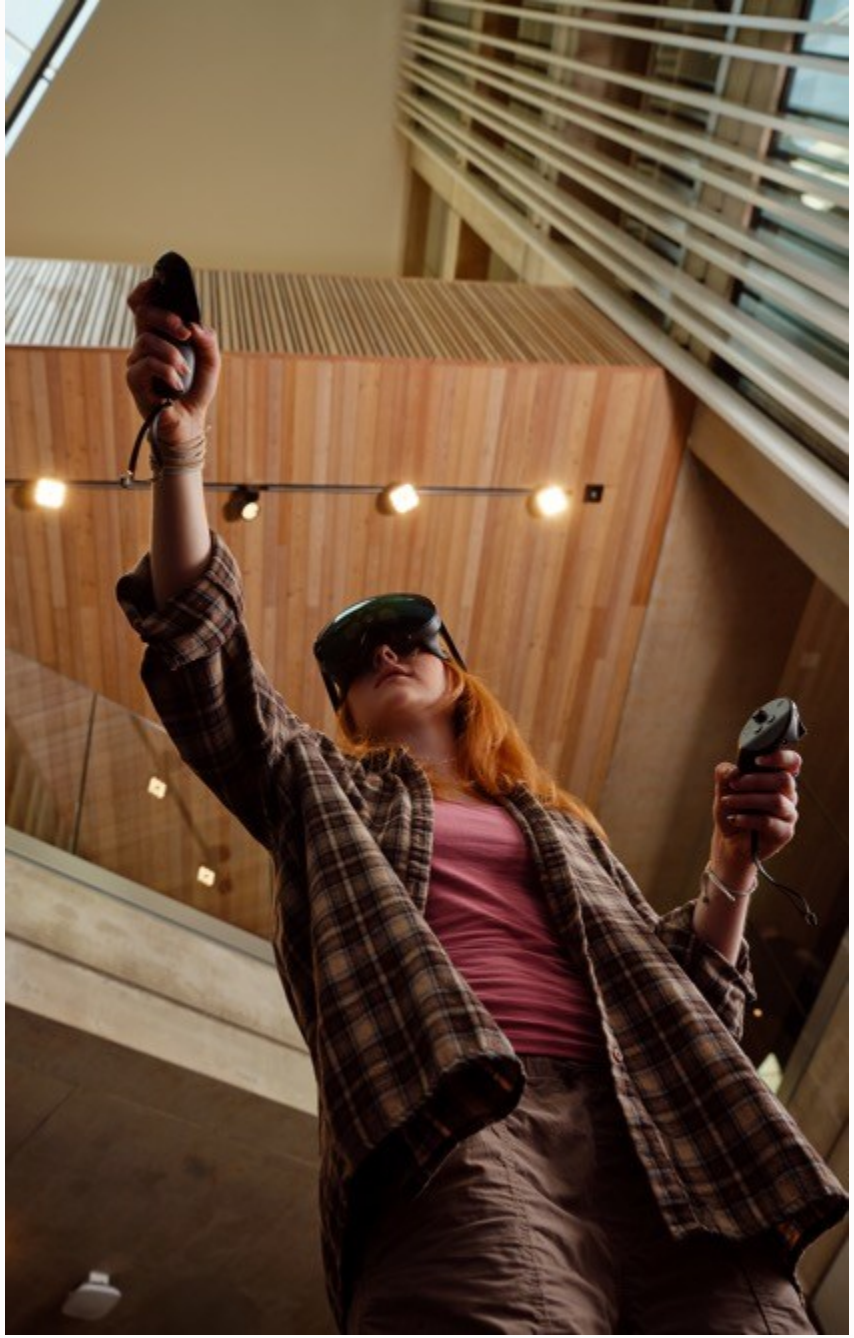
A Taste of Science and Engineering

Dr Leonardo Alves Dias



UNIVERSITY OF WARWICK

Agenda

01 Introduction to Arduino Platform

02 Programming Arduino: Logical Blocks

03 Arduino: Simulation Platforms

Learning Outcomes

01 Understand and explain concepts of microcontrollers.

02 Understand basic concepts of electronic devices.

03 Understand basic concepts of visual programming using block-based programming.

04 Simulate electronic prototypes.

Recommended Reading

01 Misra, Y. 2021, Programming and interfacing with Arduino, 1st. edn, CRC Press, Boca Raton. Available at: https://go.exlibris.link/VDHgB6pt

02 Barrett, Steven F. Arduino: Getting Started. vol. 58;58.;, Morgan & Claypool, San Rafael, California, 2020. Available at: Arduino I: Getting Started | Springer books | IEEE Xplore (warwick.ac.uk)

03 Langbridge, James A.. Arduino Sketches : Tools and Techniques for Programming Wizardry, John Wiley & Sons, Incorporated, 2015. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/warw/detail.action?docID=1895751.

04 Hughes, J. M.. Arduino: a Technical Reference : A Handbook for Technicians, Engineers, and Makers, O'Reilly Media, Incorporated, 2016. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/warw/detail.action?docID=4543968.

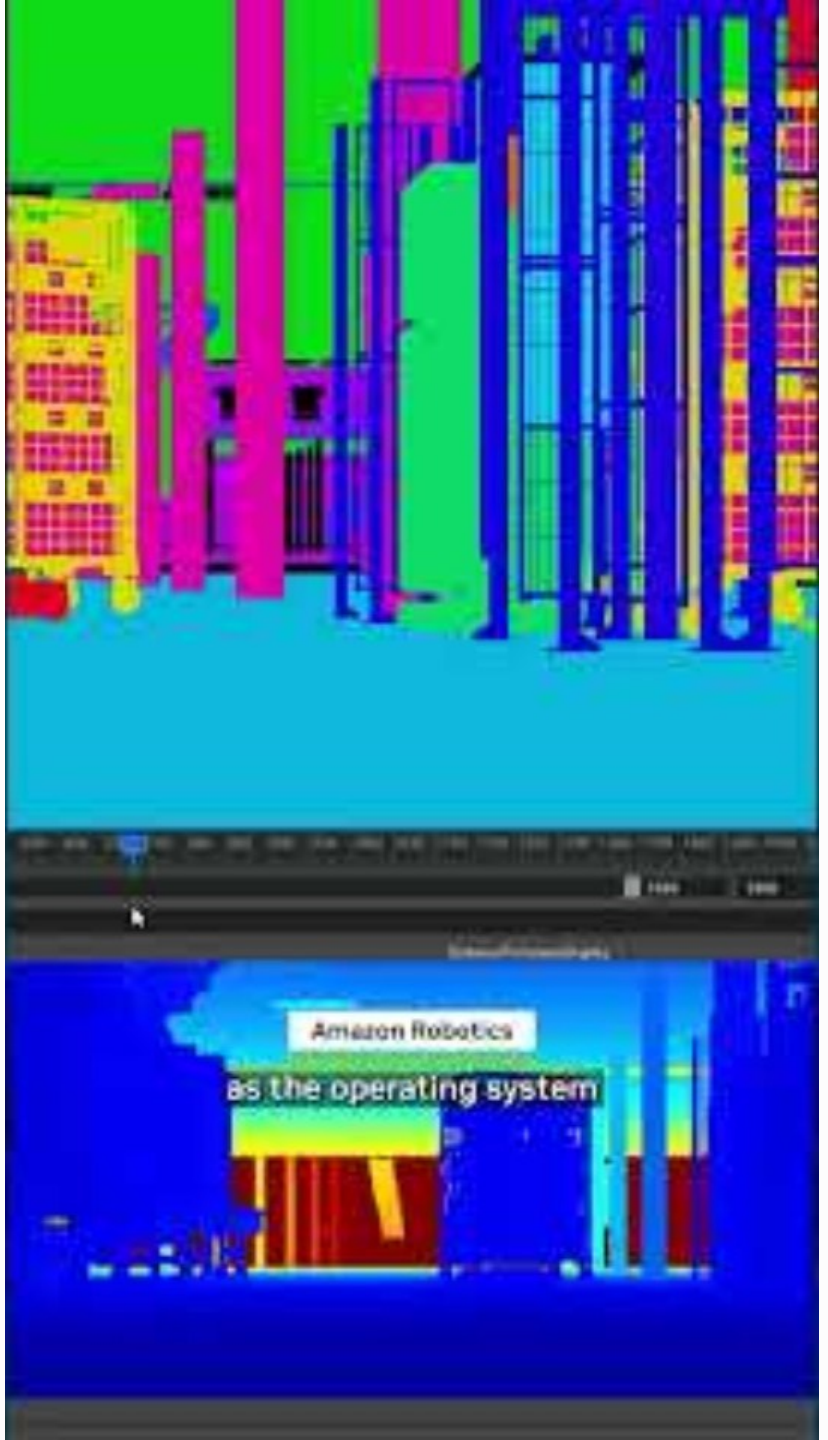
05 Johannes Wild. Arduino Projects with Tinkercad: Designing and programming Arduino-based electronics projects using Tinkercad. 2023. Available to buy at: Arduino Projects with Tinkercad: Designing and programming Arduino-based electronics projects using Tinkercad (Arduino | Introduction and Projects Book 2) eBook : Wild, M.Eng. Johannes: Amazon.co.uk: Kindle Store

06 Johannes Wild. Arduino Projects with Tinkercad: Arduino Projects with Tinkercad | Part 2: Design & program advanced Arduino-based electronics projects with Tinkercad. 2023. Available to buy at: Arduino Projects with Tinkercad | Part 2: Design & program advanced Arduino-based electronics projects with Tinkercad (Arduino | Introduction and Projects Book 3) eBook : Wild, M.Eng. Johannes : Amazon.co.uk: Kindle Store

07 Massimo Banzi and Michael Shiloh. Getting Started with Arduino. Published by Maker Media. 2014. Available to buy at: Getting Started With Arduino eBook : Banzi, Massimo, Shiloh, Michael: Amazon.co.uk: Kindle Store

Welcome to the Arduino Experience!

AI is almost everywhere, including in the physical world.



How does an AI open a door?

THE BRAIN

ChatGPT, Claude, Gemini…
Thinks, writes, decides — but lives in a data centre. It can't touch anything.

THE BODY

Sensors + microcontrollers + motors.
Sees, hears, moves, switches things on. This is what we build today.

Every robot, smart device, and self-driving car is made up of these two halves working together.

01

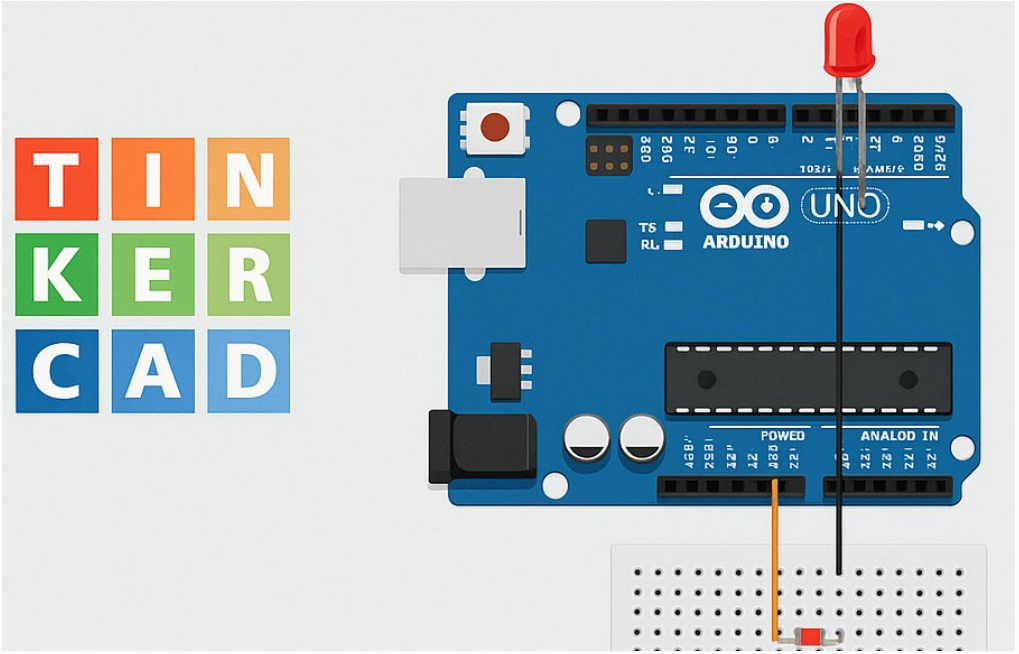
Introduction to Arduino

UNIVERSITY OF WARWICK

Welcome to the Arduino Experience!

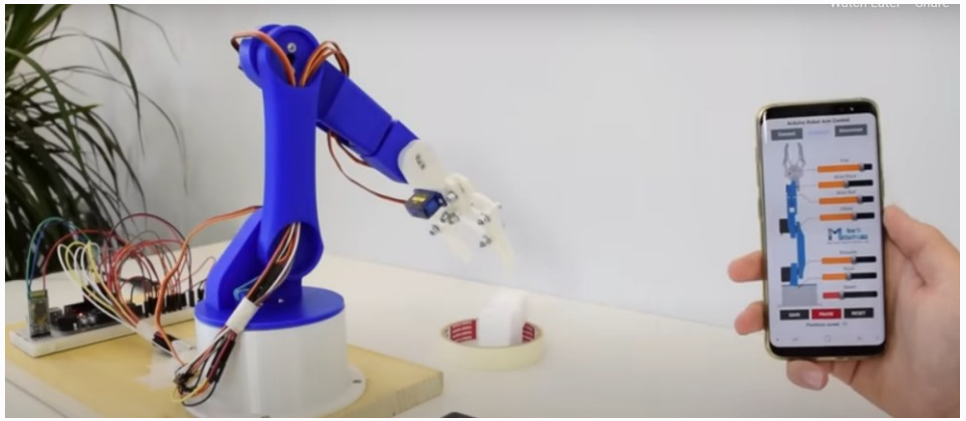
Session aims:

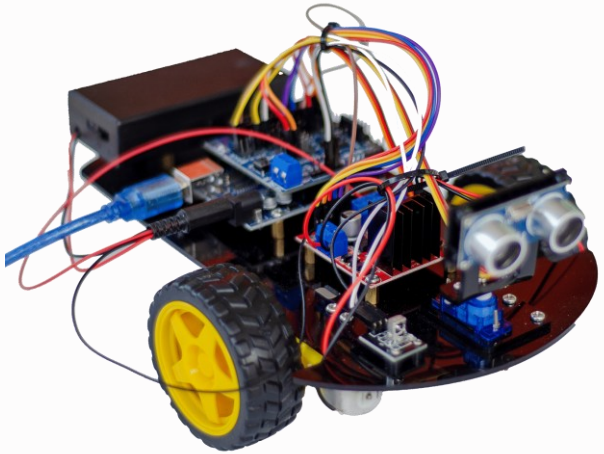
- A Taste of University Life
- Build Logic Circuits Using Arduino
- Try It Out on TinkerCAD – No Hardware Needed!
- By the end of today, you'll build your first smart system

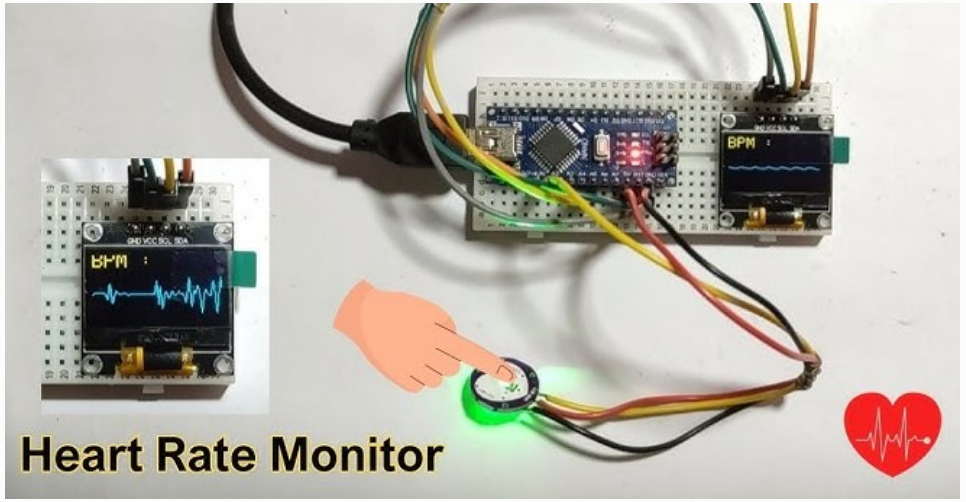


Why Arduino? Why Now?

- Used by engineers, designers, and hobbyists worldwide
- Powers real products: smart lights, alarms, robots
- You'll create your own circuit today!







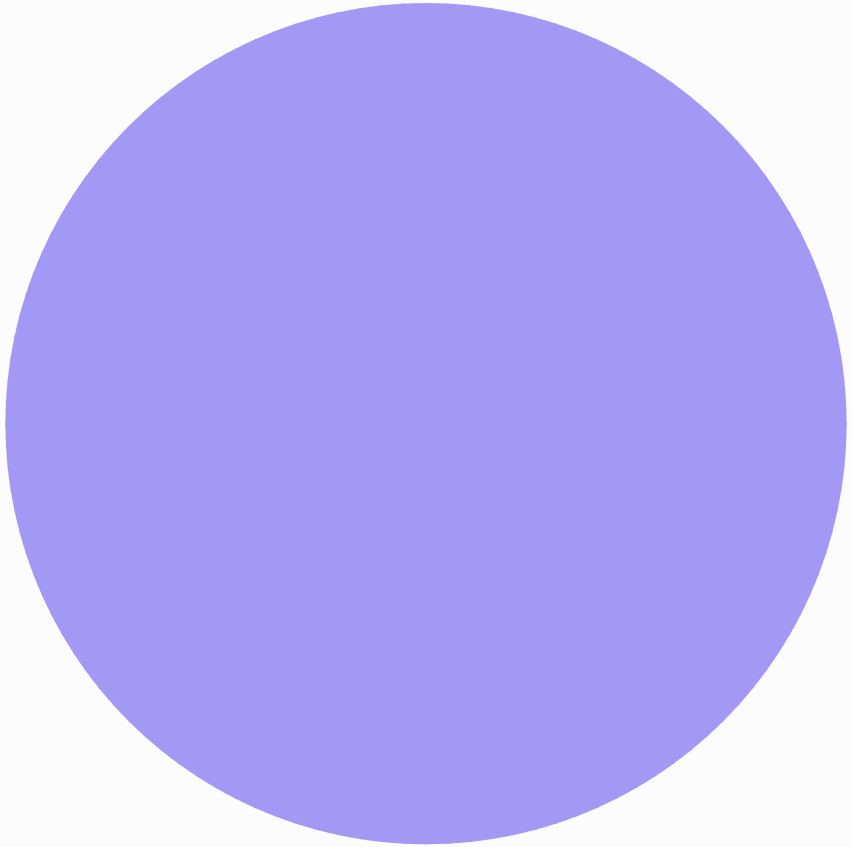
Join the Vevox session

Go to **vevox.app**

Enter the session ID: **180-439-293**

Or scan the QR code

Who has ever built a circuit or written code?



I have! 0%

I have not! 0%

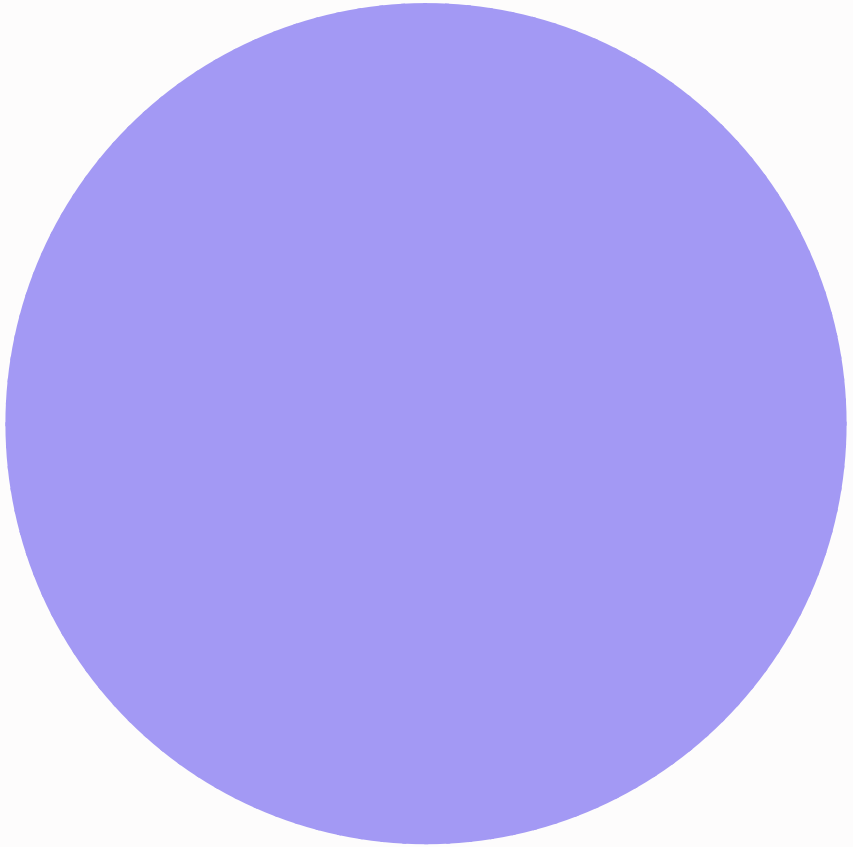
0

Join at: **vevox.app**

ID: **180-439-293**

Preparing Results

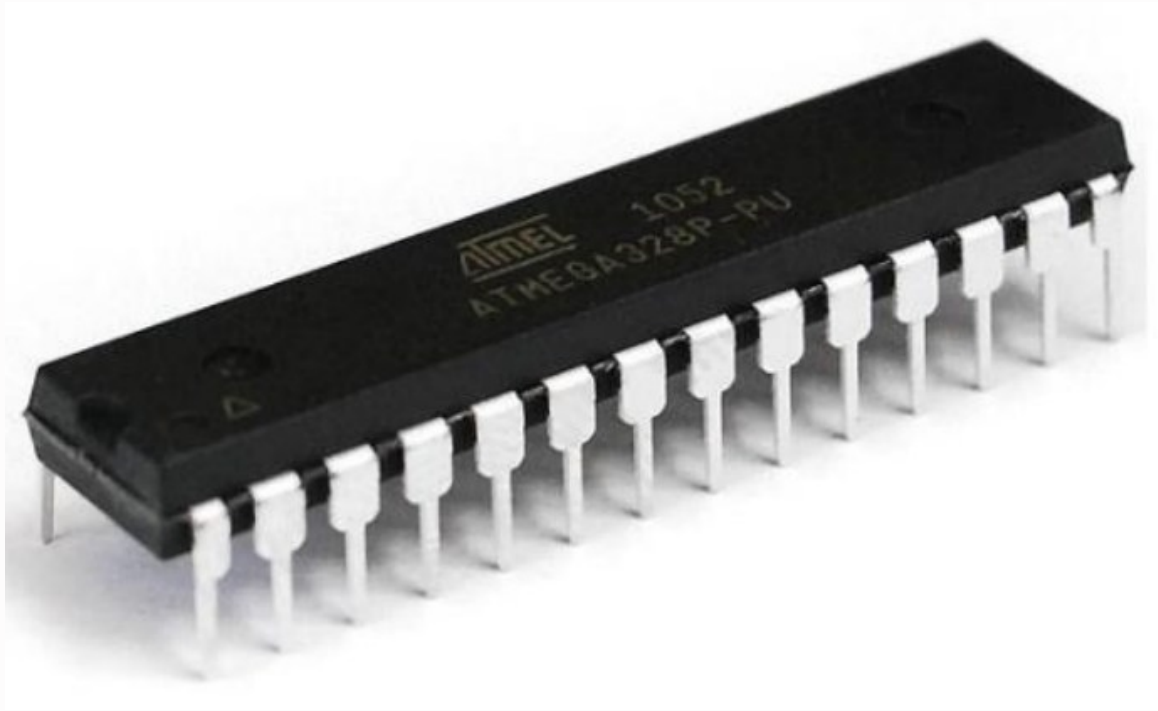
Who has ever built a circuit or written code?



I have! 0%

I have not! 0%

A chip that costs less than £3



What do you own that has one of these inside?

(Hint: you're probably touching three right now.)

It is called *a microcontroller*!

Microcontrollers run your life

Drones
flight stabilisation

E-scooters
motor + battery control

Controllers
every button press

Robot vacuums
bump + cliff sensors

Cars
50–150 per vehicle

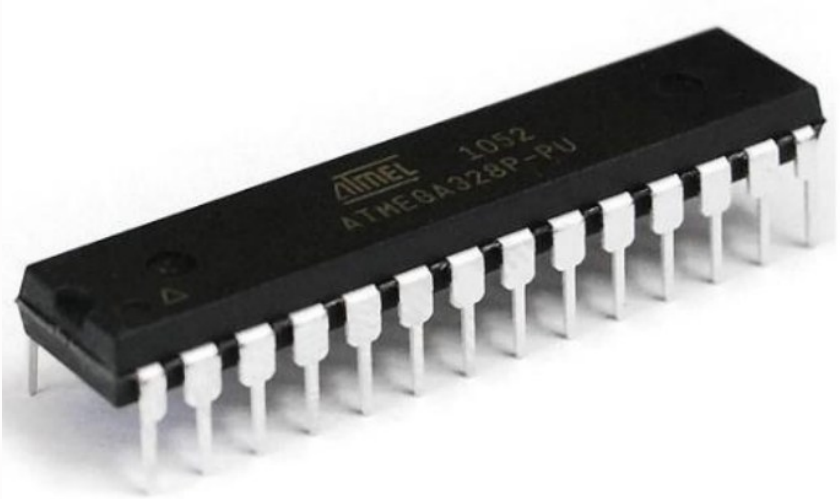
Wearables
smart rings, fitness bands

A modern car has more microcontrollers than your house has light bulbs.

So what is it?

A microcontroller is a tiny, programmable computer on a single chip.
It runs one program forever the instant power arrives.

- **Stores a program** — your instructions live in its memory
- **Executes them** — millions of steps per second, no operating system
- **Controls things** — reads sensors, switches outputs on and off



Arduino: the microcontroller for everyone

Created in mid-2005.

Arduino is a programmable electronics prototyping platform based on microcontrollers (Atmel) used in the control of logical processes.

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

Open-Source Platform: free hardware and software.

https://www.arduino.cc/

Applications of Microcontrollers

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

UNIVERSITY OF WARWICK

Have you heard about the Internet of Things (IoT)? What do you think it is?

Have you heard about the Internet of Things (IoT)? What do you think it is?

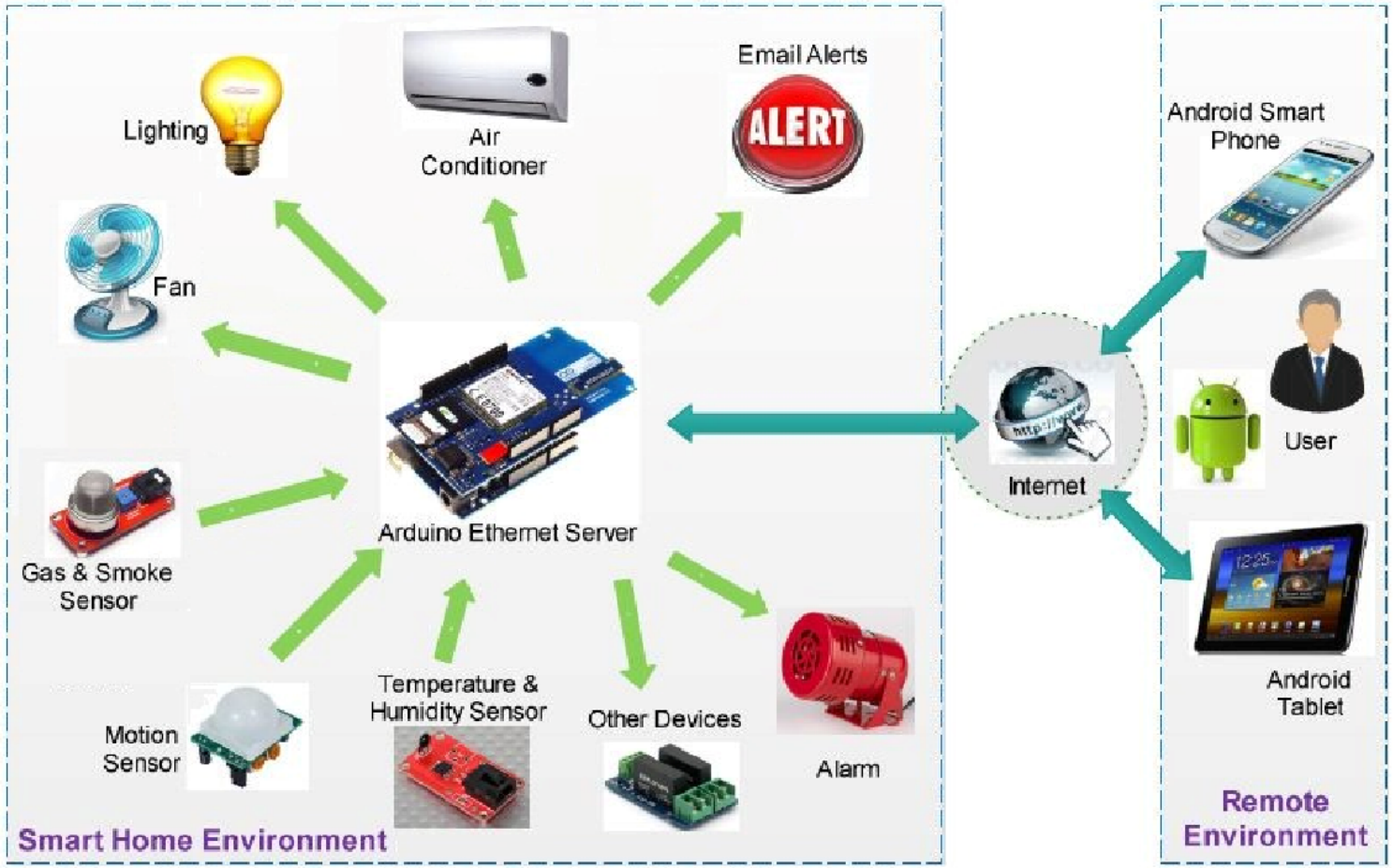
Internet of Things (IoT)

- The Internet of Things (IoT) is a concept that refers to the interconnection of everyday objects and devices usually over 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

Smart Home Devices:
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 com módulo PoE

Arduino M0

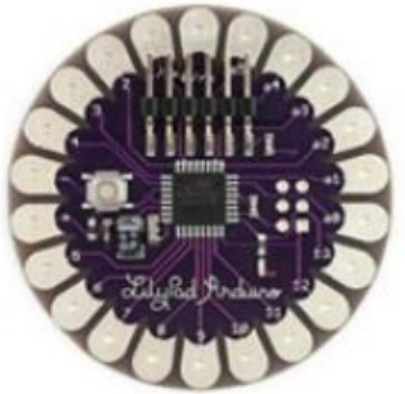
Introduction to Arduino Platform

Arduino Family:

Arduino Yún

Arduino Leonardo Ethernet

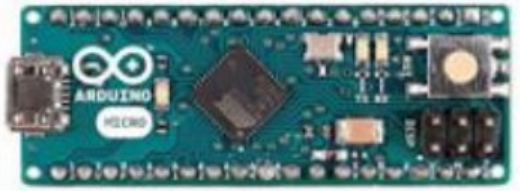
Arduino LilyPad



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.

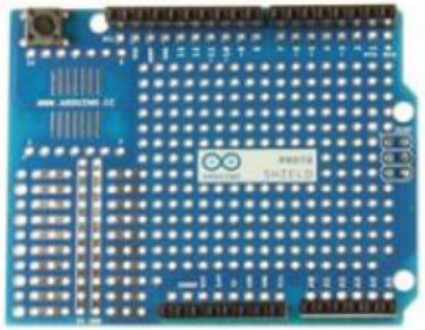
Arduino Ethernet Shield R3

Motor Shield R3

Arduino WiFi Shield

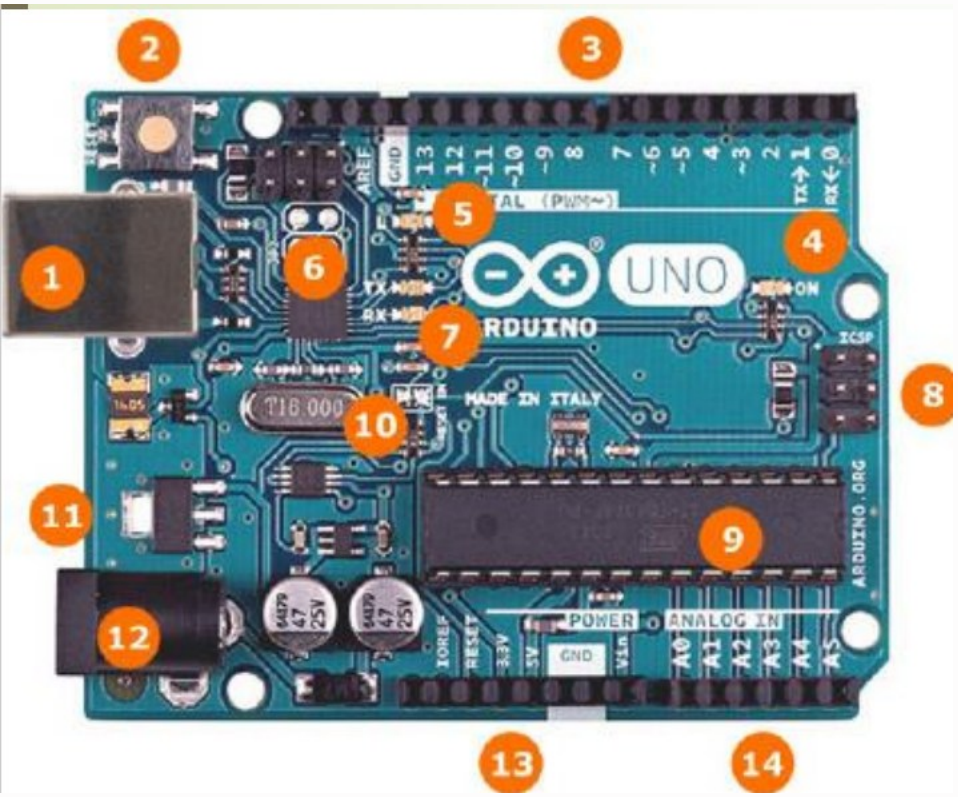
Shield Arduino 4 relês

Arduino ProtoShield R3



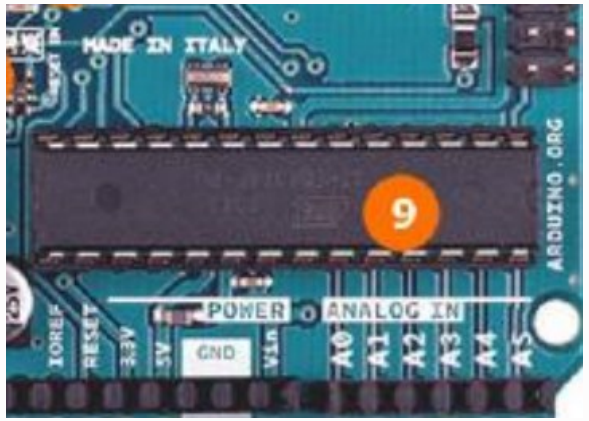
Arduino USB Host Shield

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.

So why use it?

- Costs £3, not £800
- Boots instantly — no OS
- Runs for months on a battery
- Never crashes, never updates
- **Perfect for being a body**

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.

02

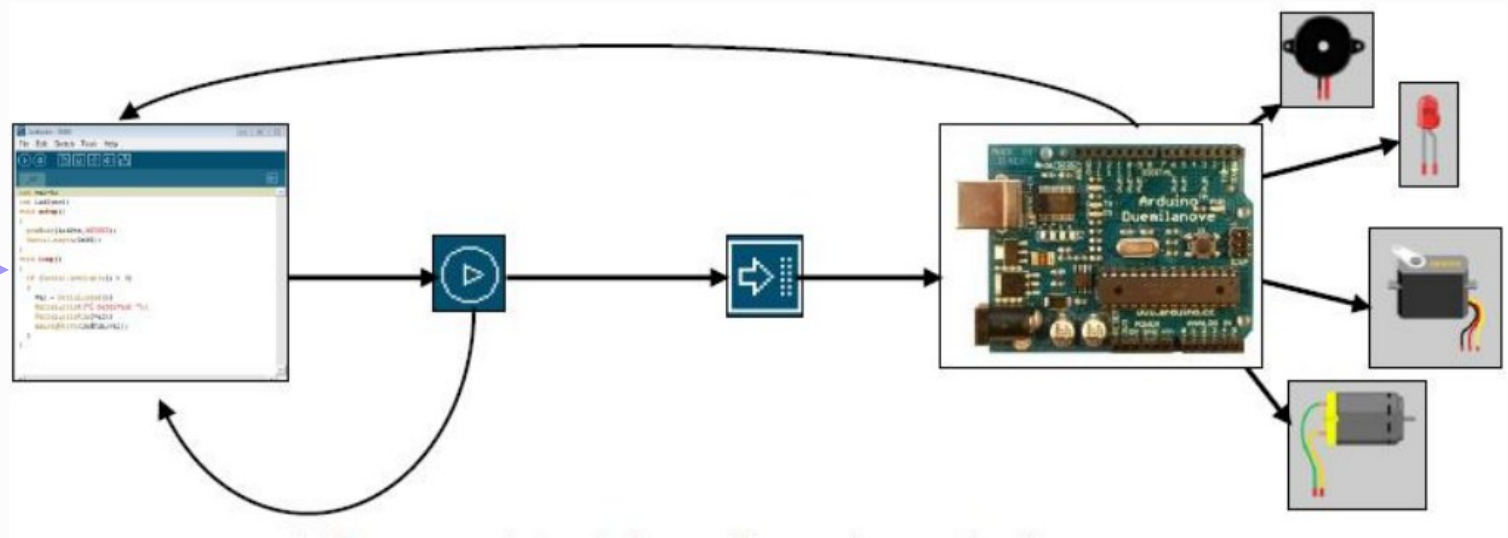
Programming Arduino: Logical Blocks

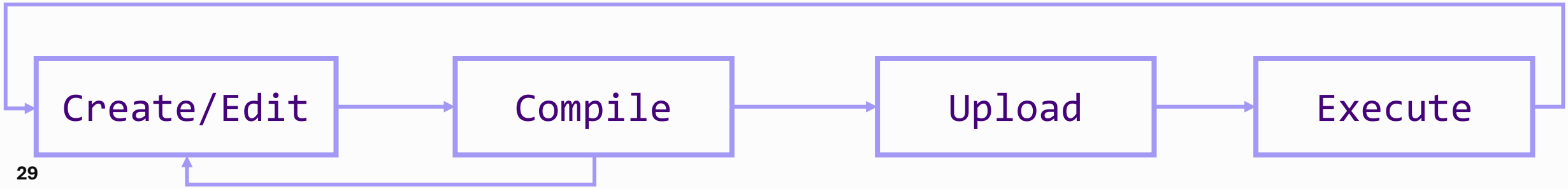
UNIVERSITY OF WARWICK

Prototyping

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

1. Tinker with the circuit
2. Program the microcontroller





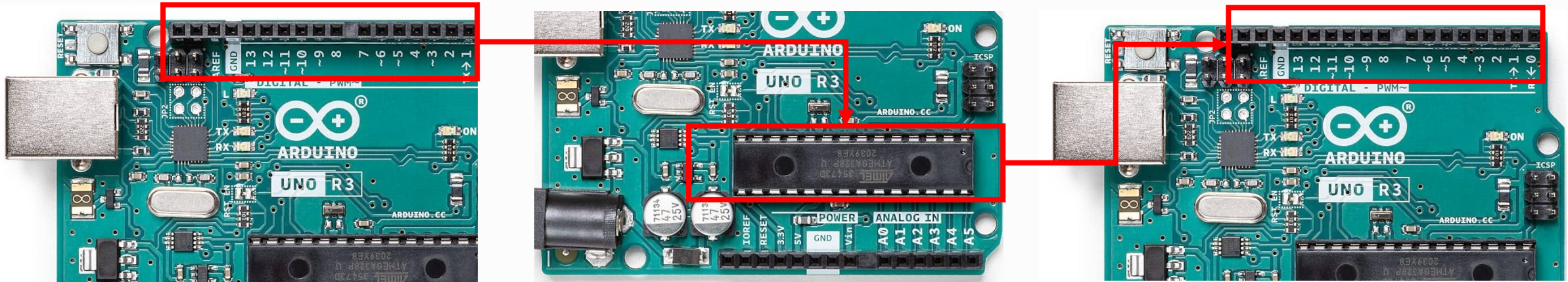
Input/Outputs

- 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

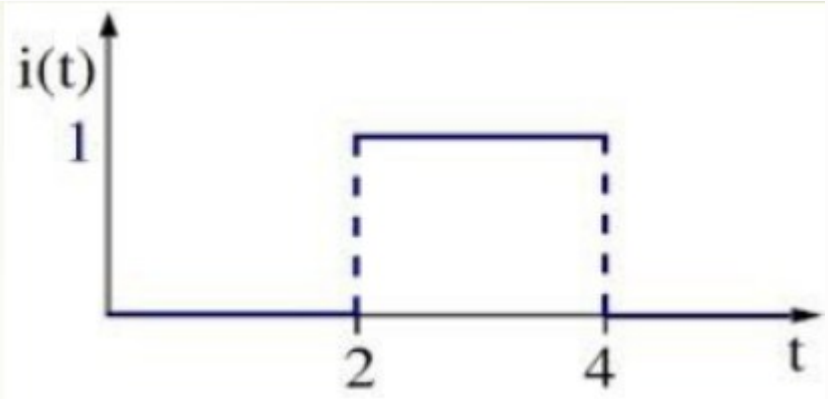
Processing:
Interpret and manipulate signals

Output:
Convert current or voltage into physically useful signals

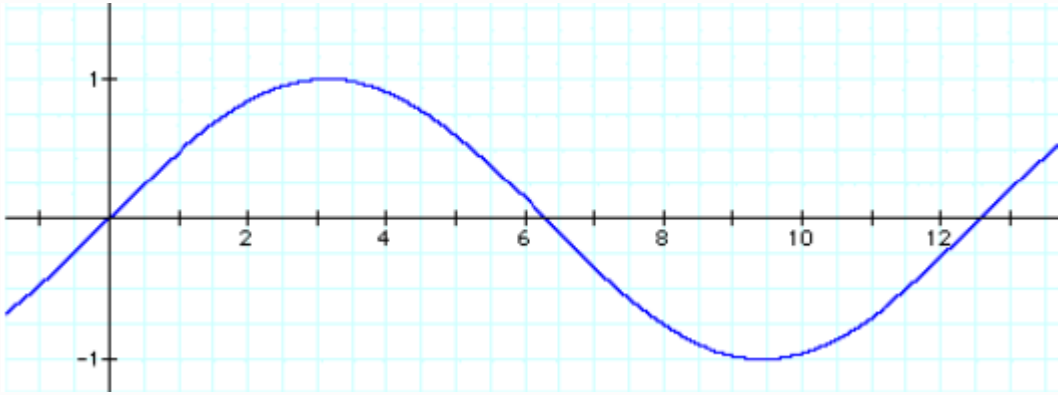


Types of Signals

- 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)



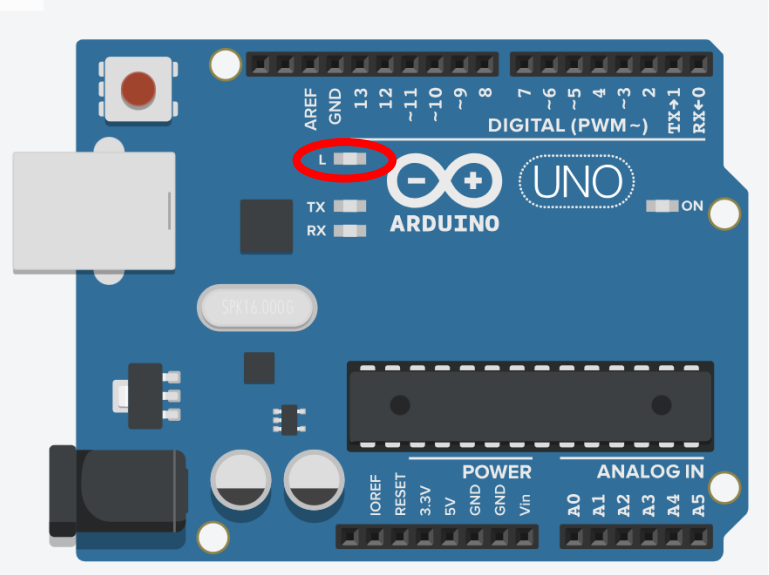
Digital Outputs: Hello World

- 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?

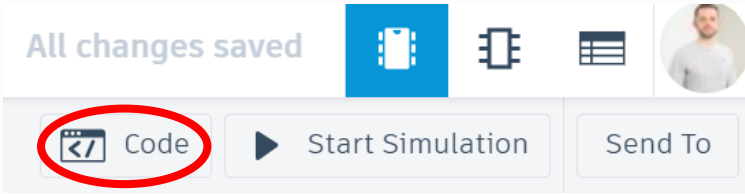
- Activate a digital output.
- Blink the on-board LED.
- The syntax of an Arduino Code.
- Use timers on an output pin.

Components

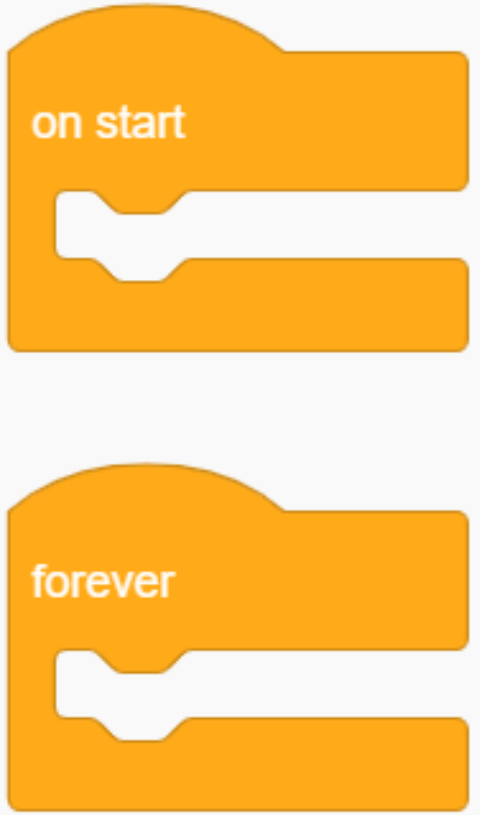


Digital Outputs: Hello World

The Code



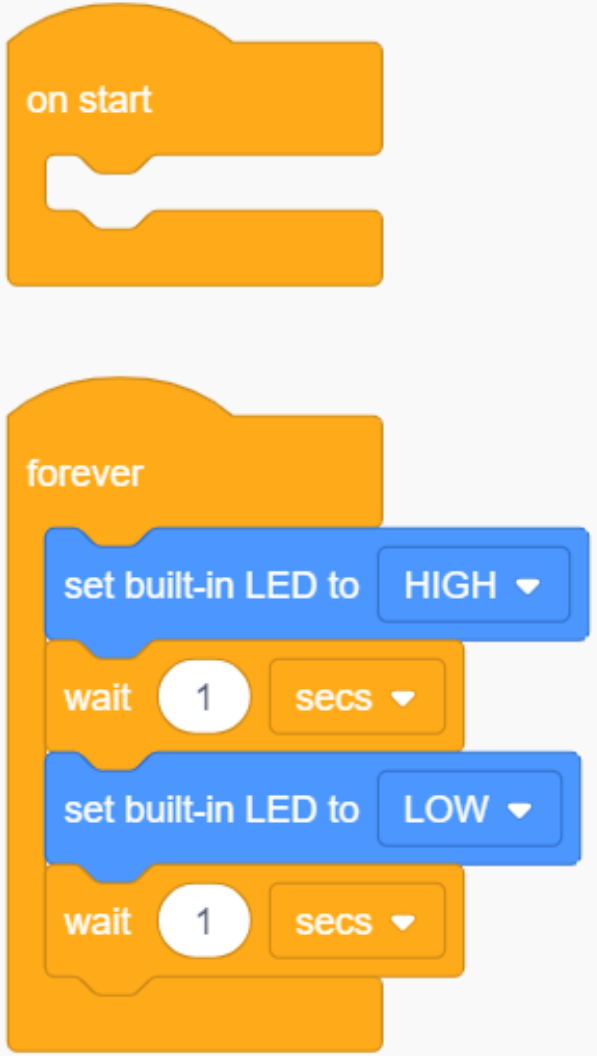
- Control Structures (Yellow):
  - *on start*: code structure that can be used to execute blocks of code only once, and when the board is powered. Use it for code that you only want to execute once when the board is turned on.
  - *forever*: code structure that can be used to execute code blocks repeatedly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

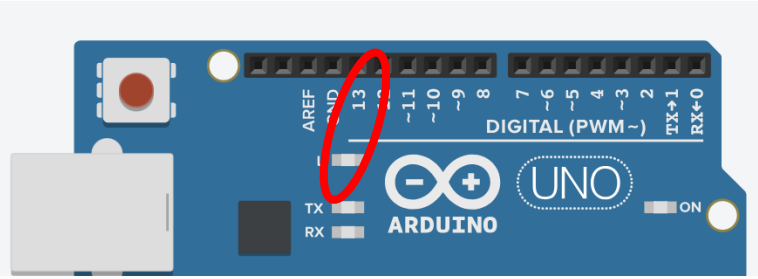


Hello World

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.

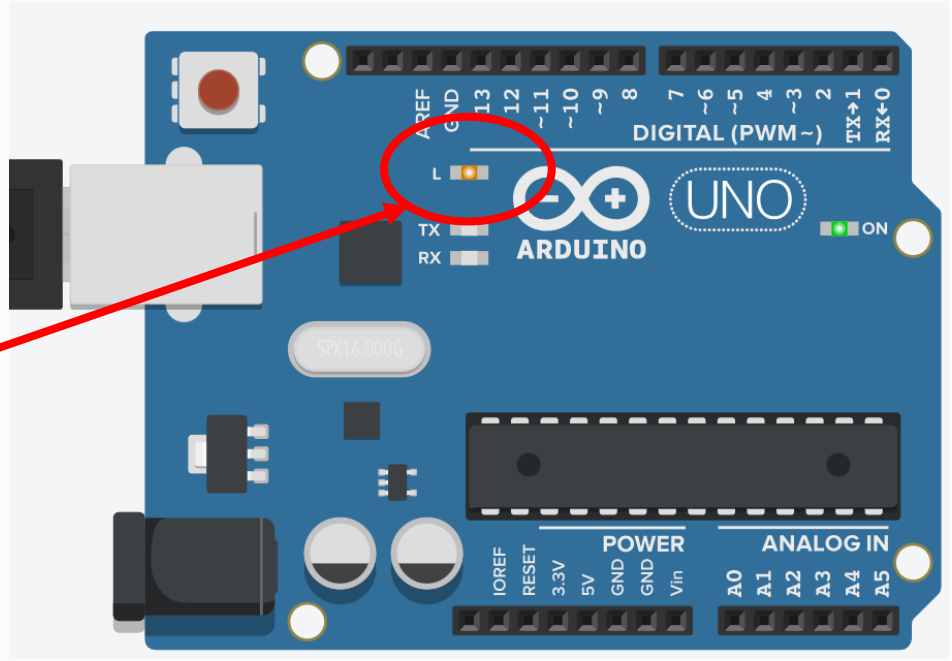


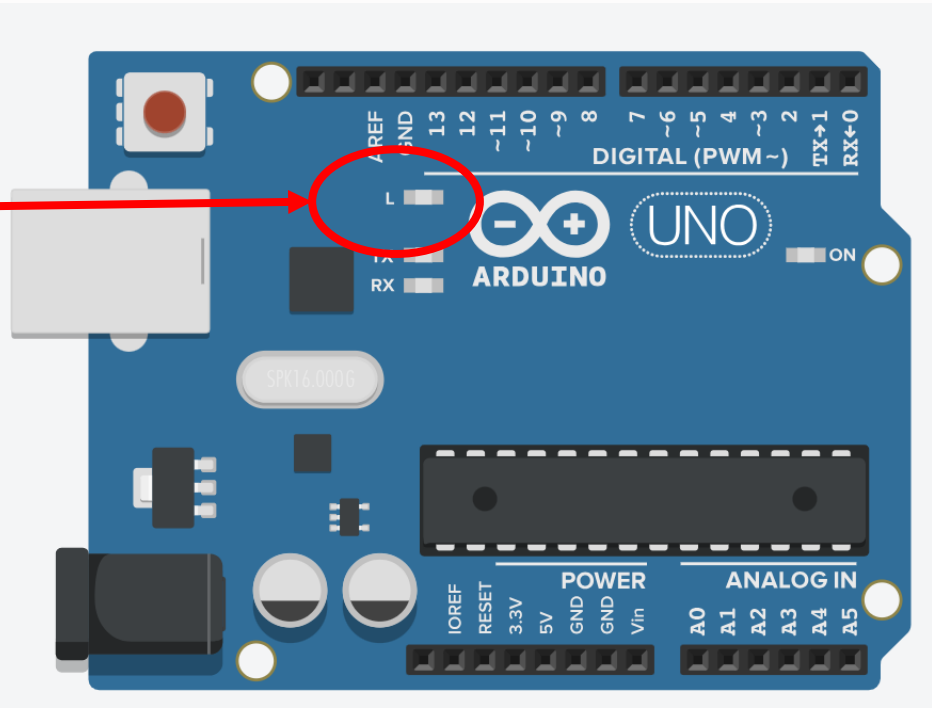


Hello World

The Code

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

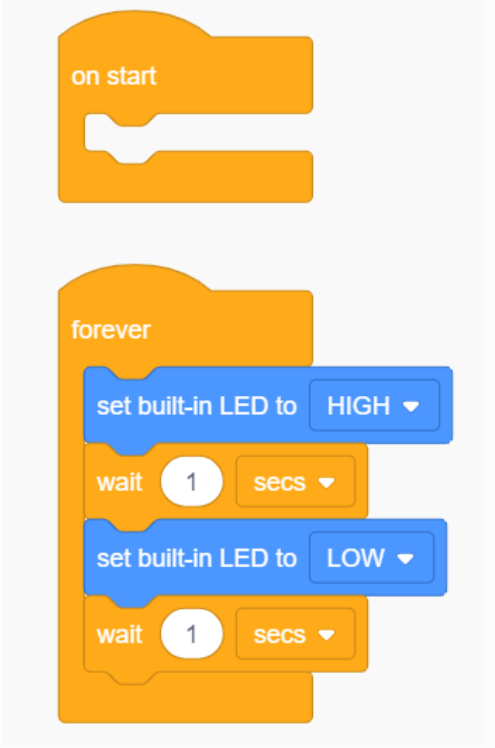




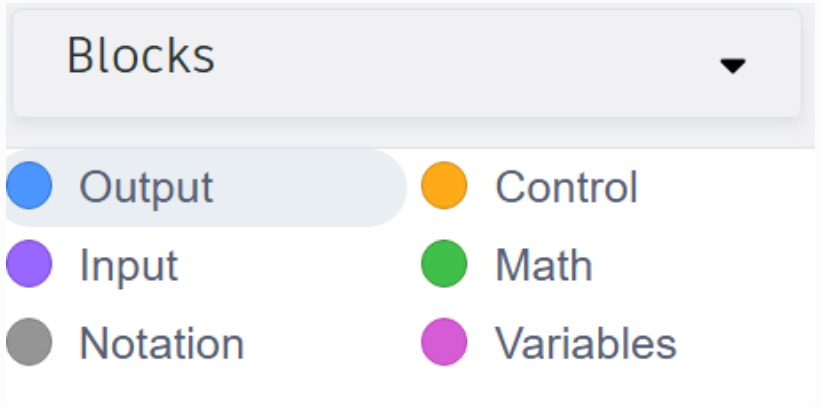
HIGH is equivalent to bit 1 (5V), while **LOW** is equivalent to bit 0 (0V).

Hello World

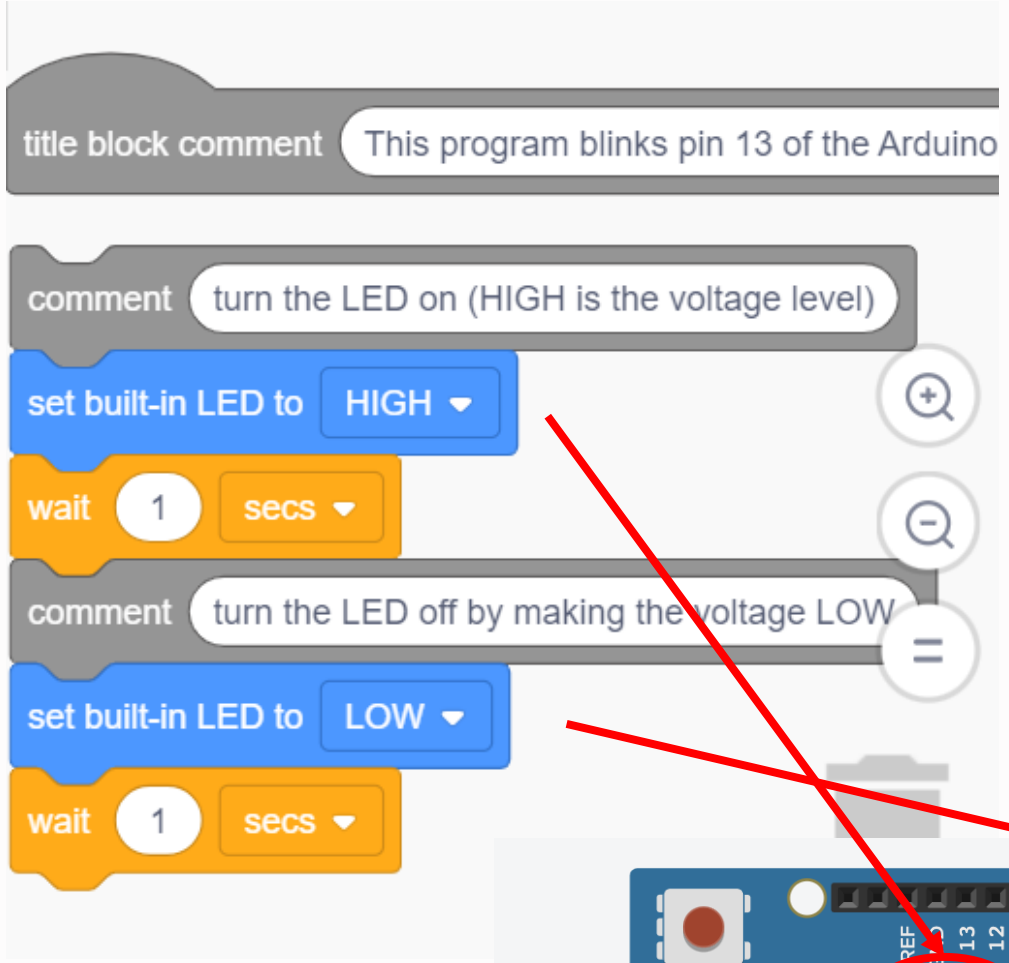
Now, you are familiar with different programming blocks: **on start, forever, set built-in LED, and wait**. Besides, you know that *HIGH* can turn something on and *LOW* can turn something off.



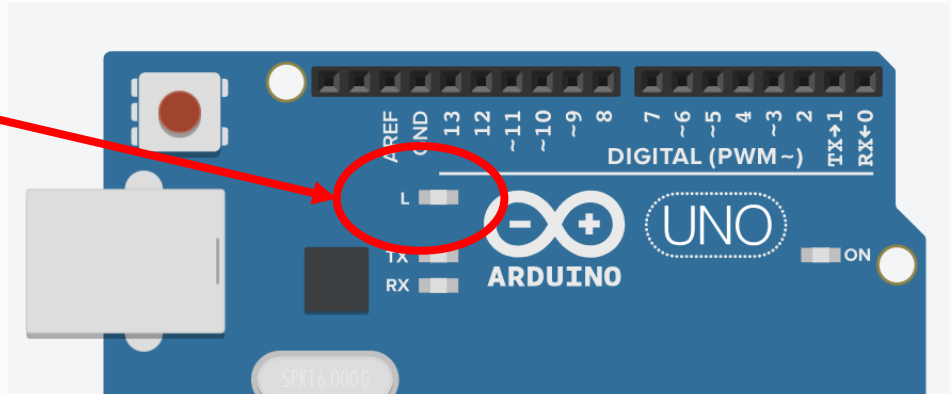
Let’s get familiar with other programming blocks!



More programming blocks: comment



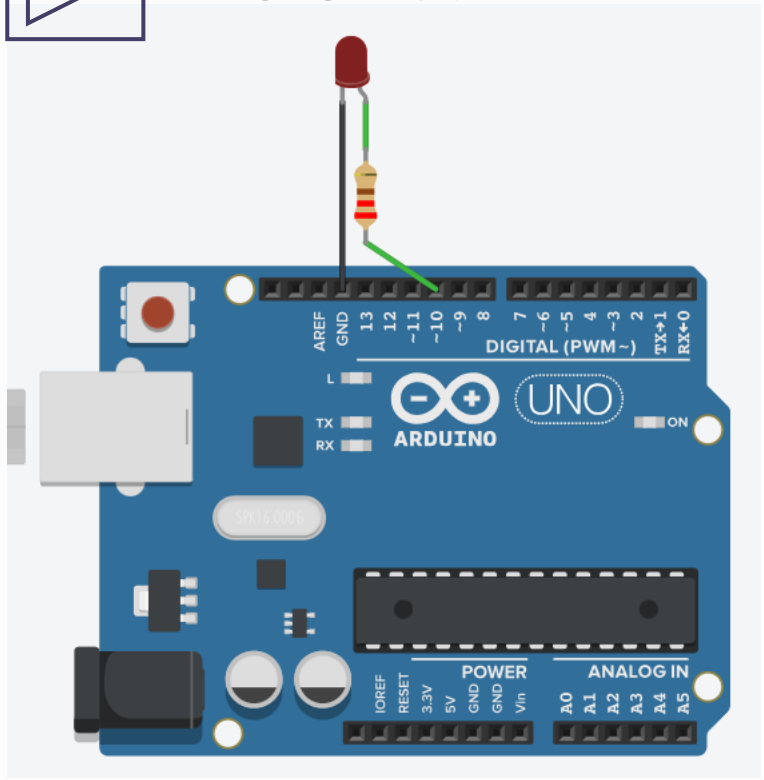
- Blocks in grey are called notation.
- The notation blocks are used to add comments to your code. Comments are helpful for others to understand your code, and they do not execute any action.
- They do not change the behaviour of the code.



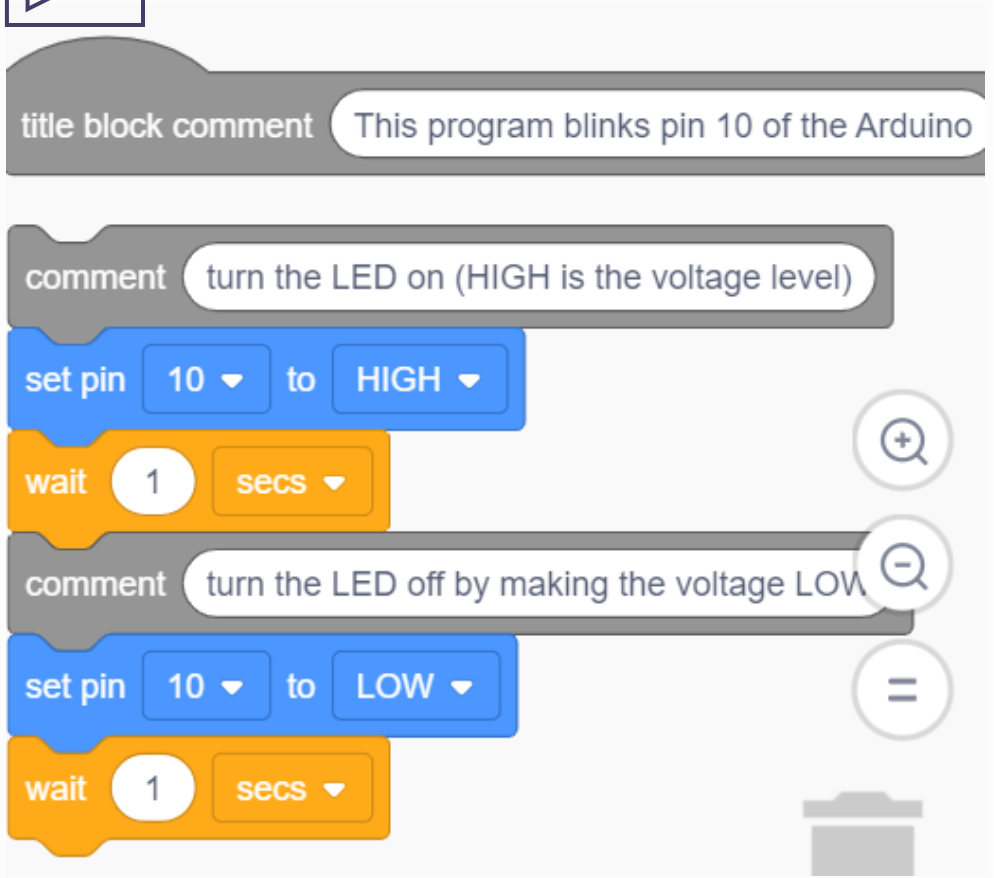
More output programming blocks: set pin

An LED can be connected to other digital pins. For example, let's replace the built-in LED with an external one connected to pin 10.

The Circuit

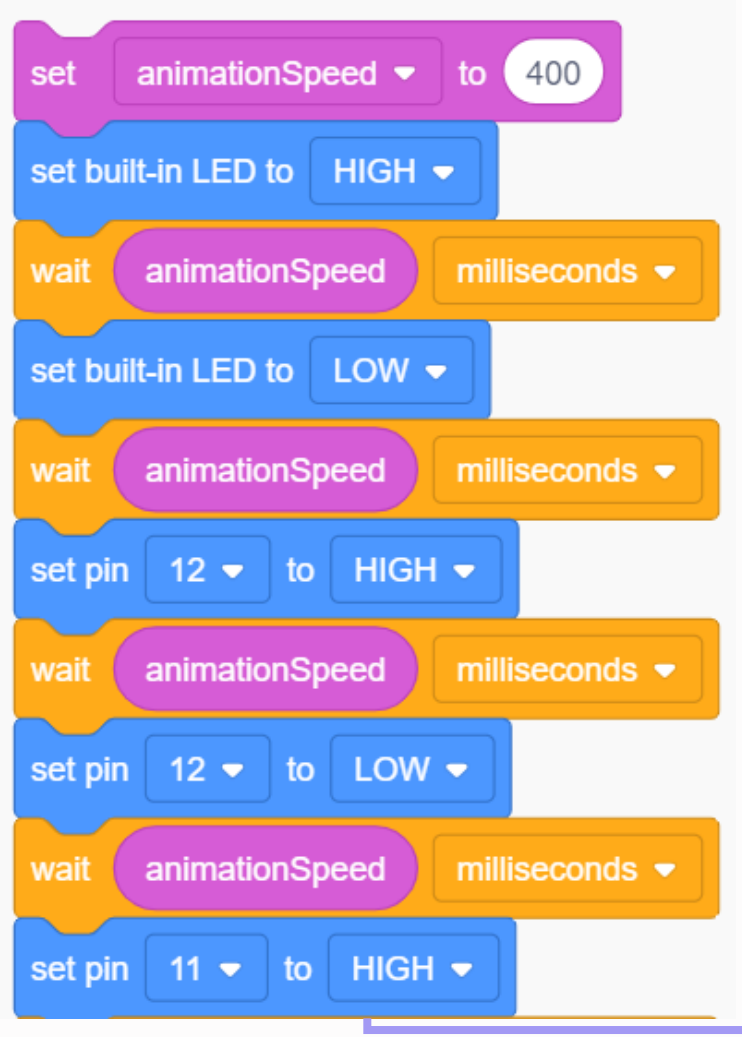


The Code

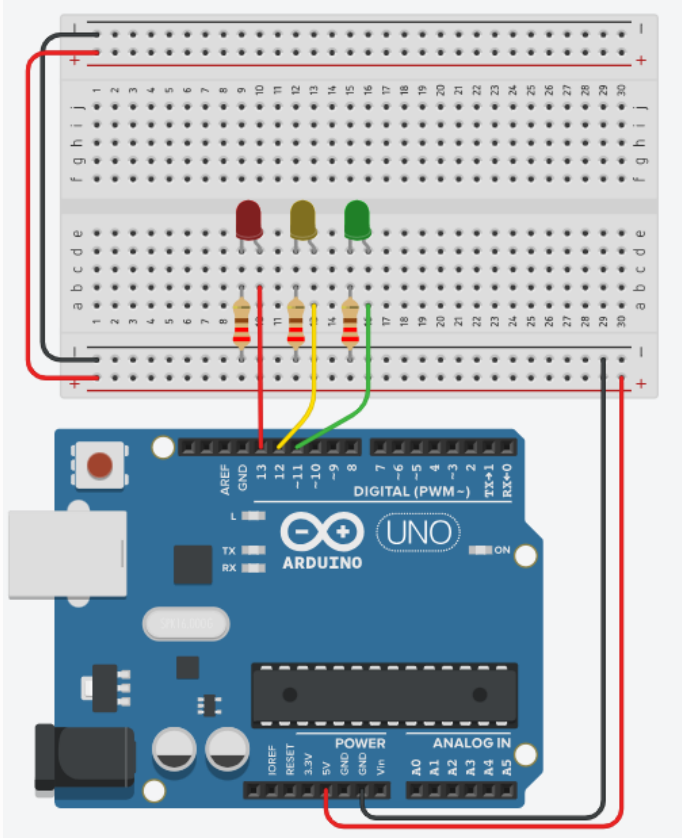


More programming blocks: variables

The Code

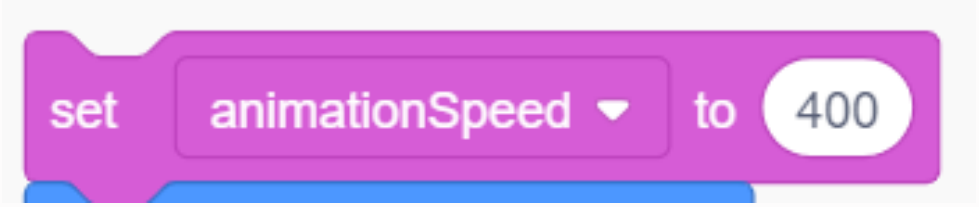


The Circuit



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.



≡

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

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

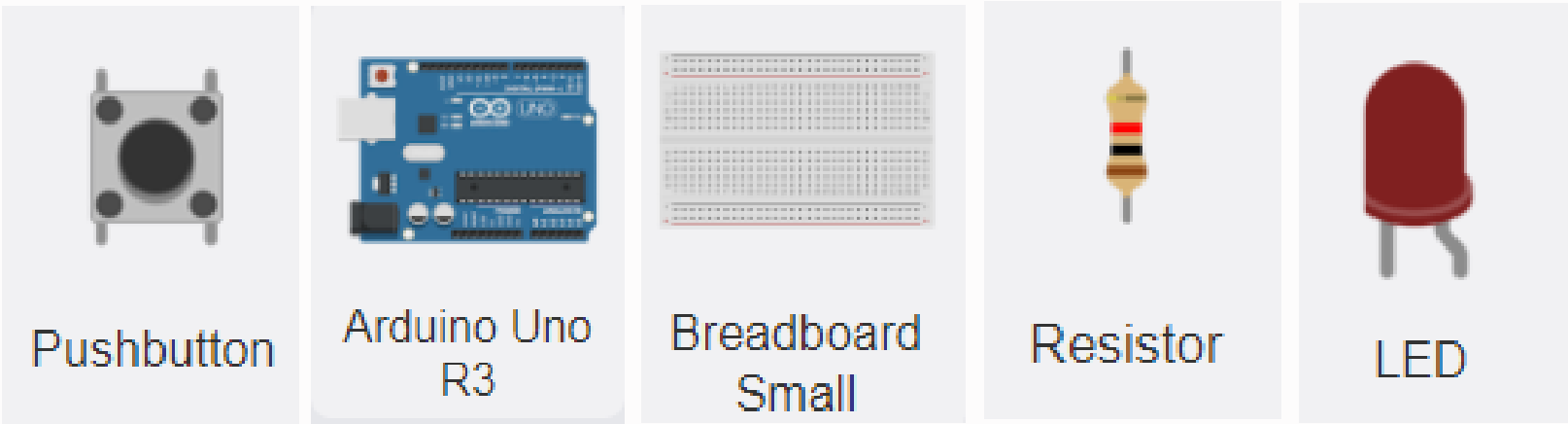
Digital Inputs: Push Button

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.

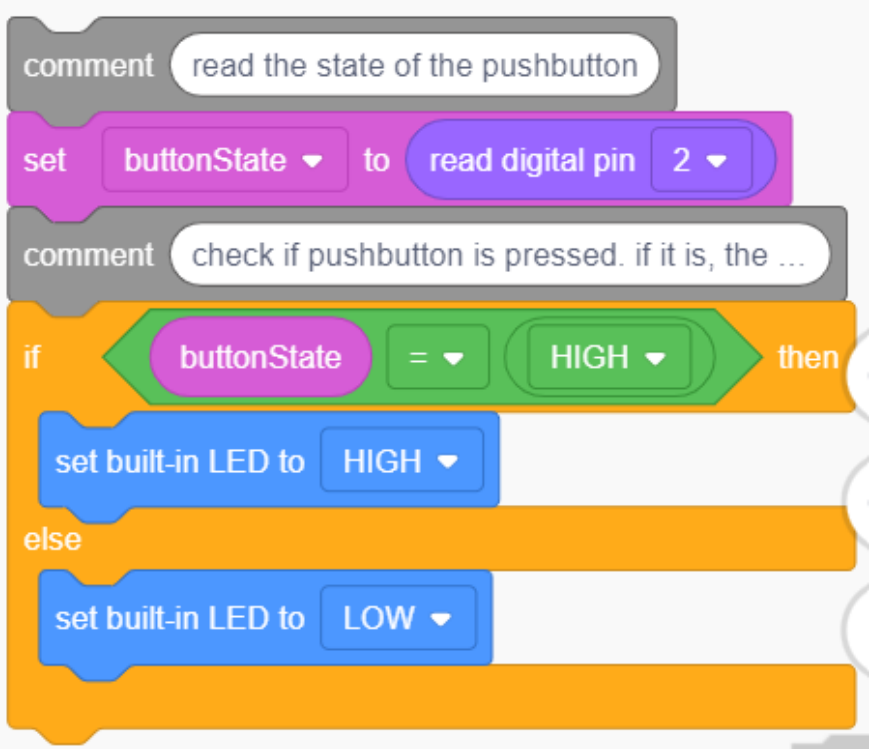
Components



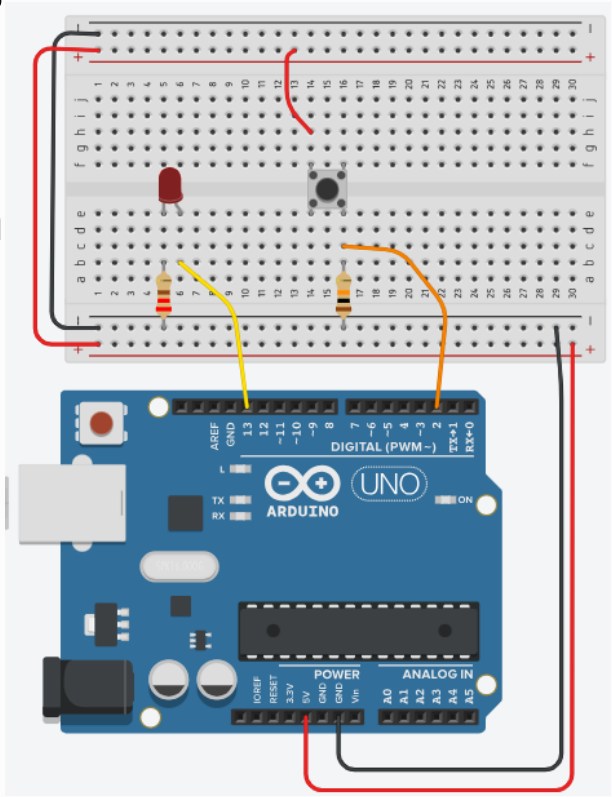
Push Button

The Code

- *if conditional*: enables the execution of 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.



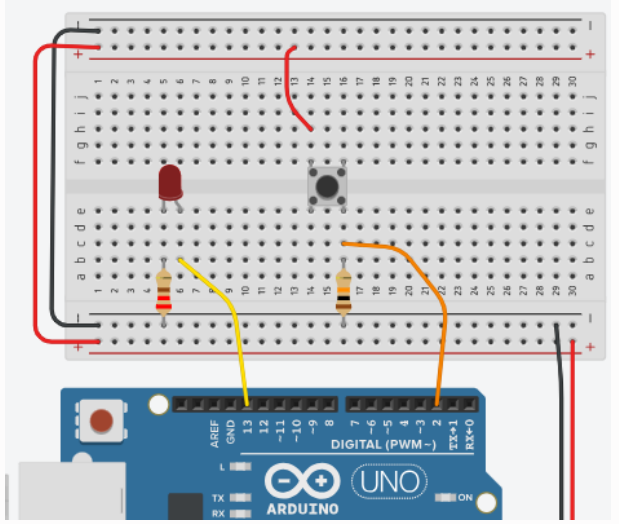
The Circuit

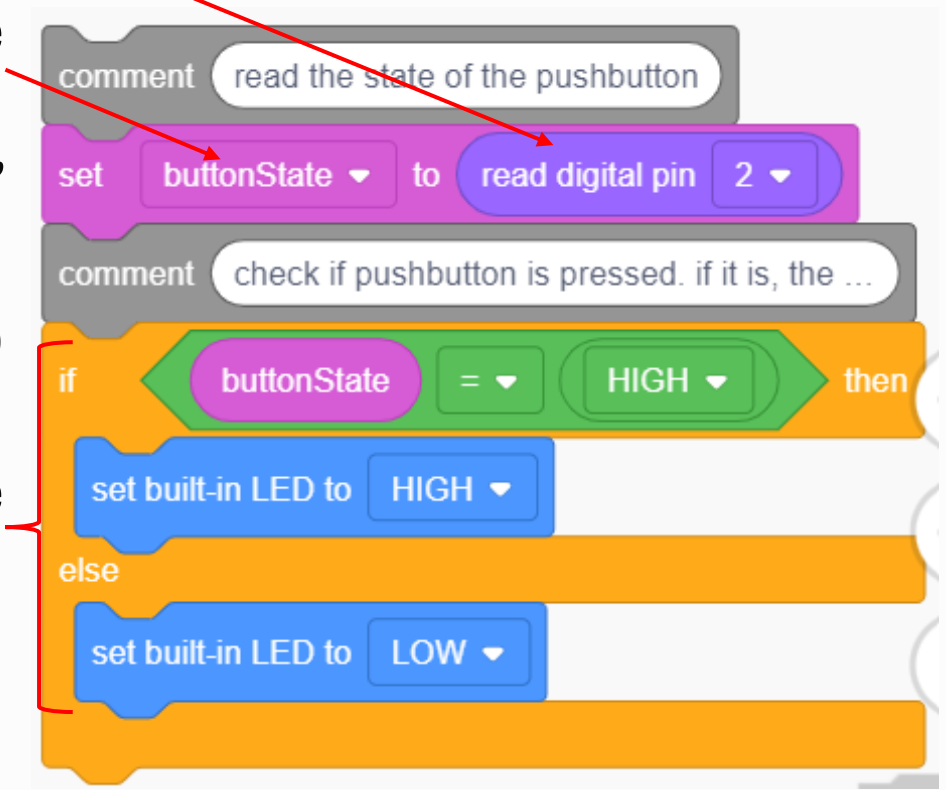


Reading Digital Inputs

The Code

- Initially, the digital pin (2) is read using the input block: "*read digital pin*".
- The value read from digital pin 2 is then stored in the variable "*buttonState*" using the "*set*" variable block.
- If the value read in the pin (2) is HIGH, "*buttonState*" value is also HIGH. Otherwise, it is set to LOW.
- The *if* conditional then turns the built-in LED (pin 13) ON when *buttonState* is equal to HIGH. Otherwise, the built-in LED (pin 13) is turned OFF by the else statement.
- Repeat.





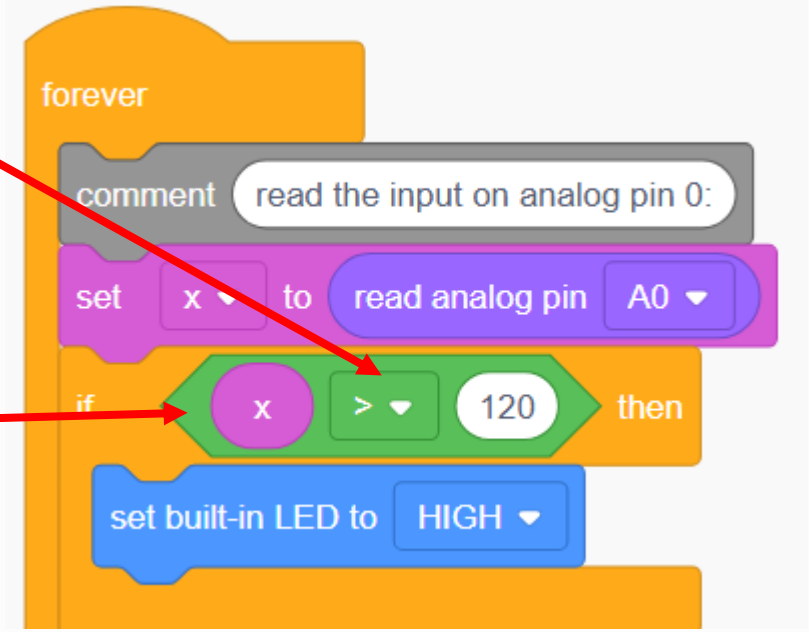
If conditional

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



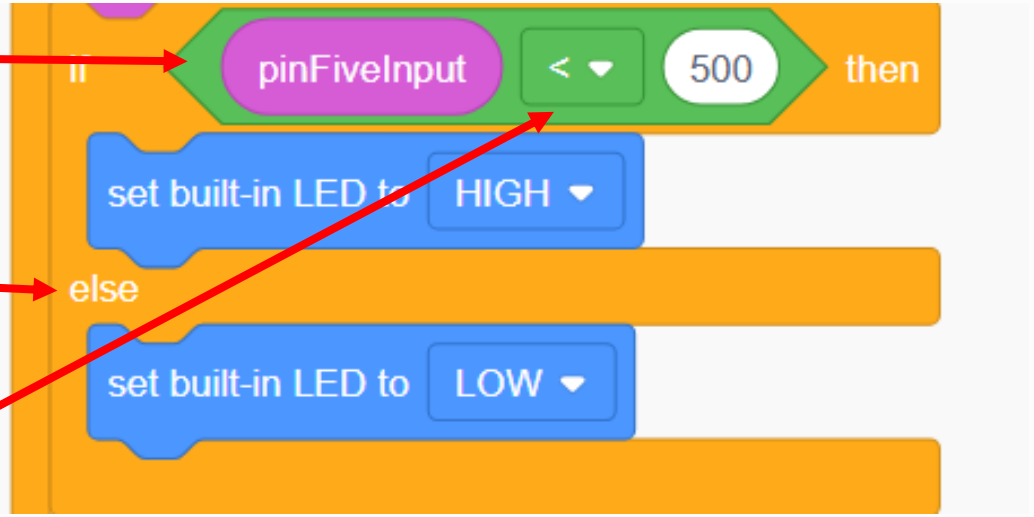
If – else conditional

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
{
  // do Thing B
}
```

Conditional

Operator



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
>	Greater than
≥	Greater than or equal to

Comparison operators

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

Arithmetic operators

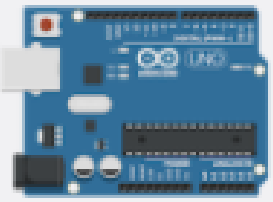
Analogue Inputs

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 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:

$$5V - 1023$$

- You can use the value read by the Arduino to determine the voltage connected to the pin. For instance, if the value read is 409, the voltage connect to pin is approximately:

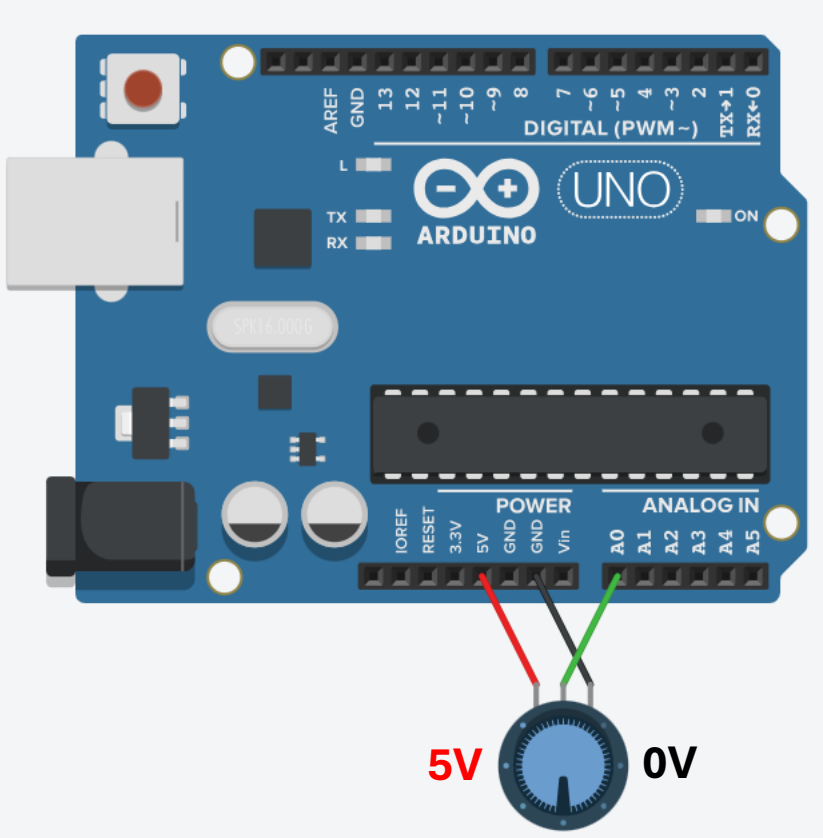
$$\frac{5 \times \mathbf{409}}{1023} \cong 2V$$

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.

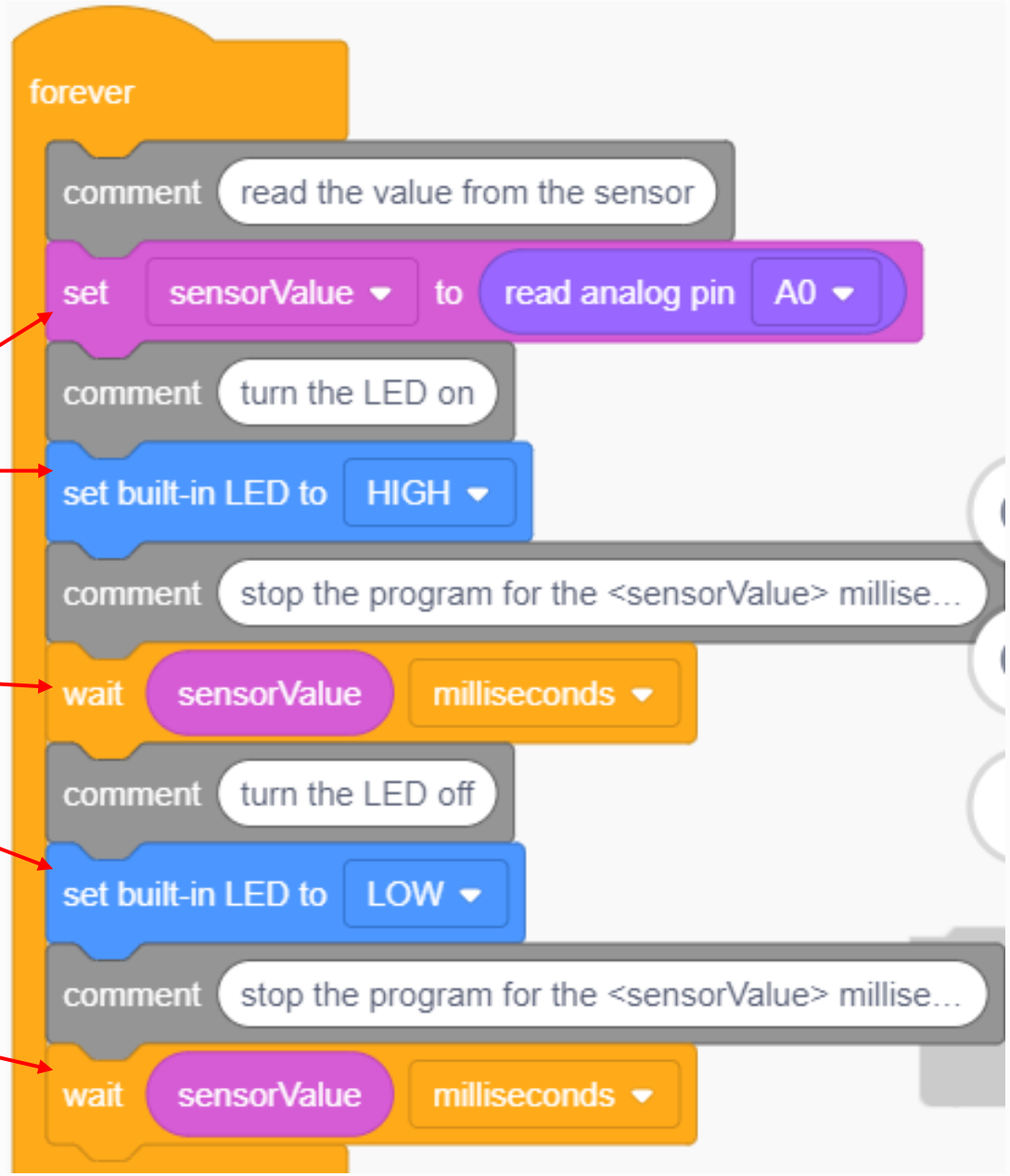
- Note that the end terminals are connected to the source (5V) and ground (GND), while the central terminal is connected to an Arduino analogue pin.



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.



What should be added to the blank circles in order to turn the built-in LED ON with less than 2.5V only?

sensorValue < 511

0%

sensorValue <= 1023

0%

sensorValue > 400

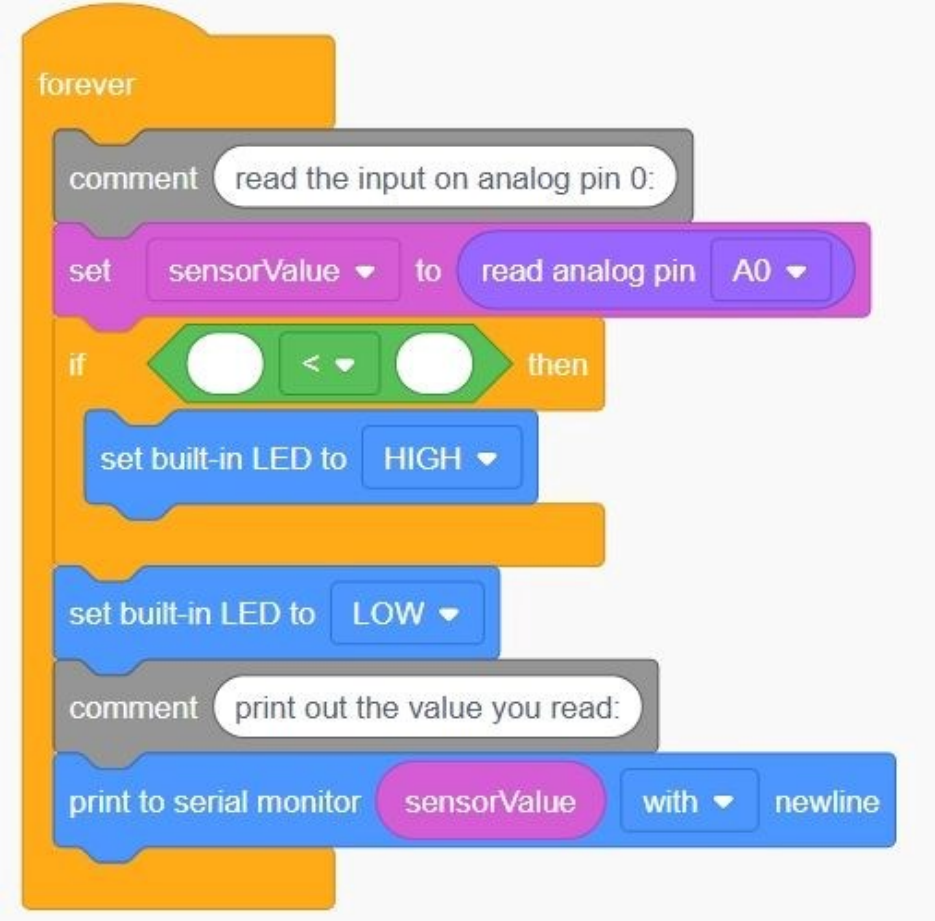
0%

sensorValue >= 511

0%

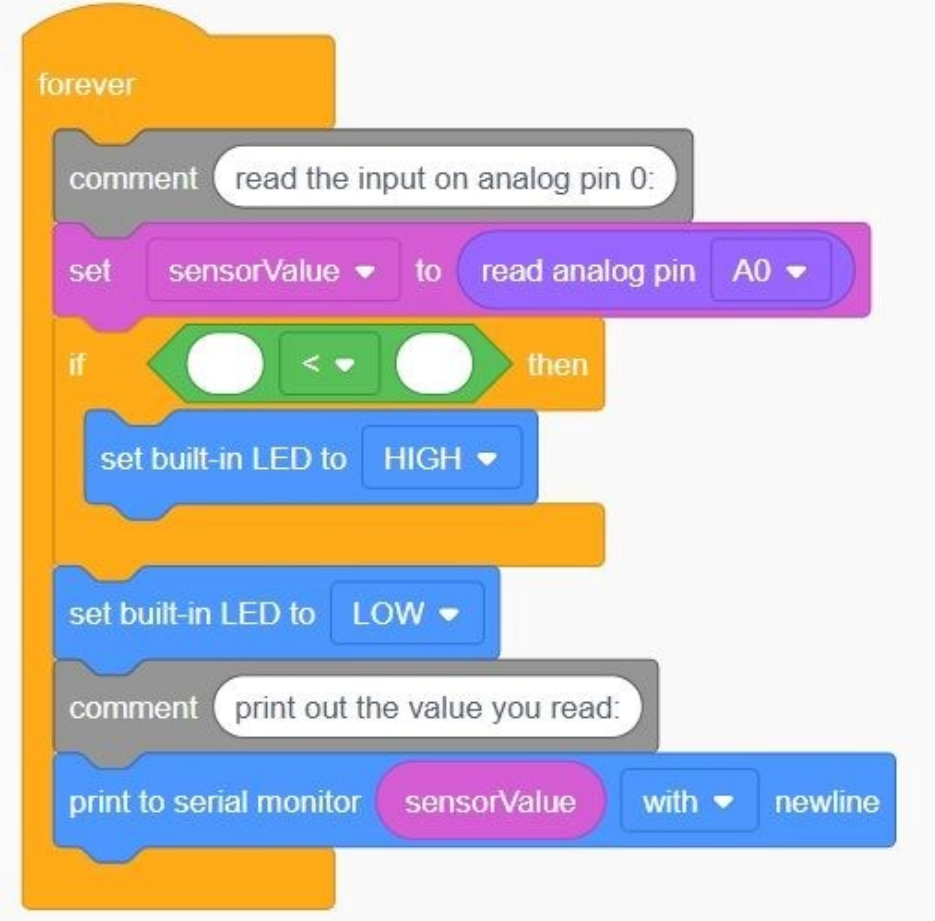
sensorvalue < 511

0%



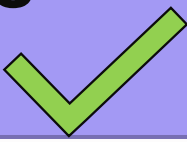
What should be added to the blank circles in order to turn the built-in LED ON with less than 2.5V only?

sensorValue < 511	77.78%
sensorValue <= 1023	3.7%
sensorValue > 400	3.7%
sensorValue >= 511	14.81%
sensorvalue < 511	0%

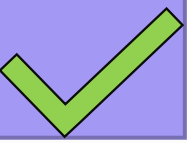


Summary

We discussed the difference between digital and analogue signals.



We introduced different TinkerCAD programming blocks.



03

Arduino: Simulation Platforms

UNIVERSITY OF WARWICK

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
- Wokwi: Wokwi - Online ESP32, STM32, Arduino Simulator

Tinkercad: Creating an account

Go to: https://www.tinkercad.com/

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

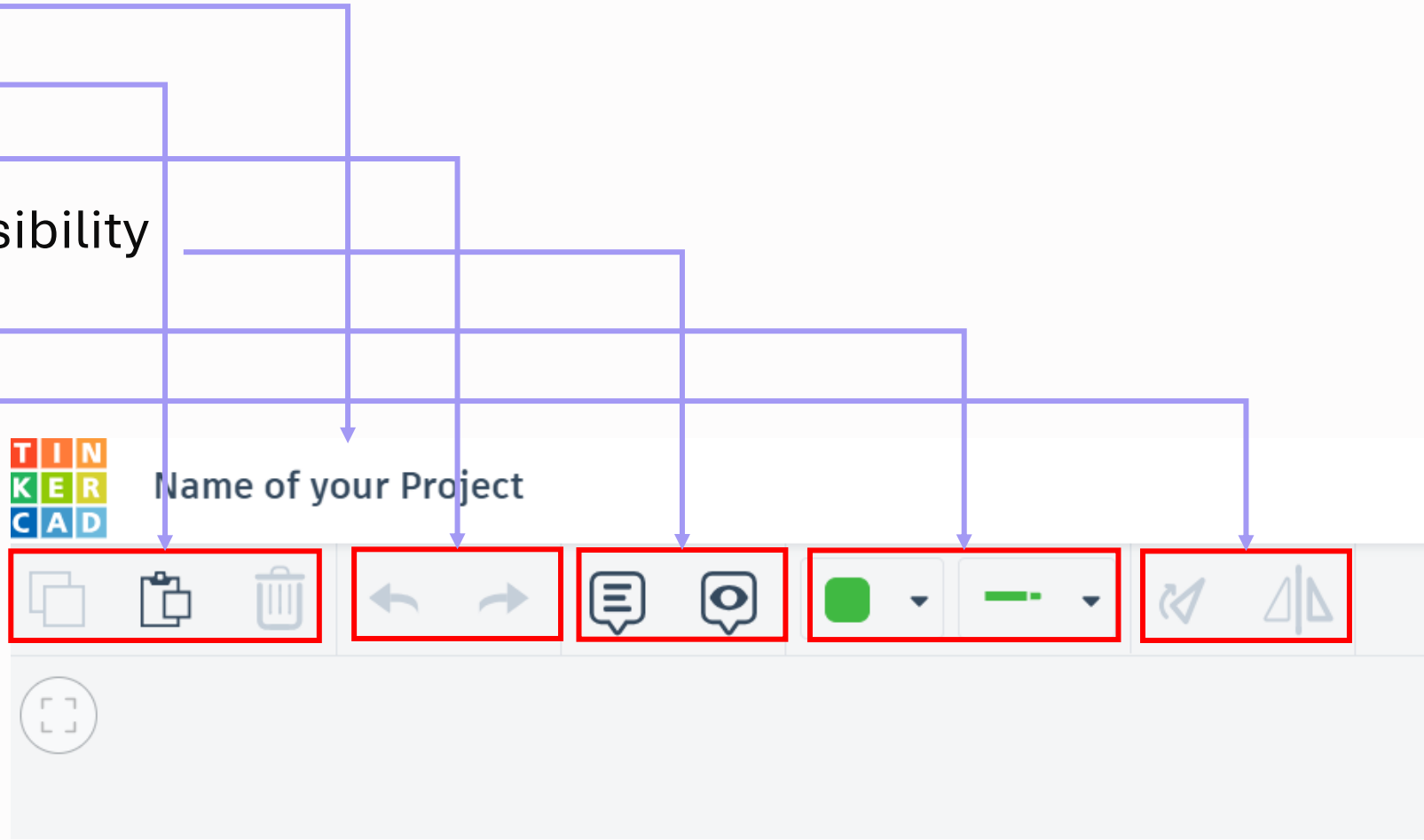
Joining a Tinkercad Class

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.

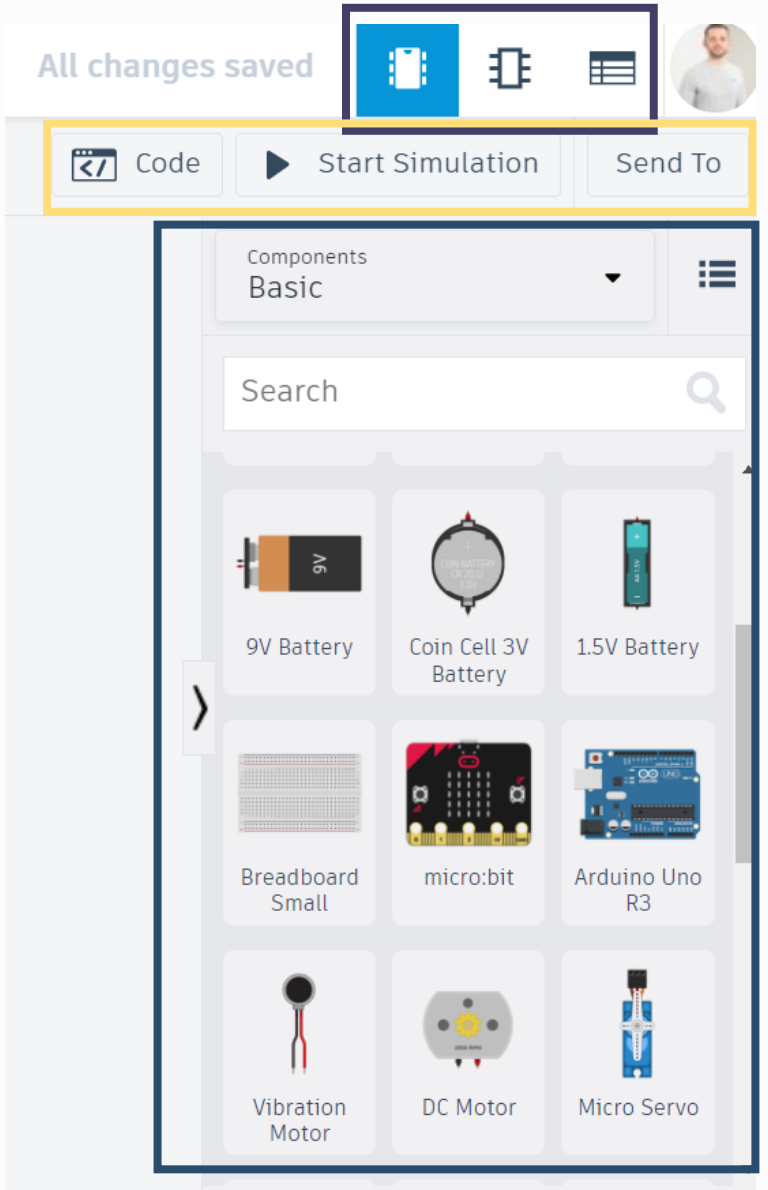
Tinkercad User Interface

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



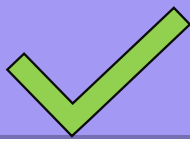
Tinkercad User Interface

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

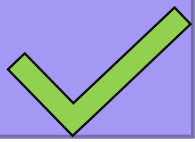


Summary

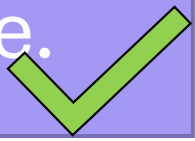
We introduced different Arduino Platforms for learning and practising.



We learned how to create and account on TinkerCAD.

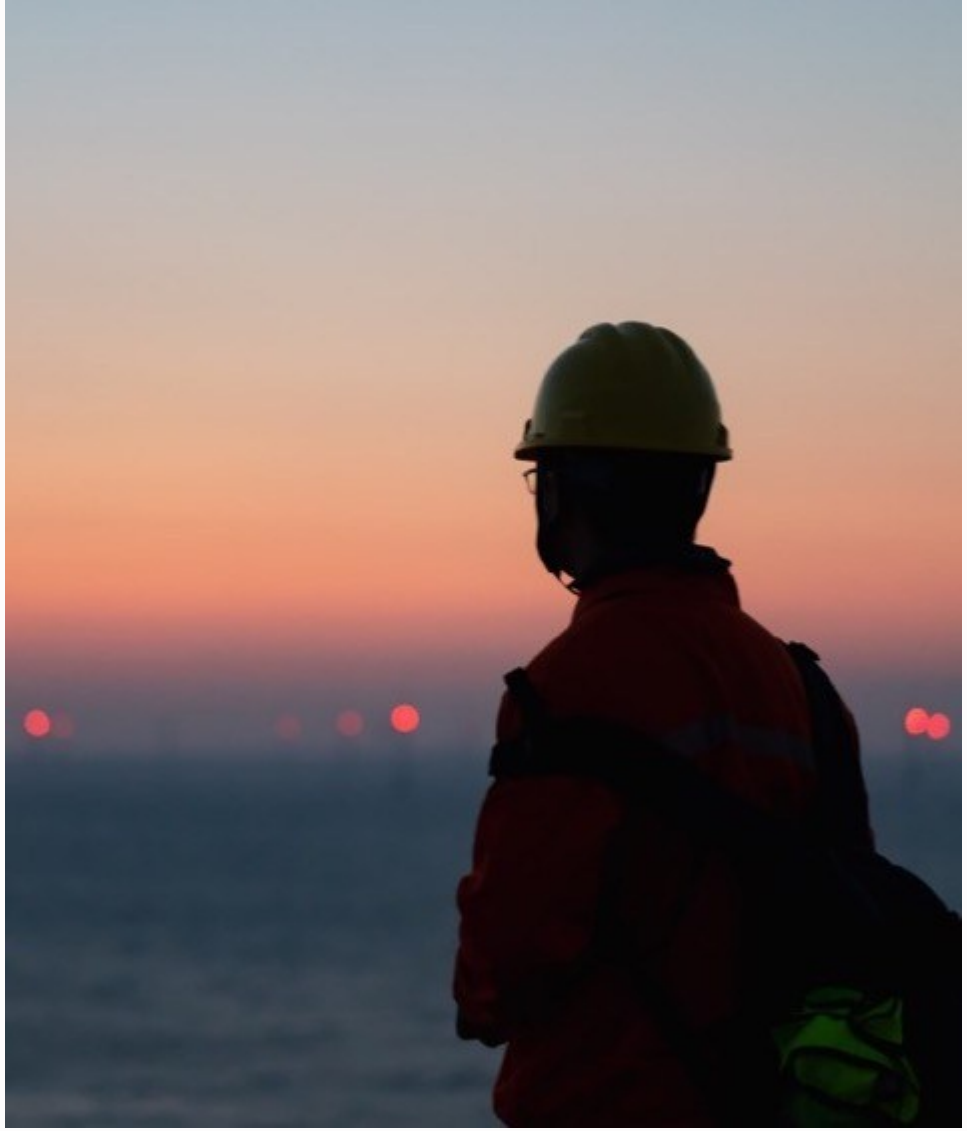


We introduced the TinkerCAD circuit simulation user interface.



Thank you

Leonardo.Alves-Dias@warwick.ac.uk



UNIVERSITY OF WARWICK