away with siren going and red and blue lights.

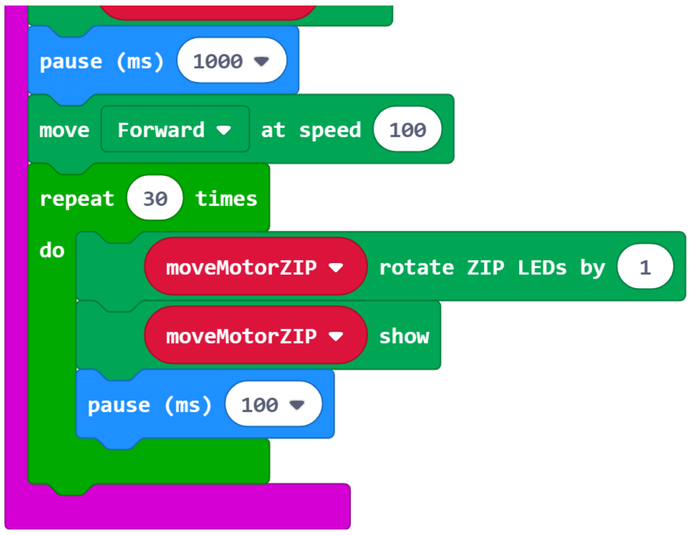


```
on start
    set moveMotorZIP to MOVE Motor with 4 ZIP LEDs
    set headlights to moveMotorZIP range from 0 with 2 leds
    set rearlights to moveMotorZIP range from 2 with 2 leds

on button A+B pressed
    turn siren on
    moveMotorZIP set ZIP LED 0 to red
    moveMotorZIP set ZIP LED 1 to blue
    moveMotorZIP set ZIP LED 2 to red
    moveMotorZIP set ZIP LED 3 to blue
    moveMotorZIP show
    pause (ms) 1000
    move Forward at speed 100
```

FLASHING LIGHTS

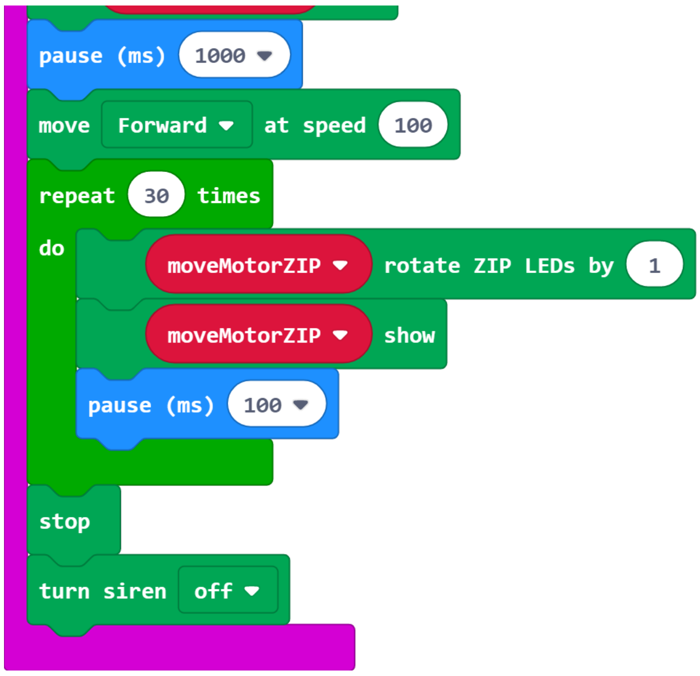
Now, that was ok, but lights on police cars are usually flashing, and you may have noticed that :MOVE Motor won't stop moving forward. There's a couple more bit to do to make the police car just right.

STEP 4: Pull in another “repeat 4 times” loop, but change the repeat number to be 30. Inside this loop add a “rotate ZIP LEDs by 1” block, followed by a “show” and then a 100ms “pause”. As the colours rotate around the 4 LEDs, this will make them look they’re flashing between red and blue.



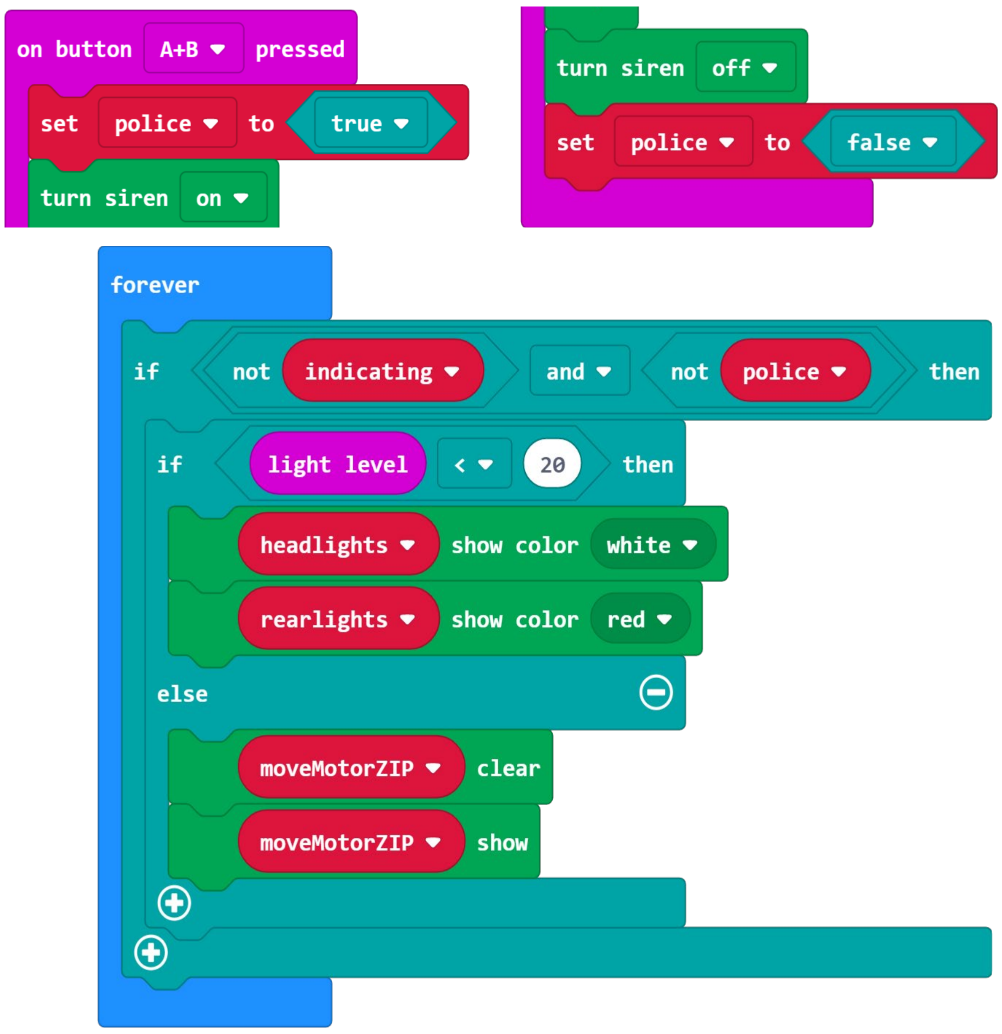
```
pause (ms) 1000
move Forward at speed 100
repeat 30 times
do
    moveMotorZIP rotate ZIP LEDs by 1
    moveMotorZIP show
    pause (ms) 100
```

STEP 5: After the lights have completed their 30 flashes, let’s “stop” :MOVE Motor and “turn siren off”.



```
pause (ms) 1000
move Forward at speed 100
repeat 30 times
do
    moveMotorZIP rotate ZIP LEDs by 1
    moveMotorZIP show
    pause (ms) 100
stop
turn siren off
```

STEP 6: When you tested the police car lights and siren earlier, you may have noticed that the headlights and rearlights were causing problems again, so we need to temporarily stop the headlights functioning.
Create a new variable called "police", set it to be "true" at the start of the "on button A+B pressed" block, and "false" at the end. Finally, in the "forever" loop first "if" statement, change the check to be "if not indicating and not police". This will make sure that when :MOVE Motor is indicating or in police car mode, the headlights won't try and turn on at the same time.



```
on button A+B pressed
    set police to true
    turn siren on
    ...
    turn siren off
    set police to false

forever
    if not indicating and not police then
        if light level < 20 then
            headlights show color white
            rearlights show color red
        else
            moveMotorZIP clear
            moveMotorZIP show
```

STEP 7: CODING COMPLETE! If you have a micro:bit connected, click "Download" to transfer your code. Try pressing "buttons A+B" now. Watch the siren turn on, the lights flash and :MOVE Motor drive forward, all with no headlight inteference.

For any further queries or support, please visit the Kitronik website: www.kitronik.co.uk/5683

Or get in touch:

Telephone	+44 (0) 115 970 4243
Sales email:	sales@kitronik.co.uk
Tech Support email:	support@kitronik.co.uk
Web:	www.kitronik.co.uk

kitronik.co.uk/twitter
kitronik.co.uk/facebook
kitronik.co.uk/youtube
kitronik.co.uk/instagram

Designed & manufactured in the UK by Kitronik

RoHS