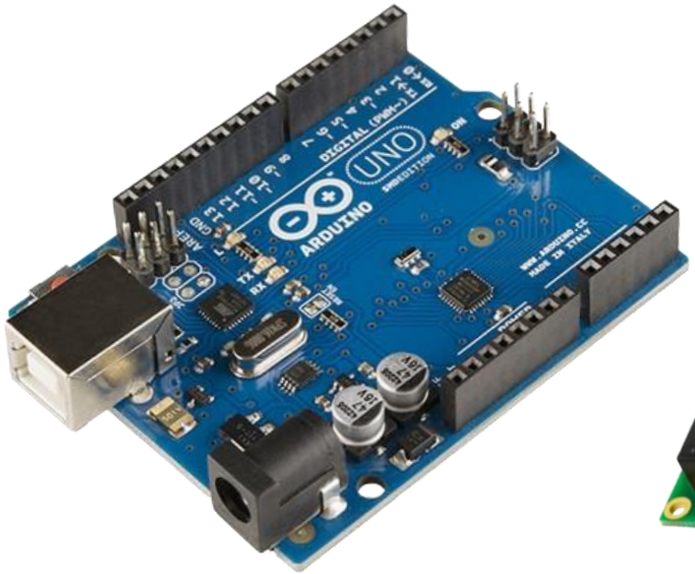
Arduino Course



kakush

Prepared by: Qossay Abu Rida

BIRZEIT UNIVERSITY

Agenda

Embedded system

ADC & DAC

Arduino environment

Basic electronic component

Interrupts

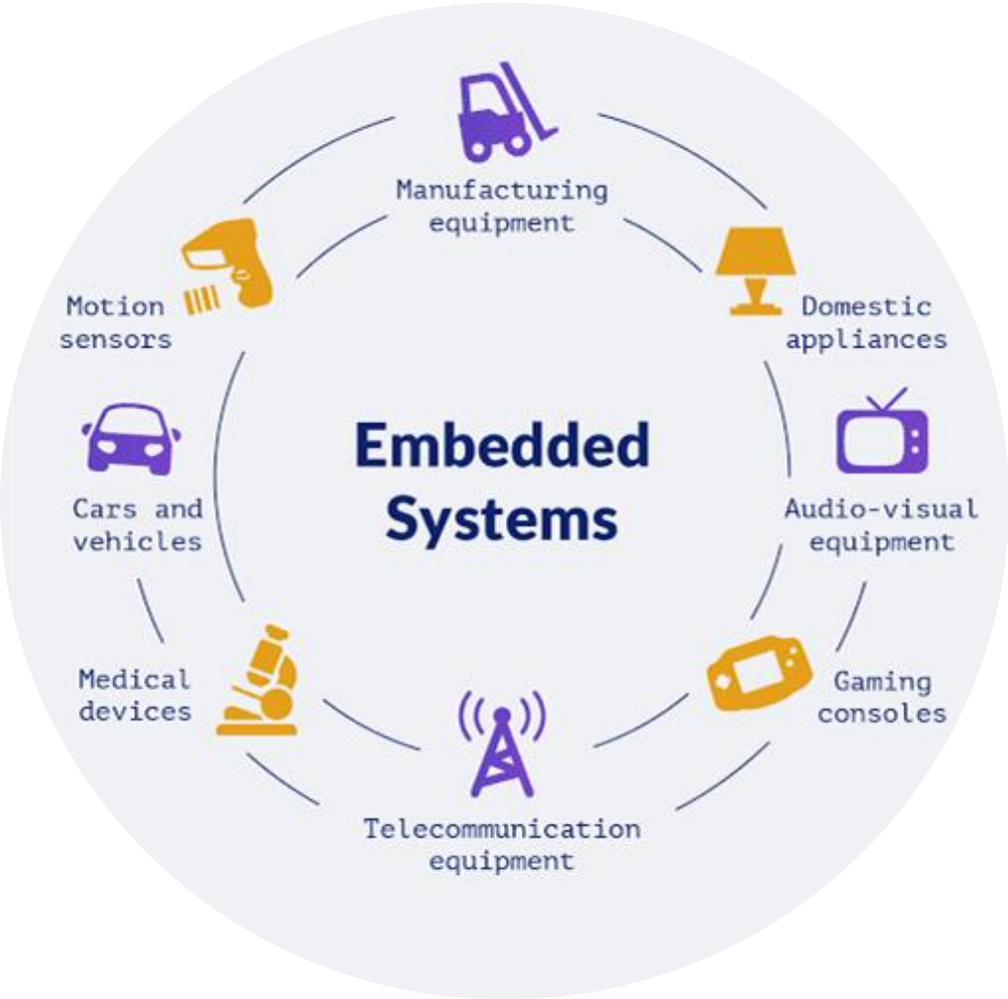
Sensors

Actuators

Communication

kakuʌh

Embedded System



kakush

What is an Embedded System?

kakush

- **Definition**: An embedded system is a specialized computer system that is designed to perform a specific task. Unlike general-purpose computers, which can run a variety of applications, embedded systems are dedicated to one or a few functions.

- **Common Use**: Embedded systems are found in many everyday devices, from household appliances to automobiles, medical equipment, and industrial machines.

Basic Structure of an Embedded System

1- Microcontroller/Microprocessor:

- The brain of the embedded system.
- Executes programmed instructions to perform specific tasks.
- Example: Arduino uses the ATmega328 microcontroller.

2- Memory:

- RAM (Random Access Memory): Temporary storage for data and instructions currently in use.
- ROM (Read-Only Memory): Permanent storage for the system's firmware and operating instructions.

3- Input Devices:

- Collect data from the external environment.
- Examples: Sensors, buttons, keyboards.

4- Output Devices:

- Deliver data or actions to the external environment.
- Examples: Displays, LEDs, motors.

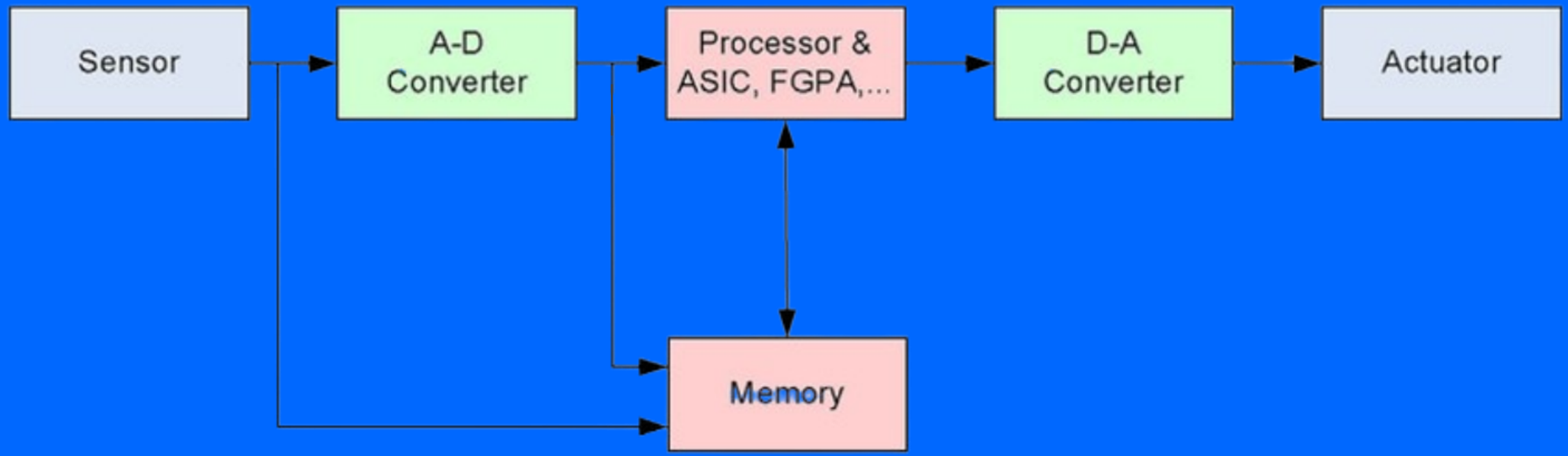
5- Power Supply:

- Provides the necessary power for the system to operate.
- Can be battery-operated or connected to a power source.

6- Communication Interfaces:

- Allow the system to communicate with other systems or devices.
- Examples: Serial ports, USB, Wi-Fi, Bluetooth.

Basic Structure (Symbolic-Based)



Example of an Embedded System: Digital Thermometer

Microcontroller: Reads temperature data from the sensor and processes it.

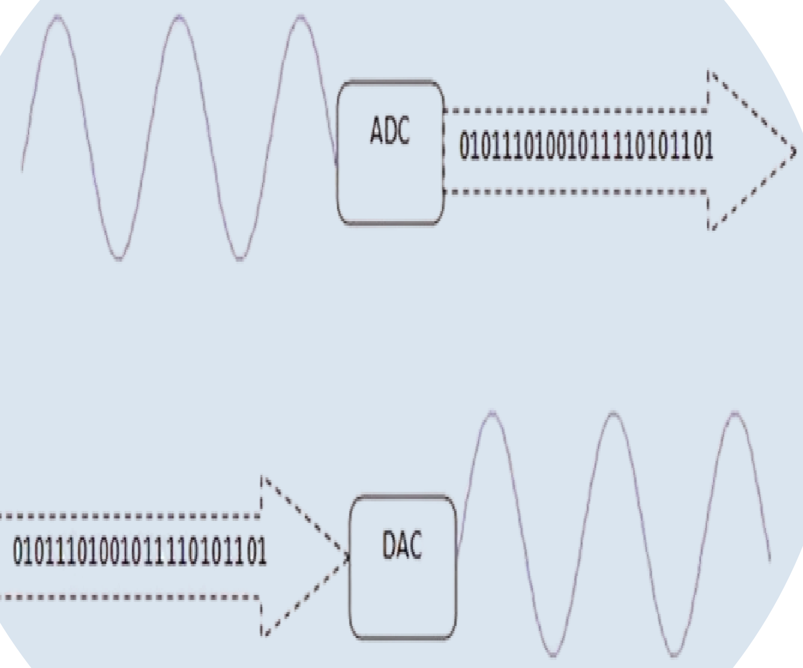
Sensor: Measures the temperature and sends data to the microcontroller.

Display: Shows the temperature reading to the user.

Buttons: Allow users to switch between Celsius and Fahrenheit or reset the device.

Power Supply: Battery that powers the thermometer.

ADC & DAC



ADC
010111010010111110101101
010111010010111110101101
DAC

What is an ADC?

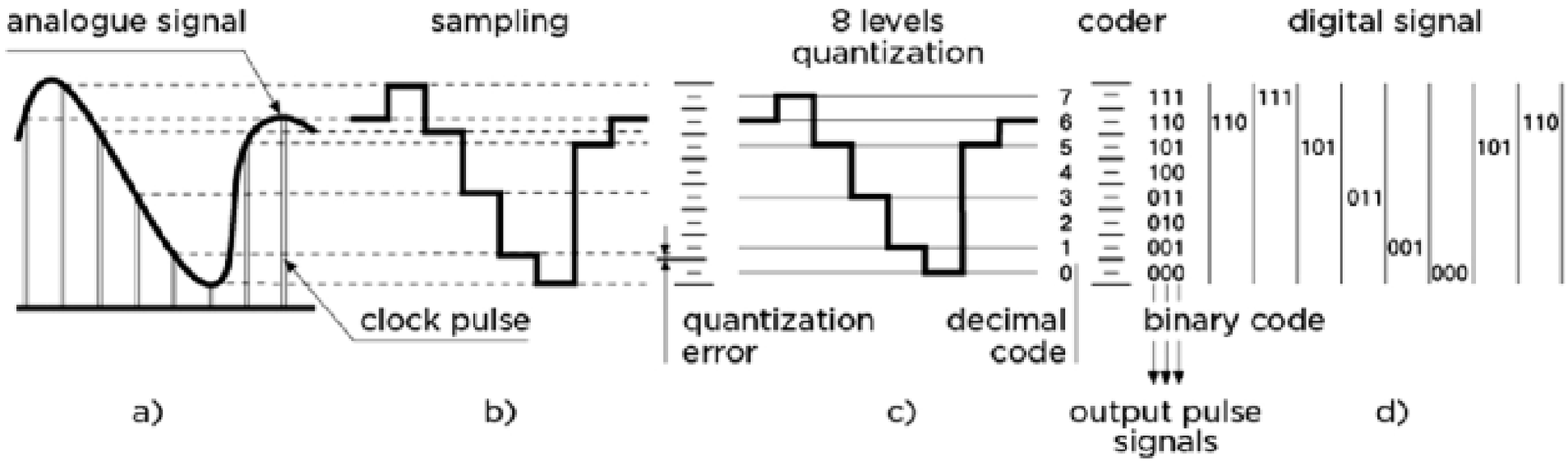
- **Definition**: An Analog-to-Digital Converter (ADC) is a device that converts analog signals (continuous signals) into digital signals (discrete signals) that a microcontroller or computer can process.
- **Purpose**: To allow digital systems to interact with the real world by converting analog inputs (e.g., temperature, sound, light) into a format they can understand and process.

What is a DAC?

- **Definition**: A Digital-to-Analog Converter (DAC) is a device that converts digital signals (discrete signals) into analog signals (continuous signals).
- **Purpose**: To allow digital systems to produce analog outputs (e.g., audio signals, light intensity control) that can be used in the real world.

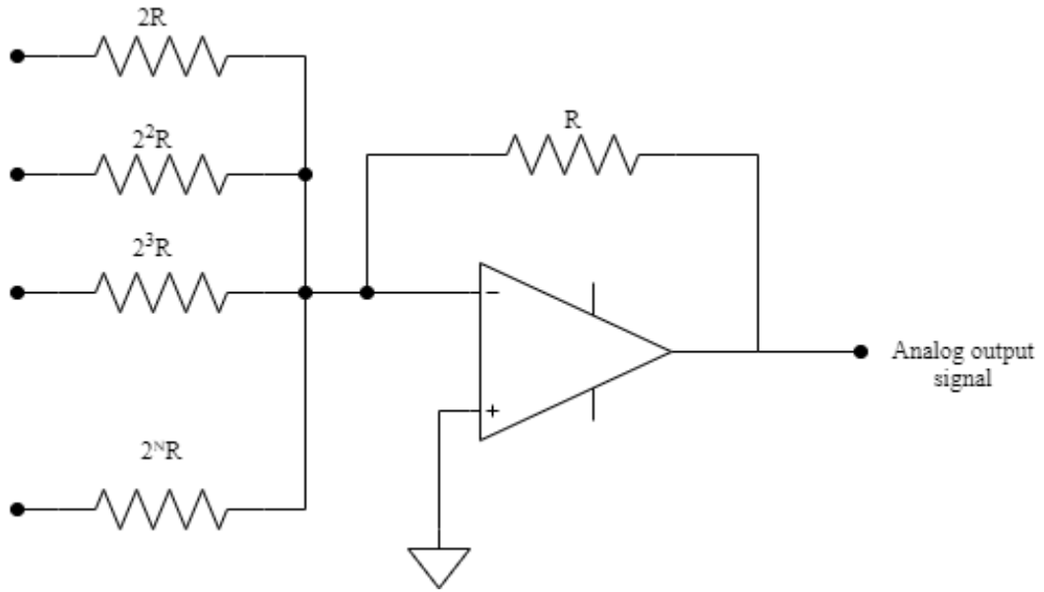
How ADC Works

- **Sampling**: The analog signal is sampled at regular intervals.
- **Quantization**: Each sample is approximated to the nearest value within a range.
- **Encoding**: The quantized values are converted into binary code.

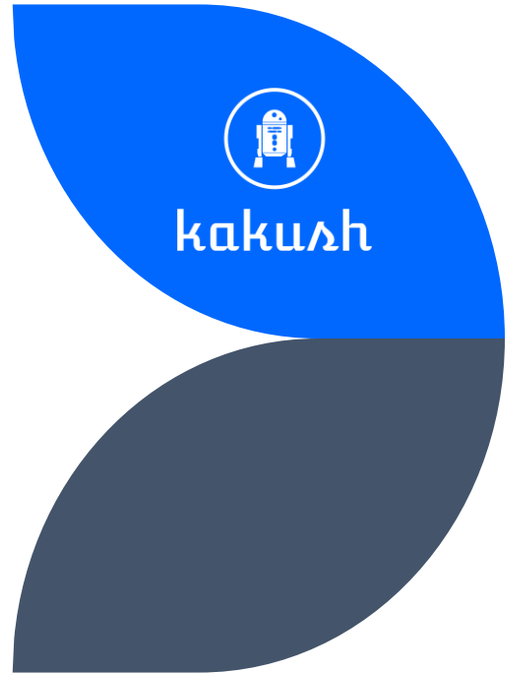


How DAC Works

- **Input:** The digital signal is received from a microcontroller or computer.
- **Conversion:** The digital values are converted to corresponding analog values.
- **Output:** The analog signal is output to drive an analog device.



Arduino Environment



Arduino Hardware

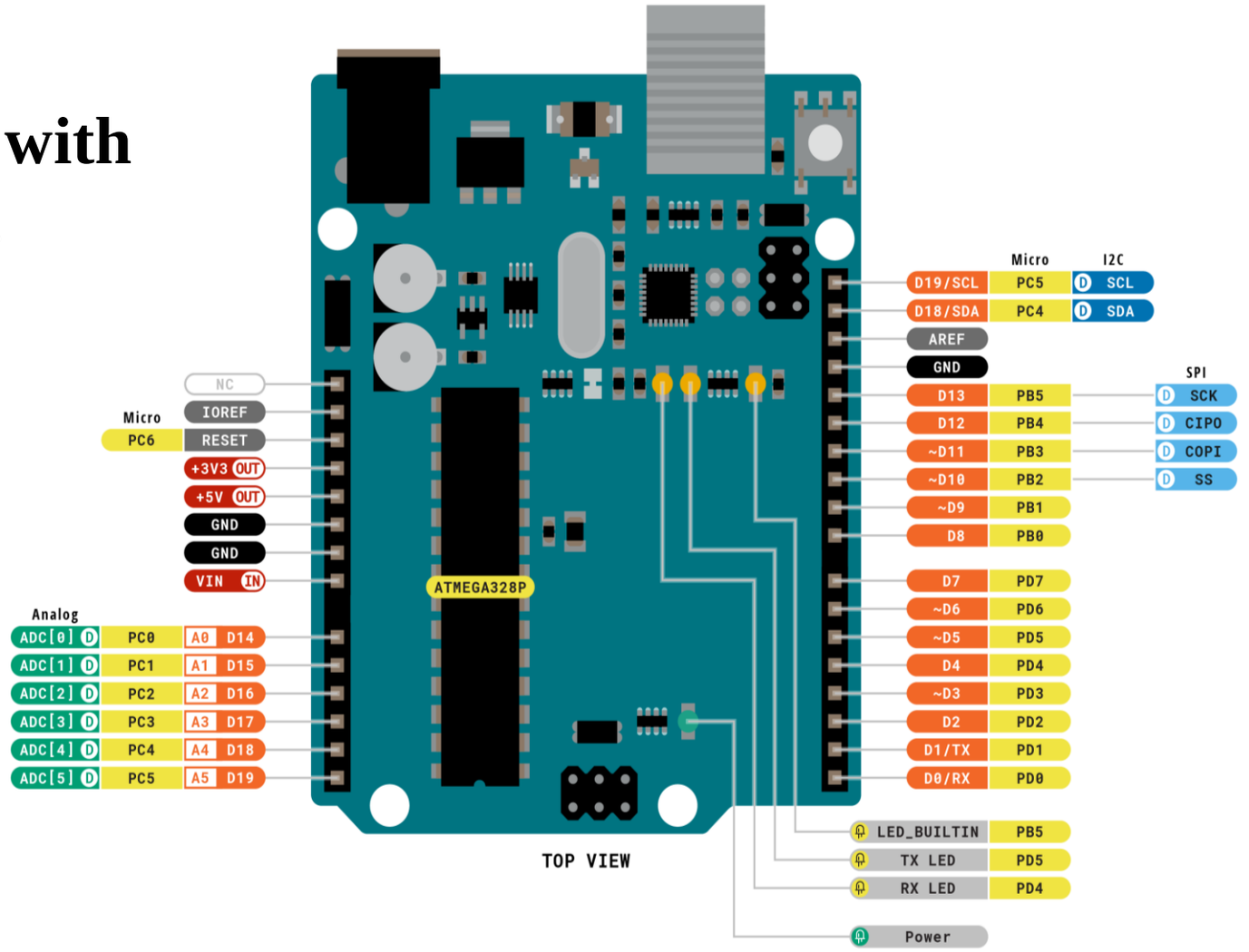
Arduino Boards

- Microcontroller-based boards with various input/output capabilities.
- Examples: Arduino Uno, Arduino Mega, Arduino Nano.

Key Components

- **Microcontroller**: The brain of the board (e.g., ATmega328 on Arduino Uno).
- **Digital Pins**: Used for digital input/output (e.g., LEDs, buttons).
- **Analog Pins**: Used for analog input (e.g., sensors).
- **Power Supply**: Can be powered via USB or external power source.
- **Communication Ports**: Serial, I2C, SPI for interfacing with other devices.

Arduino Uno board with labeled components.



Legend:	Digital	I2C
Power	Analog	SPI
Ground	Main Part	Analog

ARDUINO

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

kakush

Arduino Software (Arduino IDE)

Arduino IDE

- Integrated Development Environment for writing, compiling, and uploading code to Arduino boards.
- Available for Windows, Mac, and Linux.

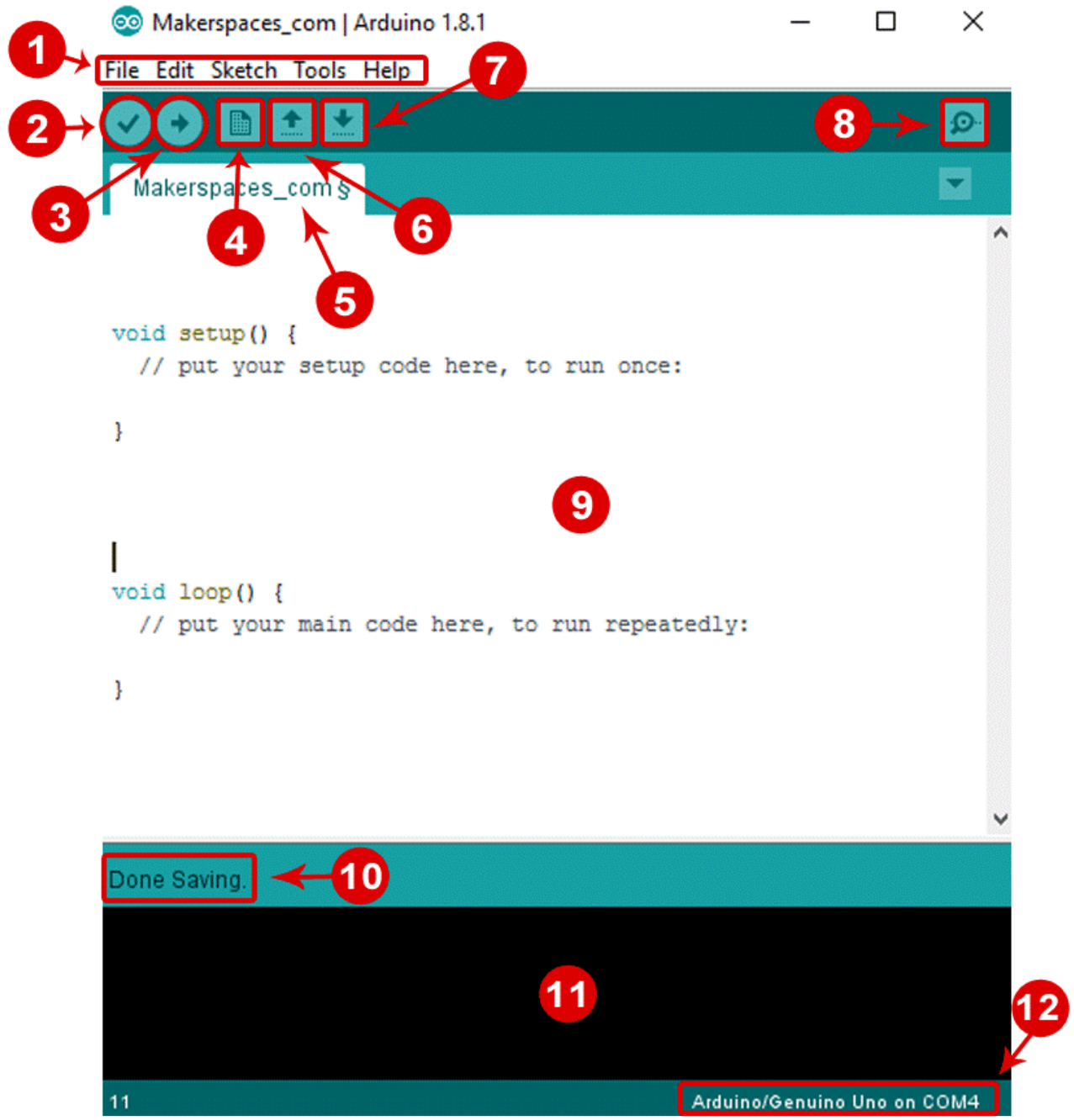
Key Features

- **Code Editor**: Write and edit Arduino sketches (programs).
- **Compiler**: Convert sketches into machine code that the microcontroller can execute.
- **Uploader**: Transfer the compiled code to the Arduino board via USB.
- **Serial Monitor**: Communicate with the board and debug sketches using serial communication.

Programming Language

- Based on C/C++ with simplified syntax and functions.
- Uses libraries to add functionality (e.g., controlling motors, reading sensors).

Arduino IDE simple sketch.



kakush

How to Download the Arduino IDE ?

Visit the Arduino Website:

Open your web browser and go to **arduino.cc**

Navigate to the Software Section:

Hover over "Software" in the main menu.

Click on "Downloads."

Choose Your Operating System:

Select the version for your OS (Windows, Mac, Linux).

Download the Installer:

Click "Just Download" if prompted for a donation.

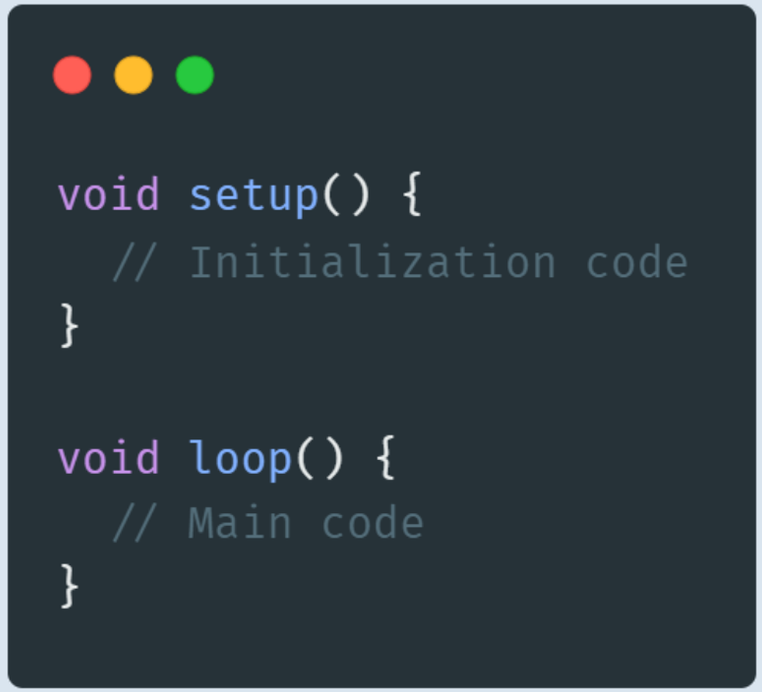
How to Program Arduino and Code Structure

Setup Function:

- **Purpose**: Initializes settings.
- **Runs Once**: This code runs once when the Arduino is powered on or reset.

Loop Function:

- **Purpose**: Contains the main logic of the program.
- **Runs Repeatedly**: This code runs in a loop after the setup function has completed.



```
void setup() {
  // Initialization code
}

void loop() {
  // Main code
}
```

Common Functions Used in Arduino Programming

pinMode():

- **Purpose**: Configures a specified pin to behave either as an input or an output.

digitalWrite():

- **Purpose**: Sets a specified digital pin to HIGH or LOW.

digitalRead():

- **Purpose**: Reads the value from a specified digital pin, either HIGH or LOW.

```
pinMode(pin, mode);

digitalWrite(pin, value);

int value = digitalRead(pin);
```

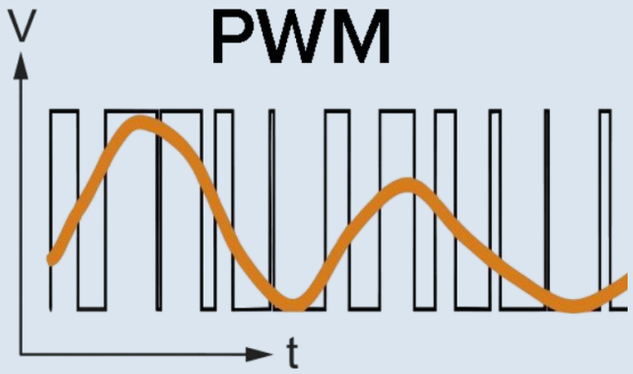
Common Functions Used in Arduino Programming

analogRead():

- **Purpose**: Reads the value from a specified analog pin.

analogWrite():

- **Purpose**: Writes an analog value (PWM wave) to a specified pin.




```
int value = analogRead(pin);

analogWrite(pin, value);
```

Common Functions Used in Arduino Programming

delay():

- **Purpose**: Pauses the program for a specified number of milliseconds.

millis():

- **Purpose**: Returns the number of milliseconds since the Arduino board began running the current program.

```
delay(milliseconds);

unsigned long currentTime = millis();
```

Common Functions Used in Arduino Programming

Serial.begin():

- **Purpose**: Sets the data rate in bits per second (baud) for serial data transmission.

Serial.print():

- **Purpose**: Prints data to the serial port as human-readable ASCII text.

Serial.println():

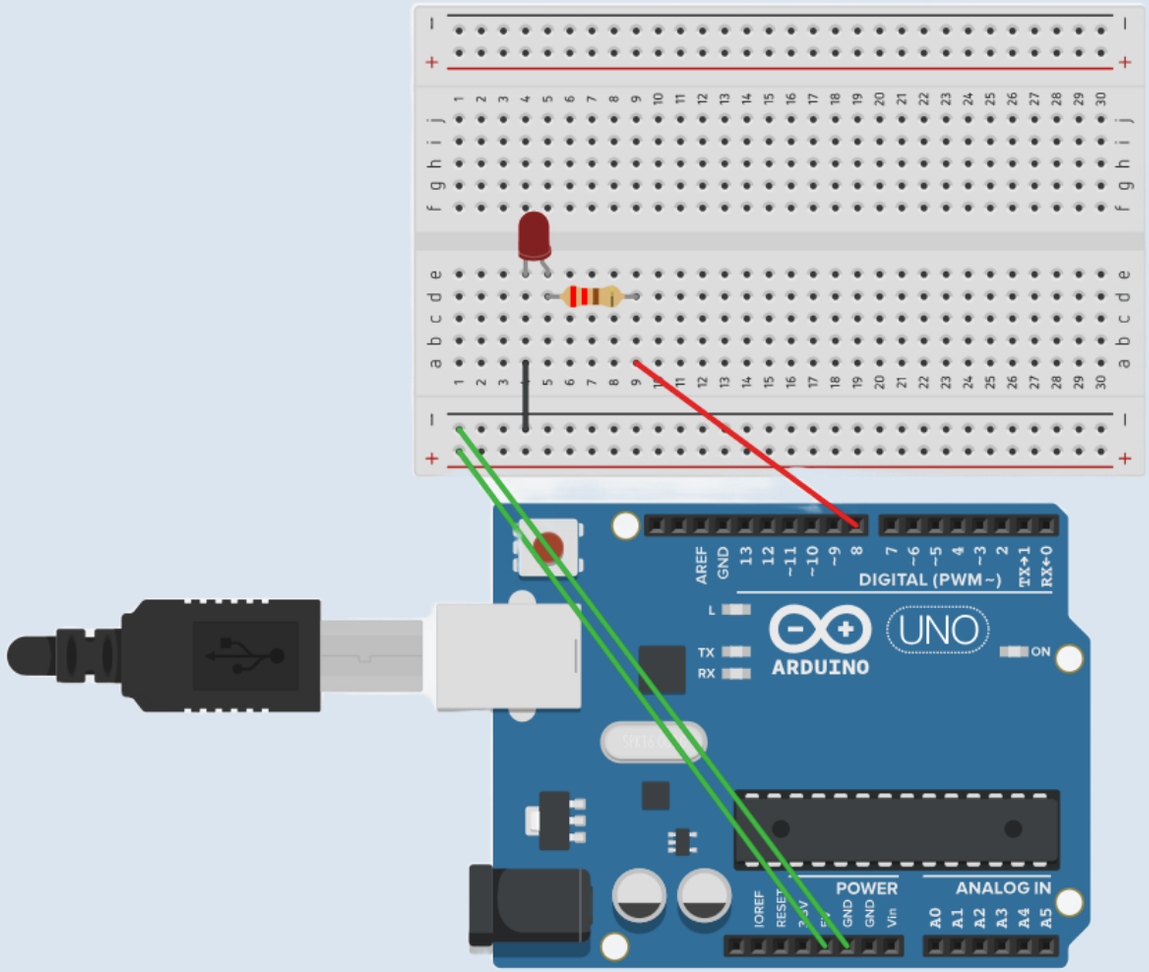
- **Purpose**: Prints data to the serial port as human-readable ASCII text, followed by a carriage return character.

```
Serial.begin(baudRate);

Serial.print(data);

Serial.println(data);
```

Example 1 => Blinking LED

Blinking LED using Delay function:

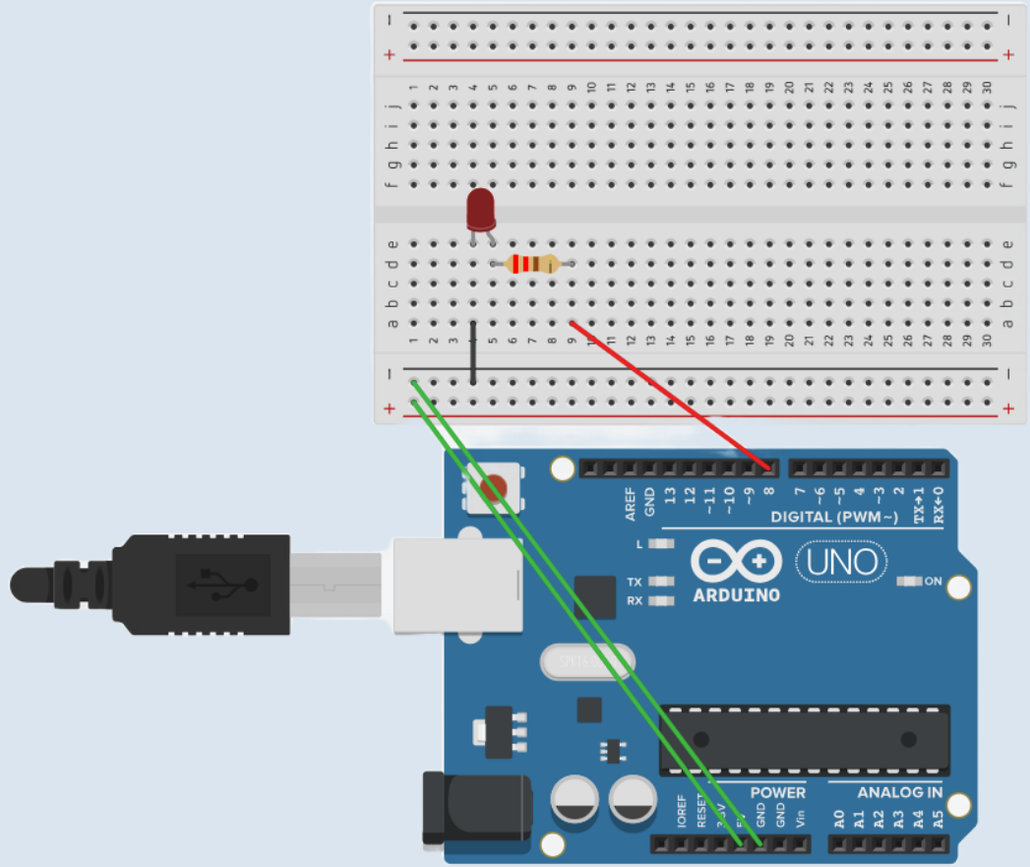
```
int ledPin=8;

void setup(){
    pinMode(ledPin,OUTPUT);
}

void loop(){
    digitalWrite(ledPin,HIGH);
    delay(1000);
    digitalWrite(ledPin,LOW);
    delay(1000);
}
```



Example 2 => Blinking LED

Blinking LED using millis function:

```
int ledPin = 8;
unsigned long previousMillis = 0;
const long interval = 1000;

void setup() {
    pinMode(ledPin, OUTPUT);
}

void loop() {
    unsigned long currentMillis = millis();

    if (currentMillis - previousMillis ⩾ interval) {
        previousMillis = currentMillis;

        if (digitalRead(ledPin) == LOW)
            digitalWrite(ledPin, HIGH);
        else
            digitalWrite(ledPin, LOW);
    }
}
```



Basic Electronic Component

kakush

LED (Light Emitting Diode)

Definition: A semiconductor device that emits light when an electric current passes through it.

Usage: Visual indicators, status lights, displays.

Connection:

- **Anode (+)**: Connected to a positive voltage.
- **Cathode (-)**: Connected to ground, usually through a current-limiting resistor.

Button

Definition: A simple switch used to make or break a connection in an electric circuit.

Usage: User inputs, control signals, event triggers.

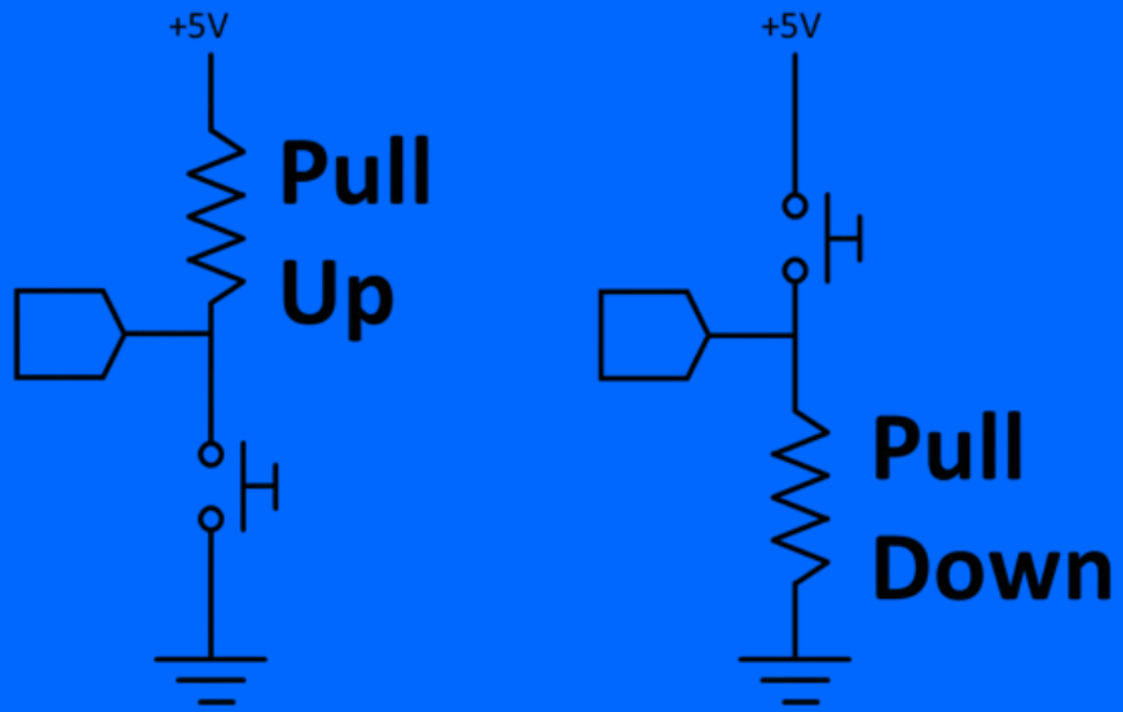
Connection:Typically connected between a digital input pin and ground or VCC.

Pull-Up and Pull-Down Resistors:

Pull-Up: Connects the pin to VCC (high voltage) through a resistor, ensuring a known state when the button is not pressed.

Pull-Down: Connects the pin to ground (low voltage) through a resistor, ensuring a known state when the button is not pressed.

Internal Pull-Up: Use pinMode(pin, INPUT_PULLUP); to enable the internal pull-up resistor.



Potentiometer

Definition: A three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider.

Usage: Variable resistors, volume controls, sensor calibration.

- **Connection**:**Wiper**: Connected to an analog input pin.
- **Other Terminals**: Connected to VCC and ground

LED (Light Emitting Diode)

LED (Light Emitting Diode)

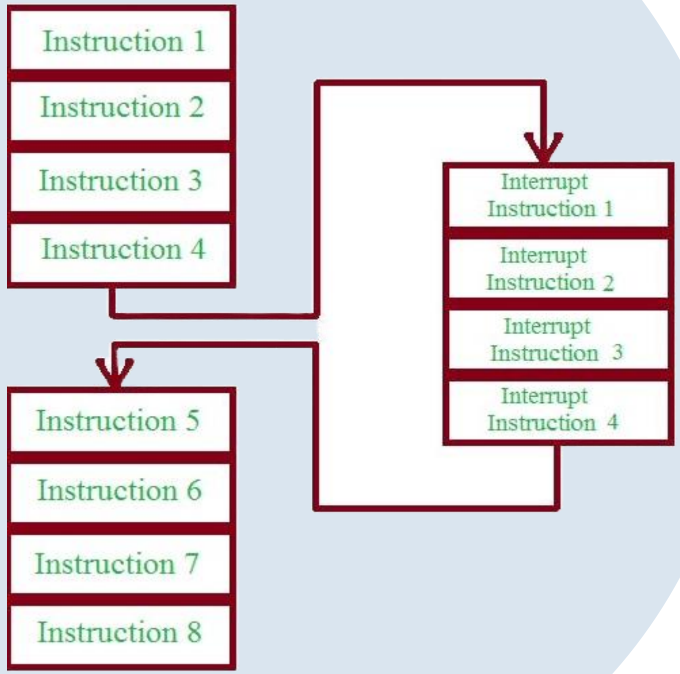
Definition: A semiconductor device that emits light when an electric current passes through it.

Usage: Visual indicators, status lights, displays.

Connection:

- **Anode (+)**: Connected to a positive voltage.
- **Cathode (-)**: Connected to ground, usually through a current-limiting resistor.

Interrupts



kakush

What are Interrupts?

kakush

- **Definition**: Interrupts are signals that temporarily halt the main program execution to immediately run a special function called an Interrupt Service Routine (ISR).
- **Purpose**: To handle time-critical tasks, respond to external events, or process real-time signals without delay.
- **Benefits**: Allows for efficient handling of tasks without continuously checking for events within the main loop.

Using Interrupts in Arduino

Attaching an Interrupt:

- Use the attachInterrupt() function to specify which pin and condition will trigger the interrupt, and which ISR to call.

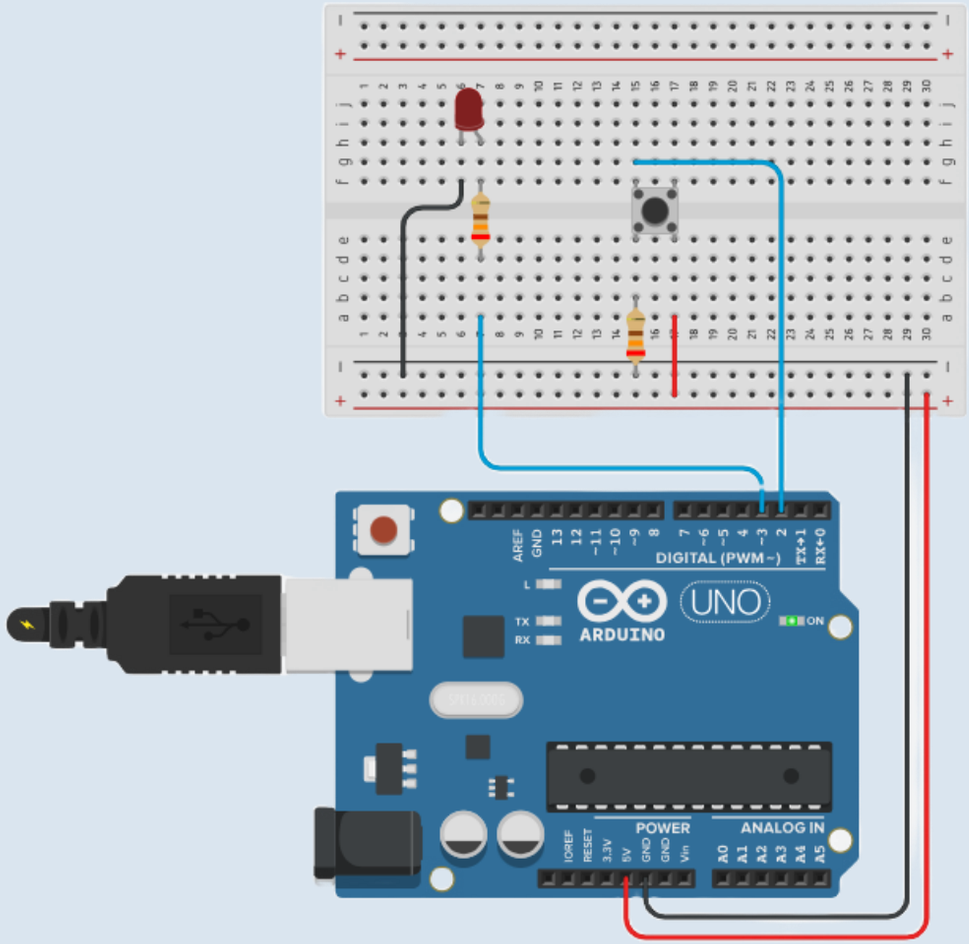
```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);
```

- Parameters:
  - pin: The pin number where the interrupt is attached.
  - ISR: The name of the function to call when the interrupt occurs.
  - mode: The condition that triggers the interrupt (LOW, CHANGE, RISING, FALLING).

Example 3 => Toggle LED

Toggle LED without interrupts:

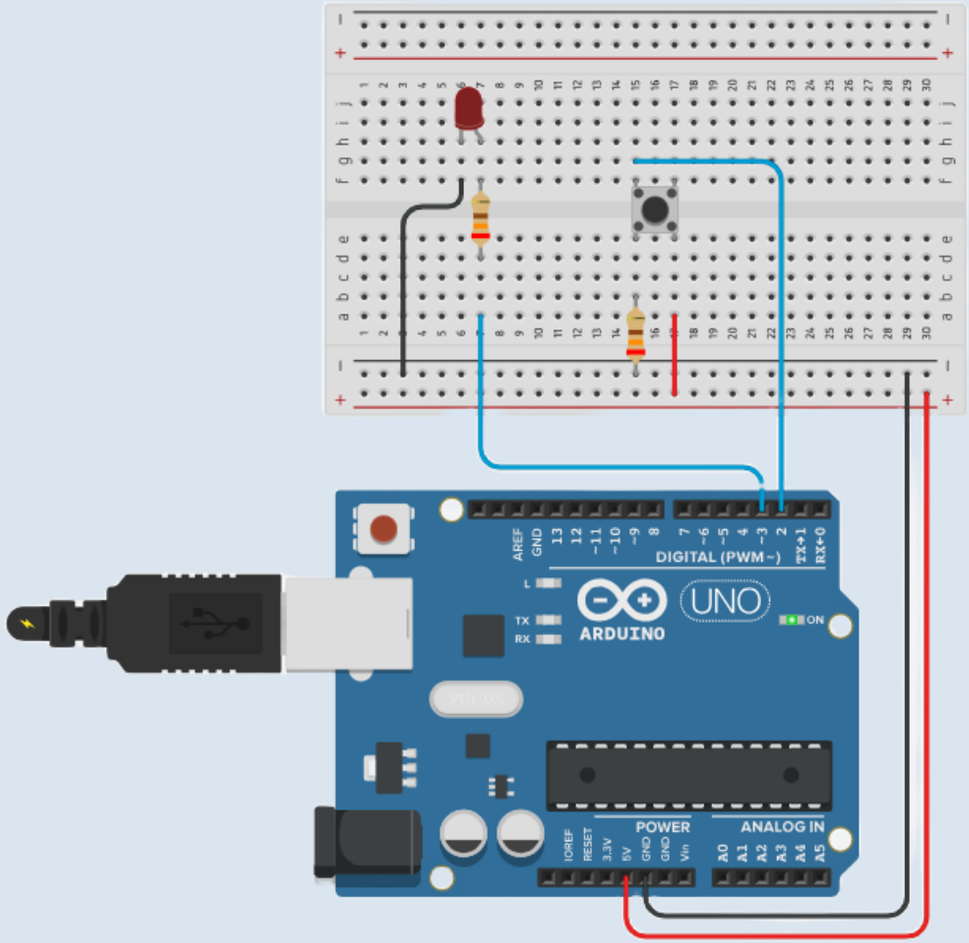
```
const int Switch = 2, LED = 3;
int state = 0, LEDstate=0;

void setup(){
  pinMode(Switch, INPUT);
  pinMode(LED, OUTPUT);
}

void loop(){
  if (state == 0 && digitalRead(Switch) == HIGH) {
    state = 1;
    LEDstate=!LEDstate;
  }
  if (state == 1 && digitalRead(Switch) == LOW) {
    state = 0;
  }
  digitalWrite(LED, LEDstate);
}
```



Example 4 => Toggle LED

Toggle LED with interrupts:

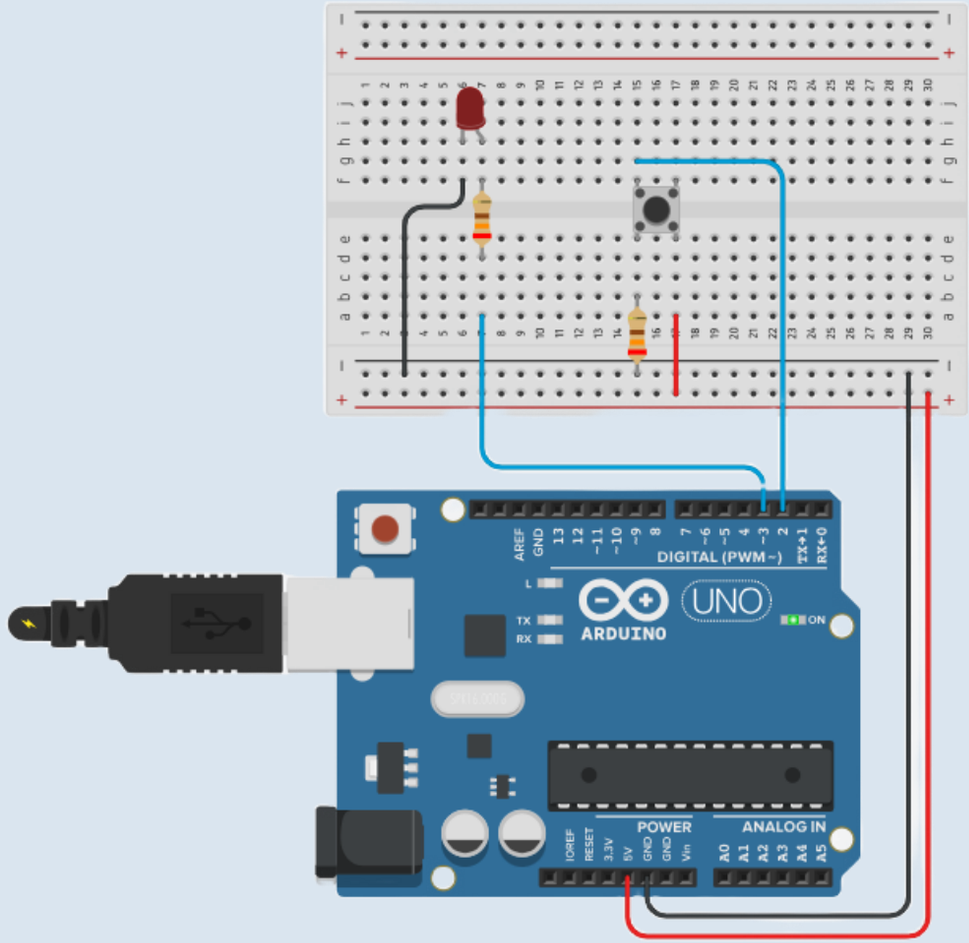
```
const int buttonPin = 2;
const int ledPin = 3;
volatile int LEDstate = LOW;

void setup() {
    pinMode(buttonPin, INPUT);
    pinMode(ledPin, OUTPUT);
    attachInterrupt(digitalPinToInterrupt(buttonPin),
                    toggleLED, RISING);
}

void loop() {
}

void toggleLED() {
    LEDstate = !LEDstate;
    digitalWrite(ledPin, LEDstate);
}
```



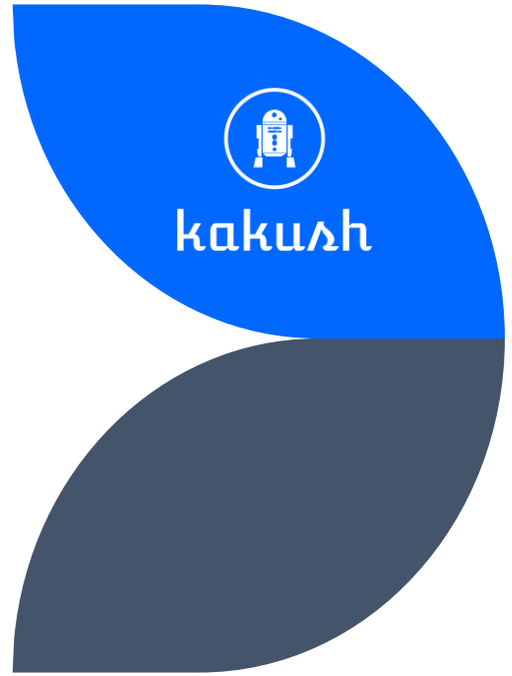
Example 5 => Toggle LED

Toggle LED with interrupts and avoid bouncing effect:

```
const int buttonPin = 2;
const int ledPin = 3;
volatile int LEDstate = LOW;
volatile unsigned long lastDebounceTime = 0;
const unsigned long debounceDelay = 50;

void setup() {
    pinMode(buttonPin, INPUT);
    pinMode(ledPin, OUTPUT);
    attachInterrupt(digitalPinToInterrupt(buttonPin),
                    toggleLED, RISING);
}

void loop() {
}

void toggleLED() {
    unsigned long currentTime = millis();
    if (currentTime - lastDebounceTime
        > debounceDelay) {
        LEDstate = !LEDstate;
        lastDebounceTime = currentTime;
        digitalWrite(ledPin, LEDstate);
    }
}
```



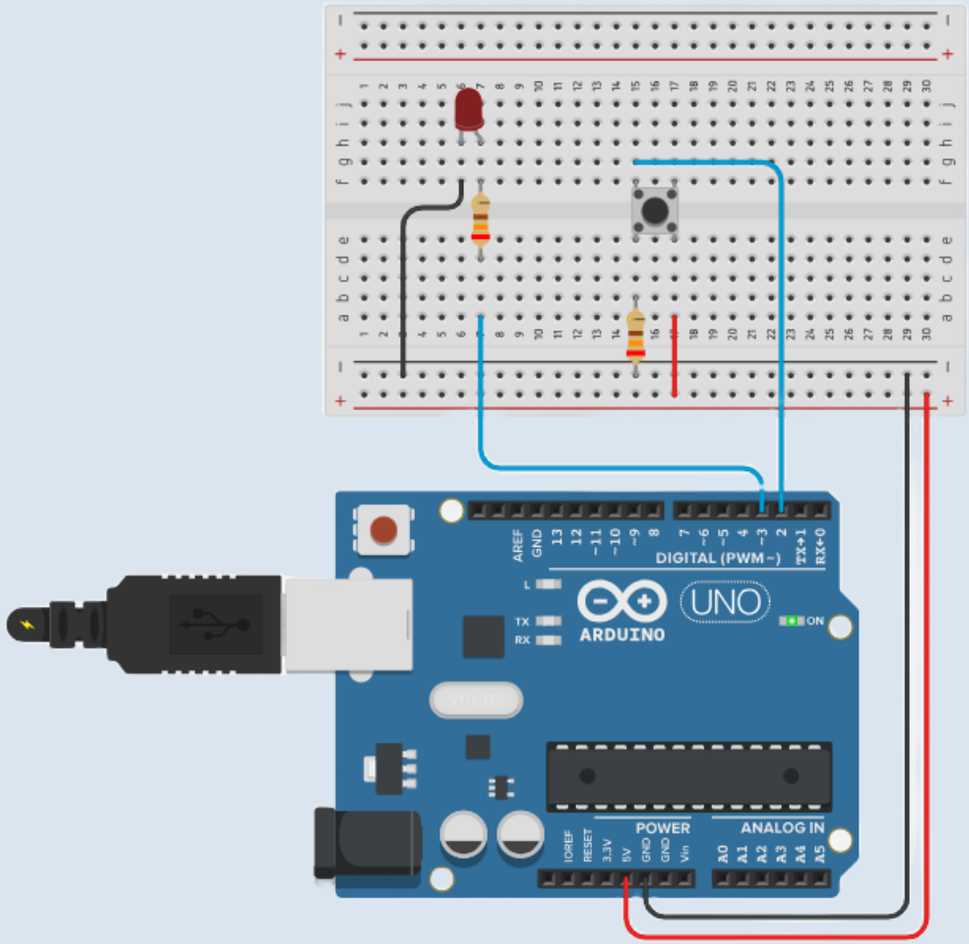
Examples for training



Example 6 => Toggle LED

Toggle LED without interrupts:

```arduino
const int Switch = 2, LED = 3;
int state = 0, LEDstate=0;

void setup(){
  pinMode(Switch, INPUT);
  pinMode(LED, OUTPUT);
}

void loop(){
  if (state == 0 && digitalRead(Switch) == HIGH) {
    state = 1;
    LEDstate=!LEDstate;
  }
  if (state == 1 && digitalRead(Switch) == LOW) {
    state = 0;
  }
  digitalWrite(LED, LEDstate);
}
```



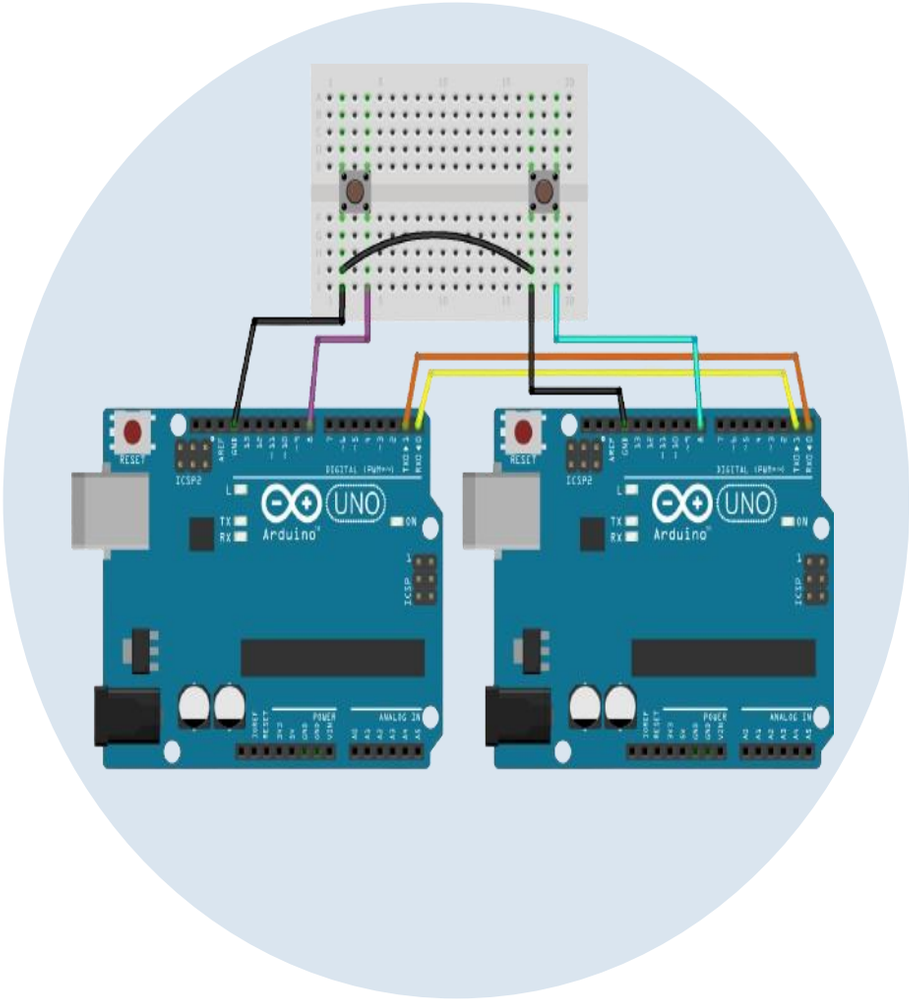
LCD liquid crystal display

kakush

Sensors

Actuators

Communication



kakuah

Contact us

kakush