WARWICK
THE UNIVERSITY OF WARWICK

University of Warwick

A Taste of Science and Engineering

Dr Leonardo Alves Dias

Topics

- Introduction to Arduino Platform: electrical engineering concepts.
- Introduction to programming concepts: using block-based programming.
- Introduction to development platforms: Tinkercad.
- Prototyping with Arduino: practising programming and electronics.
- Introduction to sensors and actuators.

Learning outcomes

- Understand and explain concepts of microcontrollers.
- Understand basic concepts of electronic devices.
- Understand basic concepts of visual programming using block-based programming.
- Simulate electronic prototypes.

1. Introduction to Arduino

Introduction

Created in mid-2005.

Developed in Italy to create projects/prototypes in a simple and faster way.

Arduino is a programmable electronics prototyping platform

based on microcontrollers (Atmel) used in the control of

logical processes.

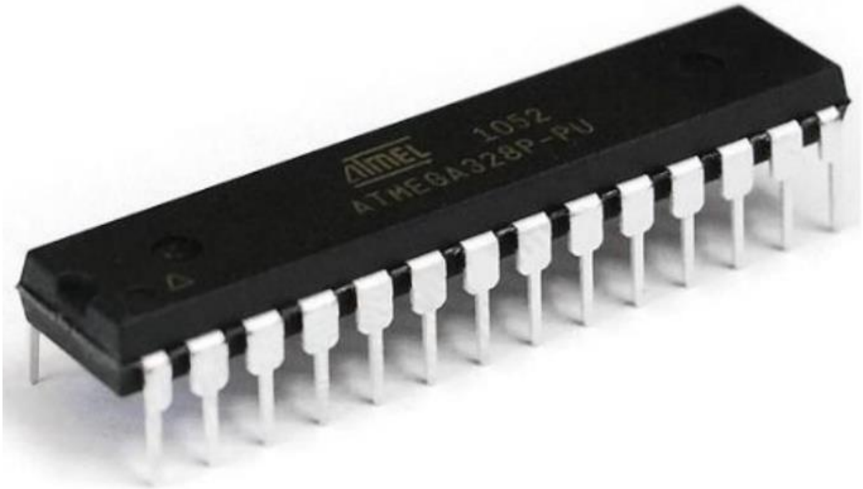
Open-Source Platform: free hardware and software.

https://www.arduino.cc/

Introduction to Microcontrollers

- A microcontroller is a programmable integrated circuit that executes orders stored in its memory. It can also be defined as an integrated circuit with “programmable intelligence” used to control logic processes.

- They are generally used in the automation and control of products and peripherals, such as automotive engine control systems, remote controls, office and home machines, toys, supervisory systems, etc.



Applications of Microcontrollers

Telecommunication, Robotics, Home appliances, Healthcare, and Industrial, among others.

Internet of Things

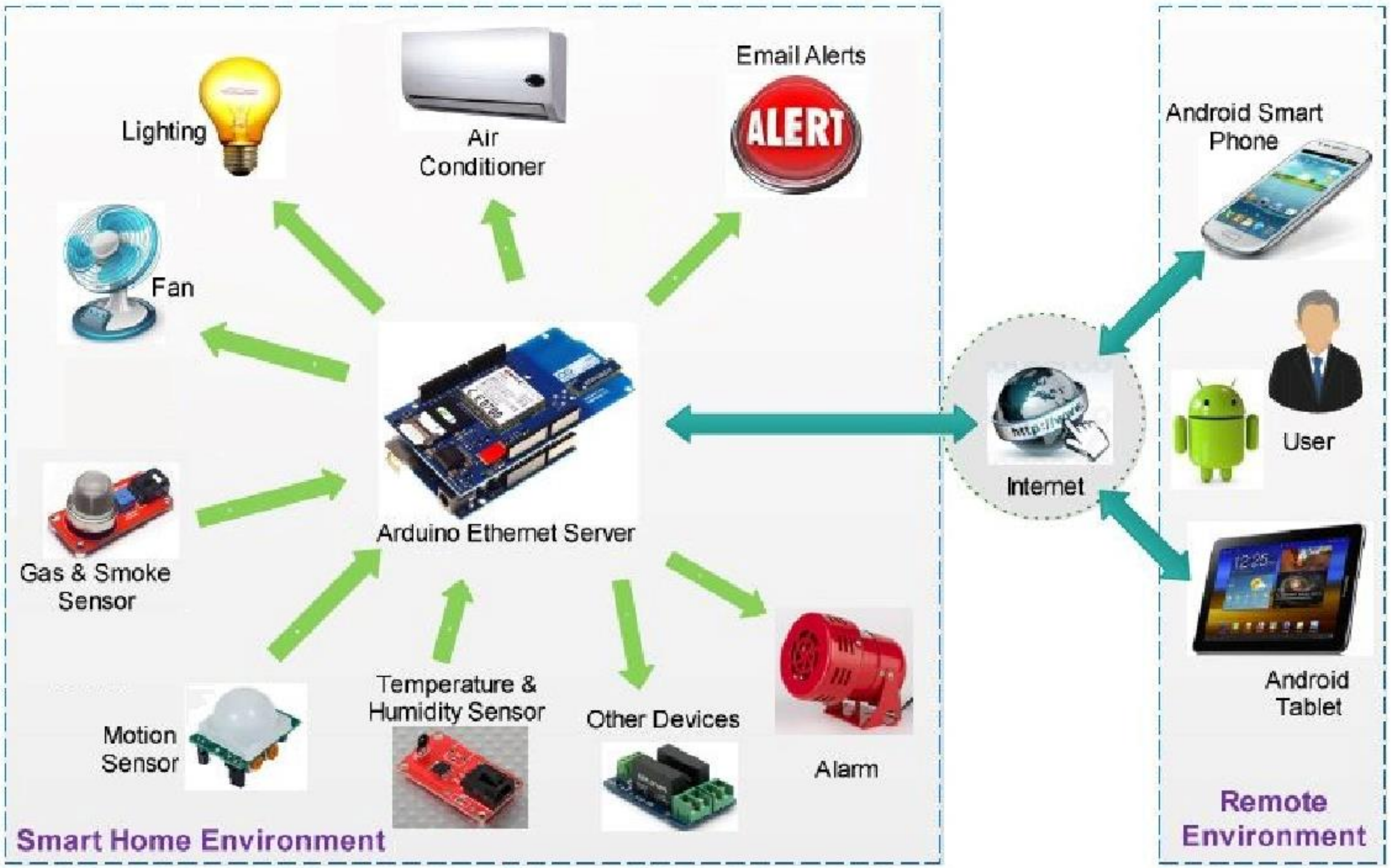
- The Internet of Things (IoT) is a concept that refers to the interconnection of everyday objects and devices to the Internet, enabling them to collect, exchange, and act on data without the need for direct human intervention.
- It extends the capabilities of traditional devices by granting them the ability to communicate, analyse, and respond to data in real-time.
- IoT has the potential to transform various industries, enhance efficiency, and improve the overall quality of life.
- Application examples: smart home devices, smart cities, healthcare, industrial, agriculture, connected vehicles, etc.

Internet of Things – Arduino Applications

<u>Smart Home Devices</u>:

IoT enables various devices to be interconnected to enhance convenience and energy efficiency.

Examples include smart thermostats, smart lighting systems, voice-controlled virtual assistants (such as Amazon Echo or Google Home), and connected security cameras, amongst others.



Introduction to Arduino Platform

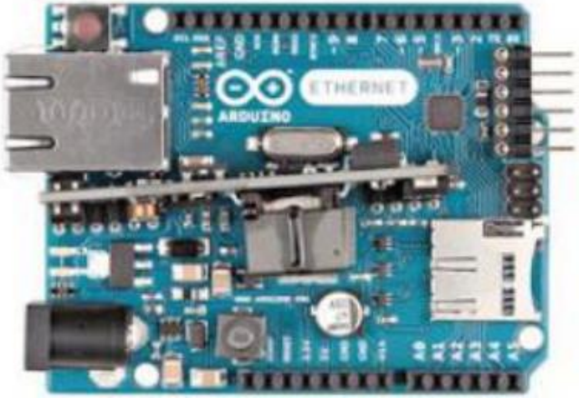
Arduino Family:

Arduino Leonardo

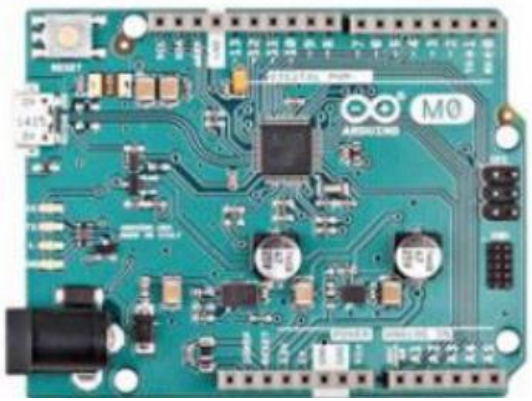
Arduino Mega2560 R3

Arduino Industrial 101

Arduino Ethernet



Arduino M0



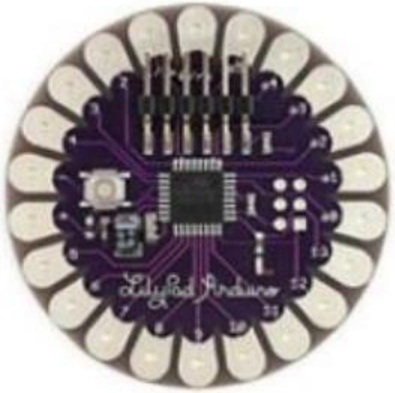
Introduction to Arduino Platform

Arduino Family:

Arduino Yún

Arduino Leonardo Ethernet

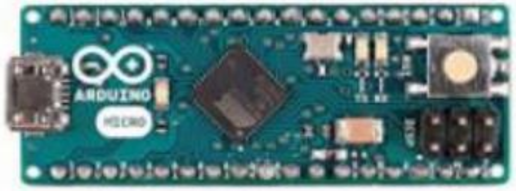
Arduino LilyPad



Arduino ISP

Arduino Yún mini

Arduino Micro



Introduction to Arduino Platform

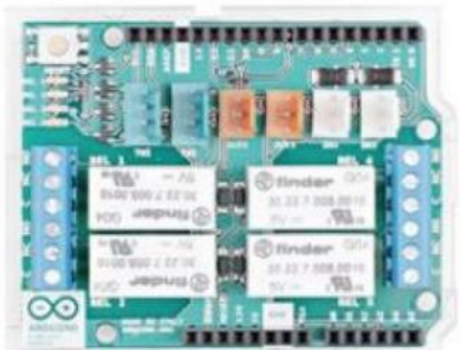
Arduino Shields: A board that allows you to expand the functionality of the Arduino.

Arduino Ethernet Shield R3

Motor Shield R3

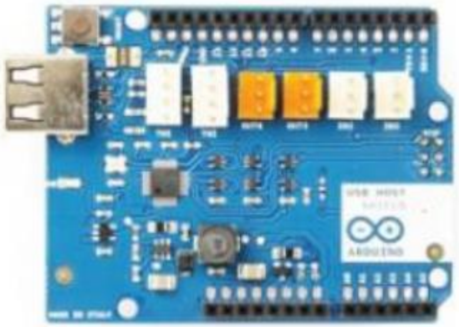
Arduino WiFi Shield

Shield Arduino 4 relês

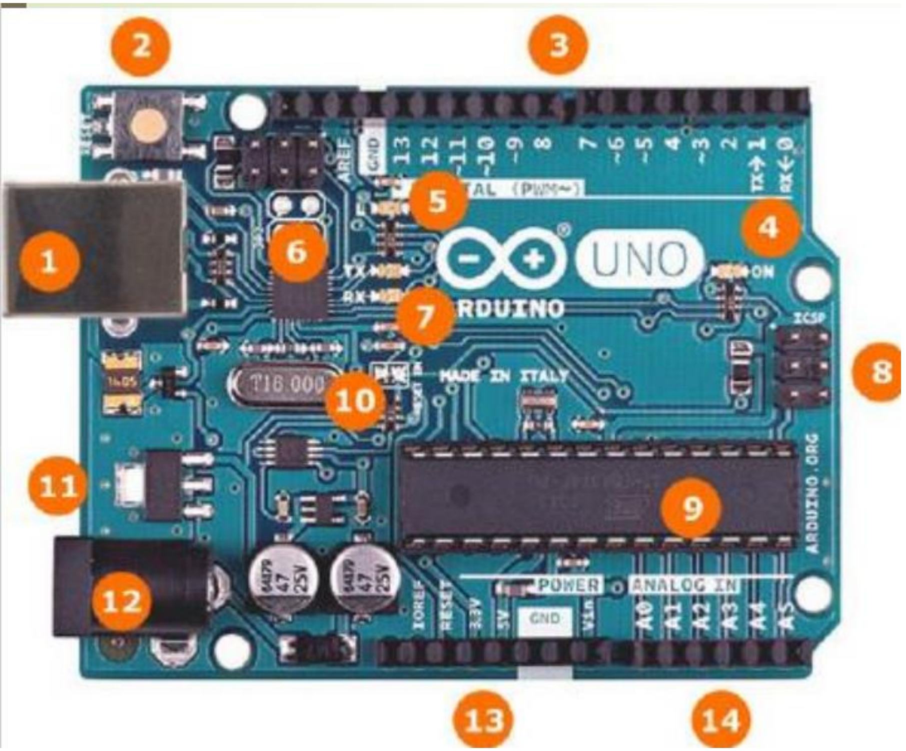


Arduino ProtoShield R3

Arduino USB Host Shield

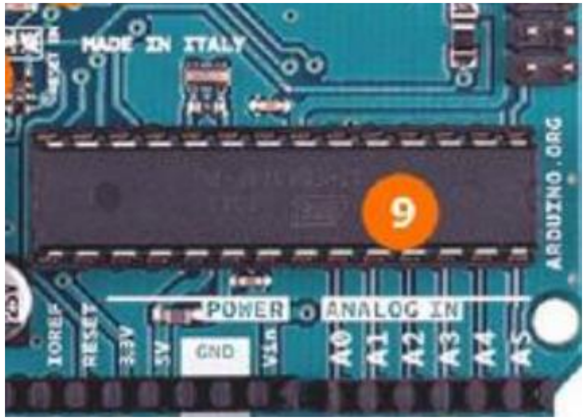


Arduino Board



1. Power (USB Connector).
2. Reset Button.
3. Digital input/output pins and PWM
4. Power LED.
5. LED connected to pin 13 (used for board testing).
6. PC Communication Microcontroller (ATMEGA).
7. Serial communication LEDs (TX/RX).
8. Serial communication (ICSP).
9. ATMEGA Microcontroller.
10. Crystal Oscillator (16MHz).
11. Voltage Regulator.
12. Power (Barrel Jack/Wall Power Supply).
13. Arduino source and ground pins.
14. Analogue inputs.

ATMEGA Microcontroller



- Microcontroller present: ATMEGA 328P.
- Operating voltage: 5V.
- I/O (Input/Output) pins: 14 (6 can be used as PWM(~)).
- Continuous current: 20mA (5V), 50mA (3.3V).
- Flash memory: 32kB.
- SRAM memory: 2kB.
- EEPROM memory: 1kB.
- Clock: 16MHz.

https://store.arduino.cc/usa/arduino-uno-rev3

Summary

We introduced the concepts of a Microcontroller. ✓

We introduced the family of Arduino Boards. ✓

We introduced the ATMEGA microcontroller present on the Arduino UNO. ✓

2. Arduino: Development platforms

Arduino Simulators

- Arduino IDE: https://www.arduino.cc/en/software
- Proteus: https://www.labcenter.com/
- SimulIDE: https://sourceforge.net/projects/simulide/
- TinkerCAD: https://www.tinkercad.com/circuits

Tinkercad: Creating an account

1. Click on Sign Up to create your account.
2. Click on Create a personal account (On your Own) and Sign Up with an Email.
3. Add your Country and Birthday.
4. Add your email and create a password.
5. Click “Done!”

Observation: You will need to check your email and verify your account

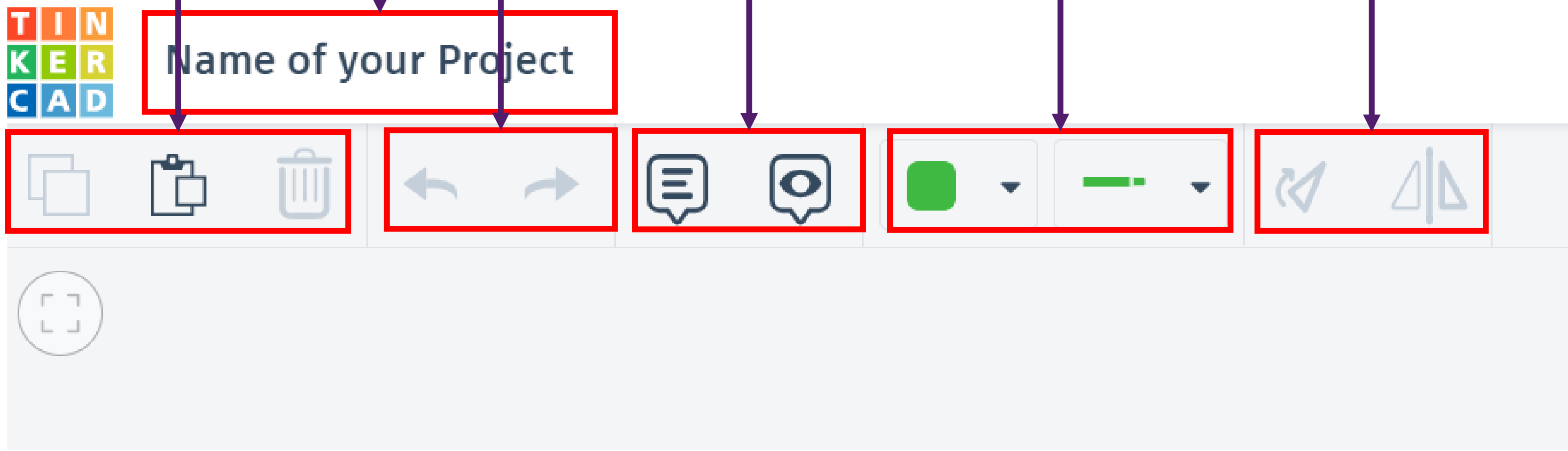
Joining a Tinkercad Class

1. Click on “Classes”
2. Click on “Join a class”
3. Use the code TWQ-I2X-2LI-PLU and click Go to My Class. Alternatively, click on the link: https://www.tinkercad.com/joinclass/TWQI2X2LIPLU
4. Click on “That’s me.”

- You now have the Warwick Summer School class. Inside this class, you will find the activities for this course.

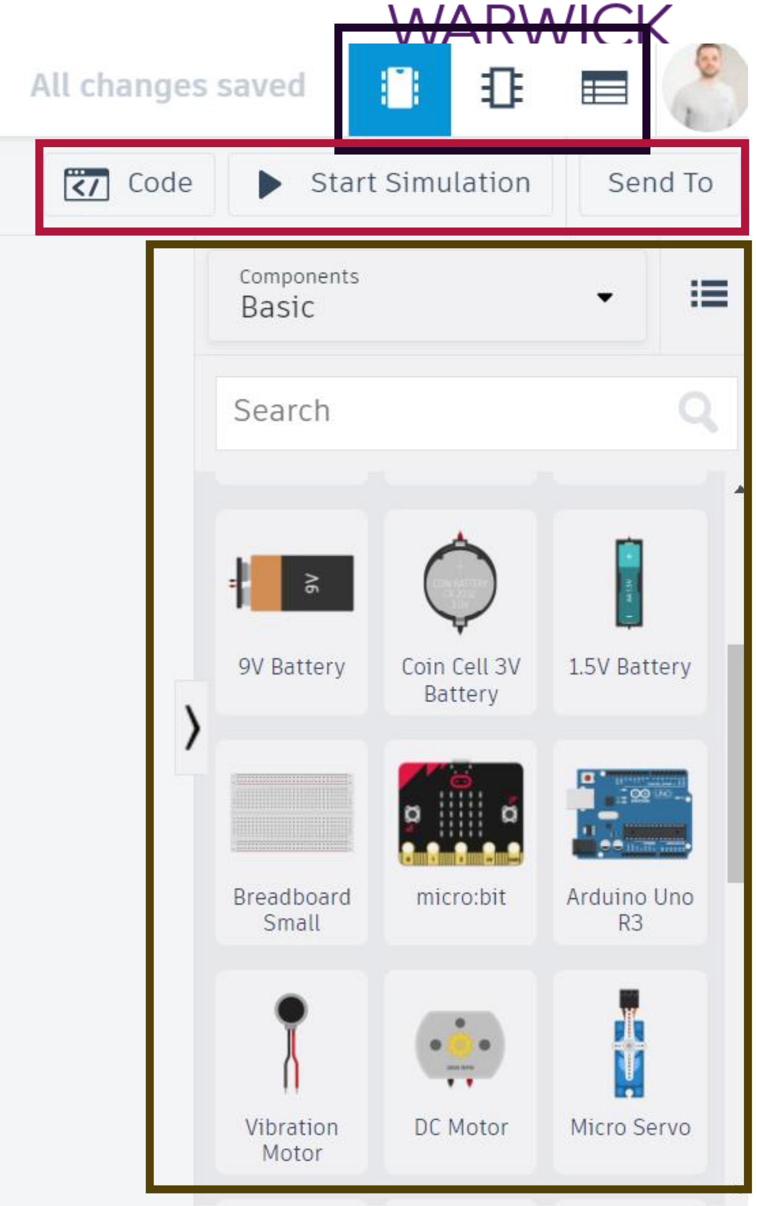
Tinkercad User Interface

- Name of your project.
- Copy, Paste, Delete
- Undo, Redo.
- Add Notes, Toggle Notes Visibility
- Wire colour and type
- Rotate and Mirror.



Tinkercad User Interface

- Circuit View
- Schematic View
- Component list.
- Code
- Start Simulation
- Send to
- Components.
- Search Bar
- Component Menu.



Summary

We introduced different Arduino Platforms for learning and practising. ✓

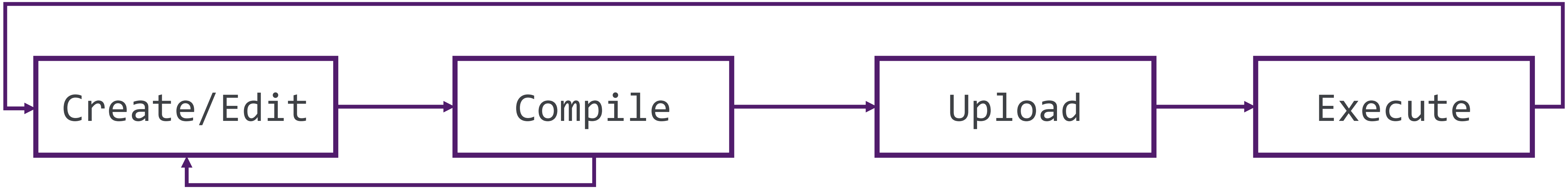
We learned how to create and account on TinkerCAD. ✓

We introduced the TinkerCAD circuit simulation user interface. ✓

3. Arduino: Prototyping

Prototyping

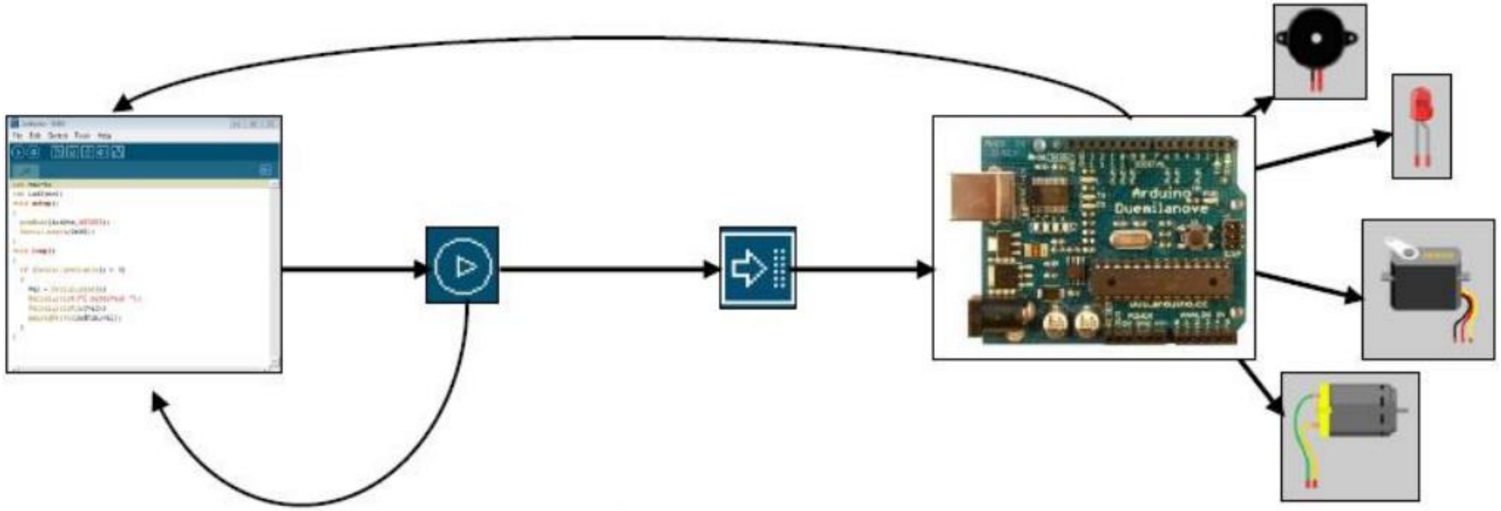
A prototype is developed based on the following step-by-step:


Create/Edit

Compile

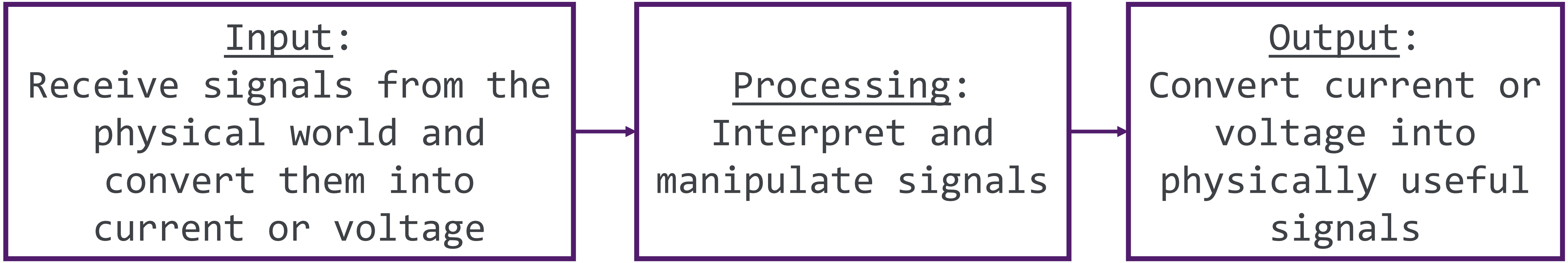
Upload

Execute



Input/Outputs

- The inputs, or inputs, are usually electronic or mechanical sensors that convert signals (in the form of temperature, pressure, humidity, contact, light, movement, pH, etc.) from the physical world and convert them into current or voltage signals. Examples of inputs are photocells, potentiometers, gas, temperature, movement sensors, etc.
- Outputs are actuators or other devices that convert current or voltage signals into physically useful signals such as movement, light, sound, force or rotation. Examples of outputs are motors, LEDs or lighting systems, a buzzer that generates different tones, etc.



Input:
Receive signals from the physical world and convert them into current or voltage

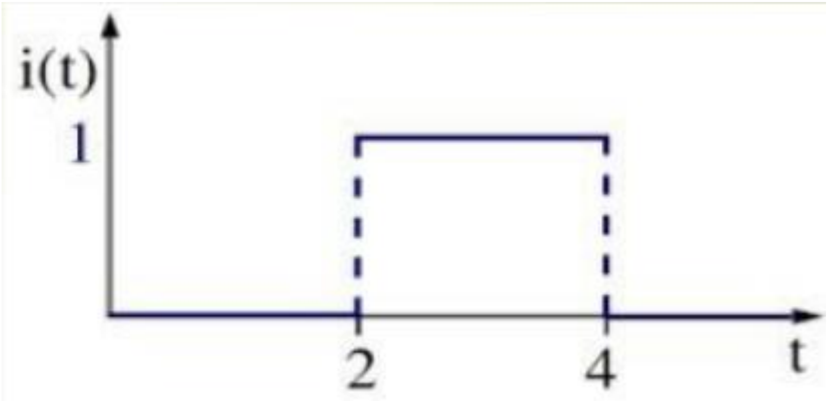
Processing:
Interpret and manipulate signals

Output:
Convert current or voltage into physically useful signals

Types of Signals

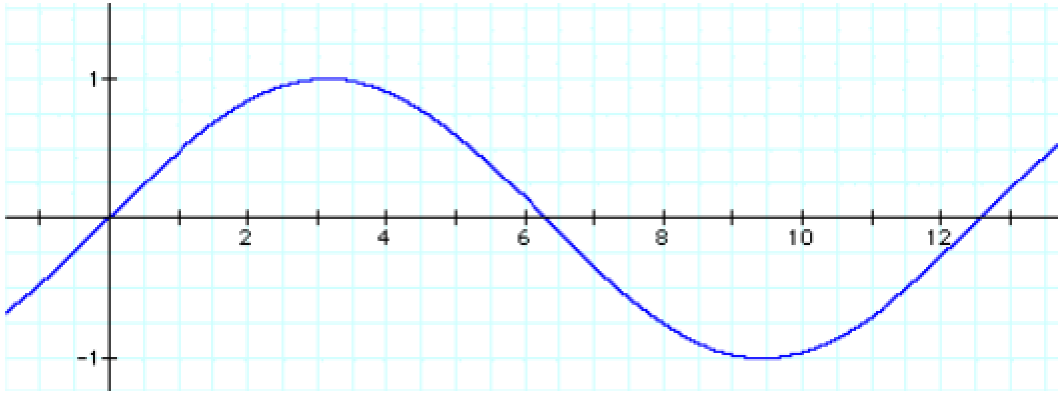
- Digital:

  Works according to Boolean principles: bit 0 represents 0V (volts), and bit 1 represents 5V (volts).



- Analogue:

  Time-varying signals (constantly take on different values)



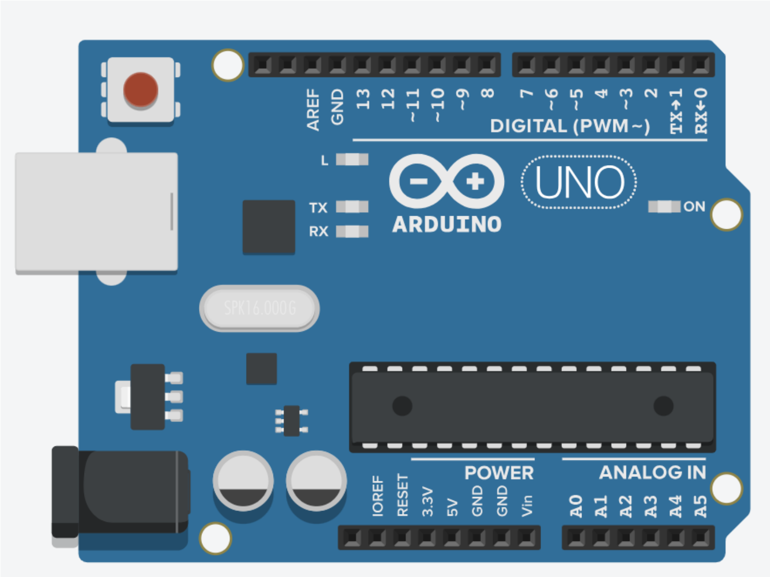
Hello World

- This example shows the simplest experiment you can do with an Arduino to verify a physical output: blinking an LED. In this example, you will learn how to drive a load (built-in LED) connected to an Arduino pin.

What will I learn?

- Activate a digital output.
- Blink the on-board LED.
- The syntax of an Arduino Code.
- Use timers on an output pin.

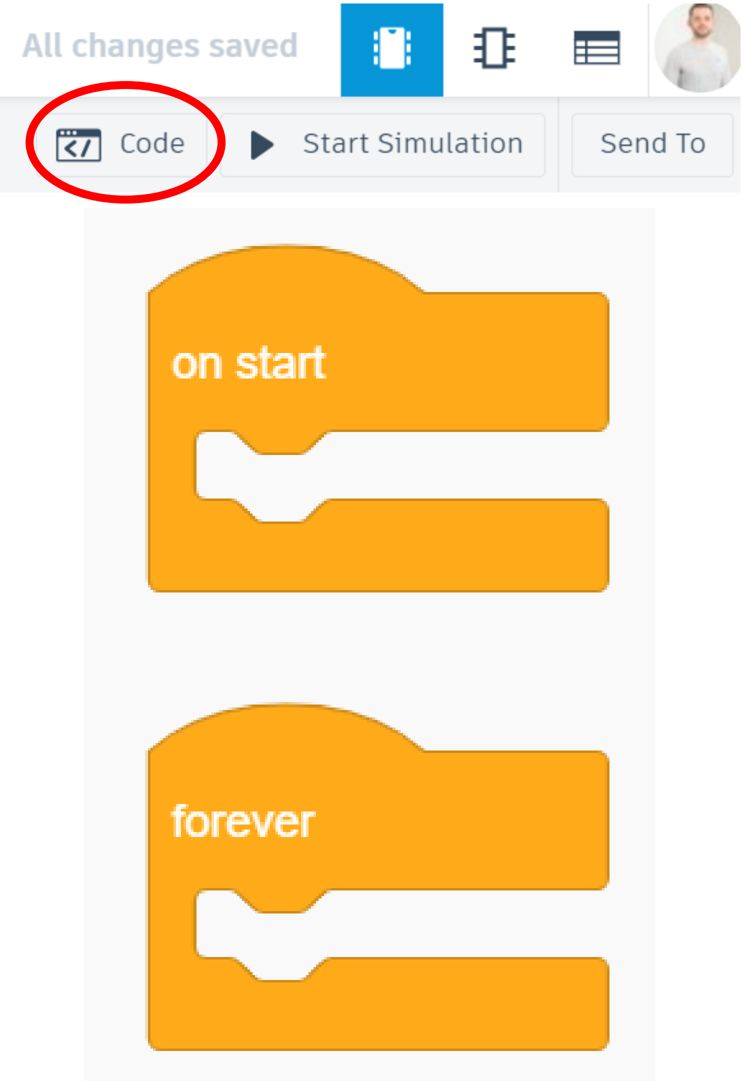
Components



Hello World

▷ The Code

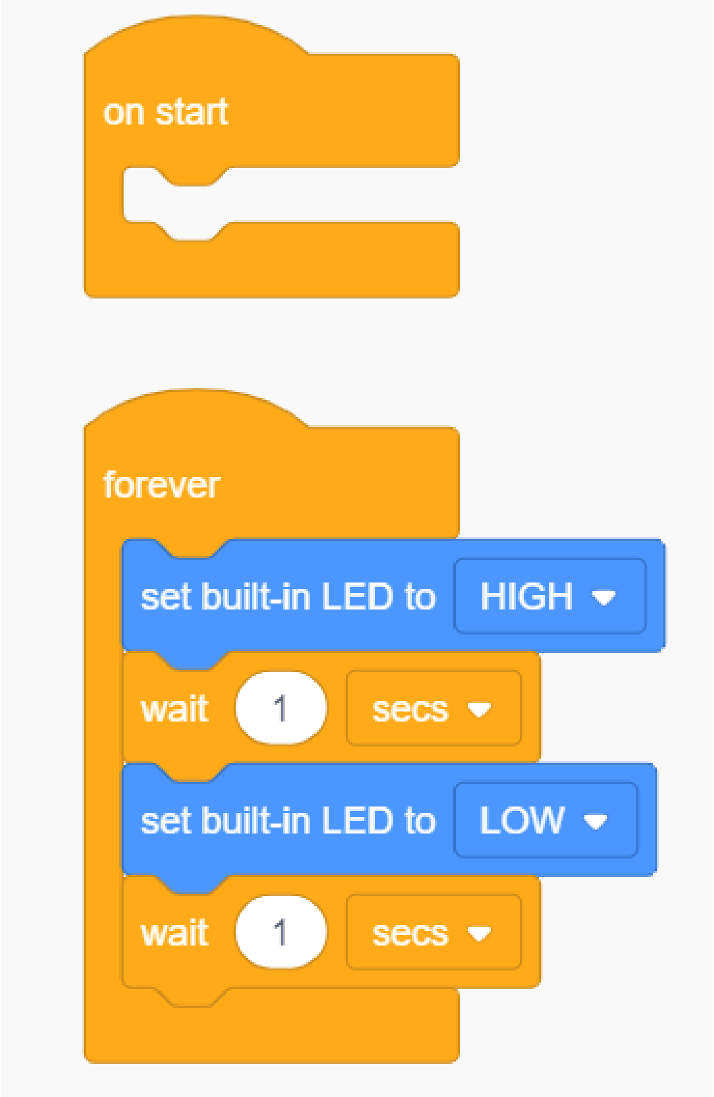
- Control Structures (Yellow):
  - *on start*: code structure that can be used to execute blocks of code only once and when the board is powered. Use it for code that you only want to execute once when the board is turned on.
  - *forever*: code structure that can be used to execute code blocks repeatedly in an infinite loop while the board is powered. Use it for code that you want to be constantly executed while the board is turned on.



Hello World

The Code

- Control Commands (Yellow):
  - *wait*: this code block activates a timer. The microcontroller waits for the defined time to be over before executing the next code block. For this example, the timer was set to 1 second.
- Output Commands (Blue):
  - *set built-in LED*: define the logical value on that pin. The pin will work as an output, controlling anything connected to it. Built-in LED is the name given to pin 13.



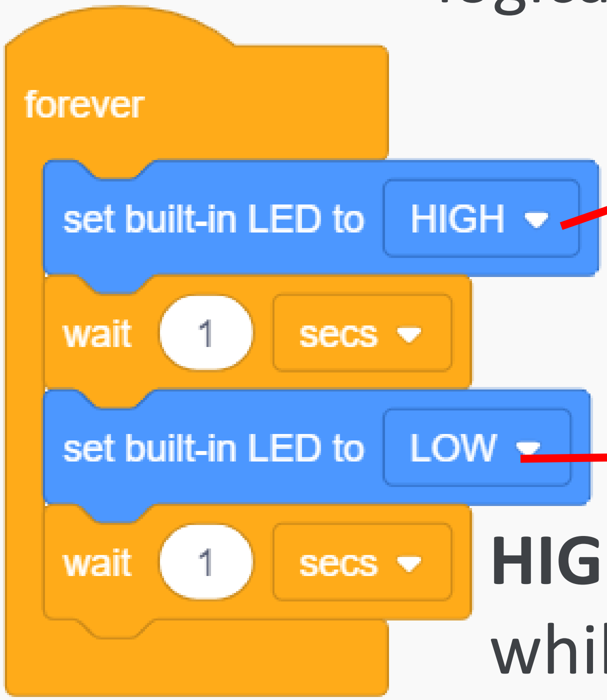
```
on start

forever
    set built-in LED to HIGH ▾
    wait 1 secs ▾
    set built-in LED to LOW ▾
    wait 1 secs ▾
```

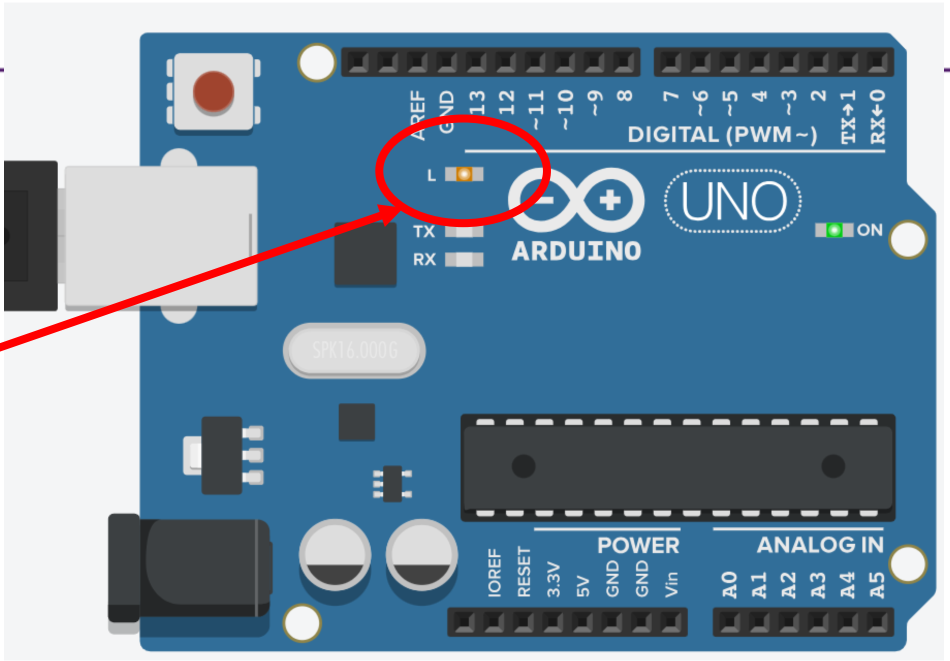
Hello World

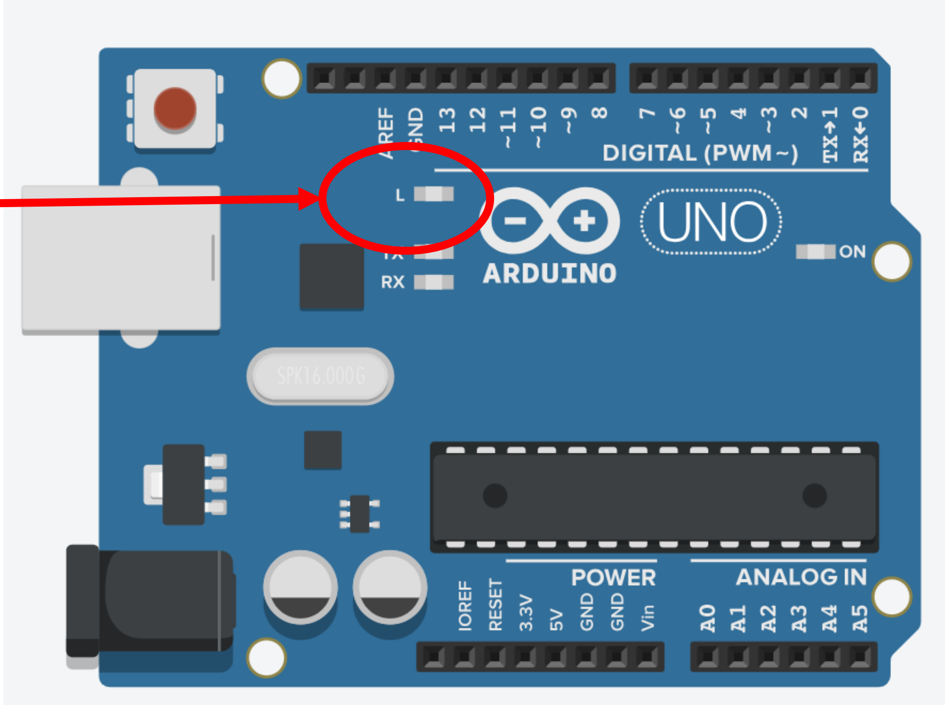
The Code

The “set” command is used to write on the pin, i.e., to send a logical value to that pin.



```
forever
    set built-in LED to HIGH
    wait 1 secs
    set built-in LED to LOW
    wait 1 secs
```





HIGH is equivalent to bit 1 (5V), while **LOW** is equivalent to bit 0 (0V).

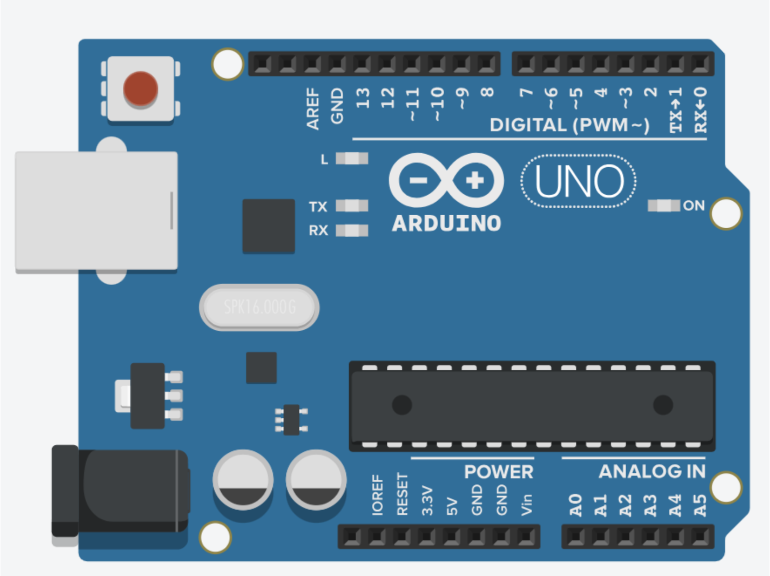
Hello World – External LED

- This example is similar to the previous one: blinking an LED. In this example, you will learn how to drive a load (an external LED) connected to an Arduino pin.

What will I learn?

- Activate a digital output.
- Blink an LED external to the board.
- The syntax of an Arduino Code.
- Use timers on an output pin.

Components



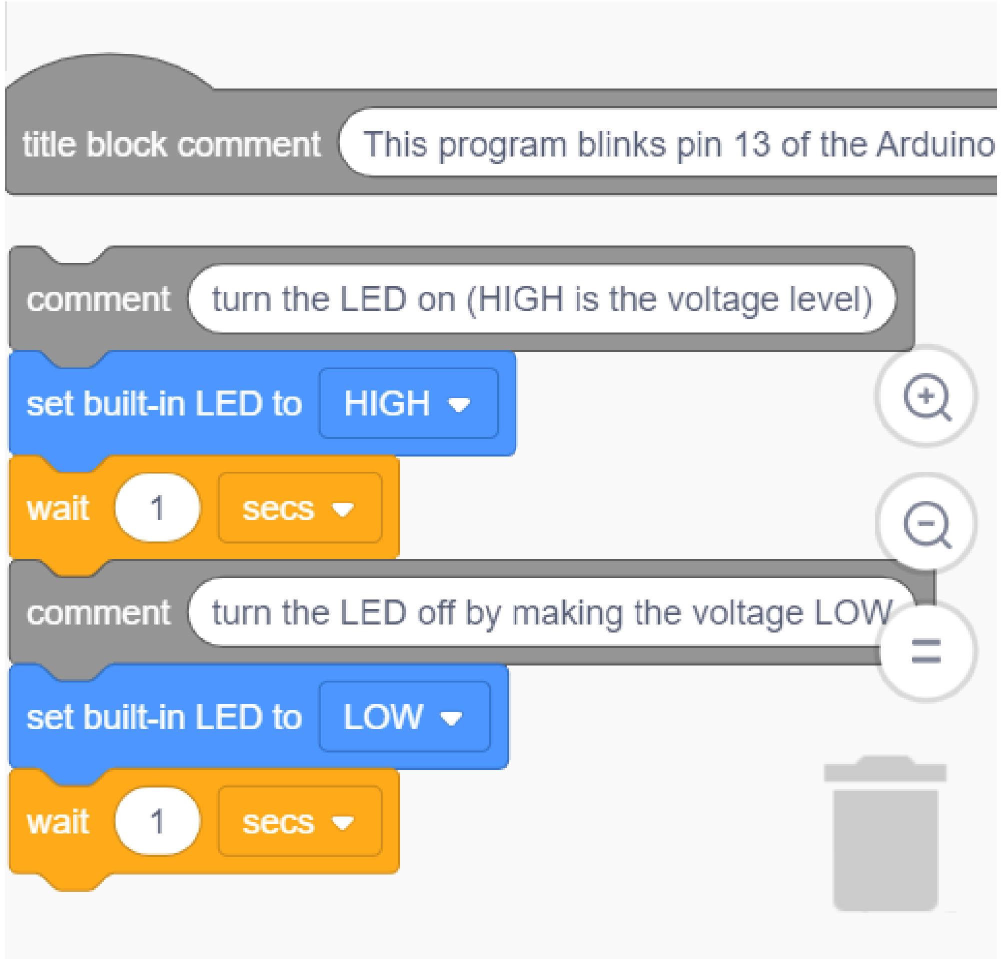
Resistor

LED

Hello World – External LED

The Code

- Notation (Grey):
  - The notation blocks are used to add comments to your code. Comments are helpful for others to understand your code, and they do not execute any action.

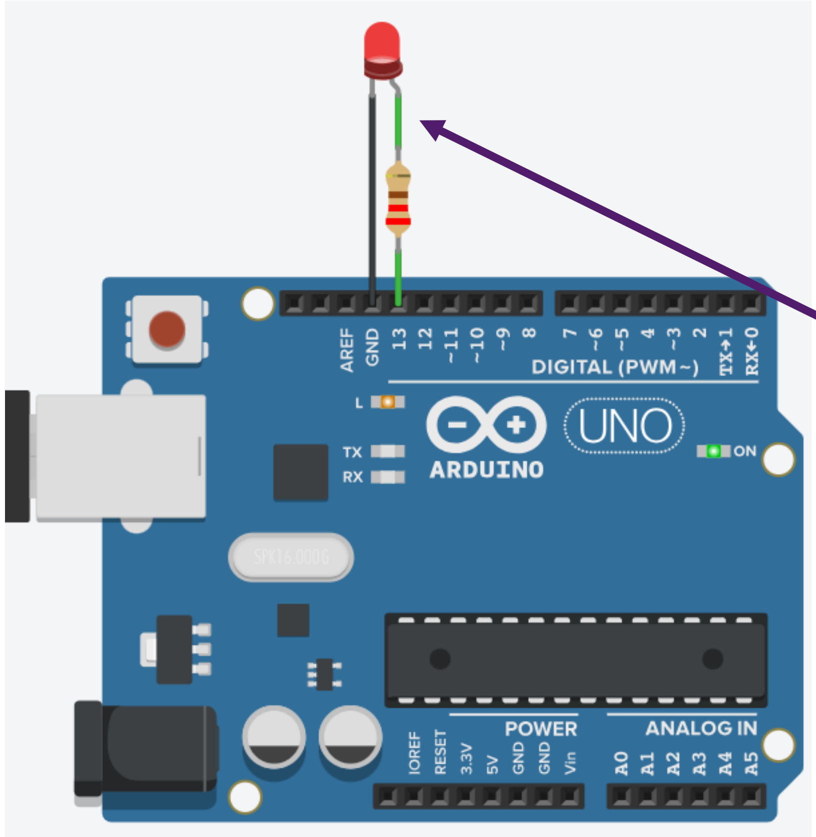


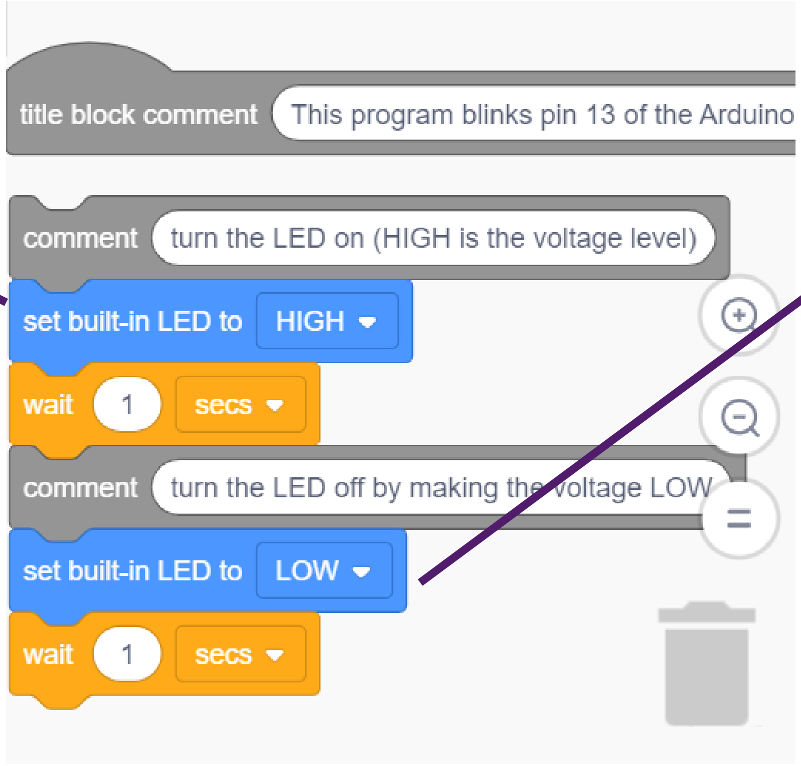
```
title block comment  This program blinks pin 13 of the Arduino
comment  turn the LED on (HIGH is the voltage level)
set built-in LED to  HIGH ▾
wait  1  secs ▾
comment  turn the LED off by making the voltage LOW
set built-in LED to  LOW ▾
wait  1  secs ▾
```

Hello World – External LED

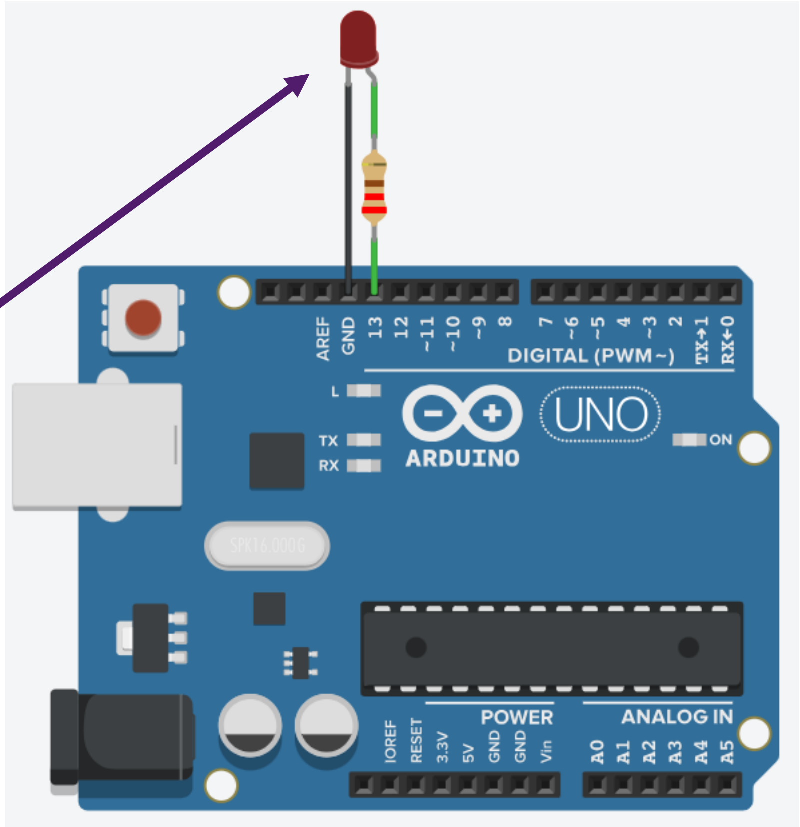
The Code

- Note that the program executes repeatedly despite not using the forever block.



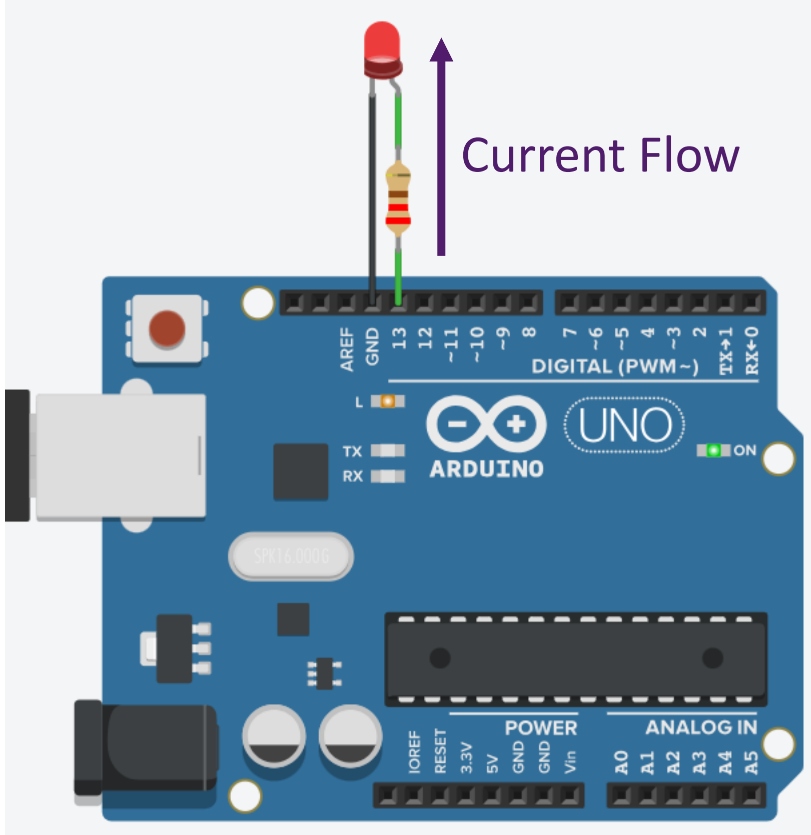


```
title block comment  This program blinks pin 13 of the Arduino
comment  turn the LED on (HIGH is the voltage level)
set built-in LED to  HIGH ▾
wait  1  secs ▾
comment  turn the LED off by making the voltage LOW
set built-in LED to  LOW ▾
wait  1  secs ▾
```



Hello World – External LED

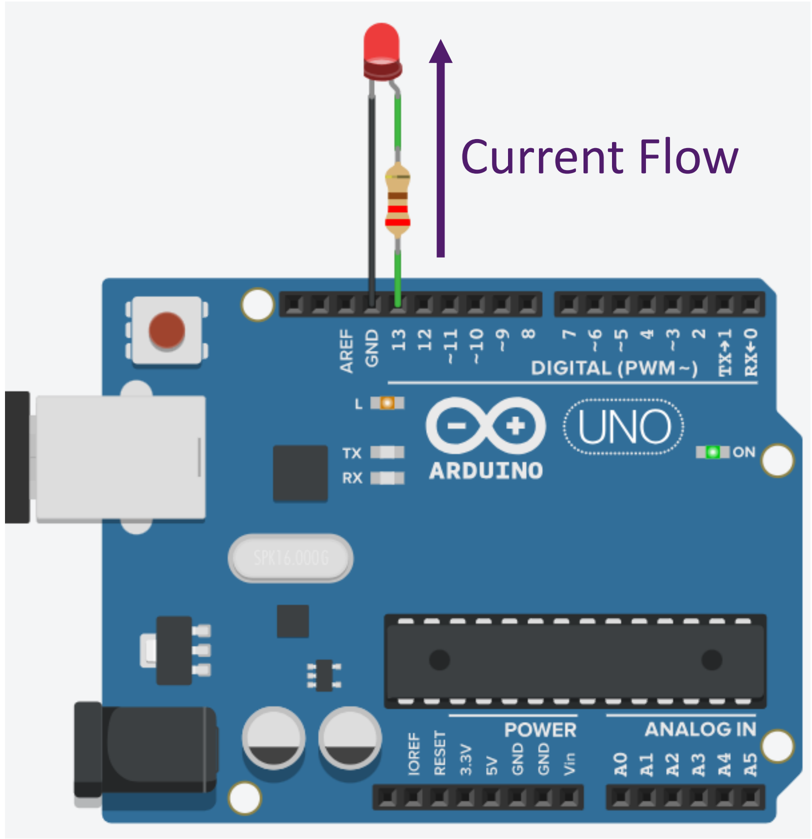
▷ The Circuit



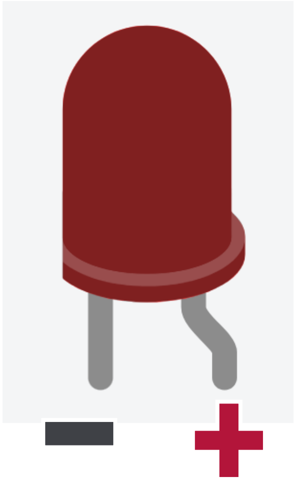
- When an Arduino pin is used as an output, it supplies the load connected to it a total of 5V and 20mA.
- This is enough to damage an LED. Therefore, a resistor is necessary to limit the amount of current that goes to the LED.
- Hence, a resistor was connected between the output pin 13, the LED and the ground (GND), reducing the current (I) from its maximum value ($I < 20$).

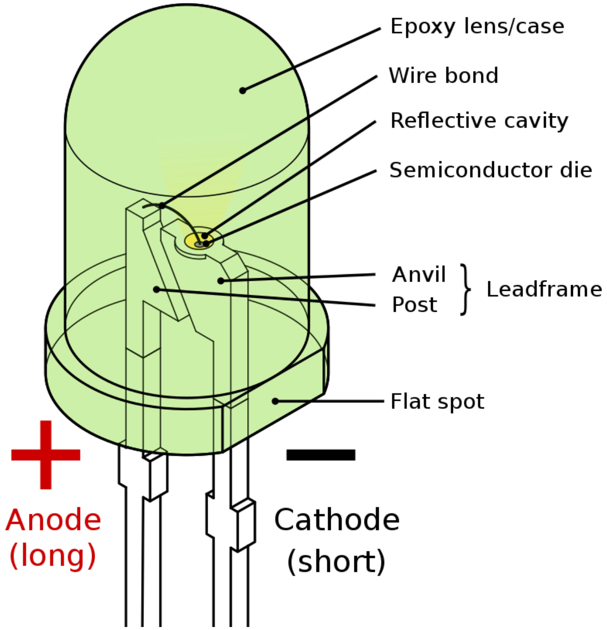
LED (Light-Emitting Diode)

The Circuit



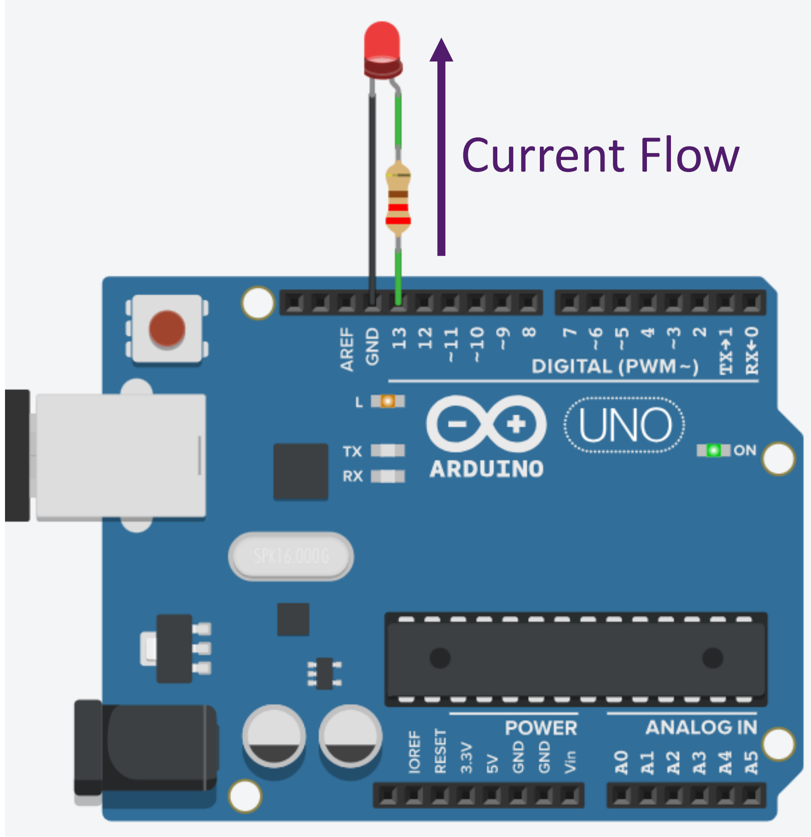
- A LED (light-emitting diode) is a semiconductor device that emits light when current flows through it.
- The LED terminals are called: anode (+) and cathode (-).



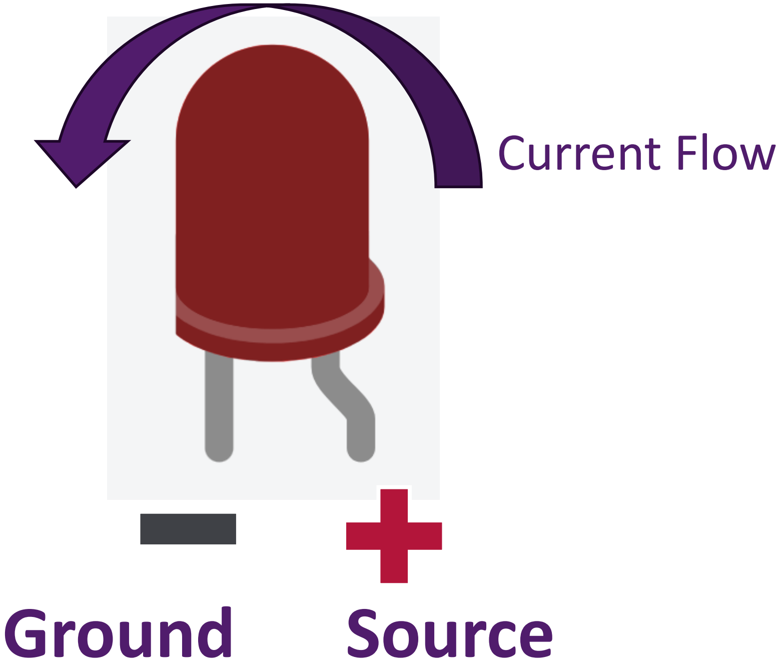


LED (Light-Emitting Diode)

The Circuit

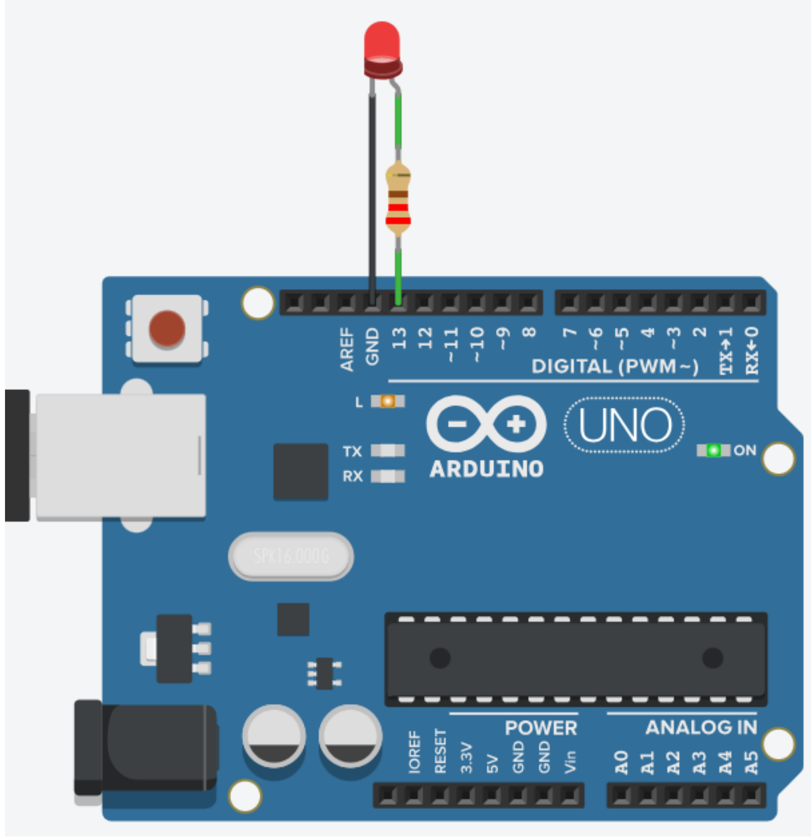


- The LED has a polarity, that is, a connection order. Changing this order will make it not work properly.
- Hence, for it to work, the current must flow from the anode (+) to the cathode (-).



Resistor

The Circuit



- A resistor is a passive two-terminal electrical component that implements electrical resistance.
- In electronic circuits, resistors are used to reduce current flow and divide voltages, among other uses.
- Ohm's law states that the resistance can be defined based on the voltage (V) and current (I), as follows:

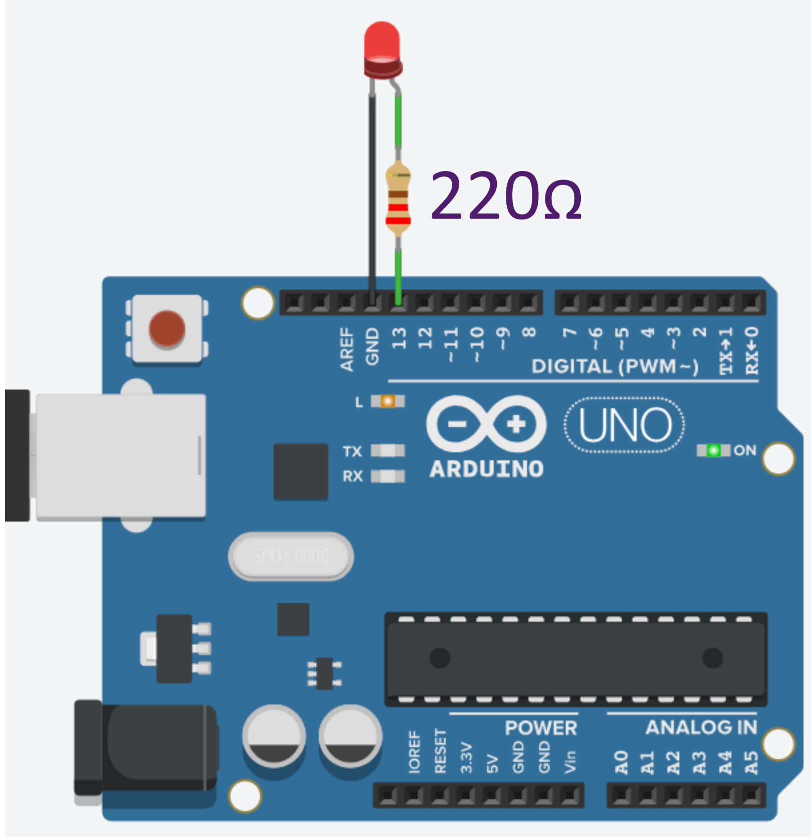
$$R = \frac{V}{I}\,\Omega$$

Resistor - Wikipedia

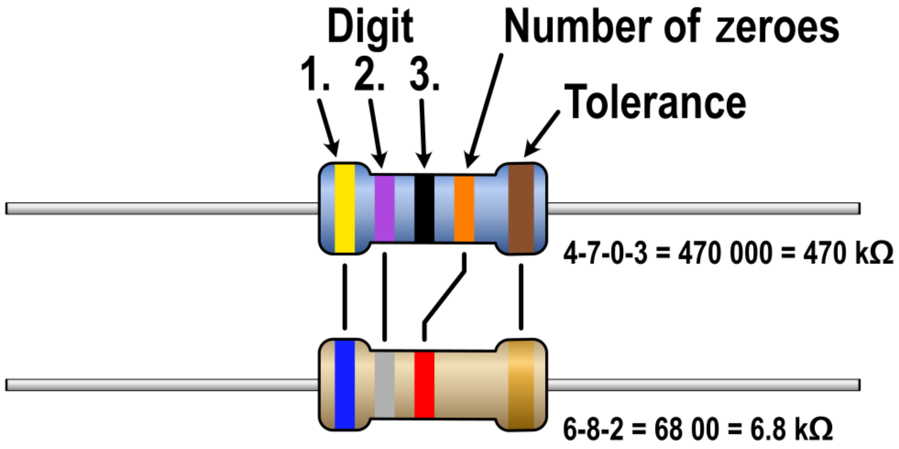
Electronic colour code - Wikipedia

Resistor

▷ The Circuit



- The resistance value of a resistor can be defined based on its colour code:
- To change its value on Tinkercad, click on the component and modify its value in the pop-up box.



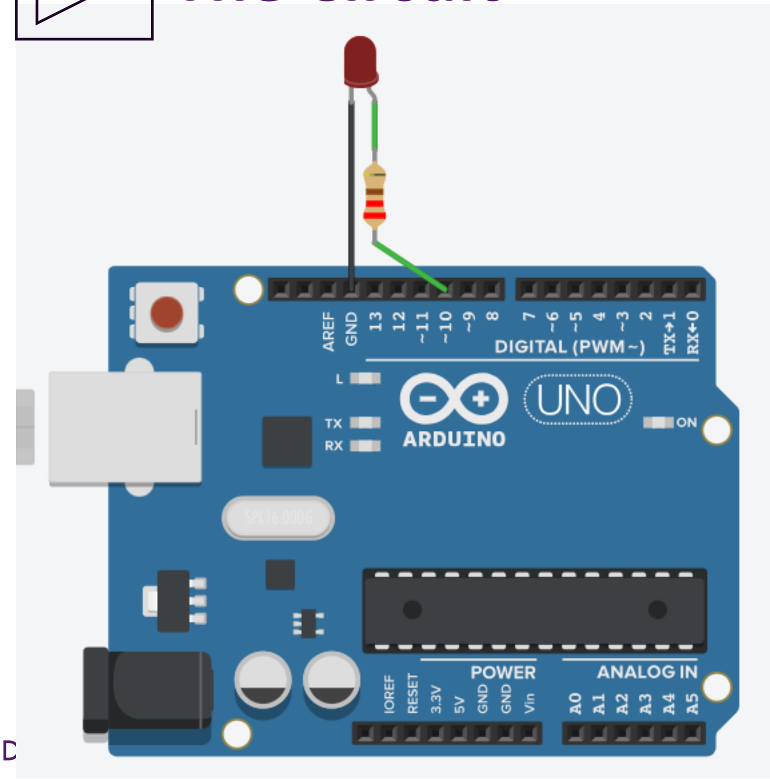
Digit	0	1	2	3	4	5	6	7	8	9

Tolerance	Silver ±10 %	Gold ±5 %	±1 %	±0.5 %	±0.1 %

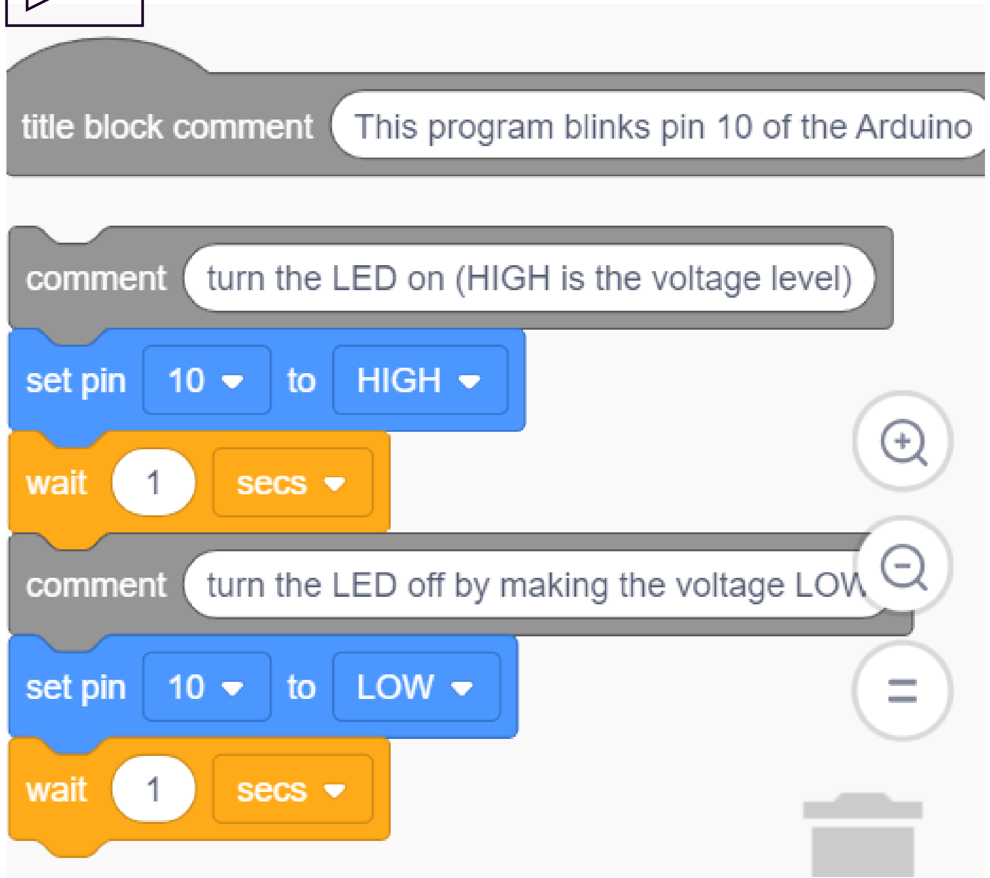
External LED in different Digital pins

The LED can be connected to other digital pins. For example, let’s replace pin 13 in the previous example to pin 10.

The Circuit



The Code



```
title block comment  This program blinks pin 10 of the Arduino

comment  turn the LED on (HIGH is the voltage level)
set pin 10 ▾ to HIGH ▾
wait 1 secs ▾
comment  turn the LED off by making the voltage LOW
set pin 10 ▾ to LOW ▾
wait 1 secs ▾
```

Practice Questions

1. Modify the Hello World with an External LED example, as follows:

a) Make the LED stay 3 seconds ON and 3 seconds OFF.

b) Make the LED stay 200 milliseconds ON and 500 milliseconds OFF.

2. Again, from Hello World with an External LED example, make the necessary changes in the code and circuit so that the LED is placed on Pin 5, and stays 2 seconds ON and 1 second OFF.

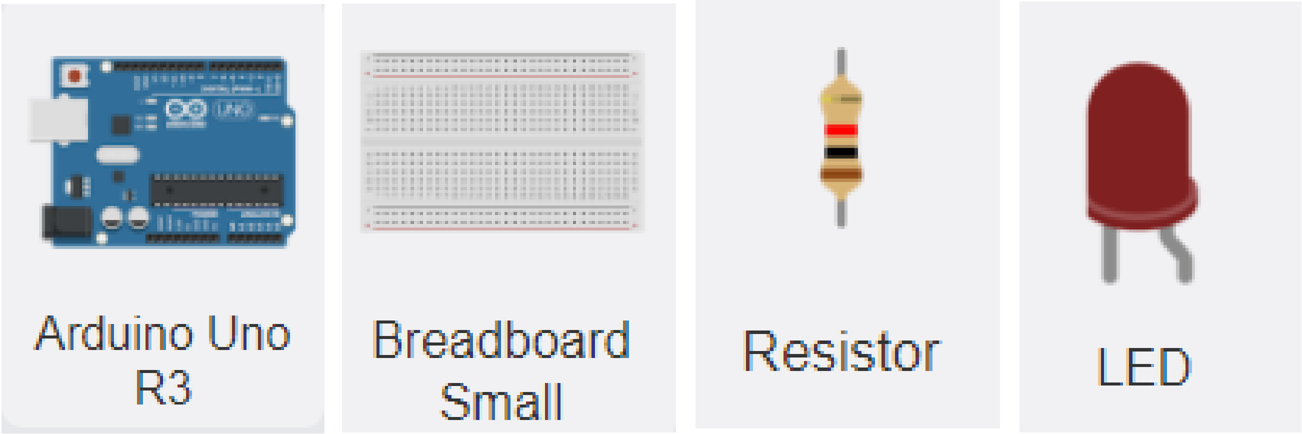
Multiple LEDs

- In this example, you will learn how to drive three loads (LEDs) connected to an Arduino pin.

What will I learn?

- Activate multiple outputs simultaneously.
- Use a breadboard.
- Use Arduino source and ground pins

Components

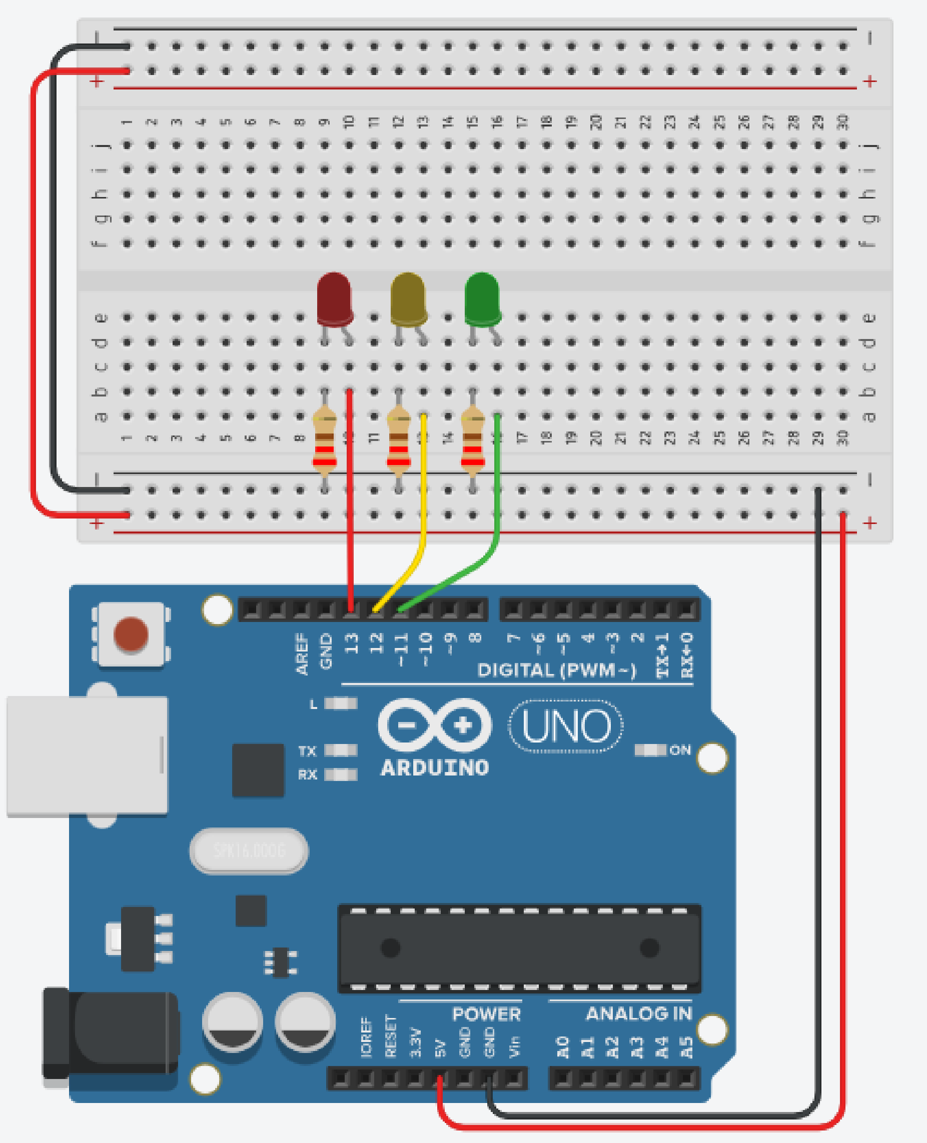


Multiple LEDs

▷ The Circuit

- You will need:
- 1 Arduino Uno board.
- 3 LEDs (Click on it the change its colour).
- 2 Resistors (220Ω).
- 1 Breadboard.

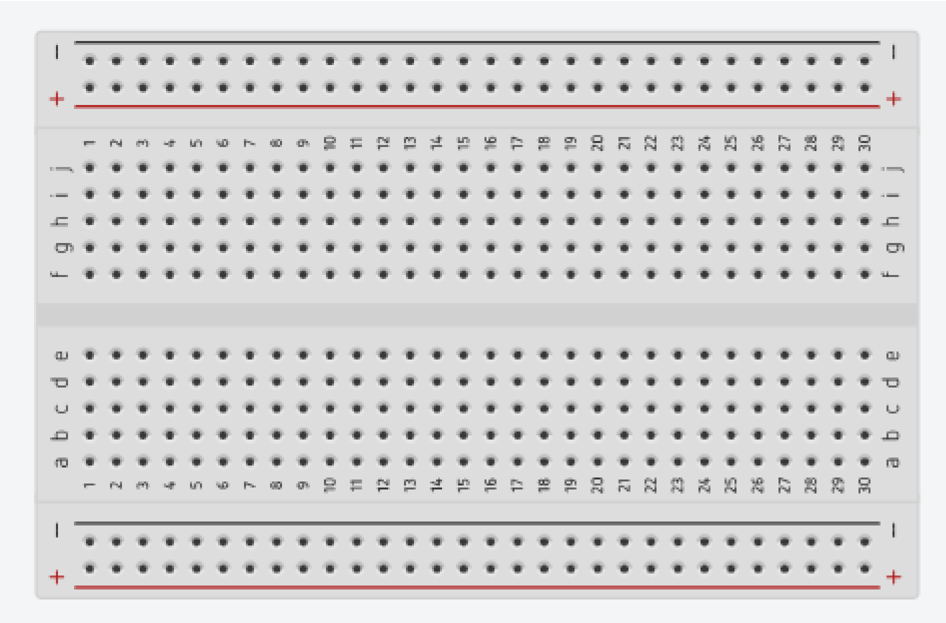
  - Connect all components like in the following image:

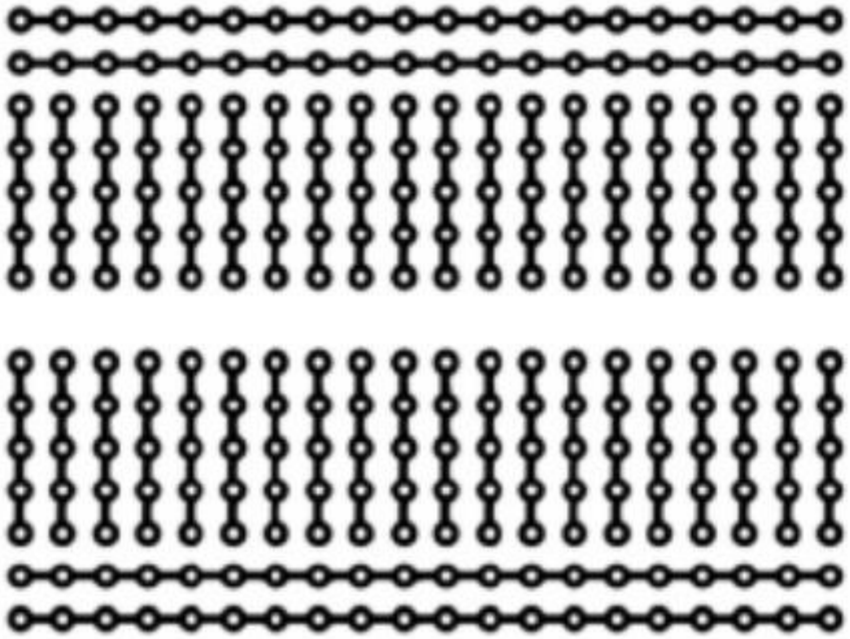


Breadboard

The Circuit

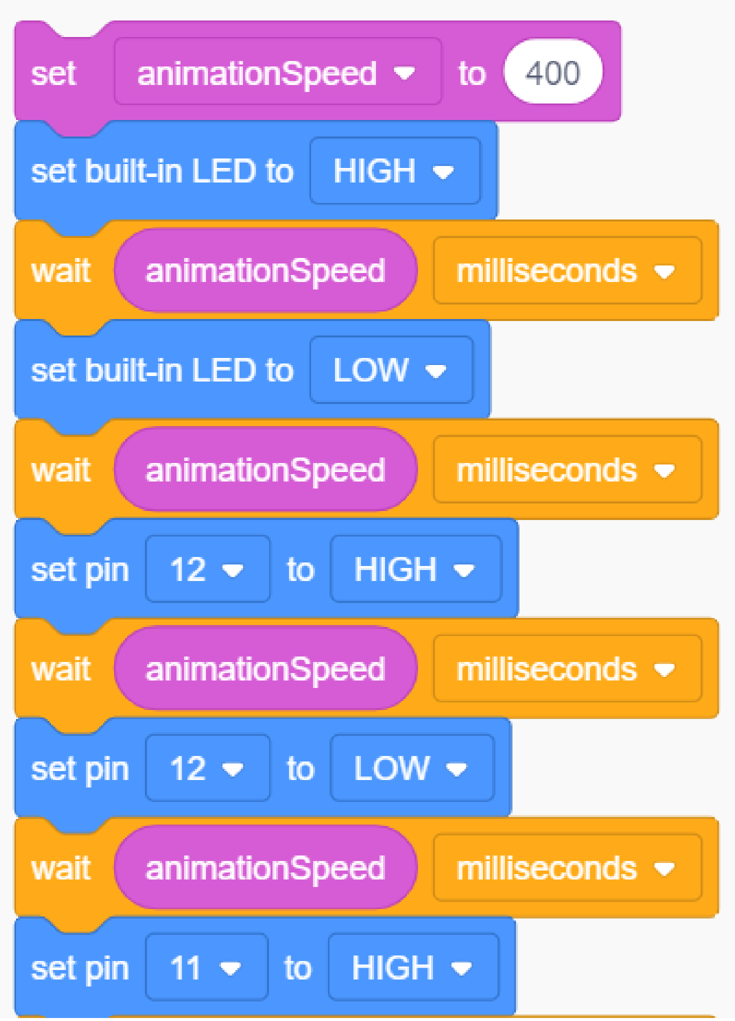
- It is a reusable board used to build electronic circuit prototypes without the need for soldering. Also called protoboard, it is made of perforated plastic blocks and several thin strips of copper metal alloy.



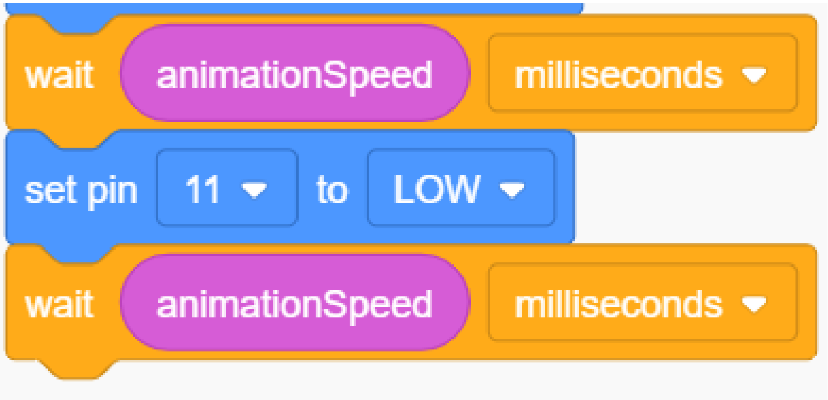


Multiple LEDs

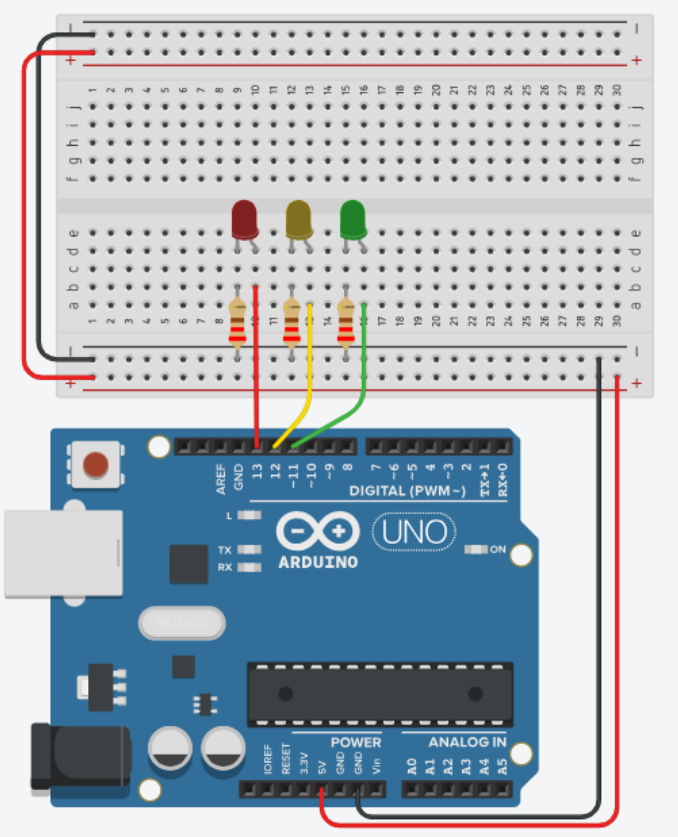
The Code



```
set animationSpeed to 400
set built-in LED to HIGH
wait animationSpeed milliseconds
set built-in LED to LOW
wait animationSpeed milliseconds
set pin 12 to HIGH
wait animationSpeed milliseconds
set pin 12 to LOW
wait animationSpeed milliseconds
set pin 11 to HIGH
```

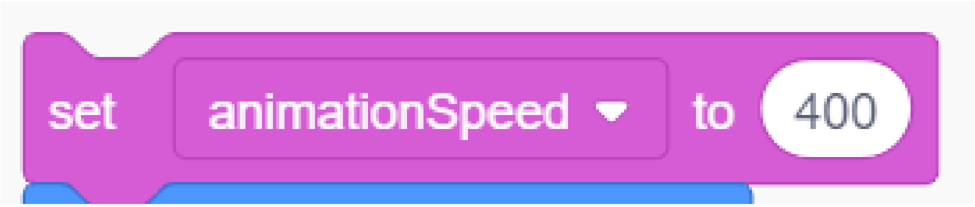


```
wait animationSpeed milliseconds
set pin 11 to LOW
wait animationSpeed milliseconds
```



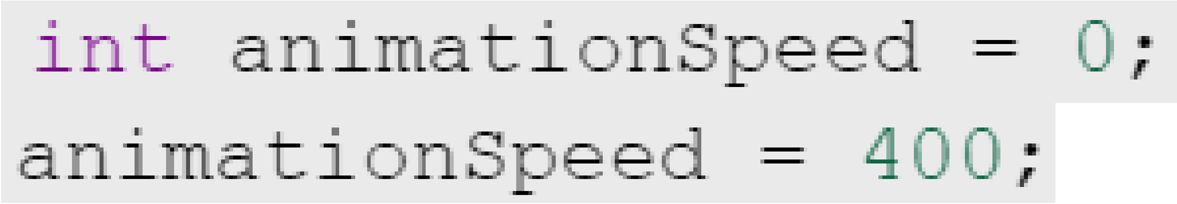
Introduction to variables

- Variables are placeholders in memory, so they are widely used to store data that needs to be used later.
- Variables need to be declared before being used, and for that, it is necessary to assign a specific type, allowing to reserve in memory the necessary space for the information that will be stored.



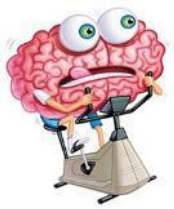
set animationSpeed ▾ to 400

≡



```
int animationSpeed = 0;
animationSpeed = 400;
```

- Integer (int) is a variable type that will hold a numeric (integer) value.



Practice Questions

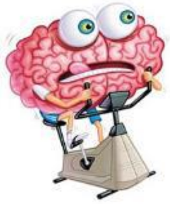
1. Create a Two-Way Traffic Light System. For this project, you need to create two different traffic lights (using three LEDs for each), and they should work as follows:

Traffic Light System 1:

1. Red LED ON;
2. Red LED ON;
3. Red LED ON;
4. Red LED ON;
5. Red LED ON for 0.5 seconds;
6. Yellow LED ON for 1 second;
7. Green LED ON for 4 seconds;
8. Yellow LED ON for 1 second;
9. Repeat.

Traffic Light System 2:

1. Red LED ON for 0.5 seconds;
2. Yellow LED ON for 1 second;
3. Green LED ON for 4 seconds;
4. Yellow LED ON for 1 second;
5. Red LED ON;
6. Red LED ON;
7. Red LED ON;
8. Red LED ON;
9. Repeat.



Practice Questions

1. A set of 4 LEDs connected to pins 2 to 5 will be activated sequentially when powering the Arduino. The LEDs will turn off after 2 seconds. And the process restarted 0.5 seconds later. Create variables to define and hold the waiting time value.

2. Make a sequence of six LEDs light up as follows: those present at BOTH ends of the sequence should already be ON when powering the Arduino. Then, sequentially turn the lights ON, one towards the other, giving the impression that they are meeting. Then, return to the ends giving the impression that they are moving away.

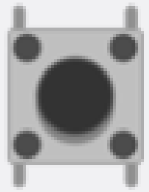
Digital Inputs – Push Button

- The button is a component that connects two circuit points when pressed. In this example, the LED is ON while the button is pressed.

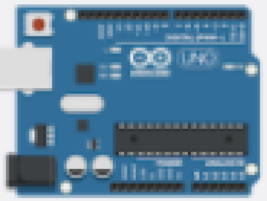
What will I learn?

- Read a digital input.
- Use If conditionals.
- Use pushbuttons.
- Pull Up setting.

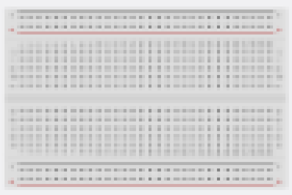
Components



Pushbutton



Arduino Uno R3



Breadboard Small

Resistor

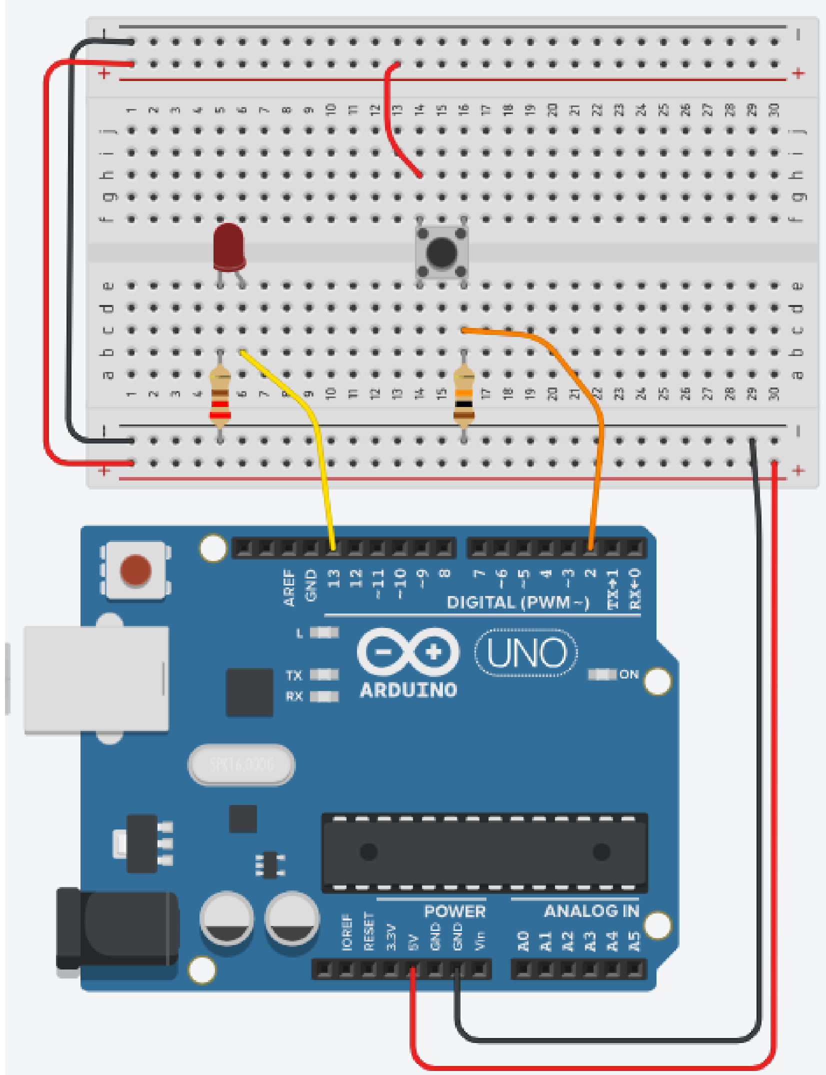
LED

Digital Inputs – Push Button

▷ The Circuit

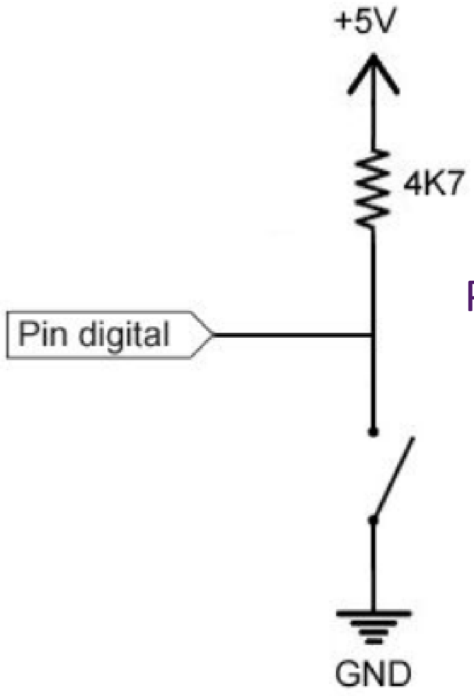
- You will need:
- 1 Arduino Uno board.
- 1 LED.
- 2 Resistors (220Ω and 10kΩ).
- 1 Breadboard.

  - Connect all components like in the following image:



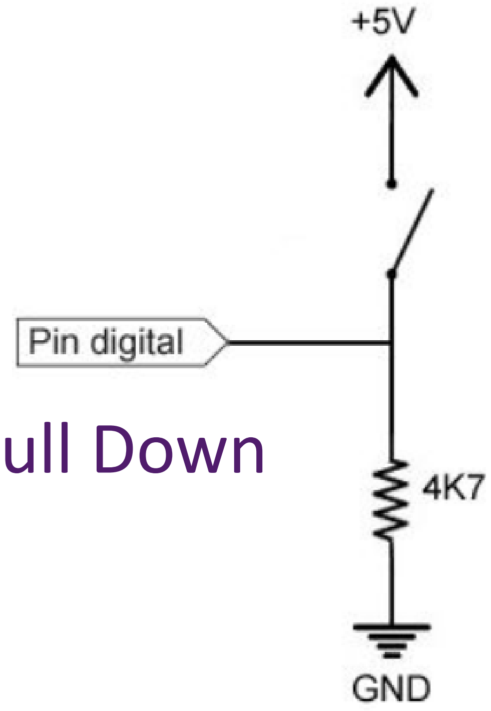
Pull Up and Pull Down

- Normally, when you connect pushbuttons to digital pins on a microcontroller, the pin switches to a floating-point state when the pushbutton is open.
- Floating point: a state is assumed at random. In the case of Arduino, it normally assumes the high level (bit 1).
- Resistors are employed in order to avoid the floating point, fixing a voltage value applied under a pin even when the button is open. When a resistor is connected to the power source (setting the pin high), it is called Pull Up. If it is connected to the ground (setting the pin at a low level), it is called Pull Down.



Pull Up

Pull Up resistors should have high values, e.g., above 10k Ω.

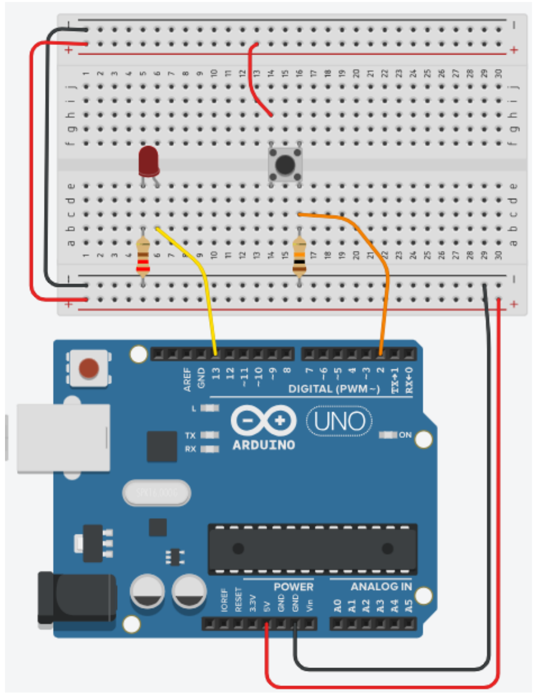


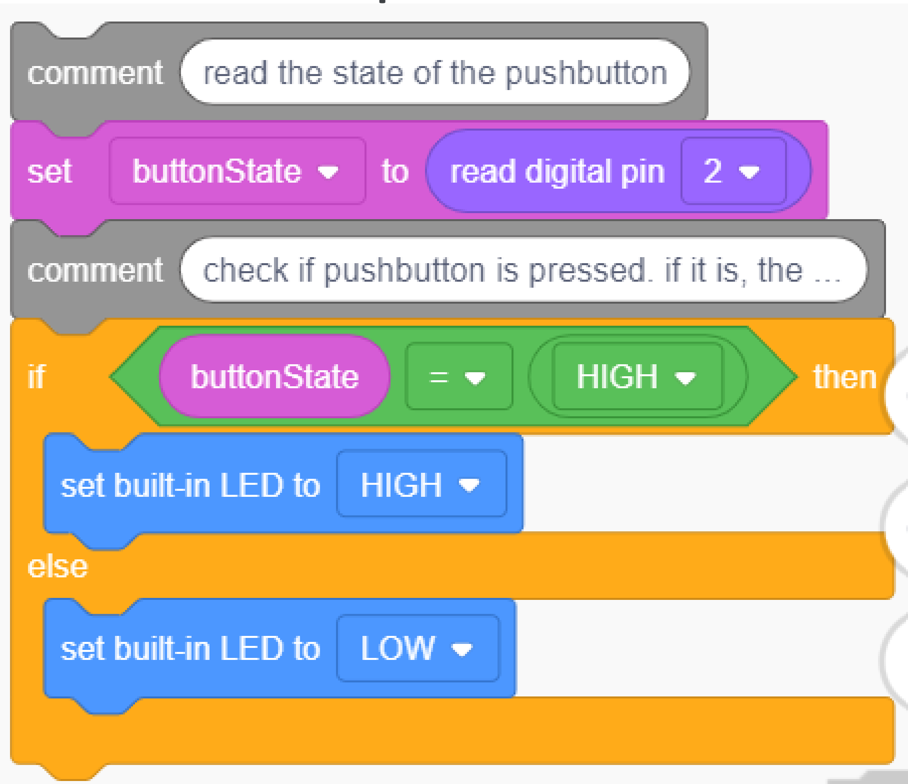
Pull Down

Digital Inputs – Push Button

The Code

- *if conditional*: enables to execute different commands based on a condition.
- *set variable (Pink)*: modify the value of a variable.
- *read digital pin Input Command (Purple)*: read the value the pin receives.

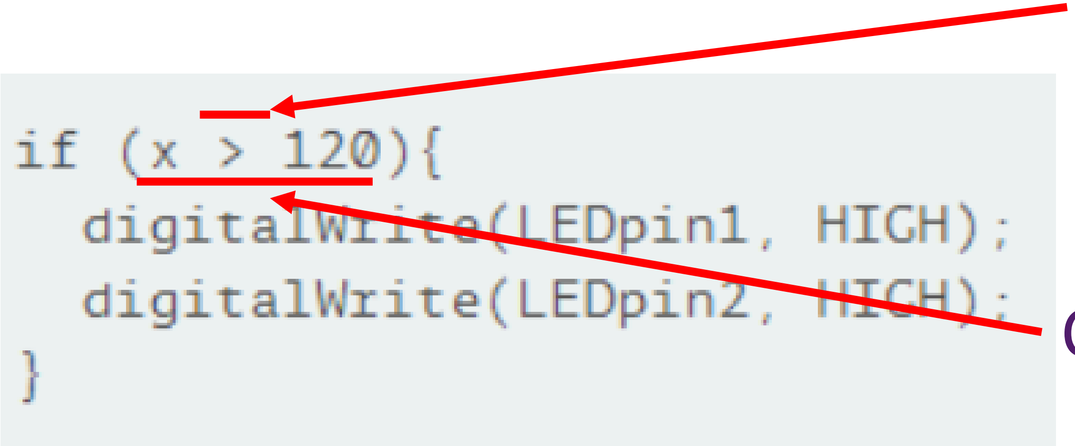




```
comment  read the state of the pushbutton
set  buttonState ▾  to  read digital pin  2 ▾
comment  check if pushbutton is pressed. if it is, the ...
if  buttonState  = ▾  HIGH ▾  then
    set built-in LED to  HIGH ▾
else
    set built-in LED to  LOW ▾
```

If conditional

- The if conditional is used to test whether a given condition is true. For example, check whether an input pin's value is above a certain number. It is usually used in conjunction with operators.



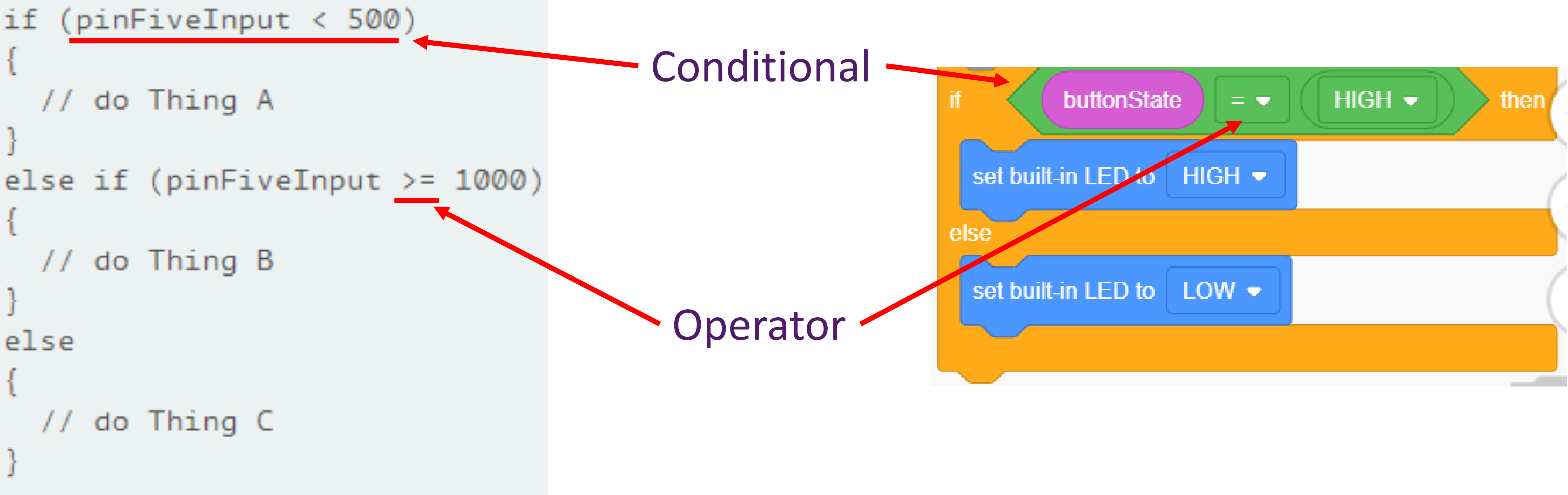
```
if (x > 120){
  digitalWrite(LEDpin1, HIGH);
  digitalWrite(LEDpin2, HIGH);
}
```

Operator

Conditional

If – else conditional

- When the if condition is false, the else is executed instead. We can also define multiple conditions using the else if.



```
if (pinFiveInput < 500)
{
  // do Thing A
}
else if (pinFiveInput >= 1000)
{
  // do Thing B
}
else
{
  // do Thing C
}
```

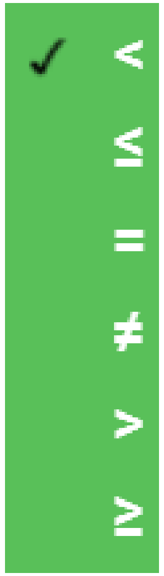
Conditional

Operator

```
if buttonState = HIGH then
  set built-in LED to HIGH
else
  set built-in LED to LOW
```

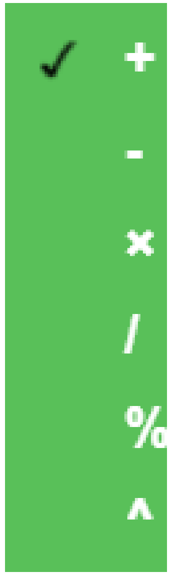
Comparison and Arithmetic Operators

- Operators are useful not only to perform operations but to build conditionals.



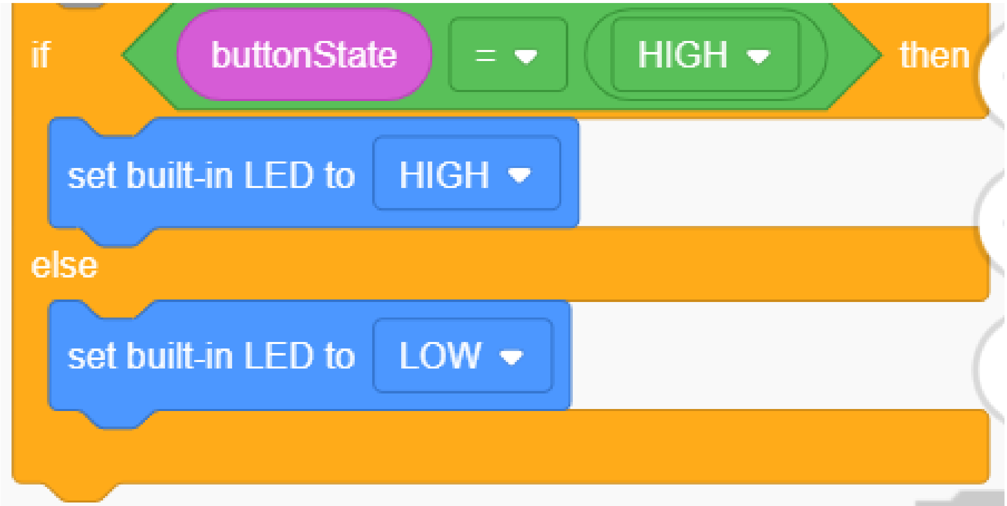
✓ <	Less than
≤	Less than or equal to
=	Equal to
≠	Not equal to
>	Less than
≥	Less than or equal to

Comparison operators

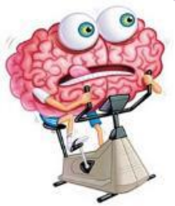


✓ +	Addition
-	Subtraction
×	Multiplication
/	Division
%	Modulus
^	Power

Arithmetic operators



```
if buttonState = HIGH then
    set built-in LED to HIGH
else
    set built-in LED to LOW
```



Practice Questions

1. Button and LED: When pressing a button connected to pin 8, an LED connected to pin 10 should turn ON in such a way that it should remain ON for 5 seconds, and after that time, the LED will turn OFF.
2. LED Bar: A set of 4 LEDs connected to pins 2 to 5 will be activated sequentially after button 1 (connected to pin 10) is pressed.
3. LED Bar: A set of 4 LEDs will activate sequentially after button 1 is pressed. The LEDs will be deactivated by a button 2.
4. Create a circuit with one LED and two buttons. The LED should be connected to pin 13, and the buttons to any digital pin you choose. Your system should turn ON the LED when pressing buttons 1 or 2.
5. Create a circuit with two LEDs and two buttons. Each LED should be turned ON by one of the buttons.

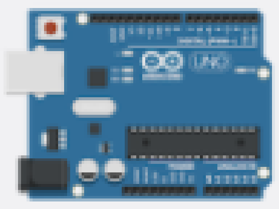
Analogue Inputs

- In this example, you will learn how to read an analogue input. The value read from the input will be used to control the blinking time of the onboard LED using a potentiometer. The potentiometer is a resistor with a varying resistance.

What will I learn?

- Read an analogue input.
- Use a potentiometer.

Components



Arduino Uno R3

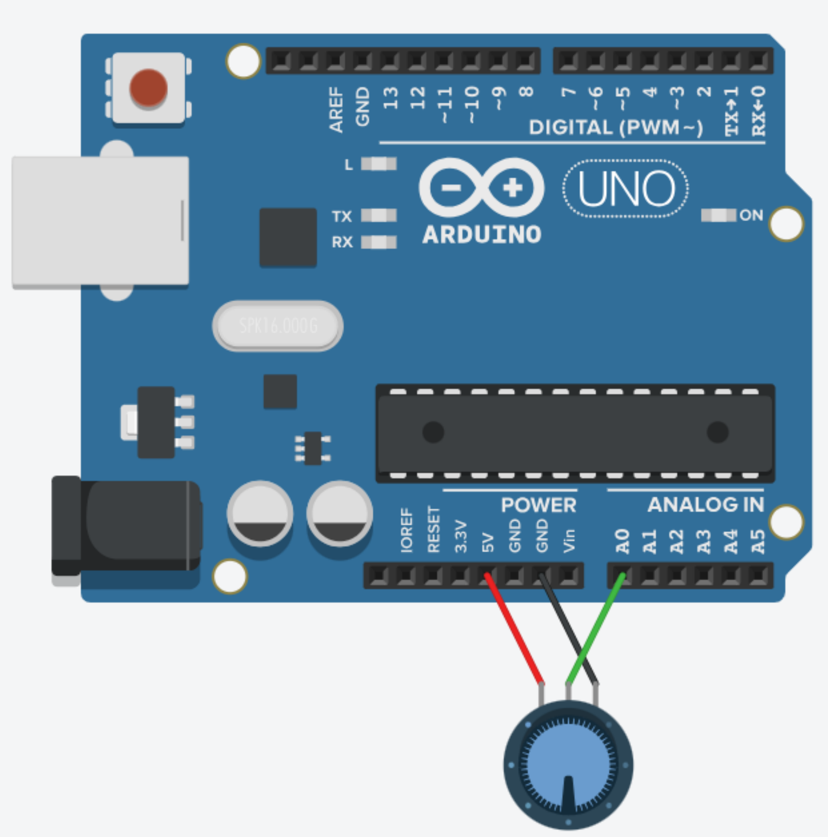
Potentiometer

Analogue Inputs

▷ The Circuit

- A potentiometer is a resistor whose value is variable, suitable for use as a control element in electronic devices. The resistance between the centre and end terminals varies according to the position of the centre control switch.

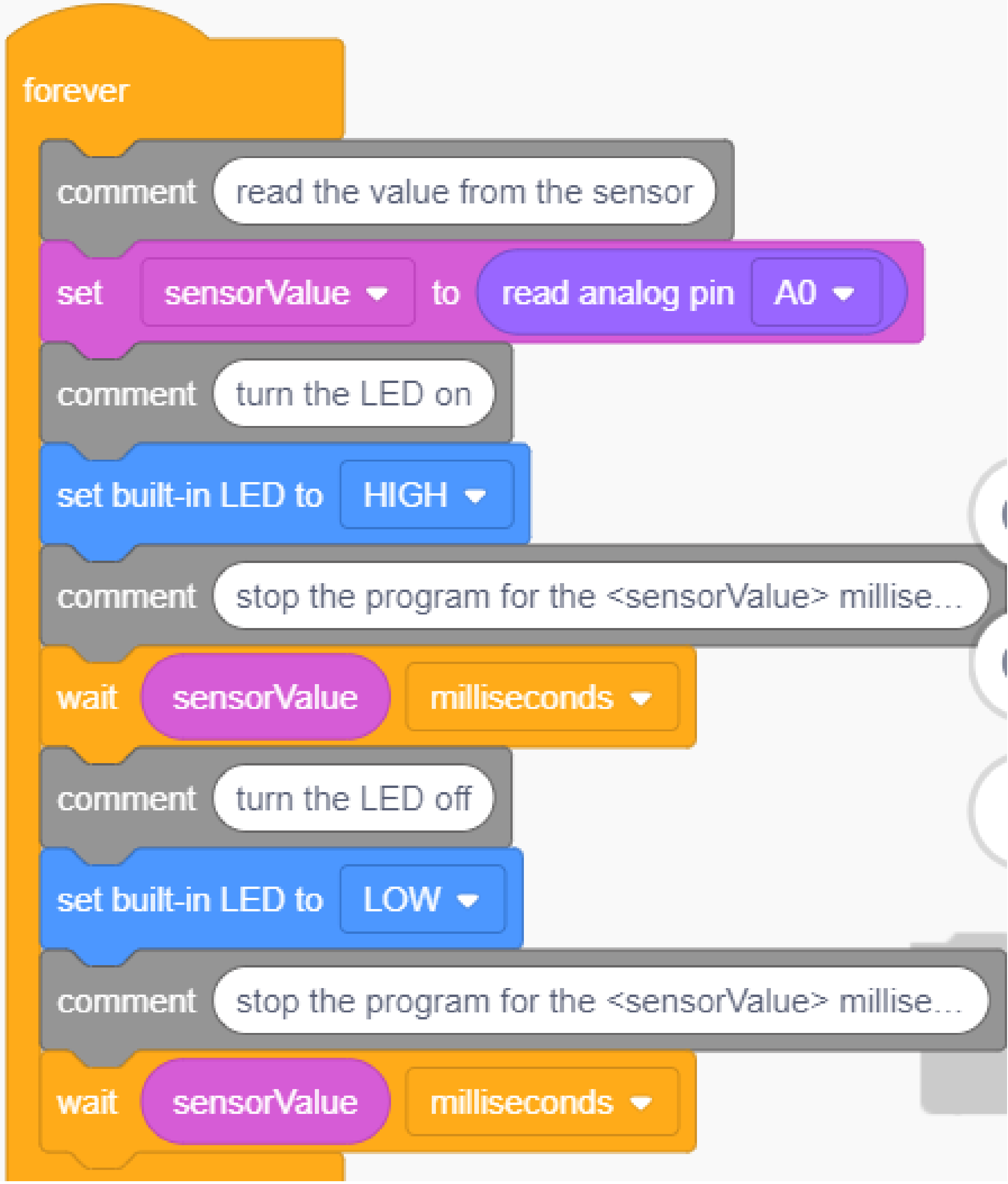
- Note that the end terminals are connected to the source (5V) and ground (GND), while the central terminal is connected to an Arduino analogue pin.



Analogue Inputs

The Code

- Initially, the analogue pin is read, and its value is stored in the variable “sensorValue.”
- The onboard LED (pin 13) turns ON.
- Pause the code for “sensorValue” milliseconds. This pause allows us to continue seeing the LED ON.
- The onboard LED (pin 13) turns OFF.
- Pause the code for “sensorValue” milliseconds. This pause allows us to continue seeing the LED OFF.
- Repeat.



```
forever
    comment read the value from the sensor
    set sensorValue ▾ to read analog pin A0 ▾
    comment turn the LED on
    set built-in LED to HIGH ▾
    comment stop the program for the <sensorValue> millise...
    wait sensorValue milliseconds ▾
    comment turn the LED off
    set built-in LED to LOW ▾
    comment stop the program for the <sensorValue> millise...
    wait sensorValue milliseconds ▾
```

Analogue to Digital Converter (ADC)

- The Arduino UNO has 1 ADC to convert the signals inserted in pins A0 to A5. This ADC has a 10-bit resolution and accepts voltages ranging from 0V to 5V (it does not accept negative values).

- The voltage values read will be converted into a range from 0 to 1023, as 2^{10} = 1024. Thus:

$$5V - 1023$$

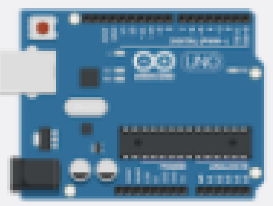
Serial Monitor

- In this example, you will learn how to display data using the serial monitor. As the name suggests, the serial monitors the data transferred between the Arduino and the computer.

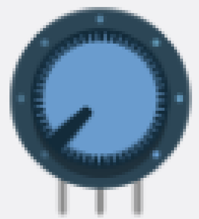
What will I learn?

- Use the serial monitor.

Components



Arduino Uno R3



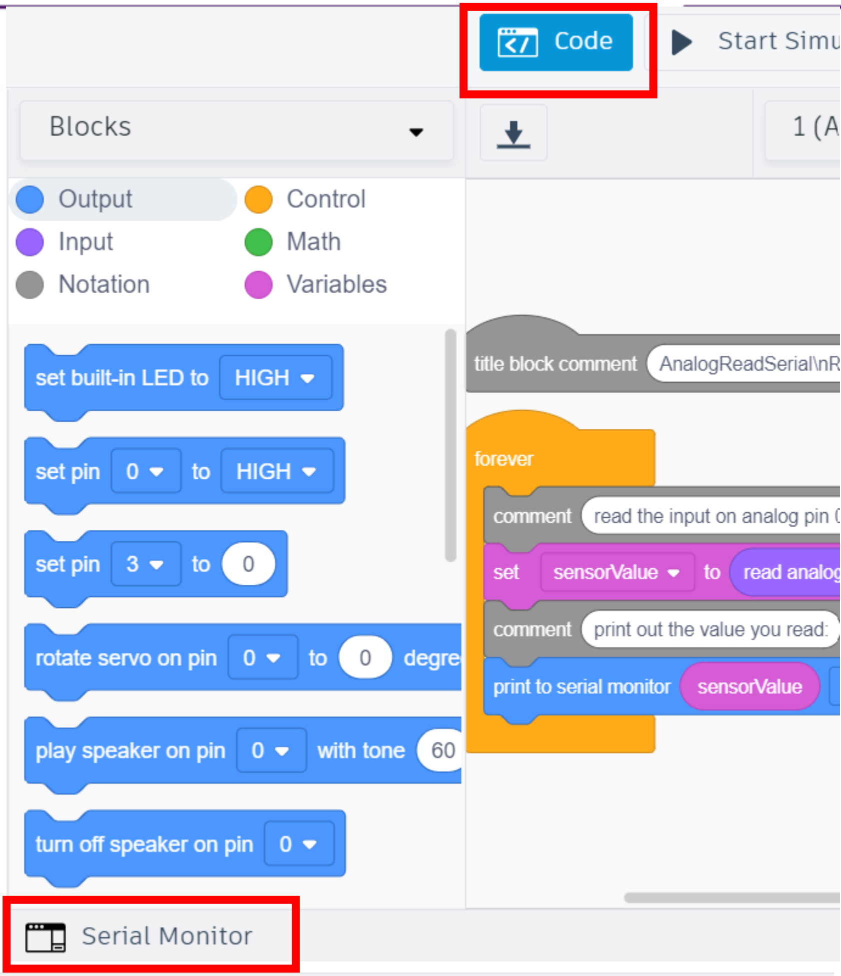
Potentiometer

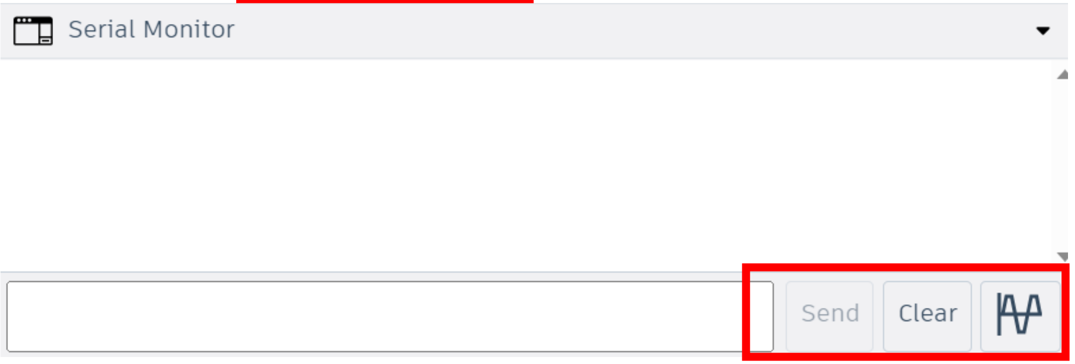
Serial Monitor

- To access the serial monitor, click on Code and Serial Monitor.

- You can use the Send and Clear buttons to send information and delete all data shown on the monitor.

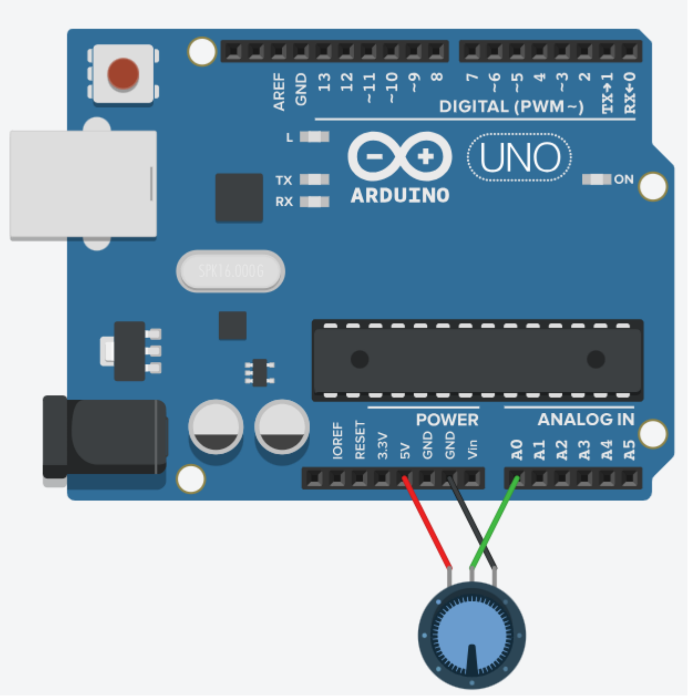
- The graph button shows the values in the form of a graph.


Code
Start Simu
Blocks
1 (A
Output
Control
Input
Math
Notation
Variables
set built-in LED to HIGH
set pin 0 to HIGH
set pin 3 to 0
rotate servo on pin 0 to 0 degre
play speaker on pin 0 with tone 60
turn off speaker on pin 0
title block comment AnalogReadSerial\nR
forever
comment read the input on analog pin
set sensorValue to read analog
comment print out the value you read:
print to serial monitor sensorValue
Serial Monitor

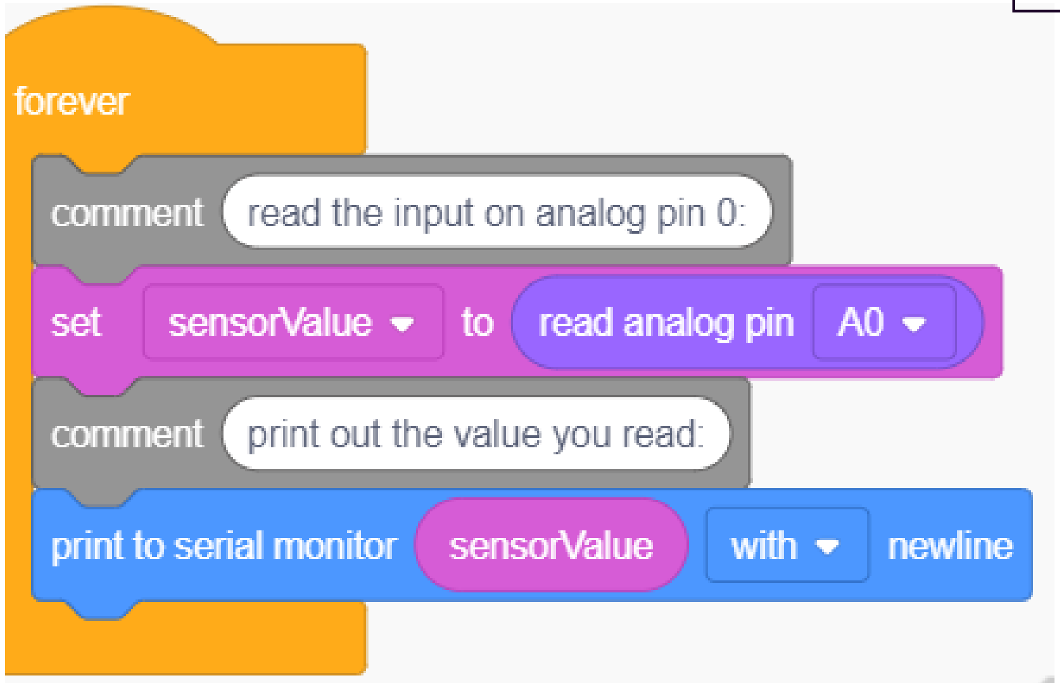


Serial Monitor

The Circuit

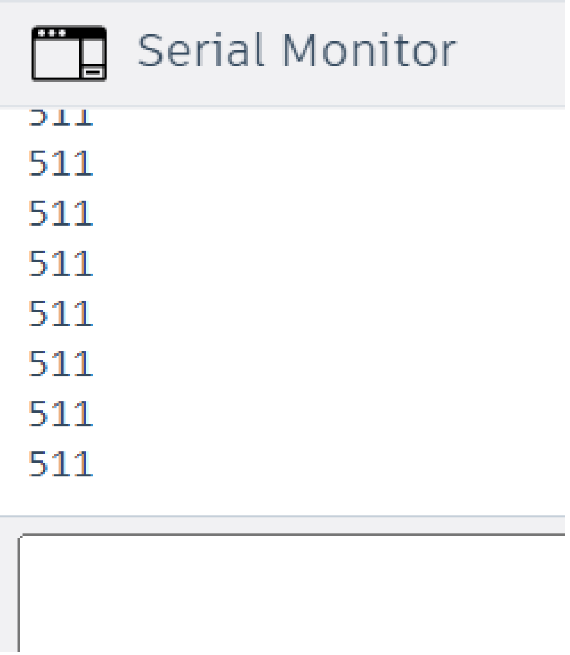


The Code



```
forever
    comment  read the input on analog pin 0:
    set  sensorValue ▾  to  read analog pin  A0 ▾
    comment  print out the value you read:
    print to serial monitor  sensorValue  with ▾  newline
```

The Monitor

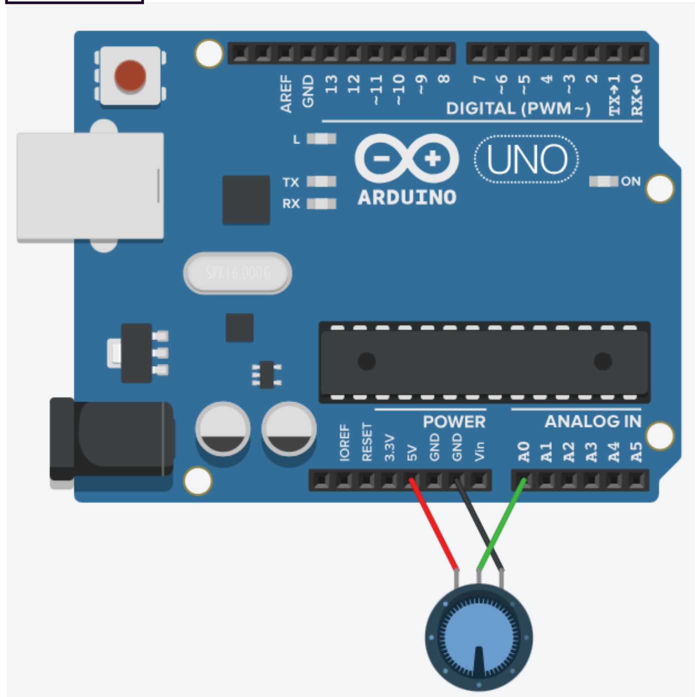


```
511
511
511
511
511
511
511
511
```

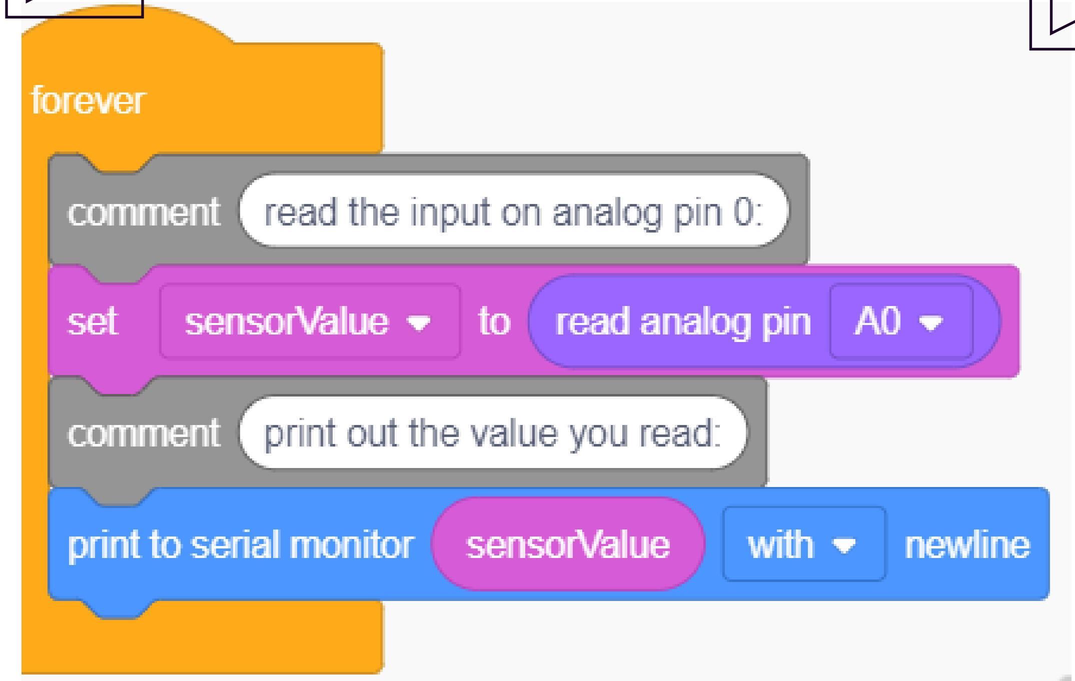
- *<u>print to serial monitor Output command (blue)</u>*: enables sending data to the serial monitor. The newline option skips to the next line after printing the data.

Serial Monitor

The Circuit

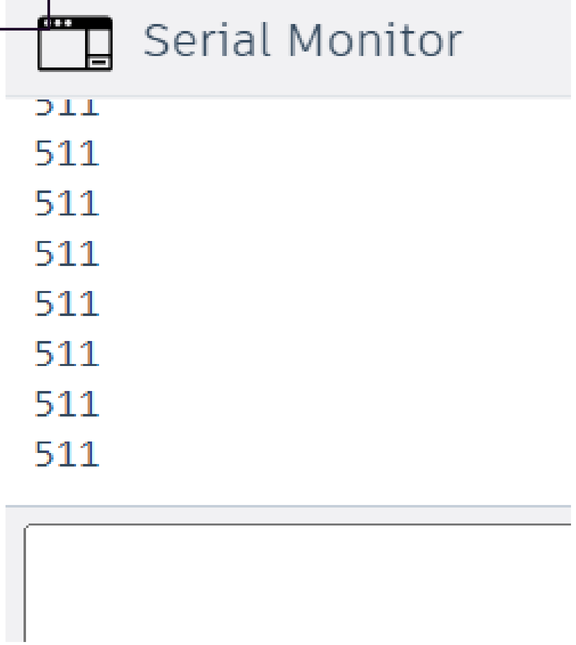


The Code



```
forever
  comment read the input on analog pin 0:
  set sensorValue ▾ to read analog pin A0 ▾
  comment print out the value you read:
  print to serial monitor sensorValue with ▾ newline
```

The Monitor



```
511
511
511
511
511
511
511
511
```

- As you rotate the central terminal of the potentiometer towards 5V, the value approaches 1023. Meanwhile, as you rotate it towards the GND, the value approaches 0.

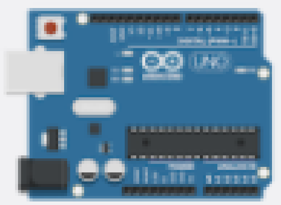
Analogue Outputs

- In this example, you will learn how to write an analogue output. For the Arduino Uno, an analogue output means using a Pulse Width Modulation (PMW). For this purpose, only digital pins with the tilde symbol can be used.

What will I learn?

- Write an analogue output.
- Learn the count code block.

Components



Arduino Uno R3

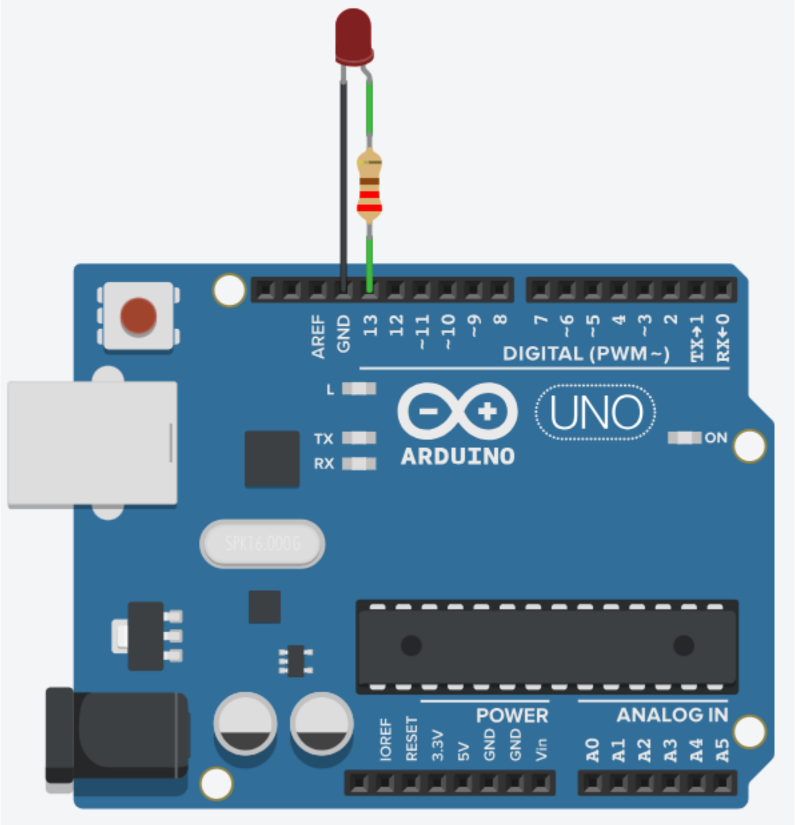
Resistor

LED

Analogue Outputs

The Circuit

- This is the same circuit we have seen for the Hello World with an External LED example.

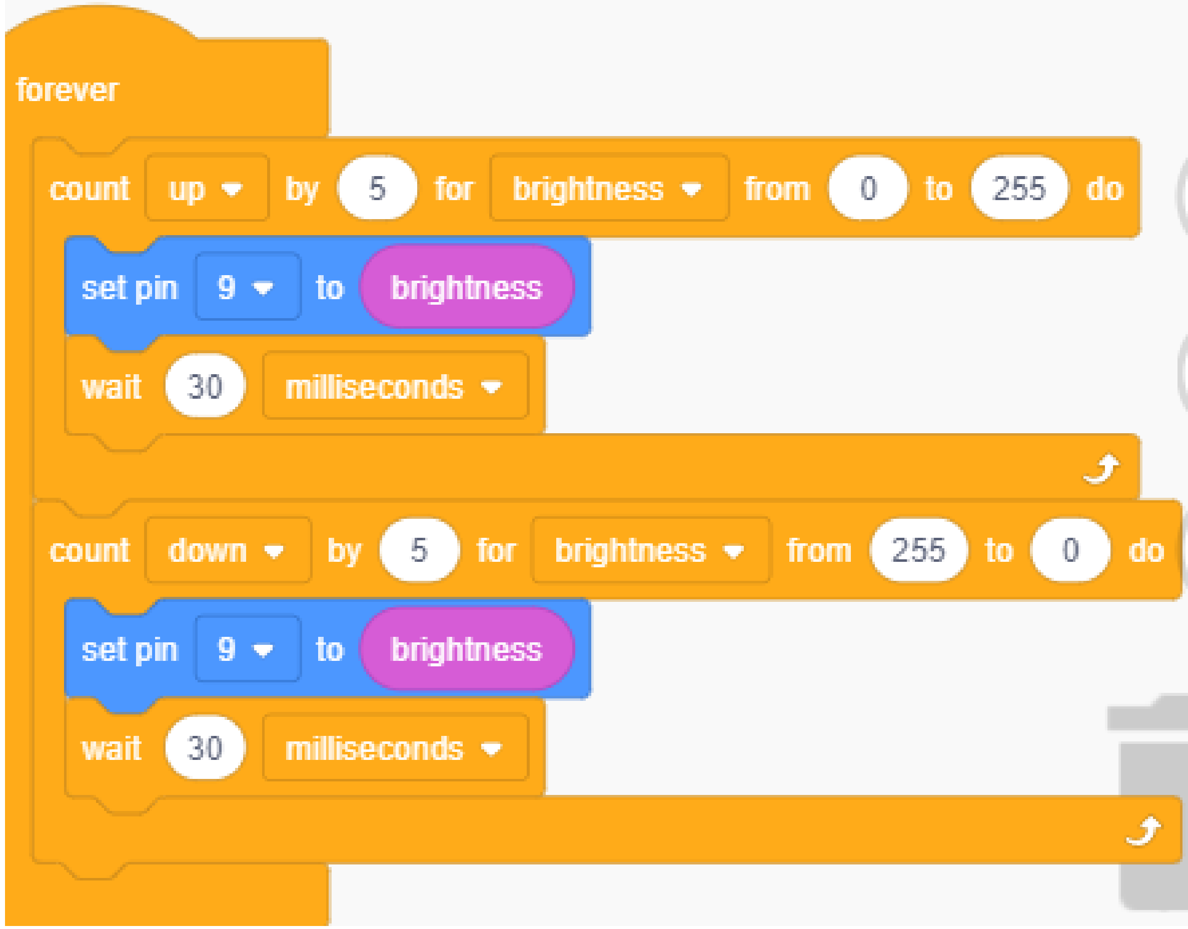


Analogue Outputs

The Code

- *count control command (yellow)*: counts (up or down) the changing step of a variable value. In addition, it implements a for loop that defines the range of values the variable value can assume.

- This creates a PWM. A PWM changes the voltage applied to a load allowing us to control its parameters. For example, if the load is an LED, we can control its brightness.



```
forever
  count up by 5 for brightness from 0 to 255 do
    set pin 9 to brightness
    wait 30 milliseconds
  count down by 5 for brightness from 255 to 0 do
    set pin 9 to brightness
    wait 30 milliseconds
```

Summary

We introduced the Arduino Platform and Microcontrollers. ✓

We simulated prototypes to practice digital and analogue input/outputs. ✓

We learned the electronic components: LED, resistor, potentiometer, breadboard, and pushbutton. ✓

We learned programming: creating variables, if conditionals, for loops, reading and writing to digital pins. ✓

We learned how to send data to the serial monitor ✓

4. Arduino: Sensors and Actuators

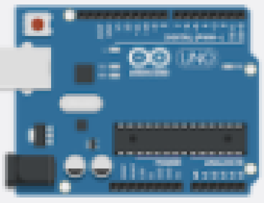
Light-Dependent Resistor (LDR)

- In this example, you will learn how to read the LDR sensor. LDR is a photoresistor whose resistance value varies according to the incident light. It has a low resistance value in the presence of light and a high value in its absence.

What will I learn?

- Write an analogue output.
- Use and LDR.

Components



Arduino Uno R3

Photoresistor

Resistor

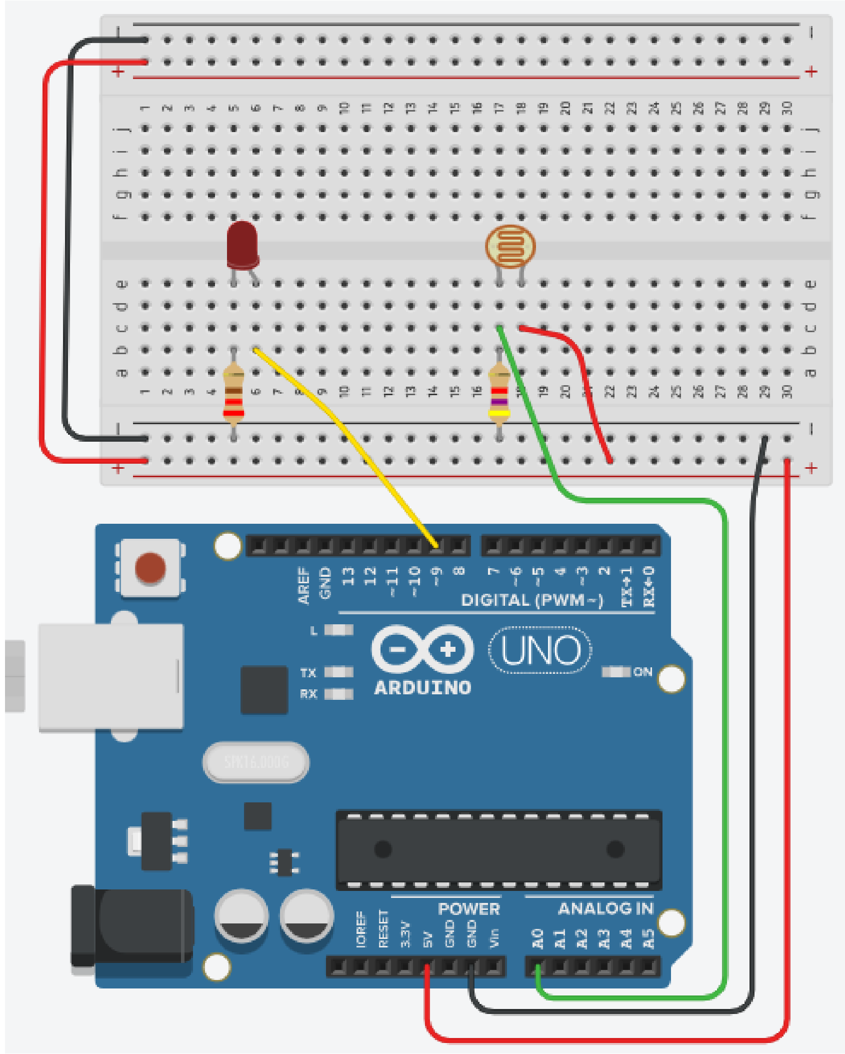
LED

LDR Sensor

The Circuit

- You will need:
- 1 Arduino Uno board.
- 1 LED.
- 2 Resistors (220Ω for the LED and 4.7kΩ for the LDR).
- 1 Breadboard.

- Connect all components like in the following image:

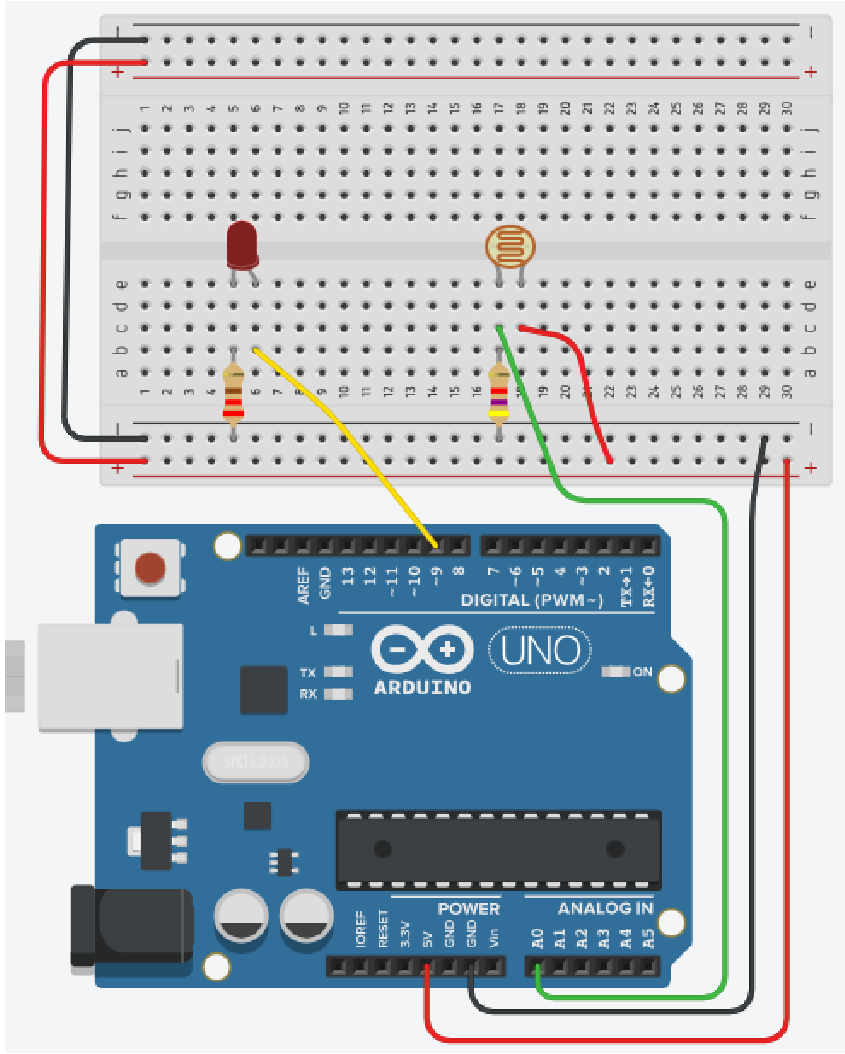


LDR Sensor

▷ The Circuit

- Note that the LDR has two terminals and can be connected in any way, like any other resistor.

- The LDR must be connected in series with another resistor and connect the Arduino pin in between them.



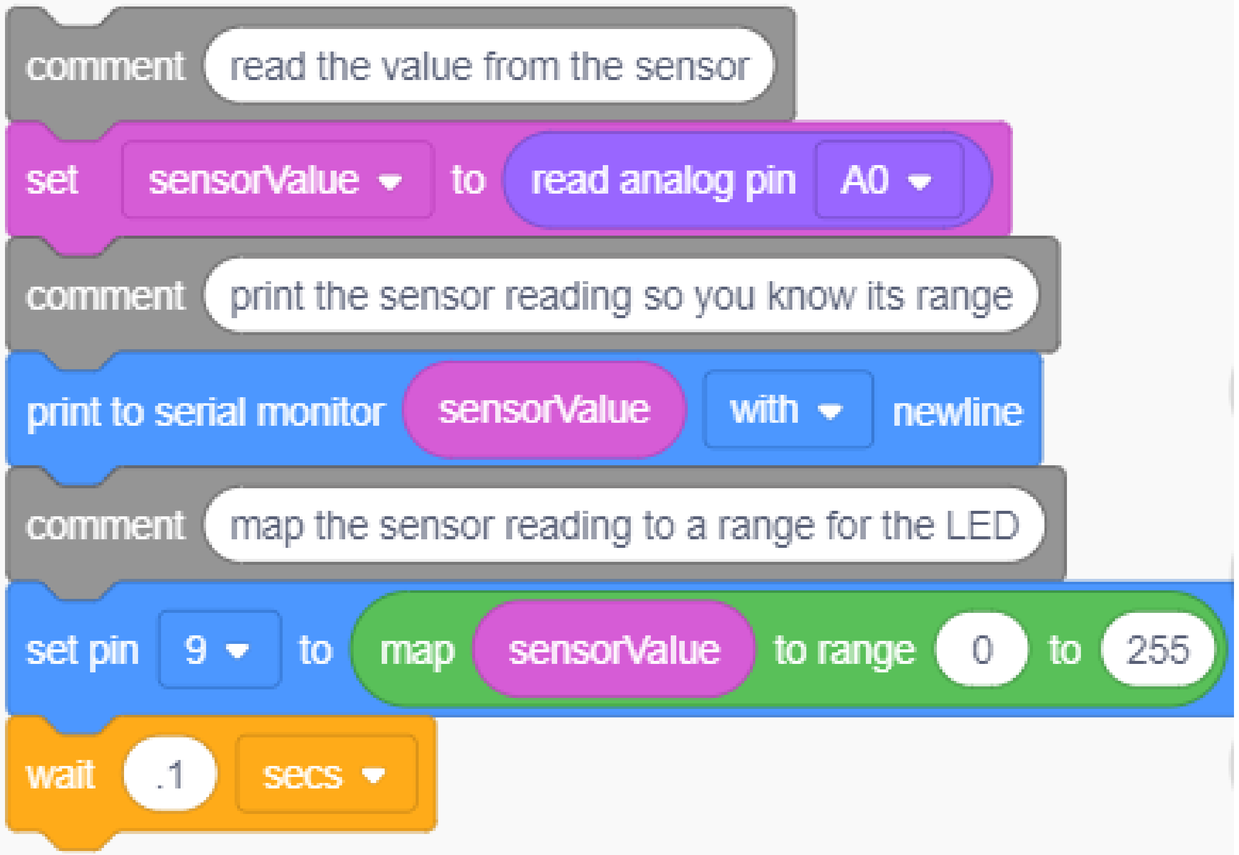
LDR Sensor

The Code

- The map function converts any value to a desirable range.

- In this example, we are converting ‘sensorValue’ in the range 0 to 255, as follows:

$$\frac{sensorValue}{255}$$



```
comment  read the value from the sensor
set  sensorValue ▾  to  read analog pin  A0 ▾
comment  print the sensor reading so you know its range
print to serial monitor  sensorValue  with ▾  newline
comment  map the sensor reading to a range for the LED
set pin  9 ▾  to  map  sensorValue  to range  0  to  255
wait  .1  secs ▾
```

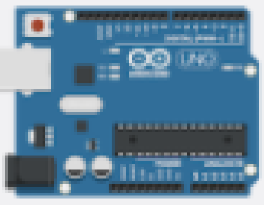
Ultrasonic Sensor

- In this example, you will learn how to read the Ultrasonic sensor. It allows measuring distances based on sound signals.

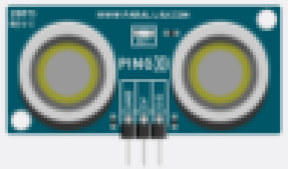
What will I learn?

- Use an Ultrasonic sensor.

Components



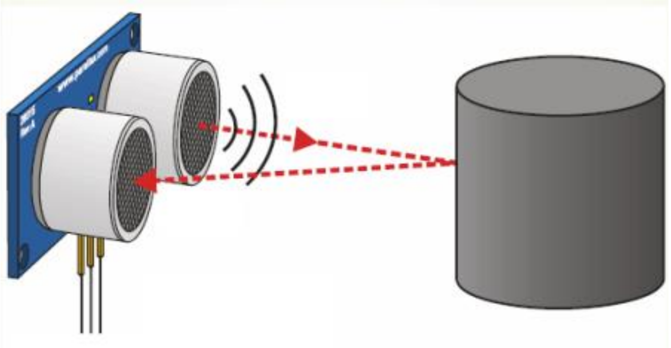
Arduino Uno R3



Ultrasonic Distance...

Ultrasonic Sensor

- The ultrasonic distance sensor consists of a transmitter and receiver capable of measuring distances based on the speed of sound in the air.

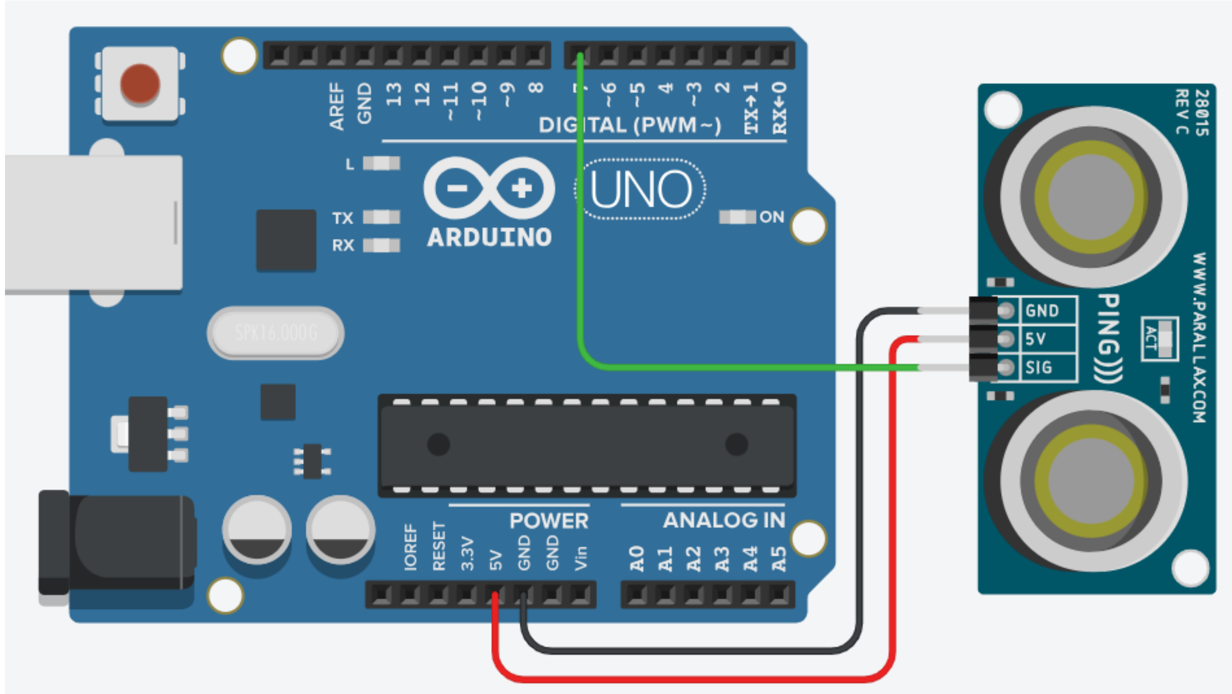


- When given a pulse on the trigger pin, the transmitter emits a sound signal. When hitting an object, this sound signal is reflected to the sensor, where the receiver captures it (echo pin). The time taken from the transmission of the sound signal to its capture by the receiver can be used to determine the object's distance based on the sound propagation speed in the air (340m/s).

Ultrasonic Sensor

The Circuit

- In this example, the transmitter and receiver are represented by the signal (SIG) pin, which is connected to pin 7.

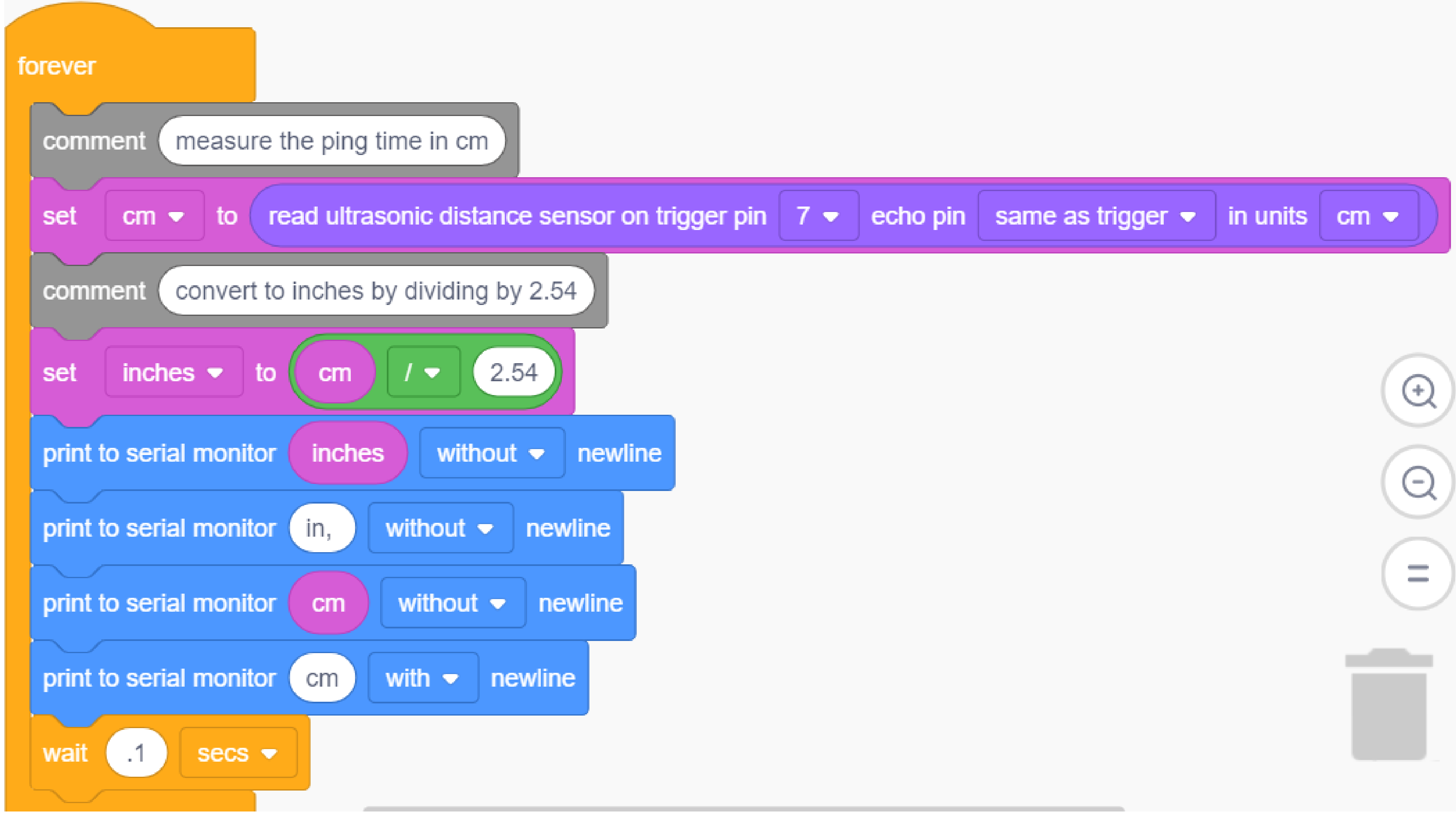


- Note the need to connect to the 5V and GND.

Ultrasonic Sensor

The Code

- The read ultrasonic input command reads the object's distance in cm.
- The value in cm is then converted to inches and stored in the variable *inches*.



```
forever
    comment  measure the ping time in cm
    set  cm ▾  to  read ultrasonic distance sensor on trigger pin  7 ▾  echo pin  same as trigger ▾  in units  cm ▾
    comment  convert to inches by dividing by 2.54
    set  inches ▾  to  ( cm  / ▾  2.54 )
    print to serial monitor  inches  without ▾  newline
    print to serial monitor  in,  without ▾  newline
    print to serial monitor  cm  without ▾  newline
    print to serial monitor  cm  with ▾  newline
    wait  .1  secs ▾
```

Practice Questions

1. Use an Ultrasound sensor to light up 2 LEDs according to the distance between an object and the sensor. If the object is close, a red LED should turn ON; otherwise, a green LED should turn ON.
2. Repeat the previous question and replace the red LED with a buzzer (search on Tinkercad for Piezo).
3. Now, use an LDR sensor to light up 2 LEDs according to the light’s intensity. If the intensity is low, a red LED should turn ON; otherwise, a green LED should turn ON.

Summary

We introduced the Arduino Platform and Microcontrollers. ✓

We simulated prototypes to practice digital and analogue input/outputs. ✓

We learned the electronic components: LED, resistor, potentiometer, breadboard, and pushbutton. ✓

We learned the programming: creating variables, if conditionals, reading and writing to digital pins. ✓

We learned how to send data to the serial monitor ✓

Thank you!

Leonardo.Alves-Dias@warwick.ac.uk