



Armstrong

School Program 2023-2024

Lesson 5



Armstrong

entertainment meets education

REGISTER NOW



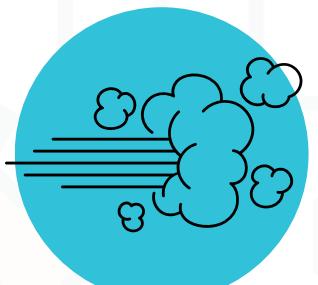
<https://armstrongedu.com/>



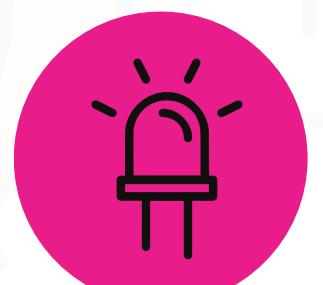
Lesson Content



Variables



Control speed



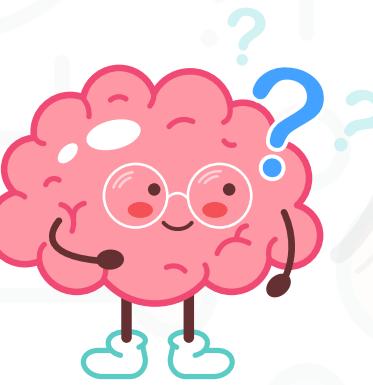
Light dimmer

Think

What is a constant?



Think



What is a constant?

A constant is a word or a symbol that represents a fixed value that does not change.

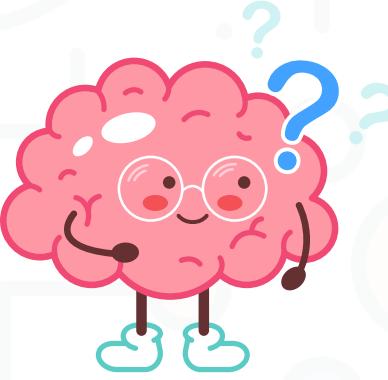
For example, the number 10 is a constant, because it always means the same thing.

Think

What is a variable?



Think



What is a variable?

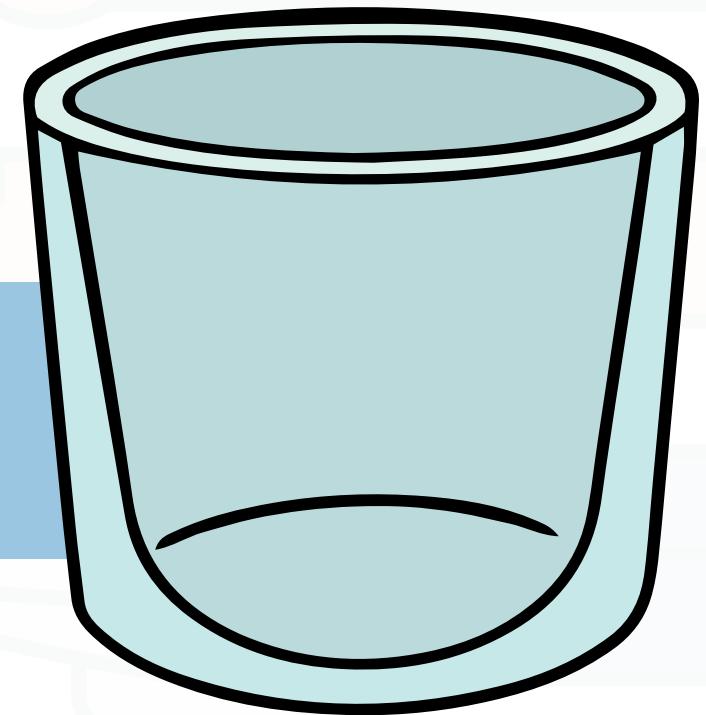


Constants are different from variables, which are words or symbols that can change their values depending on the situation.

For example, the word age is a variable, because it can mean different things for different people. The letter x is also a variable, because it can represent different numbers in different equations.

Analogy ≡

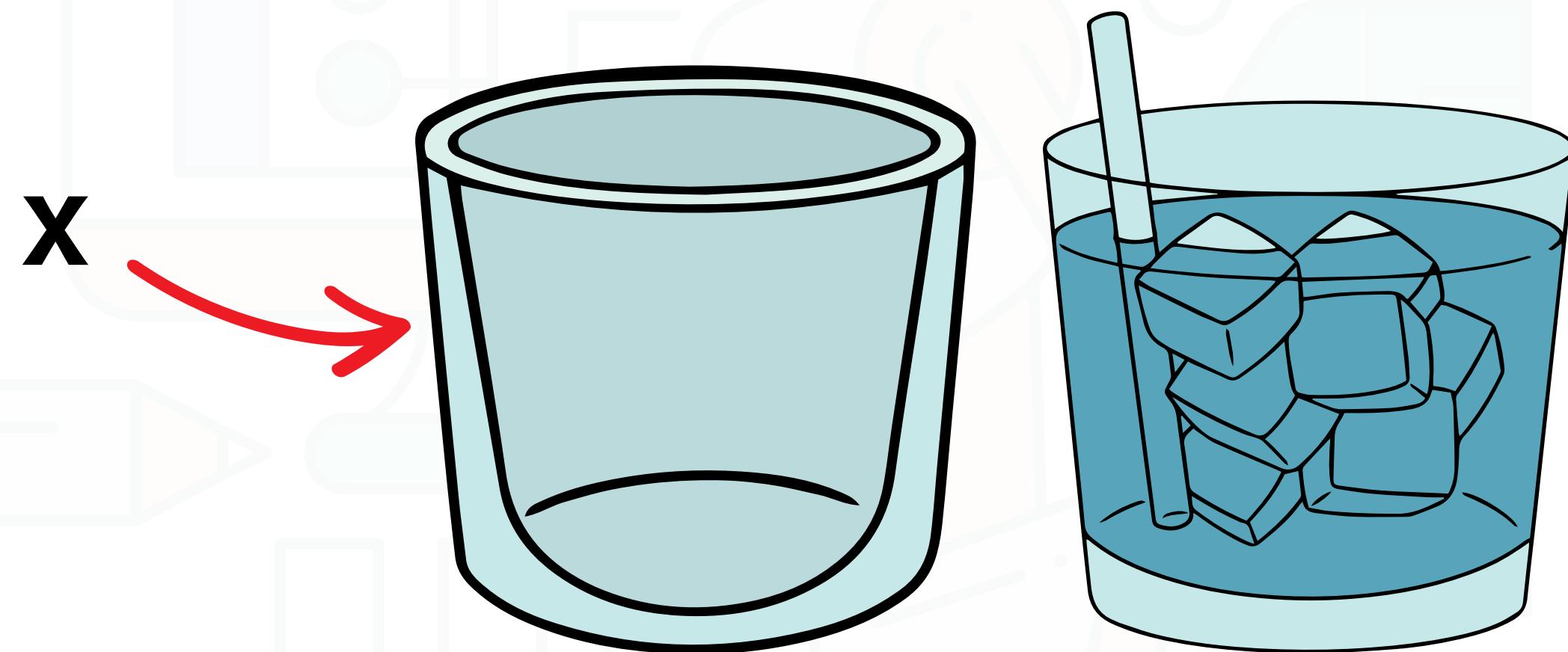
Imagine you have an empty cup.



Think about cups as containers that can hold different things, such as liquids, candies, or coins.

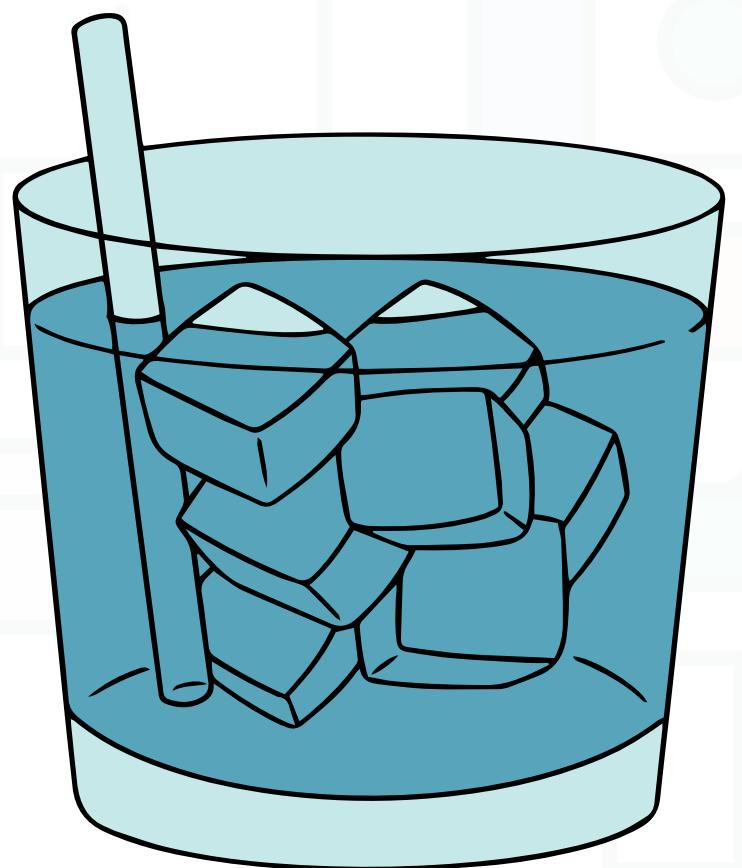
Analogy

For example, you can have a cup named **x** and fill it with water. Then you can say that **x** is a variable that has the value of water.

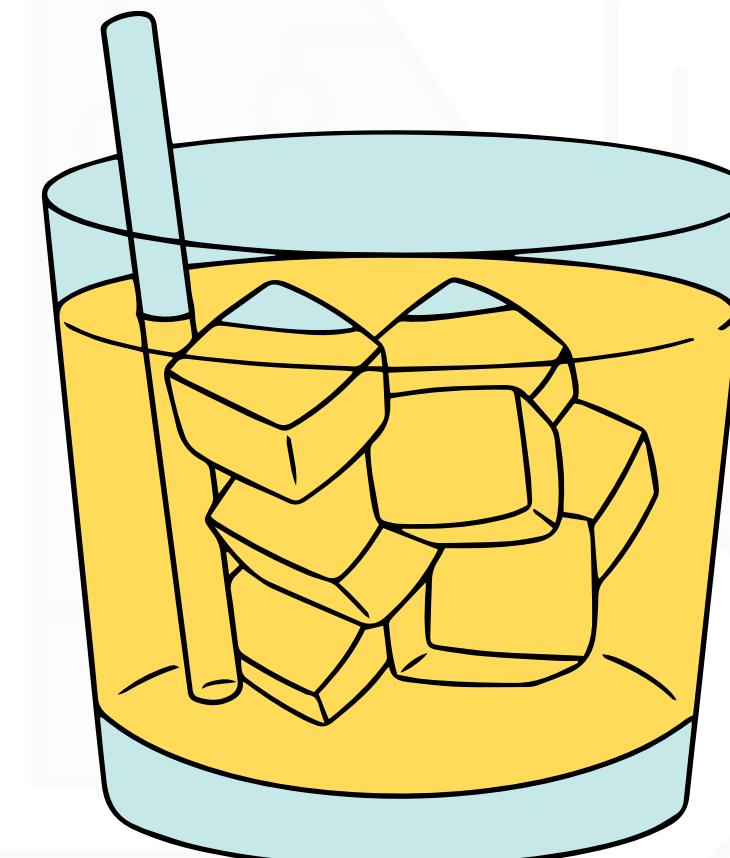
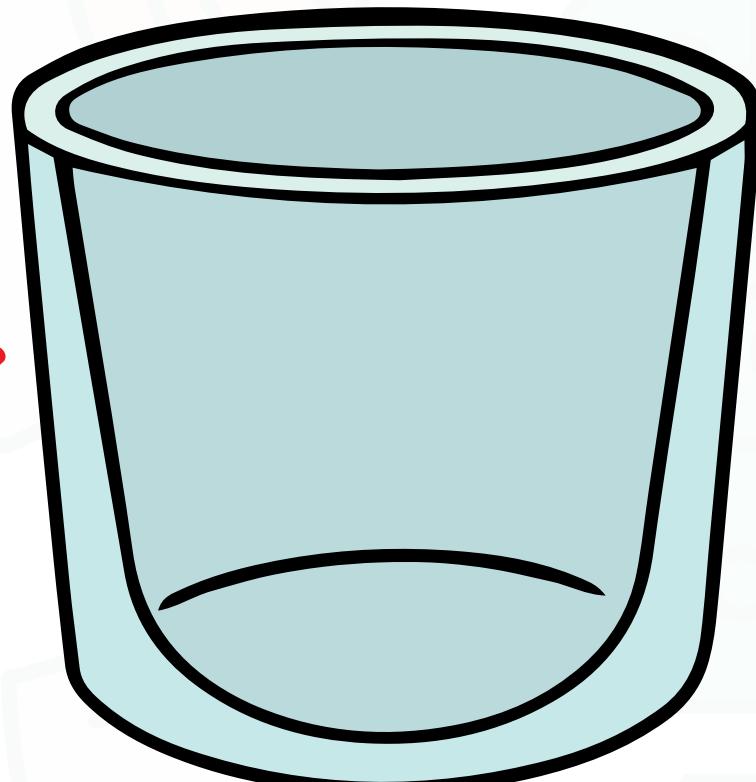


Analogy

You can also change the value of x by emptying the cup and filling it with juice. Then you can say that x is still a variable, but now it has a different value of juice.

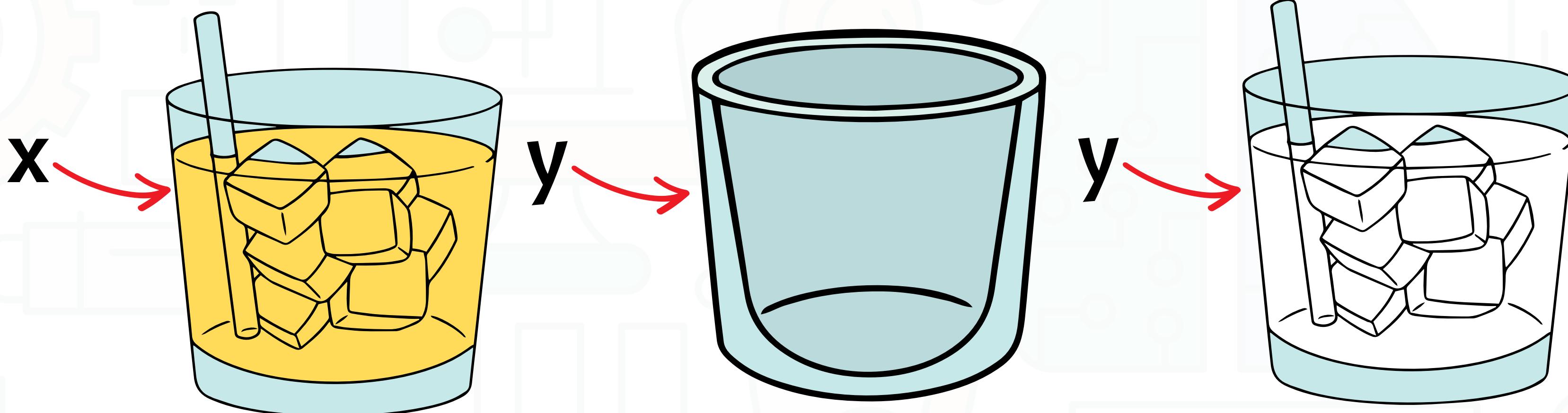


X



Analogy

You can also have another cup named y and fill it with milk. Then you can say that y is another variable that has the value of milk.



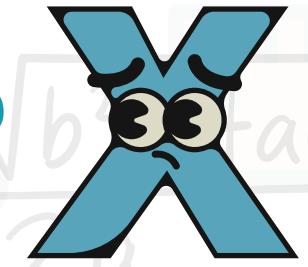
Analogy

You can also compare the values of x and y by looking at the cups and seeing which one has more or less liquid. You can also add or subtract the values of x and y by pouring the liquids from one cup to another.

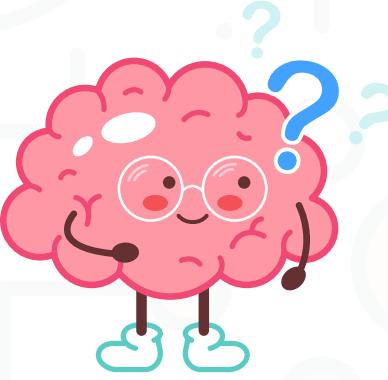


Think

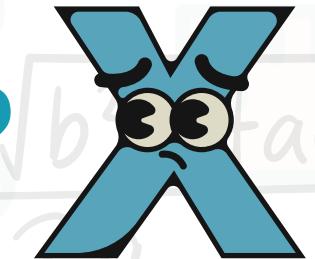
Why are variables important in coding?



Think



Why are variables important in coding?



Variables are useful in coding because they can help us store and manipulate data that can vary based on the program's requirements.

How to create variables on mBlock



makeblock | mBlock | File | Edit | Robot_2 | Save | Publish | Local file | Courses | Tutorials | Feedback | ... | Python Editor

Arduino Uno: MotorB1 0
Arduino Uno: MotorFRS 0
Arduino Uno: MotorFLS 0
Arduino Uno: MotorBRS 0
Arduino Uno: MotorBLS 0
Arduino Uno: MotorF1 0
Arduino Uno: MotorF2 0
Arduino Uno: MotorF3 0
Arduino Uno: MotorF4 0
Arduino Uno: MotorB2 0
Arduino Uno: MotorB3 0
Arduino Uno: MotorB4 0

Pin
serial port
Data
Sensor
Events
Control
Operators
Variables
My Blocks
extension

read digital pin 9
read analog pin (A) 0
read pulse pin 13 timeout 2000
set digital pin 9 output as high
set PWM 5 output as 0
play pin 9 with note C4 for 1000
set servo pin 9 angle as 90
suspend pin 2 mode rising edge
do not suspend pin 2

Blocks | Arduino C | </>

Devices | Sprites | Background | Arduino ... | add | Connect your device | How to use device? | Mode Switch | Upload | Live | Connect

The screenshot shows the mBlock 4.0 software interface. The top navigation bar includes 'makeblock | mBlock', 'File', 'Edit', 'Robot_2' (active), 'Save', 'Publish', 'Local file', 'Courses', 'Tutorials', 'Feedback', '...', 'Python Editor', and a GitHub icon. The left sidebar features a 'Devices' tab with an 'Arduino ...' entry and an 'add' button, along with tabs for 'Sprites' and 'Background'. A central workspace displays various Scratch-style blocks for controlling an Arduino Uno, such as reading pins, setting digital and PWM outputs, playing sounds, and controlling servos. A vertical palette on the right lists categories like Pin, serial port, Data, Sensor, Events, Control, Operators, Variables, My Blocks, and extension, each with corresponding blocks. A large empty grid area for placing blocks is in the center-right.

How to create variables on mBlock



Step 1: Choose Variables category

The screenshot shows the mBlock 4.0 software interface. On the left, there's a sidebar with sections for Arduino Uno pins, Devices (selected), Sprites, and Background. Below these are buttons for Mode Switch, Upload, Live, Connect, and extension. A red arrow points from the text "Click on Variables" to the "Variables" button in the sidebar, which is highlighted with a red circle. The main workspace shows a grid of blocks categorized by color: Pin (blue), serial port (light green), Data (purple), Sensor (light blue), Events (yellow), Control (orange), Operators (green), and Variables (orange). The Variables category contains blocks like "read digital pin", "set digital pin", "play pin", and "suspend pin". The top menu bar includes File, Edit, Save, Publish, Local file, Courses, Tutorials, Feedback, Python Editor, and a GitHub icon.

Click on
Variables

How to create variables on mBlock



Step 2: Create new variable

The screenshot shows the mBlock software interface. On the left, there's a palette with various blocks categorized by color: Pin (blue), serial port (green), Data (purple), Sensor (light blue), Events (yellow), Control (orange), Operators (green), Variables (orange), My Blocks (pink), and extension (grey). A red circle highlights the 'Variables' category, and a large red arrow points from it to the text 'Click on make a variable' located in the center-right area of the interface.

makeblock | mBlock | File | Edit | Robot 2 | Save | Publish | Local file | Courses | Tutorials | Feedback | Python Editor

Blocks | Arduino C

Make a Variable

- MotorB1
- MotorB2
- MotorB3
- MotorB4
- MotorBLS
- MotorBRS
- MotorF1
- MotorF2
- MotorF3
- MotorF4
- MotorFLS
- MotorFRS
- previousL
- previousR

Pin

serial port

Data

Sensor

Events

Control

Operators

Variables

My Blocks

extension

Arduino Uno: MotorB1 0

Arduino Uno: MotorFRS 0

Arduino Uno: MotorFLS 0

Arduino Uno: MotorBRS 0

Arduino Uno: MotorBLS 0

Arduino Uno: MotorF1 0

Arduino Uno: MotorF2 0

Arduino Uno: MotorF3 0

Arduino Uno: MotorF4 0

Arduino Uno: MotorB2 0

Arduino Uno: MotorB3 0

Arduino Uno: MotorB4 0

Devices Sprites Background

Arduino ...

+ add

Connect your device

How to use device?

Mode Switch ?

Upload Live

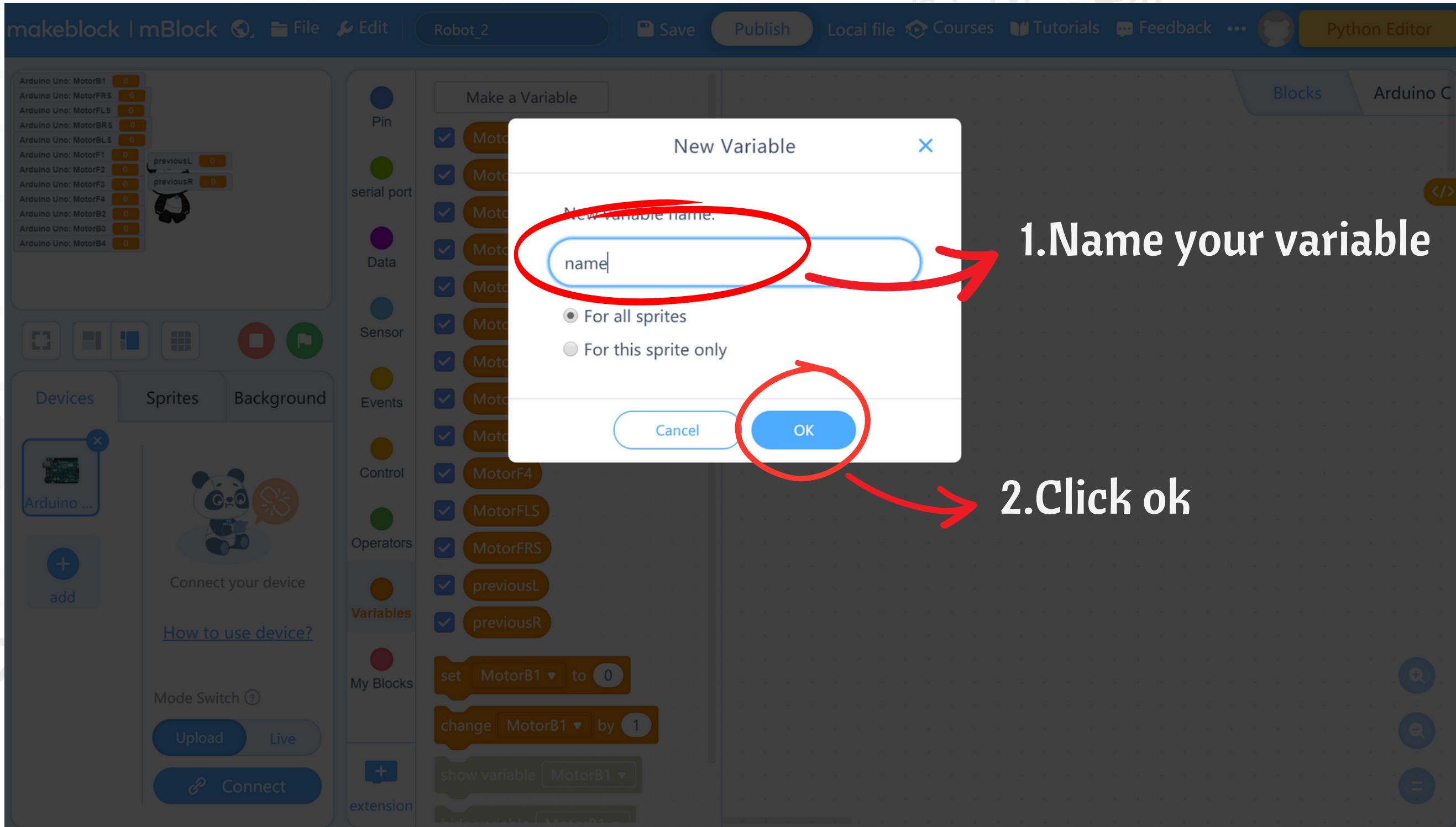
Connect

Click on make a variable

How to create variables on mBlock



Step 2: Create new variable



1. Name your variable

2. Click ok

How to create variables on mBlock



Step 3: Where can you find the new variable?

The screenshot shows the mBlock 4.0 software interface. On the left, there's a workspace with a robot sprite and some blocks. Below it, a sidebar shows device connections and a help section. In the center, a 'Blocks' palette is open under the 'Arduino C' tab. A red arrow points from the text 'After creating a new variable, you will find it here' to the 'Variables' section of the palette. This section contains several orange variable blocks, with the 'name' block highlighted by a red oval.

makeblock | mBlock | File Edit | Robot_2 | Save Publish Local file Courses Tutorials Feedback ... Python Editor

Pin
serial port
Data
Sensor
Events
Control
Operators
Variables
My Blocks
extension

Make a Variable

- MotorB1
- MotorB2
- MotorB3
- MotorB4
- MotorBLS
- MotorBRS
- MotorF1
- MotorF2
- MotorF3
- MotorF4
- MotorFLS
- MotorFRS
- name
- previousL
- previousR

set MotorB1 to 0
change MotorB1 by 1
show variable MotorB1

After creating a new variable, you will find it here

How to create variables on mBlock



Step 4: How to use variables?

The screenshot shows the mBlock 4.0 software interface. On the left, the Devices panel lists an Arduino Uno device. The main workspace shows a script editor with two blocks: "set name to 0" and "change name by 1". A red circle highlights the variable "name" in the first block. To the right, the Variables palette lists several variables: MotorB1, MotorB2, MotorB3, MotorB4, MotorBLS, MotorBRS, MotorF1, MotorF2, MotorF3, MotorF4, MotorFLS, MotorFRS, name, previousL, and previousR. A callout text points to the "name" variable in the script editor.

You can use this block to access your variable anywhere in your code

name

set name to 0

change name by 1

Variables

MotorB1
MotorB2
MotorB3
MotorB4
MotorBLS
MotorBRS
MotorF1
MotorF2
MotorF3
MotorF4
MotorFLS
MotorFRS
name
previousL
previousR

How to create variables on mBlock



Step 4: How to use variables?

The screenshot shows the mBlock 4.0 software interface. On the left, the Devices panel lists an Arduino Uno device. The main workspace shows a script editor with two blocks: "set name to 0" and "change name by 1". A red circle highlights the "set name to 0" block. To the right, the Variables palette lists various variables: MotorB1, MotorB2, MotorB3, MotorB4, MotorBLS, MotorBRS, MotorF1, MotorF2, MotorF3, MotorF4, MotorFLS, MotorFRS, name, previousL, and previousR. The "Variables" category is currently selected.

You can use this block to set your variable with an initial value or change your variable's value anywhere in your code

How to create variables on mBlock



Step 4: How to use variables?

The screenshot shows the mBlock 4.0 software interface. On the left, the device panel displays an Arduino Uno connected to pins MotorB1 through MotorB4. The main workspace shows a script with two blocks: "set name to 0" and "change name by 1". A red arrow points to the second block, which is highlighted with a red circle. To the right, the variable editor lists variables like MotorB1 through MotorB4, MotorF1 through MotorF4, and various sensor and control variables. The top navigation bar includes tabs for "Robot_2", "Save", "Publish", "Local file", "Courses", "Tutorials", "Feedback", and "Python Editor".

You can use this block to increase or decrease your variable's value

Let's try it on our robot

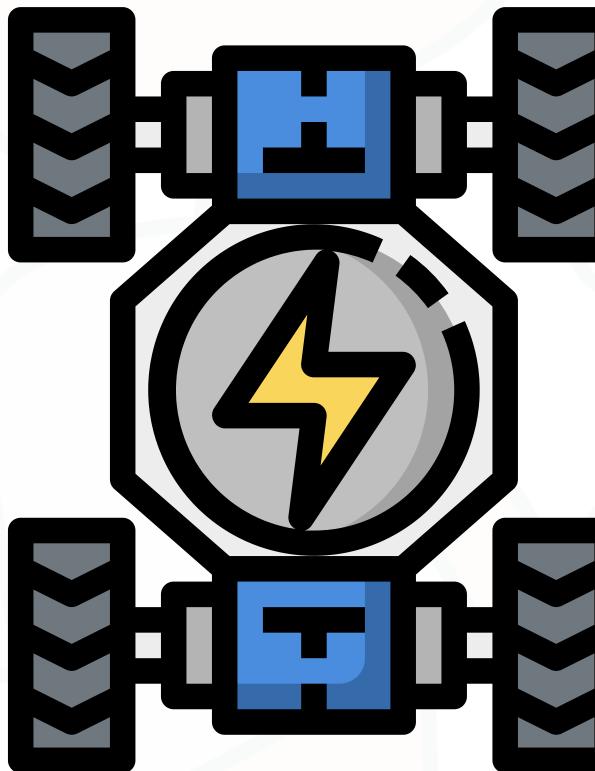


Write code to increase speed gradually

Enable -> 6

Input 1 -> 11

Input2 -> 12



Enable -> 10

Input 1 -> 15

Input2 -> 16

Enable -> 5

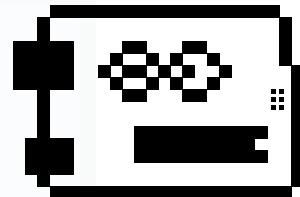
Input 1 -> 7

Input2 -> 8

Enable -> 9

Input 1 -> 13

Input2 -> 14



Increase speed gradually



Step 1: Create variable called speed

The screenshot shows the mBlock v5.4 software interface. In the center, a dialog box titled "New Variable" is open, prompting the user to enter a new variable name. The input field contains the text "speed". Below the input field are two radio buttons: "For all sprites" (which is selected) and "For this sprite only". At the bottom of the dialog are "Cancel" and "OK" buttons. In the background, the mBlock interface is visible, featuring a palette of blocks on the left and a workspace on the right. The "Variables" category is currently selected in the palette, showing various variable-related blocks like "MotorB1", "MotorFLS", "MotorFRS", "previousL", "previousR", and "speed". A "Devices" panel on the left shows an Arduino Uno connected. A central area displays a panda sprite with the text "Connect your device" and "How to use device?".

Increase speed gradually



Step 2: Import forward code

The screenshot shows the mBlock 4 software interface. On the left, there's a workspace with a panda sprite, a toolbar with various icons, and a sidebar with sections for Devices (Arduino selected), Sprites, and Background. A message says "Connect your device". Below it, there are links for "How to use device?", "Mode Switch", "Upload", "Live", and "Connect". On the right, the main workspace shows a script starting with a "when Arduino starts up" event. This event triggers a series of "set PWM" blocks for pins 6 through 14, followed by "set digital pin" blocks for pins 11 through 16, alternating between high and low states. The "Blocks" tab is selected at the top right.



ARMSTRONG
ENTERTAINMENT MEETS EDUCATION

Increase speed gradually



Step 2: Import forward code

The screenshot shows the mBlock 4 software interface. On the left, there's a workspace with a panda sprite, a toolbar with various icons, and a sidebar with sections for Devices (Arduino selected), Sprites, and Background. A message says "Connect your device". Below that, there are links for "How to use device?", "Mode Switch", "Upload", "Live", and "Connect". On the right, the main area has two tabs: "Blocks" (selected) and "Arduino C". Under "Blocks", there's a list of blocks categorized by color: Pin (blue), serial port (green), Data (purple), Sensing (light blue), Events (yellow), Control (orange), Operators (green), Variables (orange), and My Blocks (pink). A specific "repeat until" block is highlighted. Under "Arduino C", a script titled "when Arduino starts up" is shown, consisting of 15 "set PWM" and "set digital pin" blocks, each with parameters like pin number, value, and output mode (e.g., "output as 0", "high", "low").



ARMSTRONG
ENTERTAINMENT MEETS EDUCATION

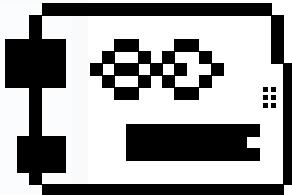
Increase speed gradually



Step 3: Set speed with 70

The screenshot shows the mBlock 4 interface. On the left, there's a workspace with a panda sprite, a device panel with an Arduino connection, and a help section about connecting a device. The center has a block palette with categories like Pin, serial port, Data, Sensing, Events, Control, Operators, Variables, and My Blocks. The right side is the script editor where a script is being built:

```
when Arduino starts up
  set speed to 70
  set PWM 6 output as 0
  set PWM 10 output as 0
  set PWM 5 output as 0
  set PWM 9 output as 0
  set digital pin 11 output as high
  set digital pin 12 output as low
  set digital pin 15 output as high
  set digital pin 16 output as low
  set digital pin 7 output as high
  set digital pin 8 output as low
  set digital pin 13 output as high
  set digital pin 14 output as low
```



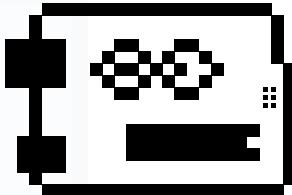
Increase speed gradually



Step 4: Set PWM pins to speed

The screenshot shows the mBlock 4 programming environment. On the left, there's a workspace with a panda sprite and a variable named "speed" set to 0. Below the workspace are tabs for Devices, Sprites, and Background. Under Devices, there's an Arduino connection status. On the right, the main area contains a script:

```
when Arduino starts up
  set speed to 70
  repeat
    set PWM 6 output as speed
    set PWM 10 output as speed
    set PWM 5 output as speed
    set PWM 9 output as speed
    set digital pin 11 output as high
    set digital pin 12 output as low
    set digital pin 15 output as high
    set digital pin 16 output as low
    set digital pin 7 output as high
    set digital pin 8 output as low
    set digital pin 13 output as high
    set digital pin 14 output as low
  end
```



ARMSTRONG
ENTERTAINMENT MEETS EDUCATION

Increase speed gradually



Step 5: Increase speed by 10 every 3 seconds

The screenshot shows the mBlock 4 programming environment. On the left, there's a workspace with a panda sprite and a variable named "speed" set to 0. Below the workspace are tabs for Devices, Sprites, and Background. A sidebar on the left lists categories like Pin, serial port, Data, Sensing, Events, Control, Operators, Variables, My Blocks, and extension. The main area shows a script starting with an "when Arduino starts up" event. It sets the variable "speed" to 70, then enters a loop where it sets PWM pins 6 through 14 to output as "speed" (values 6, 10, 5, 9, 11, 12, 15, 16, 7, 8, 13, 14), sets digital pins 11 and 15 to high, and 12 and 16 to low. After this, it waits 3 seconds and changes the speed by 10.

```
when Arduino starts up
  set speed to 70
  repeat (14)
    set PWM 6 output as speed
    set PWM 10 output as speed
    set PWM 5 output as speed
    set PWM 9 output as speed
    set digital pin 11 output as high
    set digital pin 12 output as low
    set digital pin 15 output as high
    set digital pin 16 output as low
    wait (3) seconds
    change speed by 10
```



ARMSTRONG
ENTERTAINMENT MEETS EDUCATION

Increase speed gradually



Step 6: Repeat this 5 times

The image shows the mBlock 4 software interface. On the left, the Scratch-style stage features a panda sprite. The workspace contains two scripts:

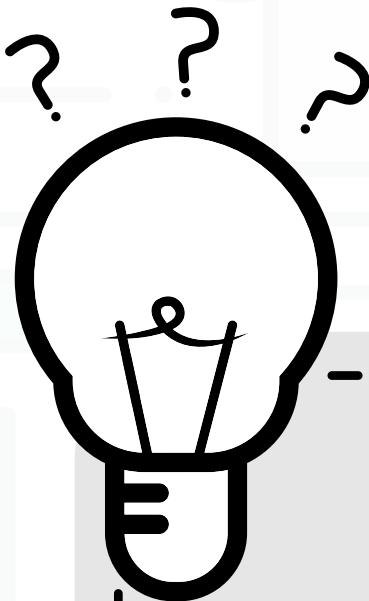
- Scratch Script:** A `when green flag clicked` script with a `repeat (10)` loop containing a `set [speed v] to [70]` block.
- Arduino C Script:** An `when Arduino starts up` script with a `repeat (5)` loop. Inside the loop, there are two sets of `set PWM` blocks (one for pin 6 and one for pin 10) followed by a series of `set digital pin` blocks (pins 11 through 14) alternating between high and low states. After the loop, there is a `wait (3 seconds)` block and a `change [speed v] by [10]` block.

The mBlock interface includes a toolbar at the top with File, Edit, Save, Publish, Courses, Tutorials, Feedback, Python Editor, and a GitHub icon. On the left, there's a Devices panel with an Arduino connection status, and a bottom section for Mode Switch, Upload, Live, and Connect.

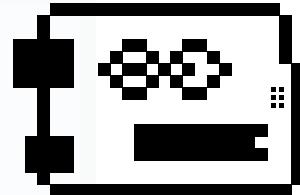
Increase speed gradually



Step 7: After 5 times make robot stop



To make robot stop:
set all motor PWM pins to 0
set all motor pins to low



Increase speed gradually



Step 7: After 5 times make robot stop

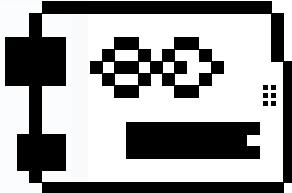
```
when Arduino starts up
  set [speed v] to [70]
repeat (5)
  set PWM [6] output as [speed]
  set PWM [10] output as [speed]
  set PWM [5] output as [speed]
  set PWM [9] output as [speed]
  set digital pin [11] output as [high ▾]
  set digital pin [12] output as [low ▾]
  set digital pin [15] output as [high ▾]
  set digital pin [16] output as [low ▾]
  set digital pin [7] output as [high ▾]
  set digital pin [8] output as [low ▾]
  set digital pin [13] output as [high ▾]
  set digital pin [14] output as [low ▾]
  wait (3) seconds
  change [speed v] by (10)
end
set PWM [6] output as [0]
set PWM [10] output as [0]
set PWM [5] output as [0]
set PWM [9] output as [0]
set digital pin [11] output as [low ▾]
set digital pin [12] output as [low ▾]
set digital pin [15] output as [low ▾]
set digital pin [16] output as [low ▾]
set digital pin [7] output as [low ▾]
set digital pin [8] output as [low ▾]
set digital pin [13] output as [low ▾]
set digital pin [14] output as [low ▾]
```

A Scratch script starting with "when Arduino starts up". It begins a "repeat (5)" loop. Inside the loop, it sets PWM pins 6, 10, 5, and 9 to the current speed value. It then sets digital pins 11, 12, 15, 16, 7, 8, 13, and 14 to high or low based on the current speed. After each iteration, it waits 3 seconds and increases the speed by 10. After the loop, it sets all PWM and digital pins to 0.



```
set PWM [6] output as [0]
set PWM [10] output as [0]
set PWM [5] output as [0]
set PWM [9] output as [0]
set digital pin [11] output as [low ▾]
set digital pin [12] output as [low ▾]
set digital pin [15] output as [low ▾]
set digital pin [16] output as [low ▾]
set digital pin [7] output as [low ▾]
set digital pin [8] output as [low ▾]
set digital pin [13] output as [low ▾]
set digital pin [14] output as [low ▾]
```

A Scratch script consisting of a vertical stack of 12 blue blocks. From top to bottom, they are: "set PWM [6] output as [0]", "set PWM [10] output as [0]", "set PWM [5] output as [0]", "set PWM [9] output as [0]", "set digital pin [11] output as [low ▾]", "set digital pin [12] output as [low ▾]", "set digital pin [15] output as [low ▾]", "set digital pin [16] output as [low ▾]", "set digital pin [7] output as [low ▾]", "set digital pin [8] output as [low ▾]", "set digital pin [13] output as [low ▾]", and "set digital pin [14] output as [low ▾]".



Light dimmer simulation

TIN
KER
CAD

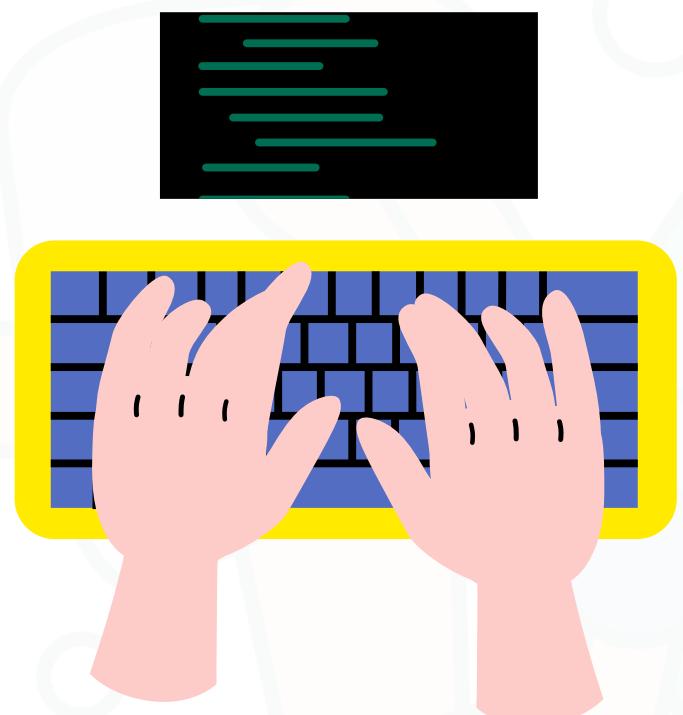


Light dimmer simulation



Step 1: Connect LED to pin 3

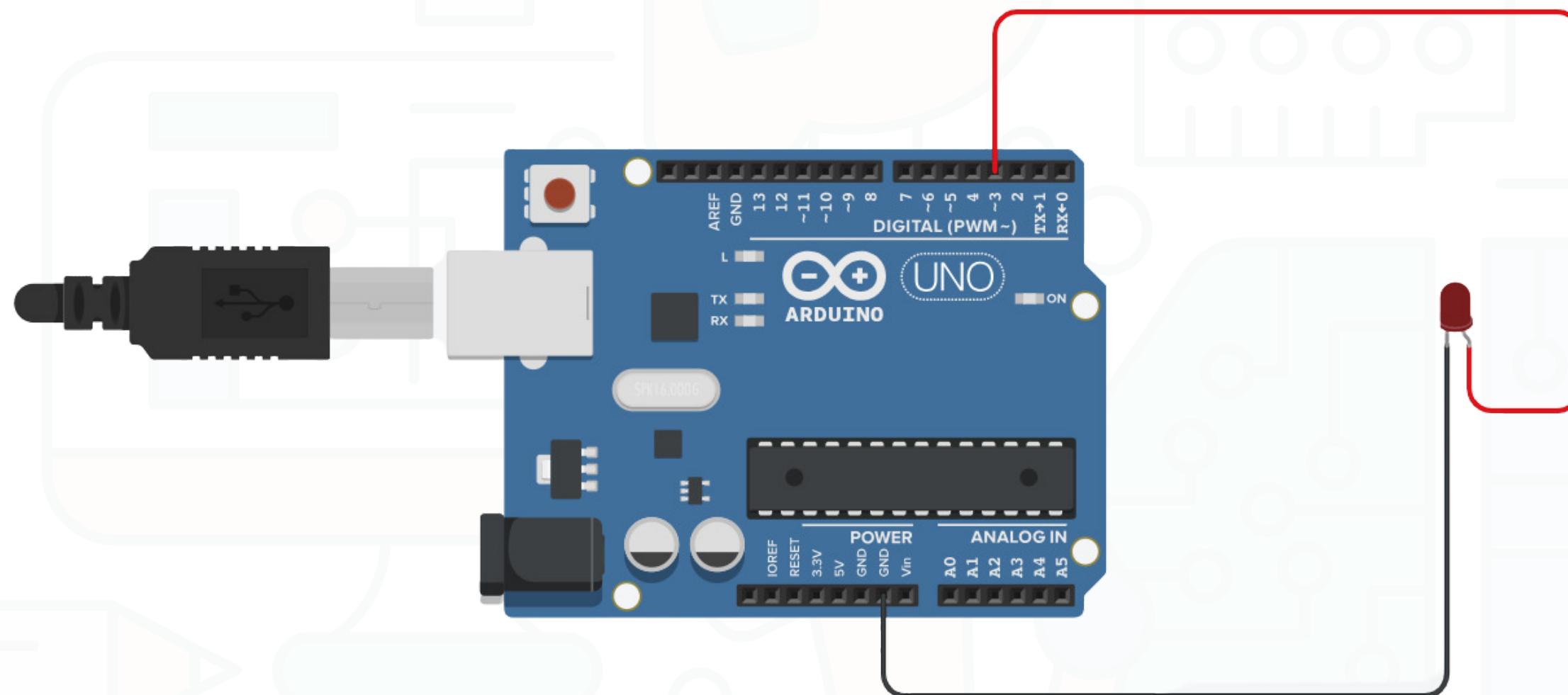
Try it by yourself



Light dimmer simulation



Step 1: Connect LED to pin 3

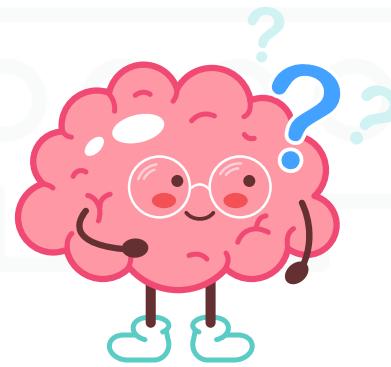


Light dimmer simulation



Step 2: Write code to control LED's brightness

Think in steps

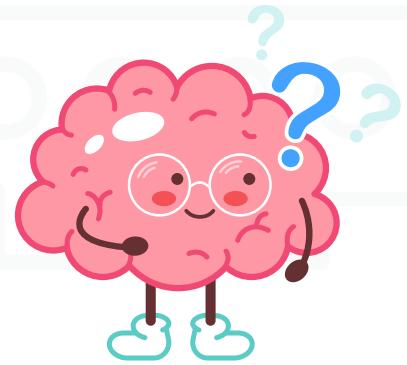


Light dimmer simulation



Step 2: Write code to control LED's brightness

Think in steps



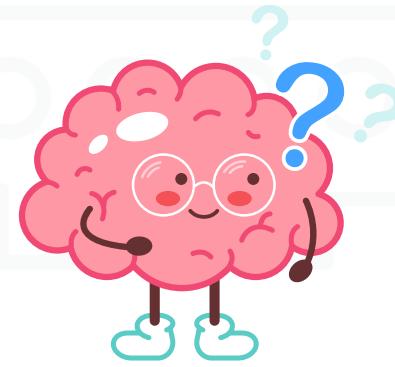
1. I want to start with a turned-off LED

Light dimmer simulation



Step 2: Write code to control LED's brightness

Think in steps



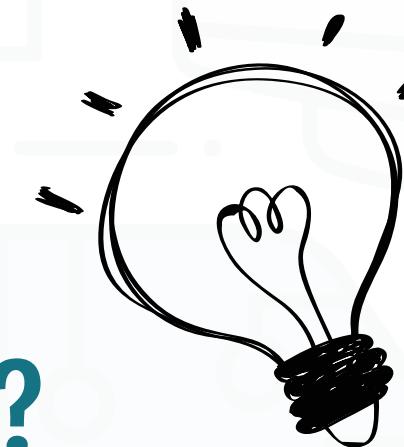
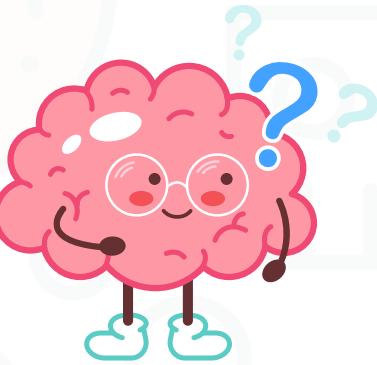
1. I want to start with a turned-off LED
2. Increase its brightness by 10 every 0.5 sec

Light dimmer simulation



Step 2: Write code to control LED's brightness

Think



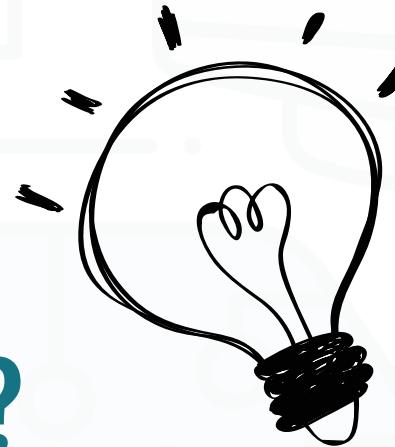
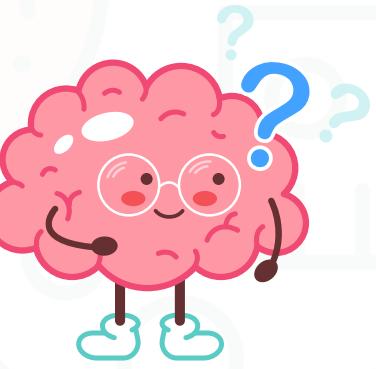
How can we control brightness?

Light dimmer simulation



Step 2: Write code to control LED's brightness

Think

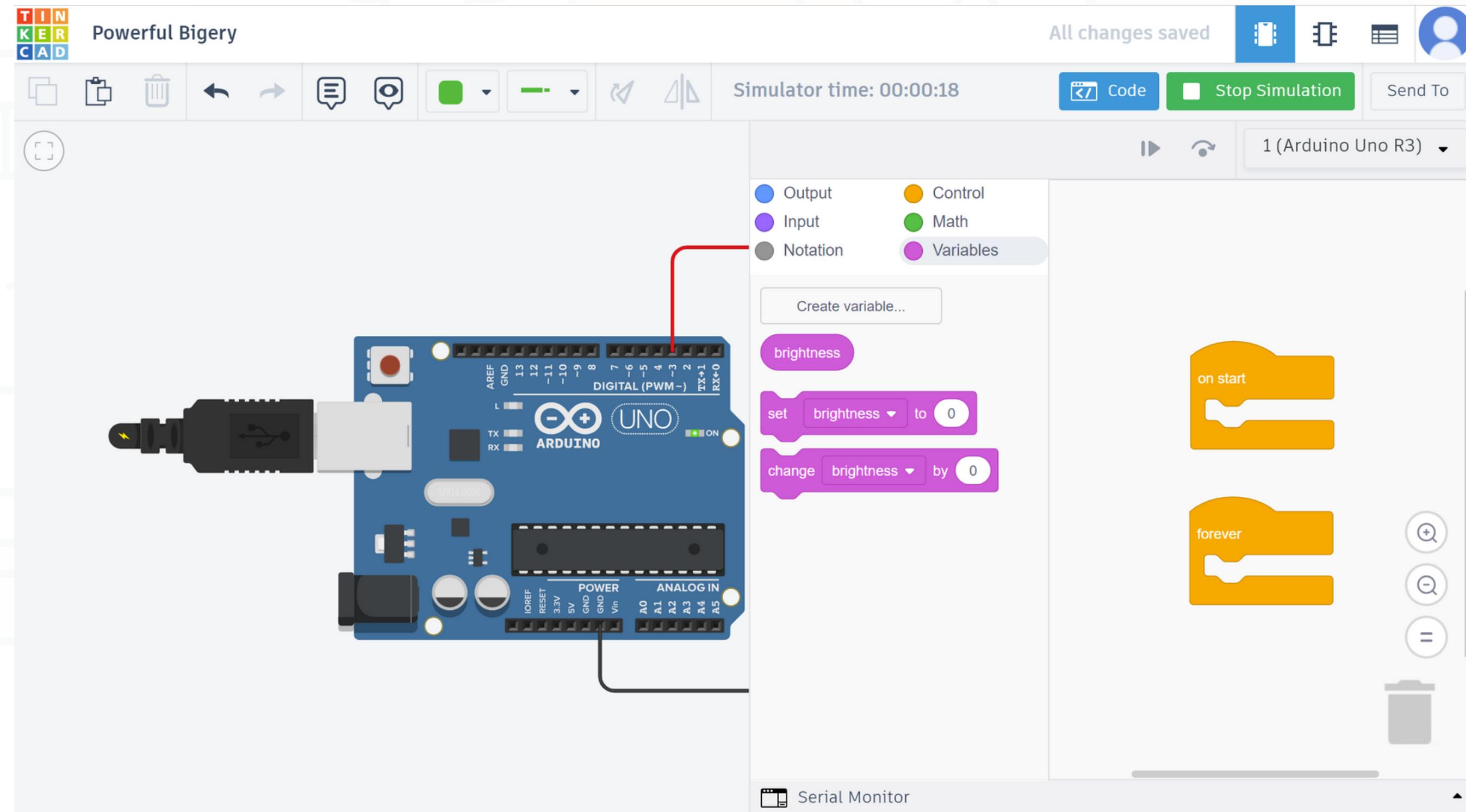


How can we control brightness?

Using a variable named brightness that can be increased or decreased over time.

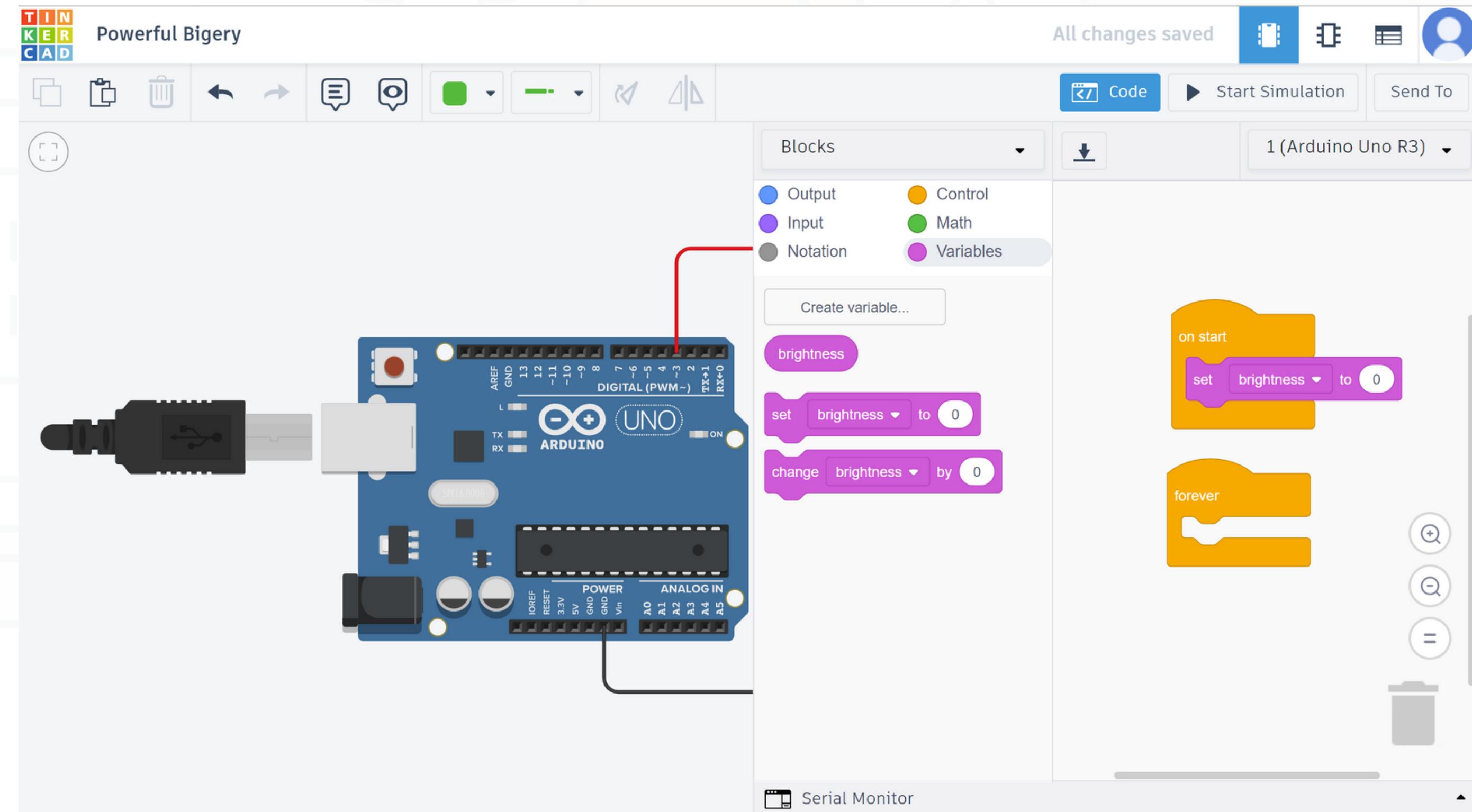
Light dimmer simulation

Step 2.1: Create brightness variable



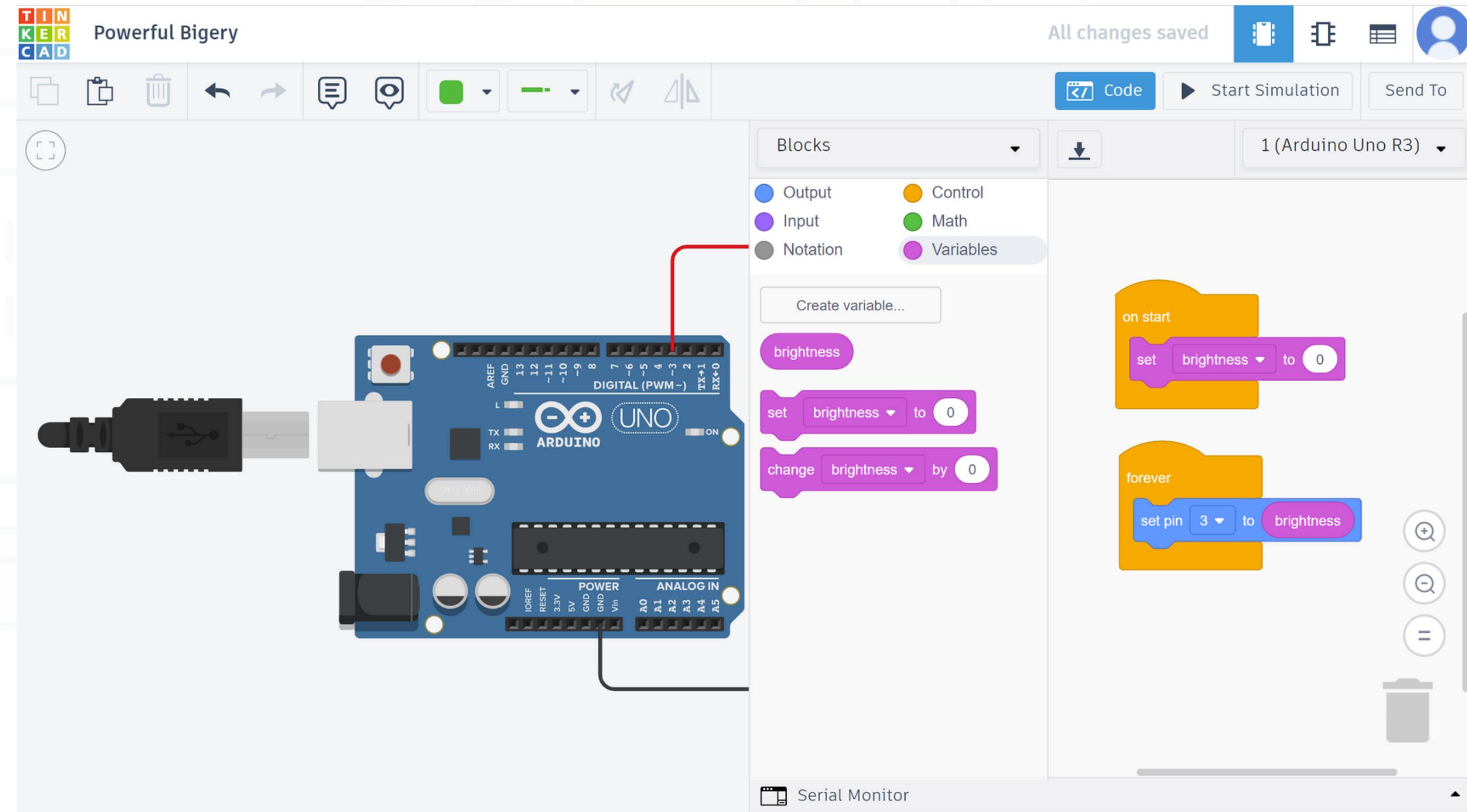
Light dimmer simulation

Step 2.2: Set variable to 0



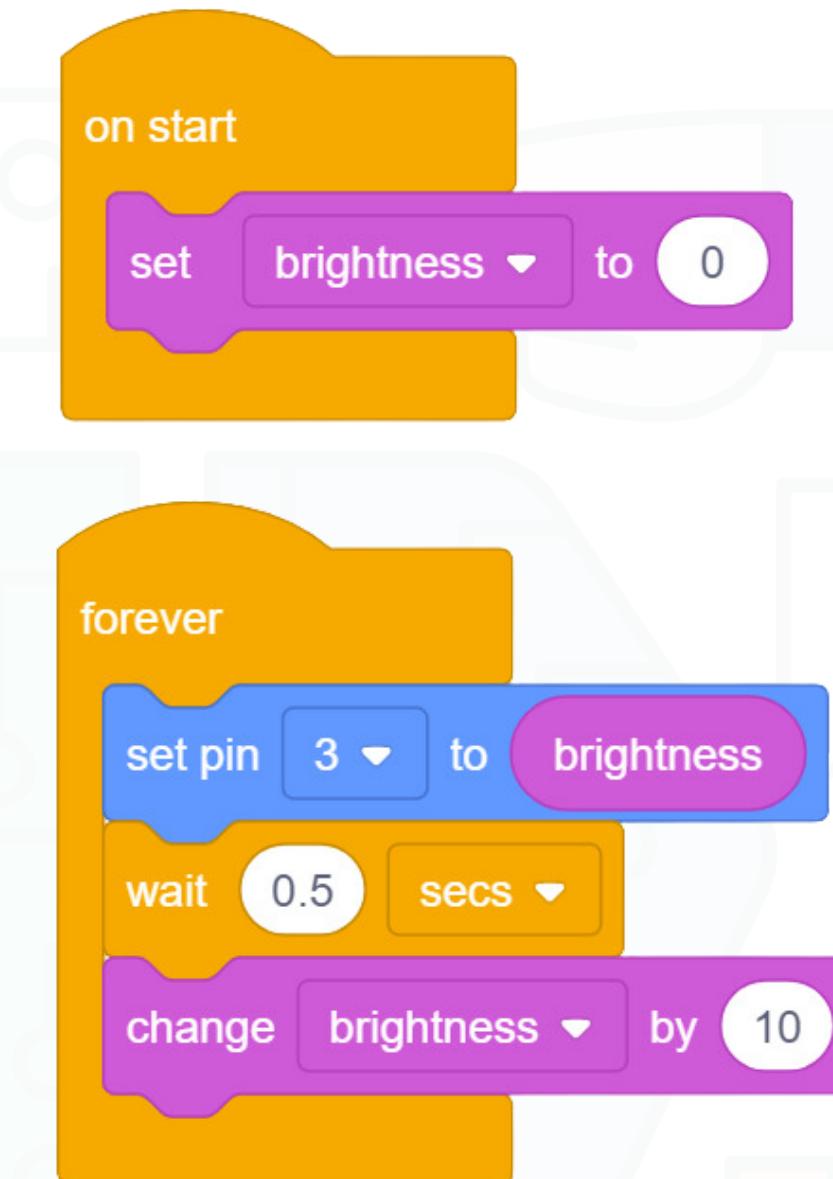
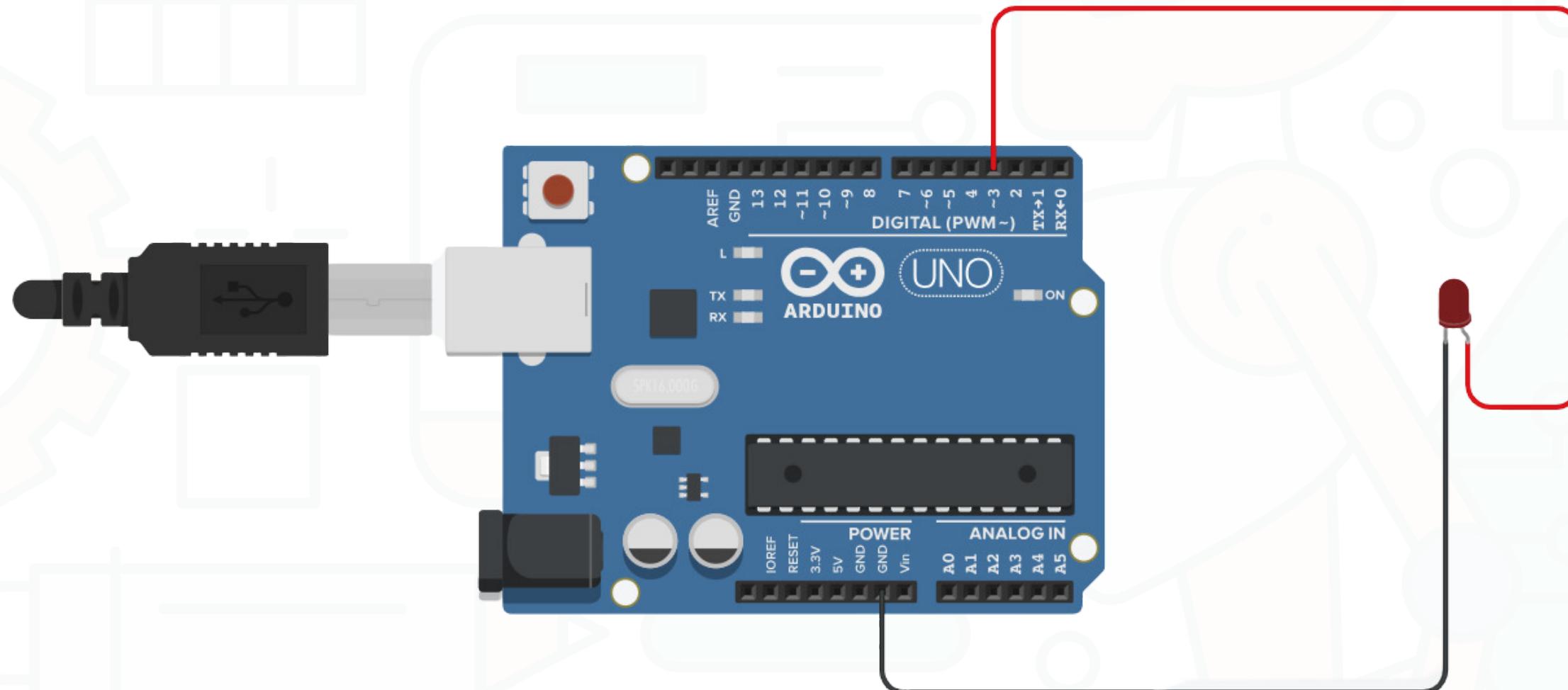
Light dimmer simulation

Step 2.3: Set pin 3 to Brightness



Light dimmer simulation

Step 2.4: Wait 0.5 sec and increase brightness by 10



```
on start
  set brightness to 0
forever
  set pin 3 to brightness
  wait 0.5 secs
  change brightness by 10
```

