Prototyping Methods for HCI

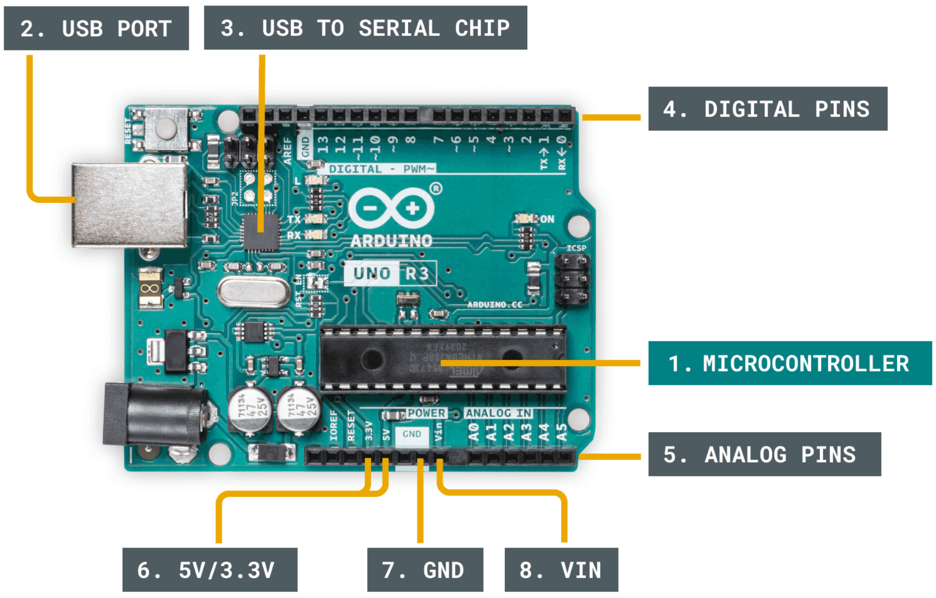
Arduino Vademecum

Designing and implementing prototypes using software and microprocessor platforms.

1) Anatomy of an Arduino Board

https://docs.arduino.cc/tutorials/uno-rev3/board-anatomy

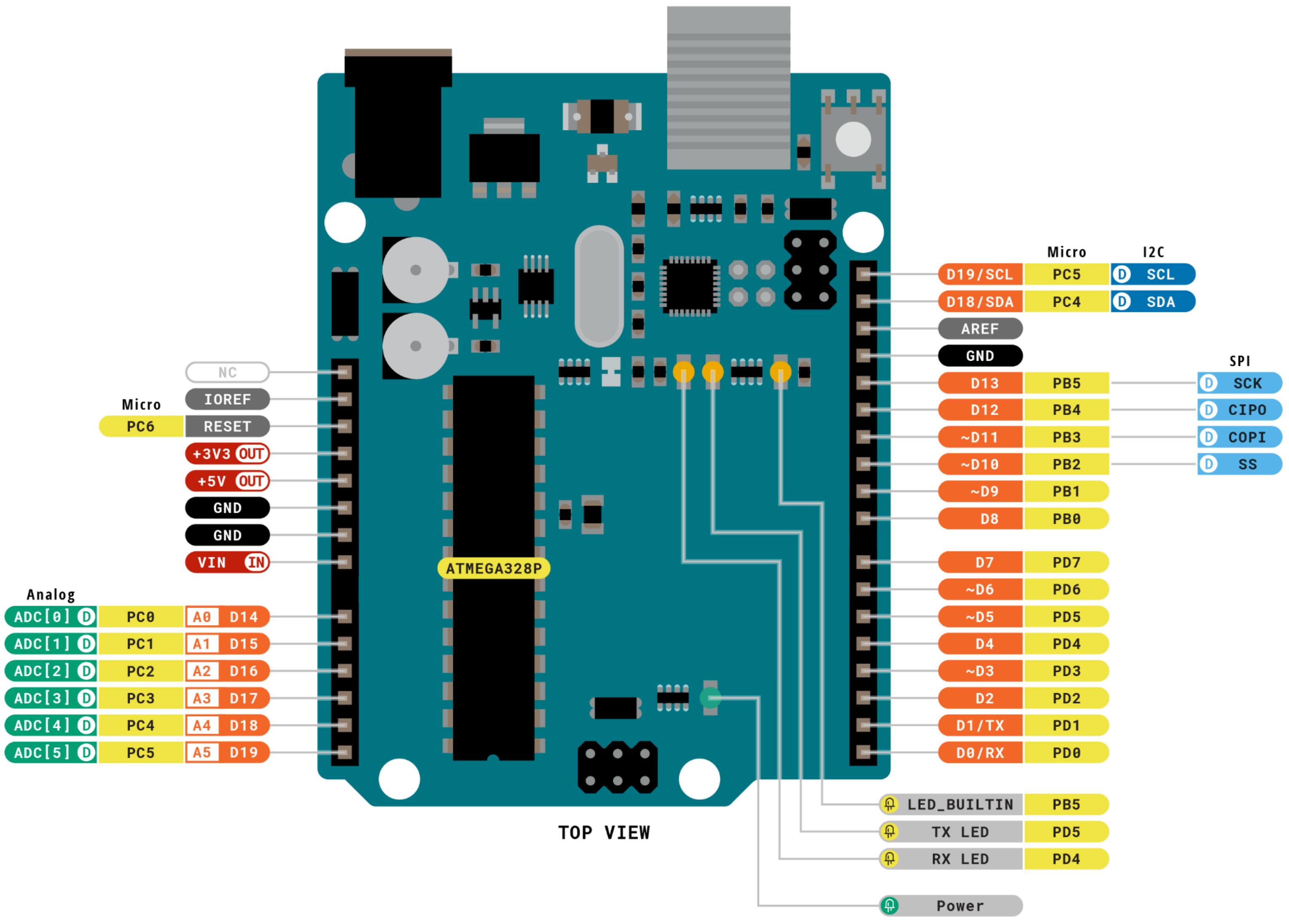
While all Arduino boards differ from each other, there are several key components that can be found on practically any Arduino. Let's take a look at the image below:



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.
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.

Pinouts

https://docs.arduino.cc/retired/boards/arduino-uno-rev3-with-long-pins



Legend:		
	🟧 Digital	🟦 I2C
🟥 Power	⬜ Analog	🟦 SPI
⬛ Ground	🟨 Main Part	🟩 Analog

ARDUINO

ARDUINO UNO REV3
SKU code: A000066
Pinout
Last update: 6 Oct, 2022

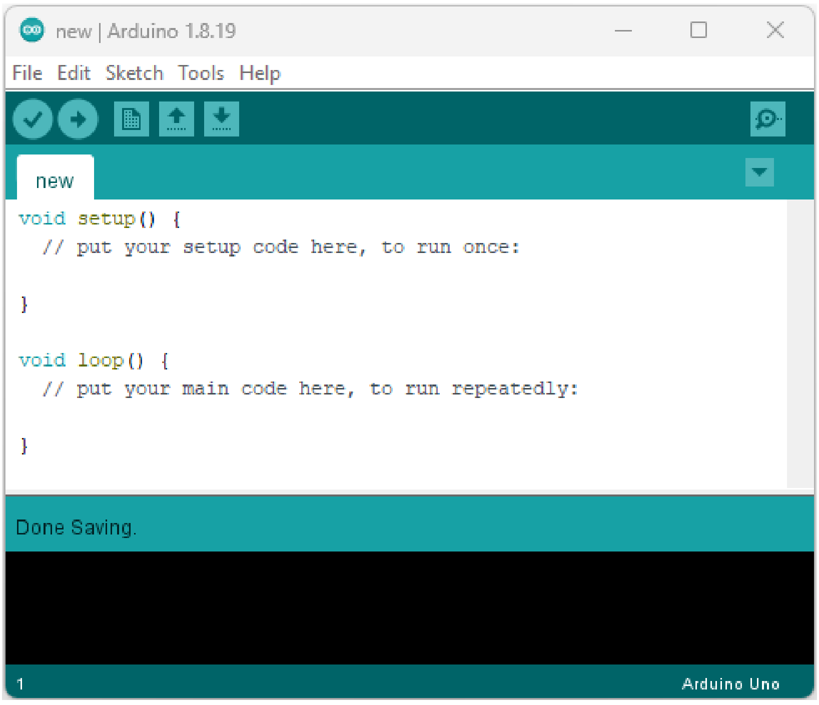
2) Language Reference

https://www.arduino.cc/reference/en/

Structure

Sketch

- ➔ **setup()**
  - ◆ Description: The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup() function will only run once, after each powerup or reset of the Arduino board.
- ➔ **loop()**
  - ◆ Description: After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.



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

}

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

}
```

Done Saving.

Arithmetic Operators

- ➔ **% (remainder)**
- ➔ *** (multiplication)**
- ➔ **+ (addition)**
- ➔ **- (subtraction)**
- ➔ **/ (division)**
- ➔ **= (assignment operator)**

Comparison Operators

- ➔ **!= (not equal to)**
- ➔ **< (less than)**

- ➔ **<= (less than or equal to)**
- ➔ **== (equal to)**
- ➔ **> (greater than)**
- ➔ **>= (greater than or equal to)**

Boolean Operators

- ➔ **! (logical not)**
- ➔ **&& (logical and)**
- ➔ **|| (logical or)**

Variables

Constants

- ➔ **HIGH | LOW**
    - ◆ When reading or writing to a digital pin there are only two possible values a pin can take/be-set-to: HIGH and LOW.
    - ◆ HIGH digitalRead():
- ➔ **INPUT | OUTPUT | INPUT_PULLUP**
- ➔ **LED_BUILTIN**
- ➔ **true | false**

Conversion

- ➔ **byte()**
- ➔ **char()**
- ➔ **float()**
- ➔ **int()**
- ➔ **long()**
- ➔ **word()**

Functions

Digital I/O

- ➔ **digitalRead()**
    - ◆ Description: Reads the value from a specified digital pin, either HIGH or LOW.
    - ◆ Syntax: digitalRead(pin)
    - ◆ Parameters: pin - the Arduino pin number you want to read
    - ◆ Returns: HIGH or LOW
- ➔ **digitalWrite()**
    - ◆ Description: Write a HIGH or a LOW value to a digital pin.
    - ◆ Syntax: digitalWrite(pin, value)
    - ◆ Parameters: pin - the Arduino pin number; value - HIGH or LOW
    - ◆ Returns: Nothing
- ➔ **pinMode()**
    - ◆ Description: Configures the specified pin to behave either as an input or an output.
    - ◆ Syntax: pinMode(pin, mode)

   - Parameters: `pin` - the Arduino pin number to set the mode of; mode: INPUT, OUTPUT, or INPUT_PULLUP.
   - Returns: Nothing

Analog I/O

- **analogRead()**
   - Description: Reads the value from the specified analog pin. Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and the operating voltage(5V or 3.3V) into integer values between 0 and 1023. On ATmega based boards (UNO, Nano, Mini, Mega), it takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.
   - Syntax: `analogRead(pin)`
   - Parameters: `pin` - the name of the analog input pin to read from
   - Returns: The analog reading on the pin. Although it is limited to the resolution of the analog to digital converter (0-1023 for 10 bits or 0-4095 for 12 bits)
   - Data type: int
- **analogWrite()**
   - Description: Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady rectangular wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite()) on the same pin.
   - Syntax: `analogWrite(pin, value)`
   - Parameters: `pin` - the Arduino pin to write to (allowed data types: int); `value` - the duty cycle: between 0 (always off) and 255 (always on) (allowed data types: int)
   - Returns: Nothing

Time

- **delay()**
- **delayMicroseconds()**
- **micros()**
- **millis()**

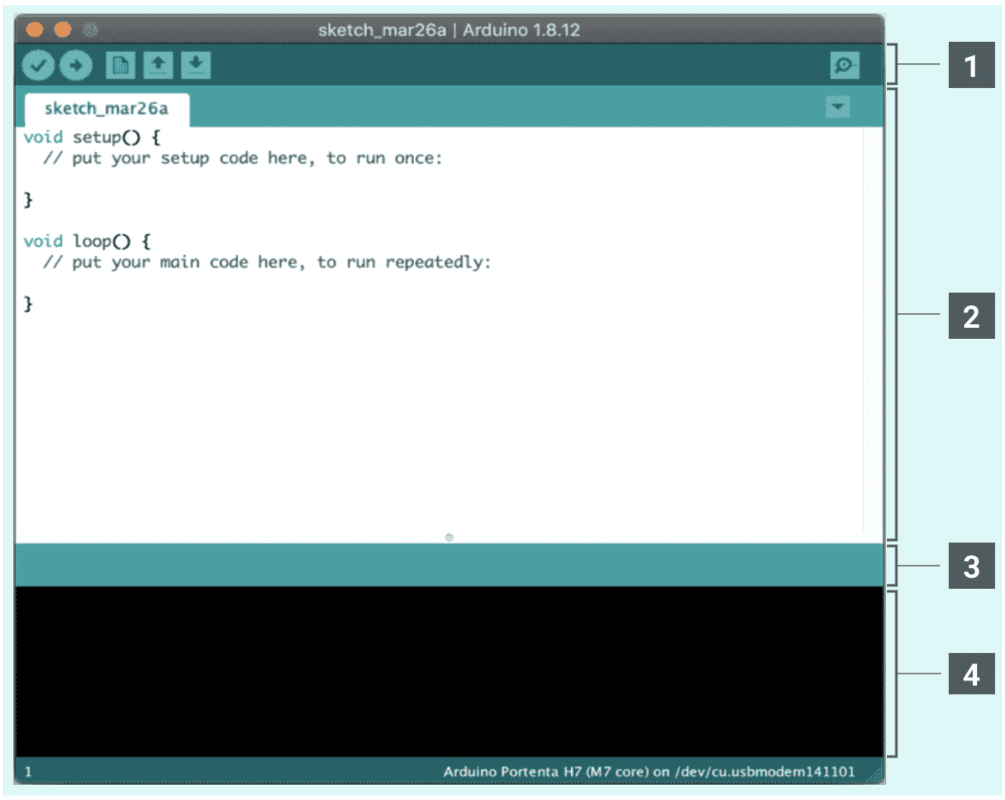
Math

- **abs()**
- **constrain()**
- **map()**
- **max()**
- **min()**
- **pow()**
- **sq()**
- **sqrt()**

Random Numbers

- ➔ random()
- ➔ randomSeed()

3) Using the Arduino Software (IDE 1.x.x)


sketch_mar26a | Arduino 1.8.12
sketch_mar26a
Arduino Portenta H7 (M7 core) on /dev/cu.usbmodem141101
1
2
3
4

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

}

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

}
```

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 programs
- ➔ upload programs
- ➔ create sketches
- ➔ open sketches
- ➔ save sketches
- ➔ open the serial monitor

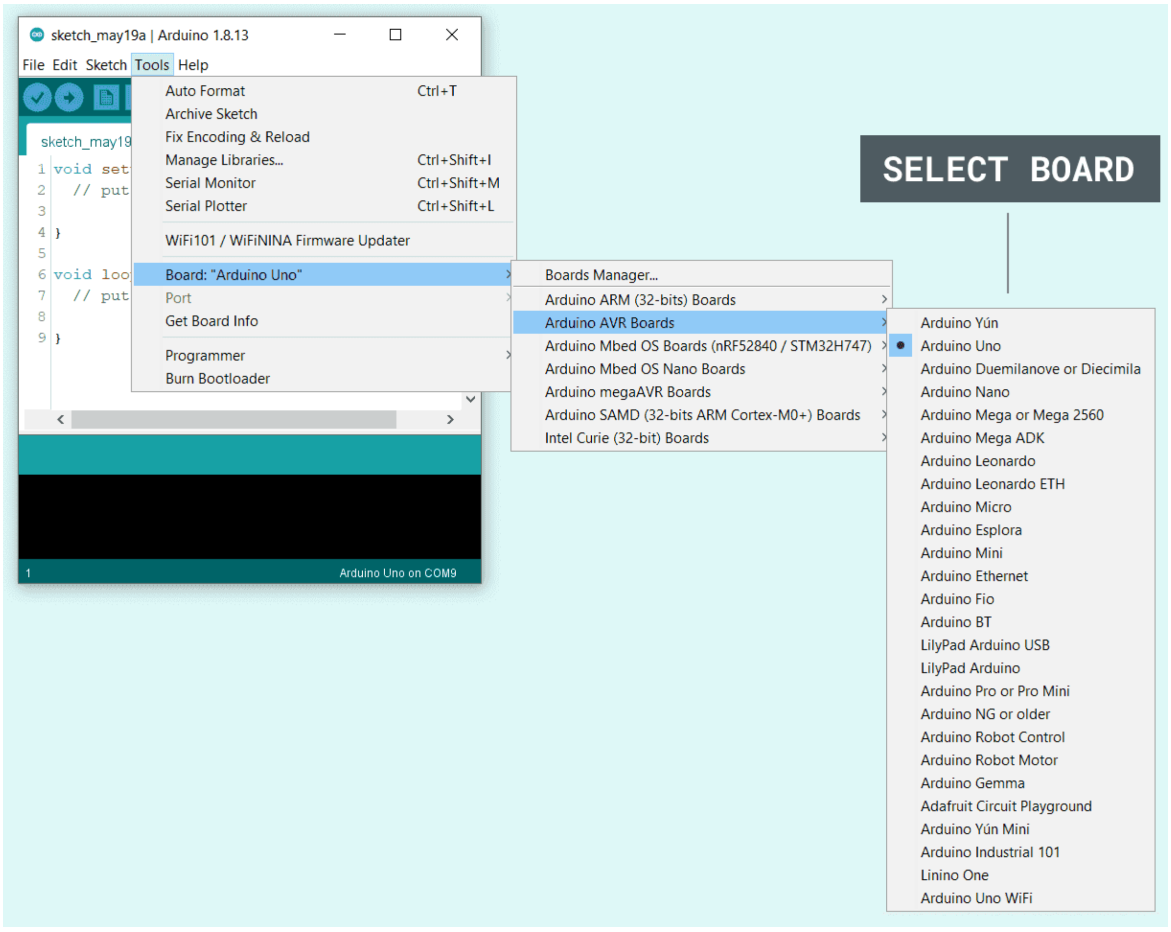
2. The **text editor** for writing your code.

3. The **message area**, gives feedback while saving and exporting and also displays errors.

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

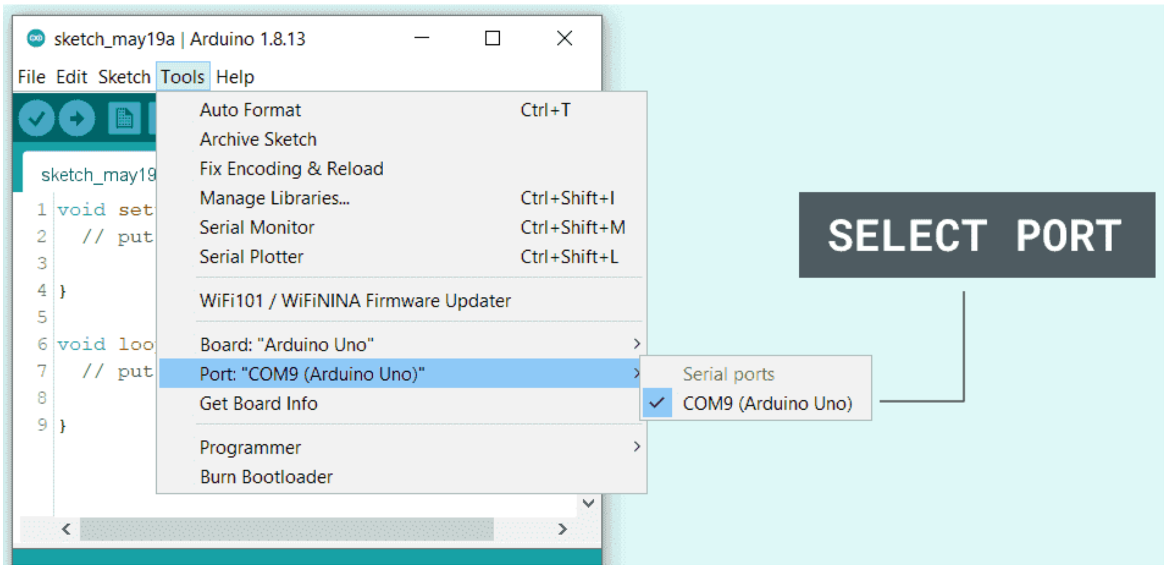
The bottom right-hand corner of the window displays the configured board and serial port.

Selecting a board

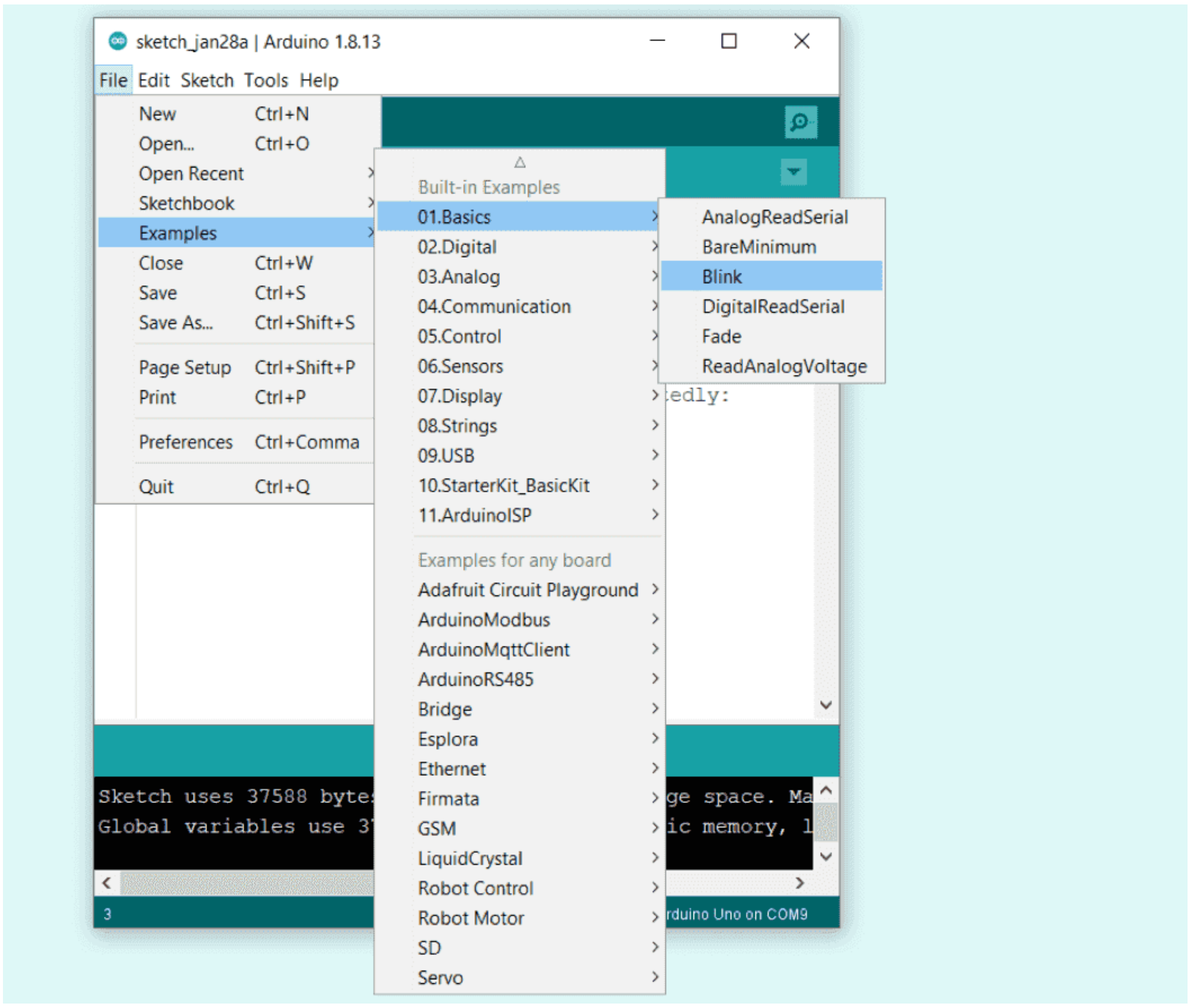


```
void set
  // put

}

void loo
  // put

}
```

Selecting the port


sketch_may19a | Arduino 1.8.13
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port: "COM9 (Arduino Uno)"
Get Board Info
Programmer
Burn Bootloader
Serial ports
COM9 (Arduino Uno)
SELECT PORT

Choosing an example



```
Sketch uses 37588 byte              ge space. Ma
Global variables use 3              ic memory, l
```

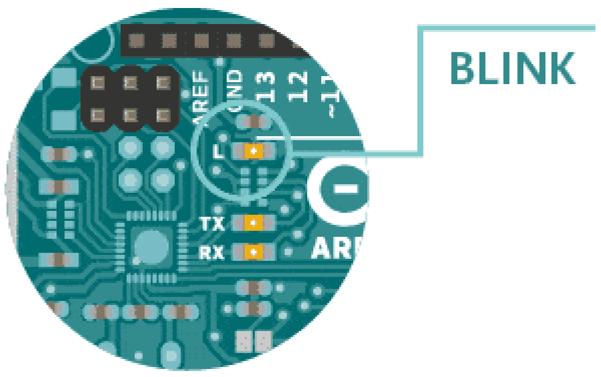
Code_0:

```
/*
```

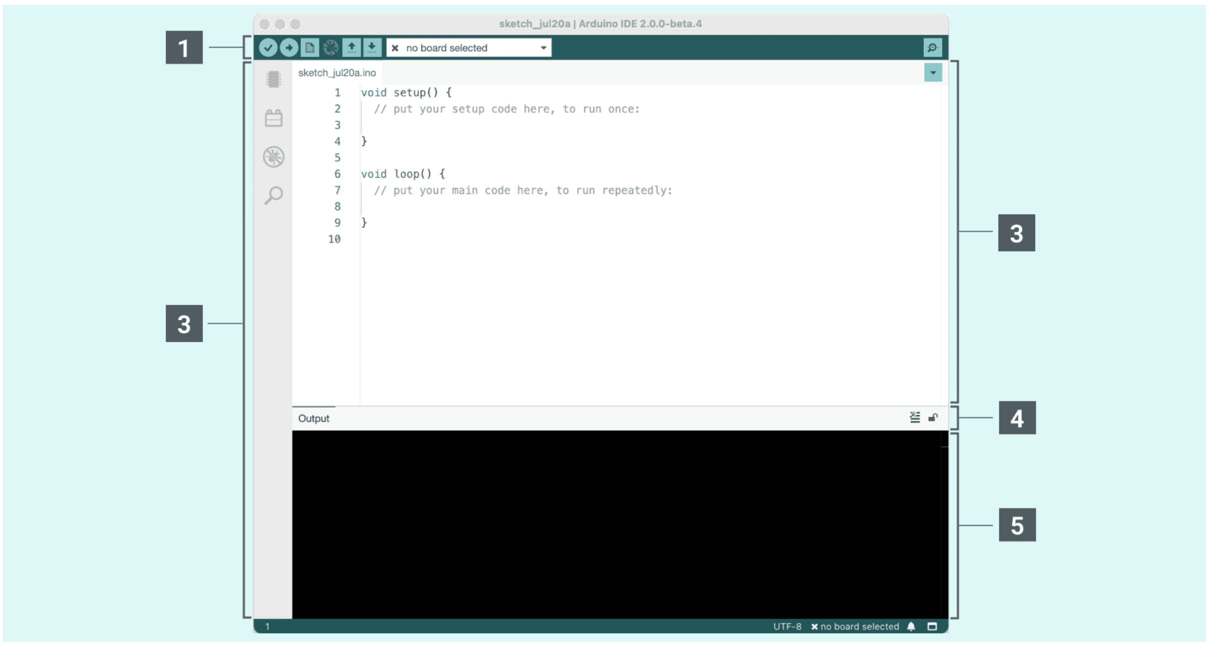
```
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

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.

Once the upload is complete, you should then see on your board the yellow LED with an 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.



4) Using the Arduino Software (IDE 2.x.x)



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

}

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

}
```

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.

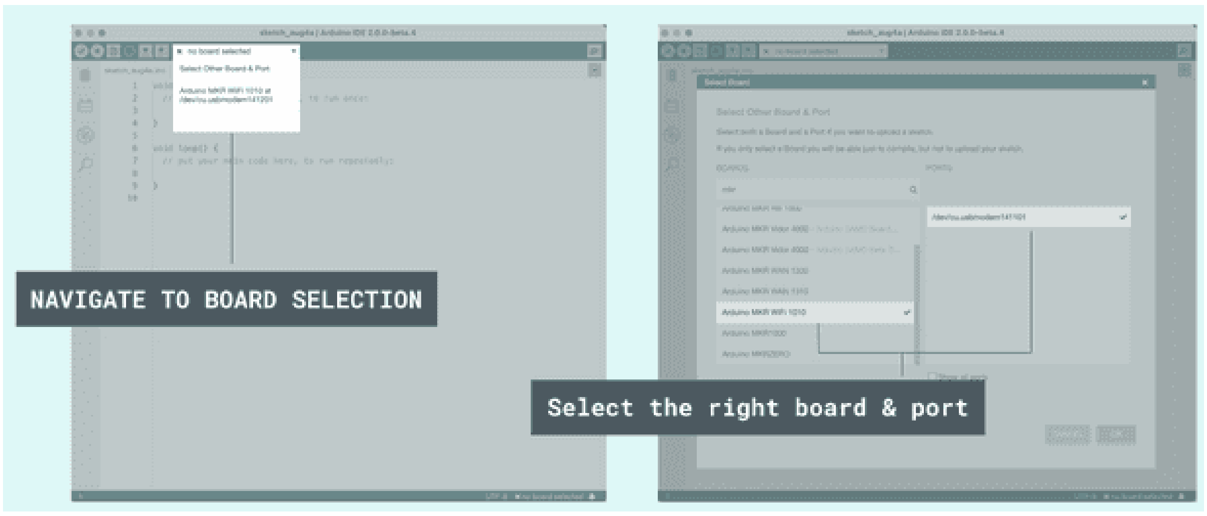
3. The text editor for writing your code.

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

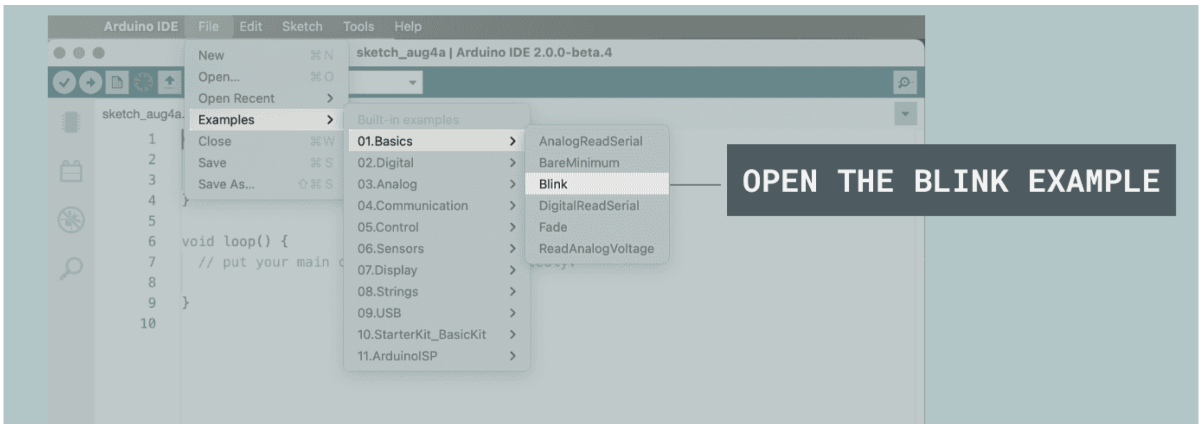
5. The text console displays text output by the Arduino Software (IDE), including complete error messages and other information.

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

Selecting a board

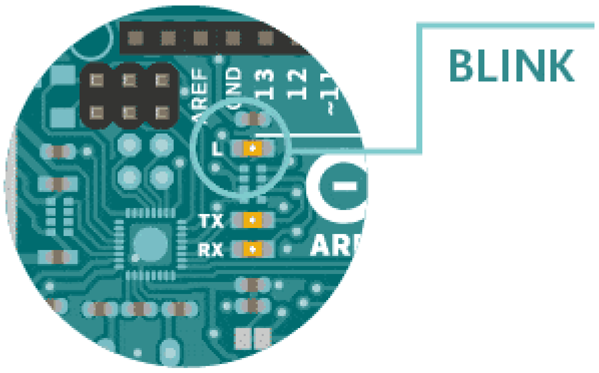


Choosing an example



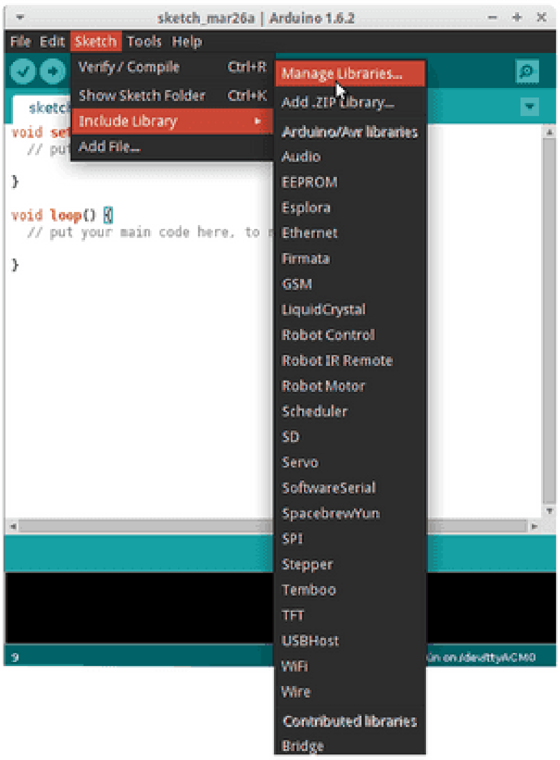
To upload it to your board, simply click on the arrow in the top left corner. This process takes a few seconds, and it is important to not disconnect the board during this process. If the upload is successful, the message "Done uploading" will appear in the bottom output area.

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.



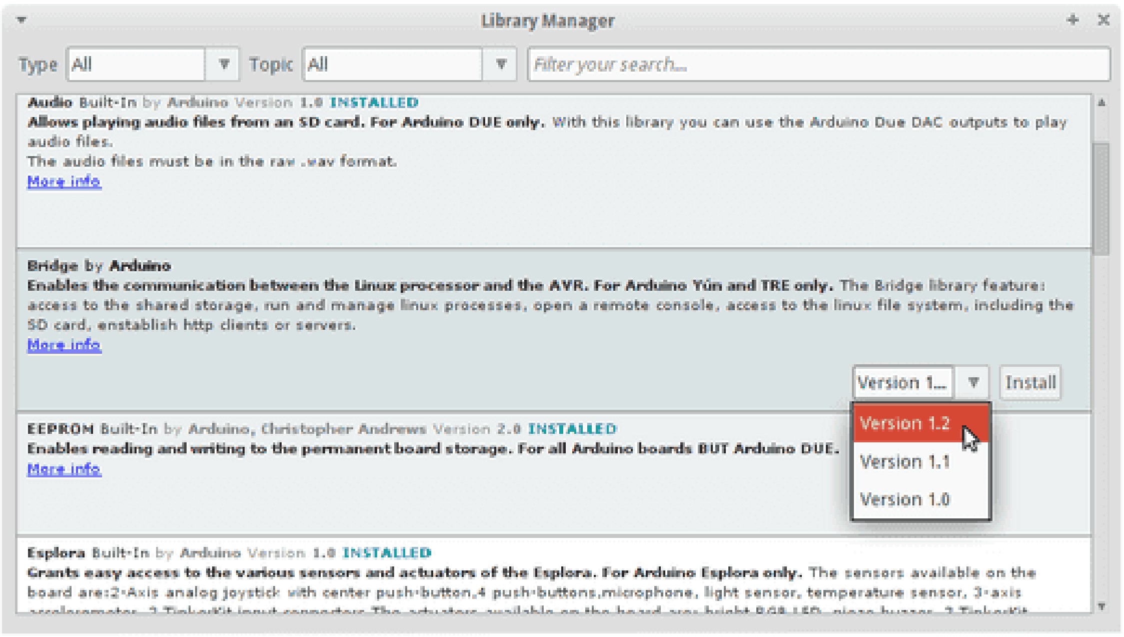
5) Install an Arduino Libraries

To install a new library into your Arduino IDE you can use the Library Manager. Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.



```
void se
  // pu
}

void loop() {
  // put your main code here, to
}
```

Then the Library Manager will open and you will find a list of libraries that are already installed or ready for installation. Scroll the list to find it, click on it, then select the version of the library you want to install. Sometimes only one version of the library is available. If the version selection menu does not appear, don't worry: it is normal.

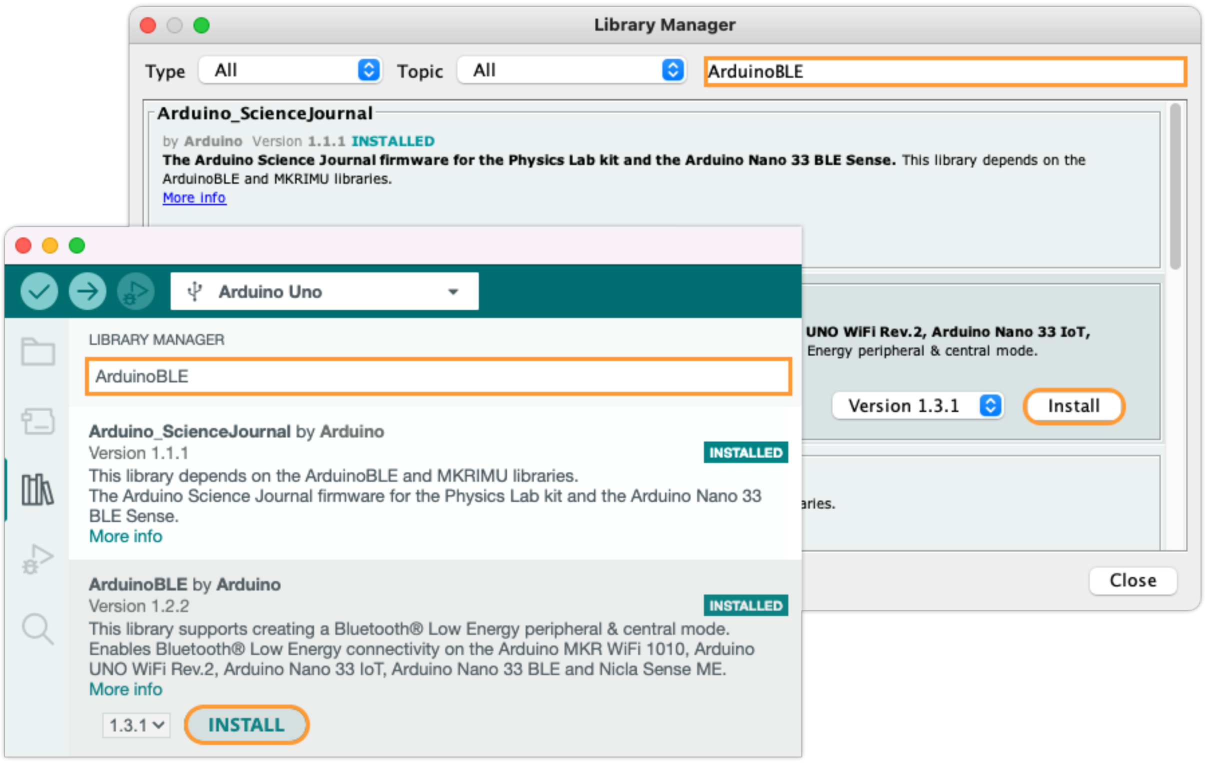


Audio Built-In by Arduino Version 1.0 INSTALLED
Allows playing audio files from an SD card. For Arduino DUE only. With this library you can use the Arduino Due DAC outputs to play audio files.
The audio files must be in the raw .wav format.
More info

Bridge by **Arduino**
Enables the communication between the Linux processor and the AVR. For Arduino Yún and TRE only. The Bridge library feature: access to the shared storage, run and manage linux processes, open a remote console, access to the linux file system, including the SD card, enstablish http clients or servers.
More info

EEPROM Built-In by Arduino, Christopher Andrews Version 2.0 INSTALLED
Enables reading and writing to the permanent board storage. For all Arduino boards BUT Arduino DUE.
More info

Esplora Built-In by Arduino Version 1.0 INSTALLED
Grants easy access to the various sensors and actuators of the Esplora. For Arduino Esplora only. The sensors available on the board are:2-Axis analog joystick with center push-button,4 push-buttons,microphone, light sensor, temperature sensor, 3-axis



Arduino_ScienceJournal
by **Arduino** Version **1.1.1** INSTALLED
The Arduino Science Journal firmware for the Physics Lab kit and the Arduino Nano 33 BLE Sense. This library depends on the ArduinoBLE and MKRIMU libraries.
More info

UNO WiFi Rev.2, Arduino Nano 33 IoT,
Energy peripheral & central mode.

Arduino_ScienceJournal by **Arduino**
Version 1.1.1
INSTALLED
This library depends on the ArduinoBLE and MKRIMU libraries.
The Arduino Science Journal firmware for the Physics Lab kit and the Arduino Nano 33 BLE Sense.
More info

ArduinoBLE by **Arduino**
Version 1.2.2
INSTALLED
This library supports creating a Bluetooth® Low Energy peripheral & central mode. Enables Bluetooth® Low Energy connectivity on the Arduino MKR WiFi 1010, Arduino UNO WiFi Rev.2, Arduino Nano 33 IoT, Arduino Nano 33 BLE and Nicla Sense ME.
More info