Building a Photometer

In this session you will:

1. understand what microcontrollers are and give examples;
2. understand what Arduino is and know how to set it up;
3. understand what a light sensor is and learn how to run a light detection experiment with the Arduino board.

Part 1: Review

Goal: Participants understand what microcontrollers are and give examples.

What is a microcontroller?

Have you ever wondered how a robot knows how to move or how a traffic light knows when to change colors? That's thanks to something called a **microcontroller**!

A microcontroller is like a tiny **computer brain** inside many of the gadgets we use every day. It's small enough to fit on your fingertip but smart enough to tell machines what to do.

How does it work?

- A microcontroller has three important parts:
    1. **Brain (Processor):** It thinks and makes decisions based on instructions.
    2. **Memory:** It remembers the instructions and some information for later.
    3. **Connectors (Inputs and Outputs):** They let the microcontroller talk to sensors, lights, buttons, or motors.

What can microcontrollers do?

They can make:

- A robot follows a line on the floor.
- A lamp turns on when it gets dark.
- A toy car drive itself.
- A door lock when you press a button.

Why are microcontrollers cool?

Microcontrollers are everywhere—inside toys, cars, and even your microwave! They help us build amazing things and solve problems. With a little bit of coding, you can teach a microcontroller to do all sorts of fun stuff. It's like being a magician for machines!

What about you? Can you think of any microcontrollers you may have already worked with or heard about?

Part 2: Getting started with Arduino[1]

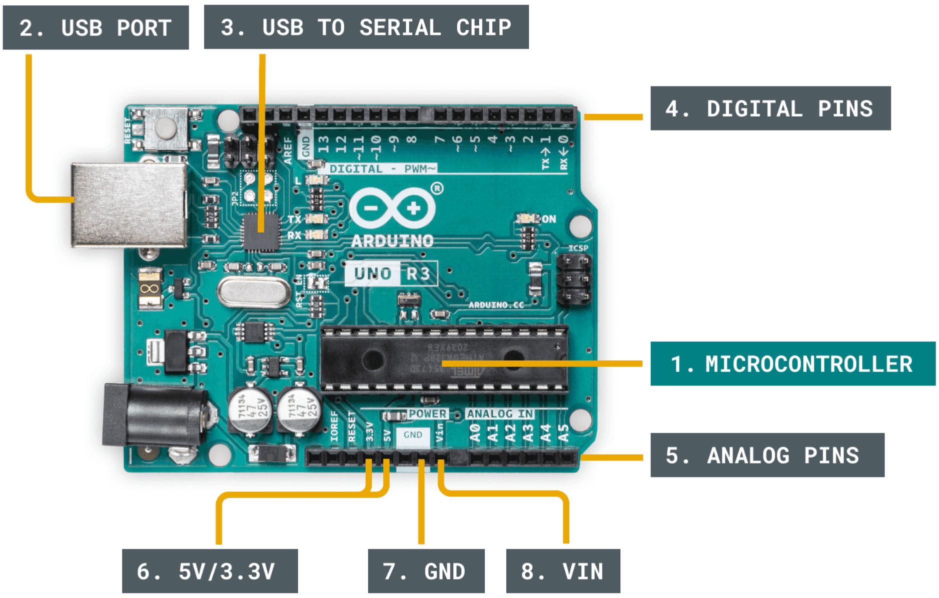
Goal: Participants understand what Arduino is and know how to set it up.

What is Arduino?

Arduino is an open-source electronics platform that makes it easy to create projects using both hardware and software. With an Arduino board, you can read inputs like light, button presses, or even messages and turn them into outputs like controlling a motor, lighting up an LED, or sharing data online. By using the Arduino programming language and software, you can program the board to follow your instructions. Arduino was designed to help students without any experience in electronics or coding, and it has grown into a global community where people share knowledge and create amazing projects, from simple gadgets to advanced tools for IoT, wearables, and more.

Anatomy of an Arduino Board

While all Arduino boards differ from each other, there are several key components that can be found on practically any Arduino. Let's take a look at the image below:



- **1. Microcontroller** - this is the brain of an Arduino, and is the component that we load programs into. Think of it as a tiny computer, designed to execute only a specific number of things.
- **2. USB port** - used to connect your Arduino board to a computer.
- **3. USB to Serial chip** - the USB to Serial is an important component, as it helps translating data that comes from e.g. a computer to the on-board microcontroller. This is what makes it possible to program the Arduino board from your computer.

[1] Part of this content was extracted from https://docs.arduino.cc/

- **4. Digital pins** - pins that use digital logic (0,1 or LOW/HIGH). Commonly used for switches and to turn on/off an LED.
- **5. Analog pins** - pins that can read analog values in a 10 bit resolution (0-1023).
- **6. 5V / 3.3V pins**- these pins are used to power external components.
- **7. GND** - also known as ground, negative or simply -, is used to complete a circuit, where the electrical level is at 0 volt.
- **8. VIN** - stands for Voltage In, where you can connect external power supplies.

Depending on the Arduino board, you will find many more components. The items listed above are generally found on any Arduino board.

How to set up an Arduino board

Before you start controlling the world around you, you'll need to set up the software to program your board. Let's watch this short video to have a better idea on how to do it:

https://youtu.be/-hUrgce4n4A

On this video we heard about some new things. Let's understand better what they are:

- **Firmware:** a small program inside a device that tells it how to work properly.
- **Arduino IDE:** the Arduino Integrated Development Environment (IDE) is a software that allows you to write code and upload it to your Arduino hardware.
- **Sketch:** a file that you write your program inside. It has the .ino extension, and is always stored in a folder of the same name.

Besides the convenient code editing functions, the Arduino Software (IDE) is equipped with a list of libraries that provide extra functionality for use in sketches, making it easier for you to connect sensors, displays, modules, etc.

You can write programs and upload them to your board with the browser IDE (Arduino Cloud Editor), or the desktop one (**Arduino Software IDE**). For practical reasons, in this workshop we're going to use the latest version of the desktop IDE (2.3.3).

Using the Arduino Software IDE

The Arduino Software (IDE) makes it easy to write code and upload it to the board offline. We recommend it for users with poor or no internet connection. This software can be used with any Arduino board.

There are currently two versions of the Arduino IDE, one is the IDE 1.x.x and the other is IDE 2.x. The IDE 2.x is new major release that is faster and even more powerful to the IDE 1.x.x. In addition to a more modern editor and a more responsive interface it includes advanced features to help users with their coding and debugging.

The following steps can guide you with using the offline IDE:

1. Download and install the Arduino Software IDE (to save time, we have already done this step for you ;D)

2. Connect your Arduino board to your device.

3. Open the Arduino Software (IDE): look for this symbol ∞ on the taskbar.

The Arduino Integrated Development Environment - or Arduino Software (IDE) - connects to the Arduino boards to upload programs and communicate with them. Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension .ino.

Using the offline IDE 2.3.3

The editor contains the four main areas:

1. A **toolbar with buttons** for common functions and a series of menus. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, choose your board and port and open the serial monitor.

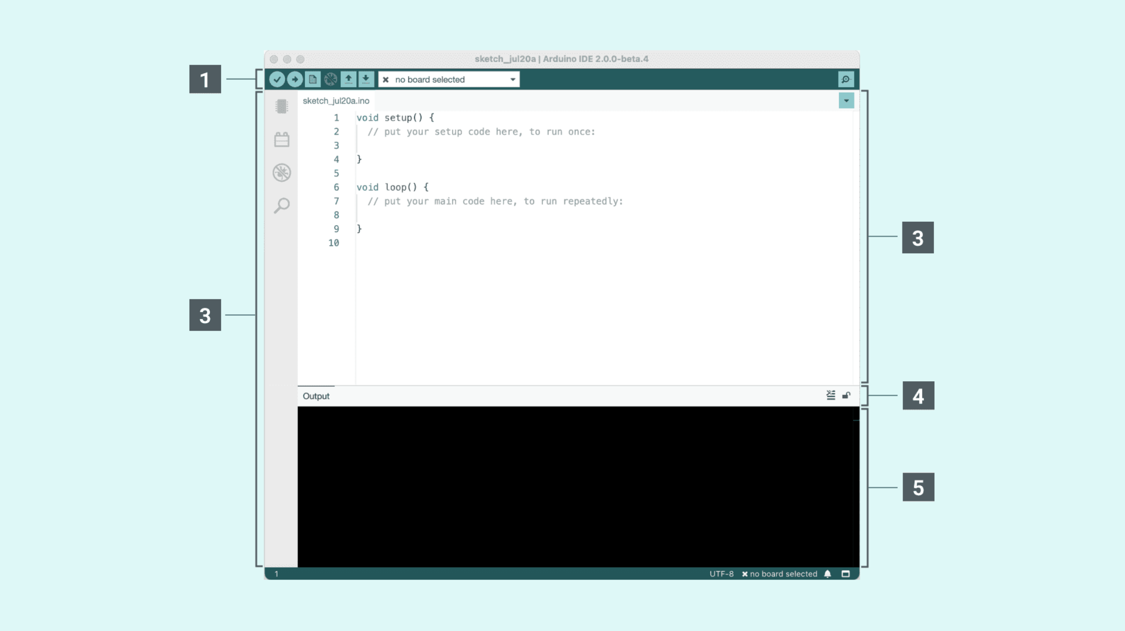
2. The **Sidebar** for regularly used tools. It gives you quick access to board managers, libraries, debugging your board as well as a search and replacement tool.

3. The **text editor** for writing your code.

4. Console controls gives control over the output on the console.

5. The **text console** displays text output by the Arduino Software (IDE), including complete error messages and other information.

The bottom right-hand corner of the window displays the configured board and serial port.

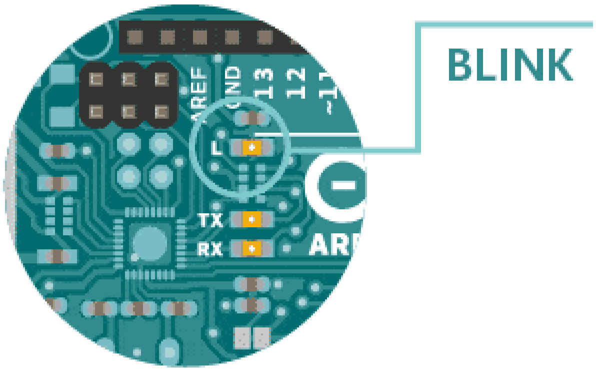


```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

The Arduino Software IDE

Now that you are all set up, **let's try to make your board blink!**

1. **Connect your Arduino** board to your computer.
2. Now, you need to **select the right board & port**. This is done from the toolbar:
   a. For the **board**: Tools>Board>Arduino AVR Boards>Arduino Uno
   b. For the **port**: Tools>Port>COM5
3. Let's **try an example**: navigate to **File > Examples > 01.Basics > Blink.**
4. To **upload it to your board**, simply click on the arrow in the top left corner. This process takes a few seconds, and it is important to not disconnect the board during this process. If the upload is successful, the message "Done uploading" will appear in the bottom output area.
5. Once the upload is complete, you should see on your board the yellow LED with the letter **L** next to it, start blinking. You can **adjust the speed of blinking** by changing the delay number in the parenthesis to 100, and upload the Blink sketch again. Now the LED should blink much faster.



Congratulations! You have successfully programmed your board to blink its on-board LED!

***Small hint:** if you want the on-board LED to stop blinking, you can just fade it back by using the sketch "fade", also from the basic examples menu.

Part 3: Let's start experimenting!

Goal: Participants understand what a light sensor is and learn how to run a light detection experiment with the Arduino board.

What is a sensor?

A sensor, in simple terms, is used to *sense* its environment, meaning it records a physical parameter, for example temperature, and converts it into an electronic signal.

There are many types of sensors, and several ways of recording data from them. Perhaps the easiest to use is an analog sensor, where we communicate a range of values through altering the voltage input fed into an Arduino analog pin (usually between 0-5 volts). This simply gives you a range between 0-1023 (a 10-bit resolution).

Now, what is a light sensor?

A light sensor measures the brightness of light and converts this information into an electronic signal. In this workshop, we will use a special type of light sensor called **spectral sensor** (the AS7262 from SparkFun) to build a simple photometer.

The spectral sensor doesn't just measure how bright the light is – it can also detect different colors of light, just like our eyes, but it measures them more precisely. It detects specific colors in the visible spectrum at 450 nm (violet), 500 nm (blue), 550 nm (green), 570 nm (yellow), 600 nm (orange), and 650 nm (red). By connecting this sensor to an Arduino Uno, we can record and analyze this light data for experiments. This is a powerful tool to explore how light interacts with different materials!

Now that you understood what a light sensor is, let's try something out with our Arduino board!

Experiment: Detecting Calliope Mini Onboard LED Colors with the AS7262 Sensor

What You'll Need

- Arduino Uno board
- USB cable
- AS7262 Spectral Sensor
- Connecting wires
- Calliope Mini kit

Step 1: Set Up the Circuit

1. **Connect the AS7262 sensor to the Arduino board** using the following pins:
   - **3V3** on the sensor to **3.3V** on the Arduino;
   - **GND** on the sensor to **GND** on the Arduino;

- **SCL** on the sensor to **SCL** on the Arduino;
- **SDA** on the sensor to **SDA** on the Arduino.

Step 2: Program the Calliope Mini

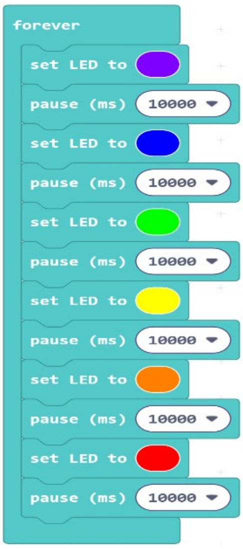
Use **MakeCode for Calliope Mini** to program the onboard RGB LED to light up in different colors.

Program Flow:

- The Calliope Mini will light up its onboard RGB LED in one color at a time (e.g., red, green, blue).
- Each color will stay on for a few seconds, giving the AS7262 sensor time to measure the light.

Here's an example on how to do this in MakeCode:

1. Open MakeCode for Calliope Mini in your web browser.
2. Create a new project.
3. Set Up the LED Colors: use the "**set LED to**" block (found in the Basic > RGB LED section of MakeCode) to control the onboard RGB LED.
4. The block program should look like this (you can choose the colors as you wish, up to 6 colors):
    - Set the RGB LED to **violet**.
    - Pause for **10000 ms**.
    - Set the RGB LED to **blue**.
    - Pause for **10000 ms**.
    - Set the RGB LED to **green**.
    - Pause for **10000 ms**.
    - Set the RGB LED to **yellow**.
    - Pause for **10000 ms**.
    - Set the RGB LED to **orange**.
    - Pause for **10000 ms**.
    - Set the RGB LED to **red**.
    - Pause for **10000 ms**.



```
forever
    set LED to 🟣
    pause (ms) 10000
    set LED to 🔵
    pause (ms) 10000
    set LED to 🟢
    pause (ms) 10000
    set LED to 🟡
    pause (ms) 10000
    set LED to 🟠
    pause (ms) 10000
    set LED to 🔴
    pause (ms) 10000
```

5. Create a loop: use the **"forever" block** (found in the **Loops** section) to repeat the lighting pattern continuously.
6. Upload the program to the Calliope Mini as usuall.

Step 3: Load the code to program the Arduino

Sensors normally have their own library, which provide extra functionality for use in sketches. A number of libraries come installed with the IDE, but you can also download or create your own. So here are the steps to install the SparkFun AS726X library:

1. Open the Arduino IDE on your computer.
2. Go to **Sketch > Include Library > Manage Libraries** and search for "AS726X." Install the library by SparkFun.

3. Load the example code:
    a. Go to **File > Examples > SparkFun_AS726X > Example1_Readings.**
    b. This code will read the light intensity values for different colors (spectral bands).
4. Upload the code to your Arduino.

Step 4: Run the Experiment

1. Power on the Calliope Mini, which will cycle through your colors sequence on its onboard RGB LED.
2. Open the Arduino Serial Monitor on the upper right corner to observe the AS7262 sensor readings.
3. Point the AS7262 sensor at the RGB LED (it has to be stable) and observe the values printed in the Serial Monitor.
4. Note how the spectral data changes with each color of the RGB LED:
    - It may be easier to observe only one spectral channel at a time: e.g. observe only the measurements taken by the "V" channel first, and write them down for all the colors at the table below. When the cycle starts again, observe only the next channel "B" and so on and so forth.

	Spectral Channels					
RGB LEDs	**V [450 nm]**	**B [500 nm]**	**G [550 nm]**	**Y [570 nm]**	**O [600 nm]**	**R [650 nm]**

Step 5: Analyze the Results

- Compare the sensor readings for all the different LED colors. What do you observe? What happens to the V, B, G, Y, O and R readings from a same color?
- Discuss which spectral channels (V, B, G, Y, O, R) correspond to which RGB colors.
- Reflect on how the AS7262 sensor can be used to detect light wavelengths and colors in various real-world applications.