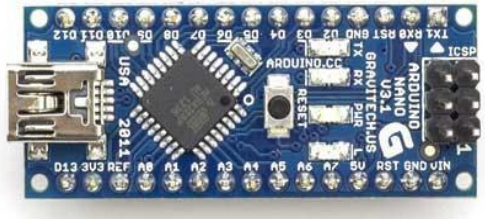
What is Arduino

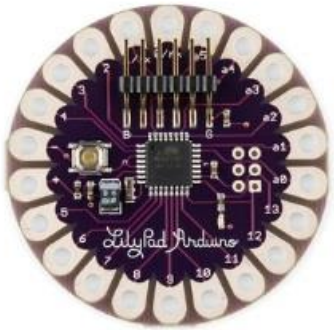
- A microcontroller board, contains on-board power supply, USB port to communicate with PC, a microcontroller chip.

- It simplifies the process of creating any control system by providing the standard board that can be programmed and connected to the system without the need to any sophisticated PCB design and implementation.

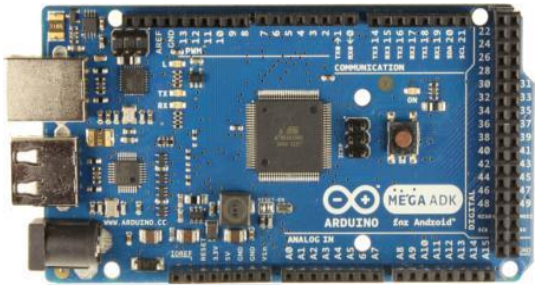
Types of Arduino Boards



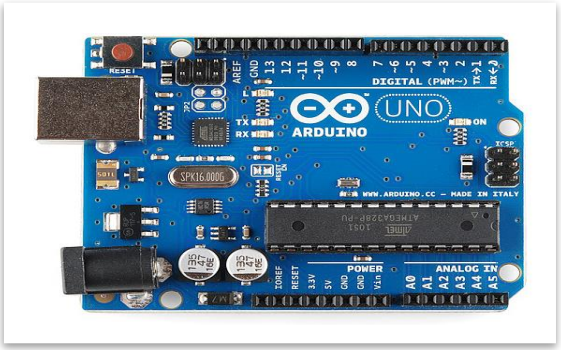
Arduino Nano



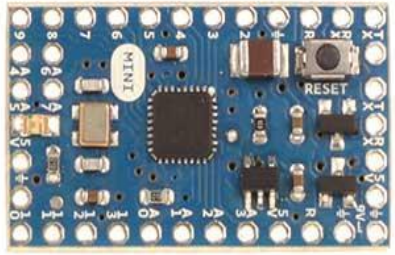
Arduino LilyPad



Arduino Mega



Arduino Uno



Arduino Mini

Arduino Leonardo

Types of Arduino board

2013: 700.000 official boards were sold.

2016: 17 versions of the Arduino board have been commercially produced.

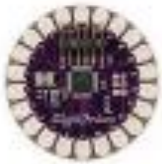
Arduino Uno

Arduino Leonardo

Arduino Mega ADK

Arduino Ethernet

LilyPad Arduino SimpleSnap



LilyPad Arduino

Arduino Due

Arduino Yún

Arduino Mega 2560

Arduino Mini

Arduino Nano

Arduino Pro Mini

Arduino Tre

Arduino Micro

LilyPad Arduino USB

LilyPad Arduino

Arduino Pro

Arduino Fio

Types of Arduino board

UNO

- Current official reference of Arduino Boards
- Most used and documented board

Mega

- Designed for more complex projects
- 54 digital I/O pins, 16 analog inputs
- ATmega2560

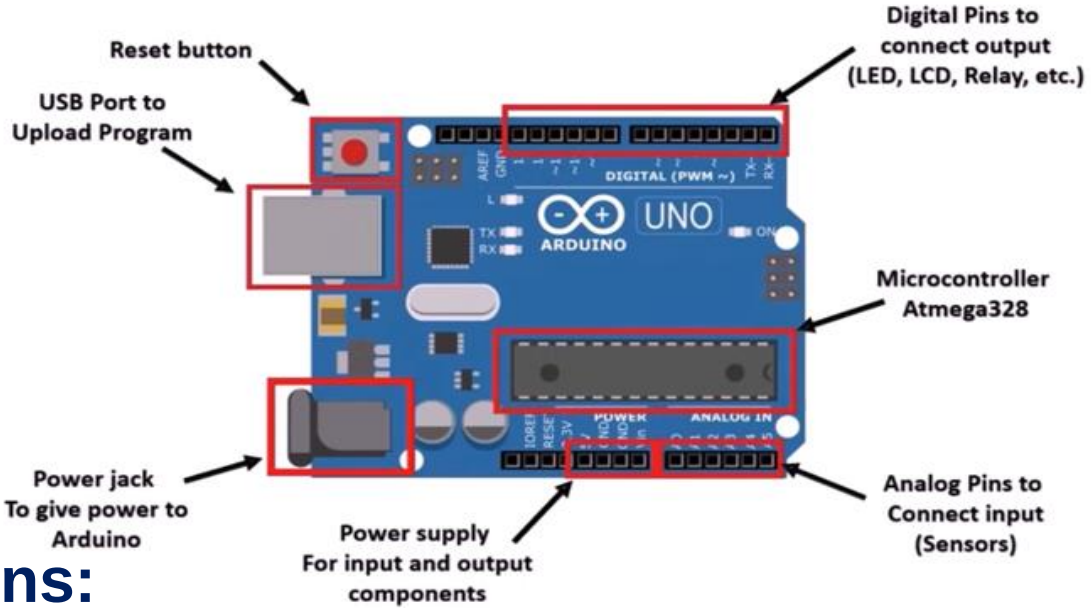
LilyPad

- Designed for e-textiles and wearables projects
- Can be sewn to fabric and to power supplies

Nano

- Compact board similar to the UNO

Arduino UNO



Digital pins:

14 digital IO pins

6 are PWM pins (3, 5, 6, 9, 10, and 11).

Analog pins:

6 analog pins(A0, A1, A2, A3, A4, and A5)

Takes analog values as an input

Board Breakdown

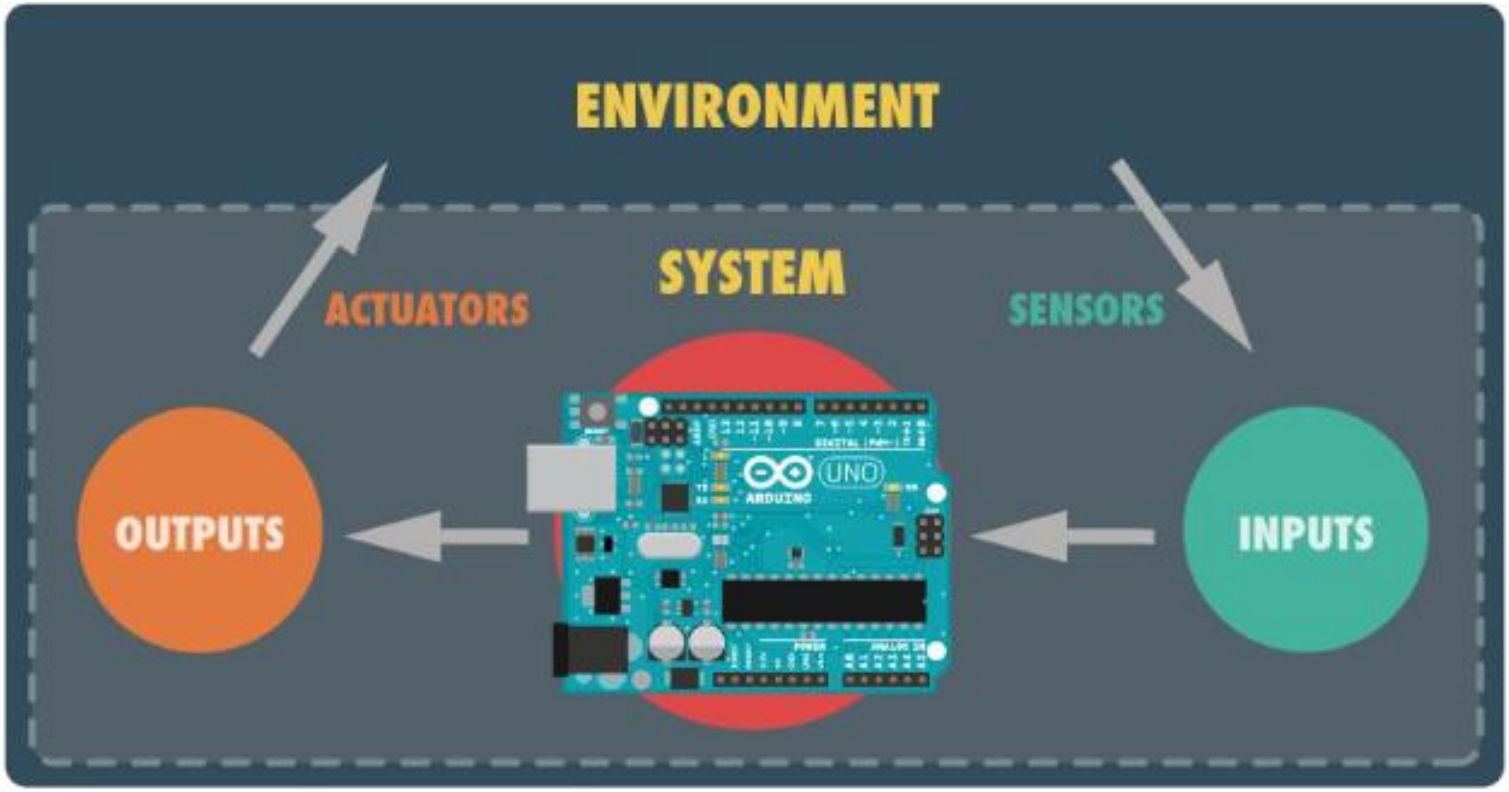
Here are the components that make up an Arduino board and what each of their functions are.

1. Reset Button – This will restart any code that is loaded to the Arduino board
2. AREF – Stands for “Analog Reference” and is used to set an external reference voltage
3. Ground Pin – There are a few ground pins on the Arduino and they all work the same
4. Digital Input/Output – Pins 0-13 can be used for digital input or output
5. PWM – The pins marked with the (~) symbol can simulate analog output
6. USB Connection – Used for powering up your Arduino and uploading sketches

Board Breakdown

7. TX/RX – Transmit and receive data indication LEDs
8. ATmega Microcontroller – This is the brains and is where the programs are stored
9. Power LED Indicator – This LED lights up anytime the board is plugged in a power source
10. Voltage Regulator – This controls the amount of voltage going into the Arduino board
11. DC Power Barrel Jack – This is used for powering your Arduino with a power supply
12. 3.3V Pin – This pin supplies 3.3 volts of power to your projects
13. 5V Pin – This pin supplies 5 volts of power to your projects
14. Ground Pins – There are a few ground pins on the Arduino and they all work the same
15. Analog Pins – These pins can read the signal from an analog sensor and convert it to digital

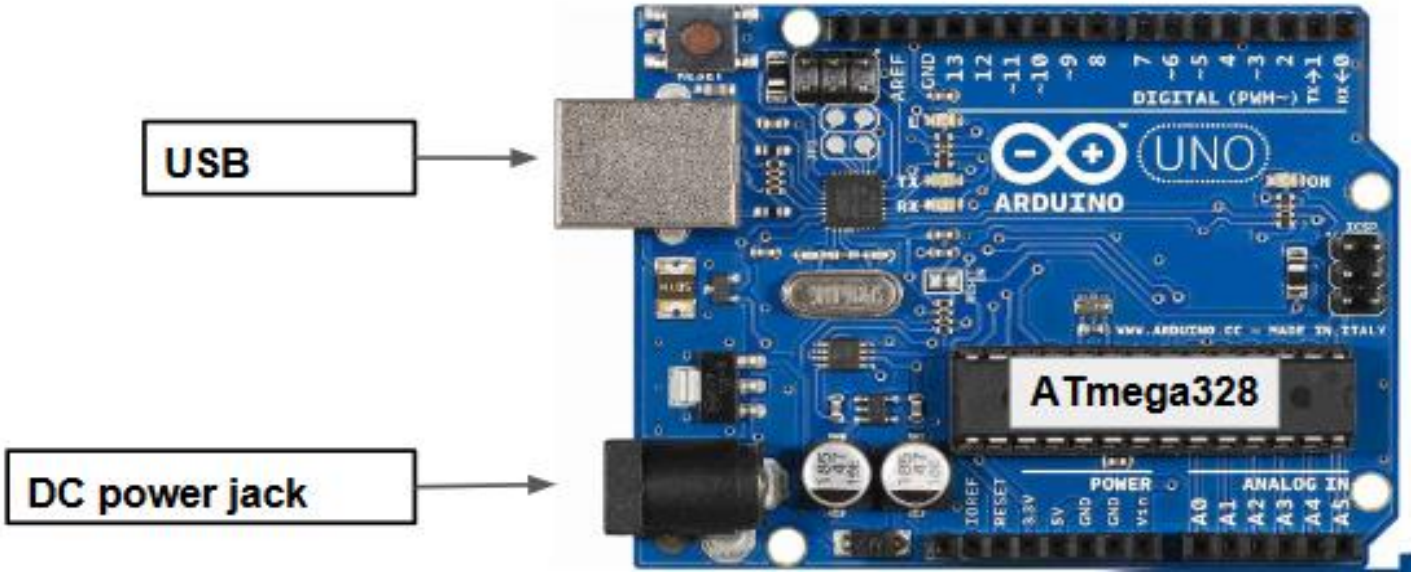
Arduino as Physical Computing Platform



Physical Computing involves the design of interactive objects that can communicate with humans using sensors and actuators controlled by a behaviour implemented as software running inside a microcontroller.

General Architecture

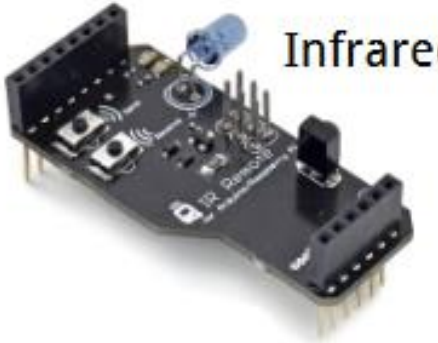
- **Main components**
  - AVR Microcontroller
  - Analog and digital I/O pins
  - Flash memory
    - Integrated in the microcontroller
  - USB port for serial communication



Arduino Communication Devices

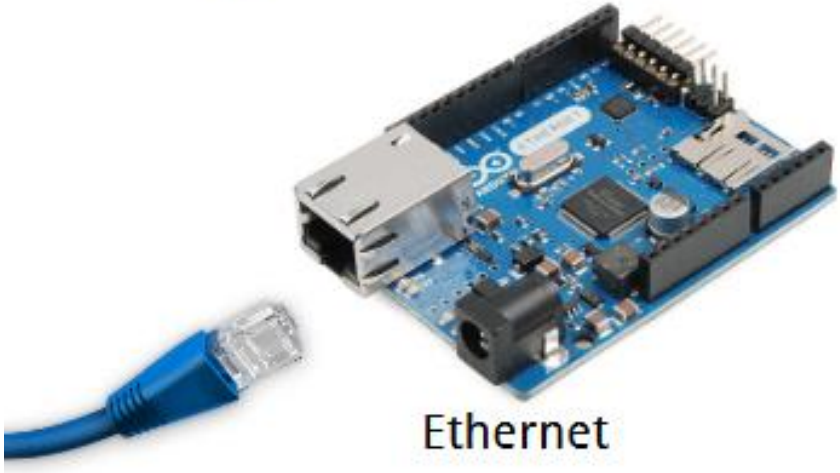
Wired and Wireless comm. devices

WiFi



Infrared

Bluetooth

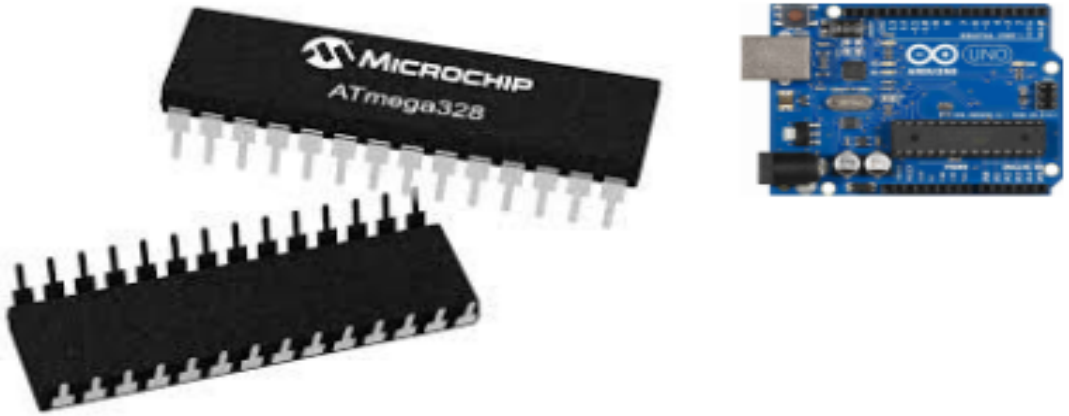


Ethernet

Zig-Bee

Characteristics

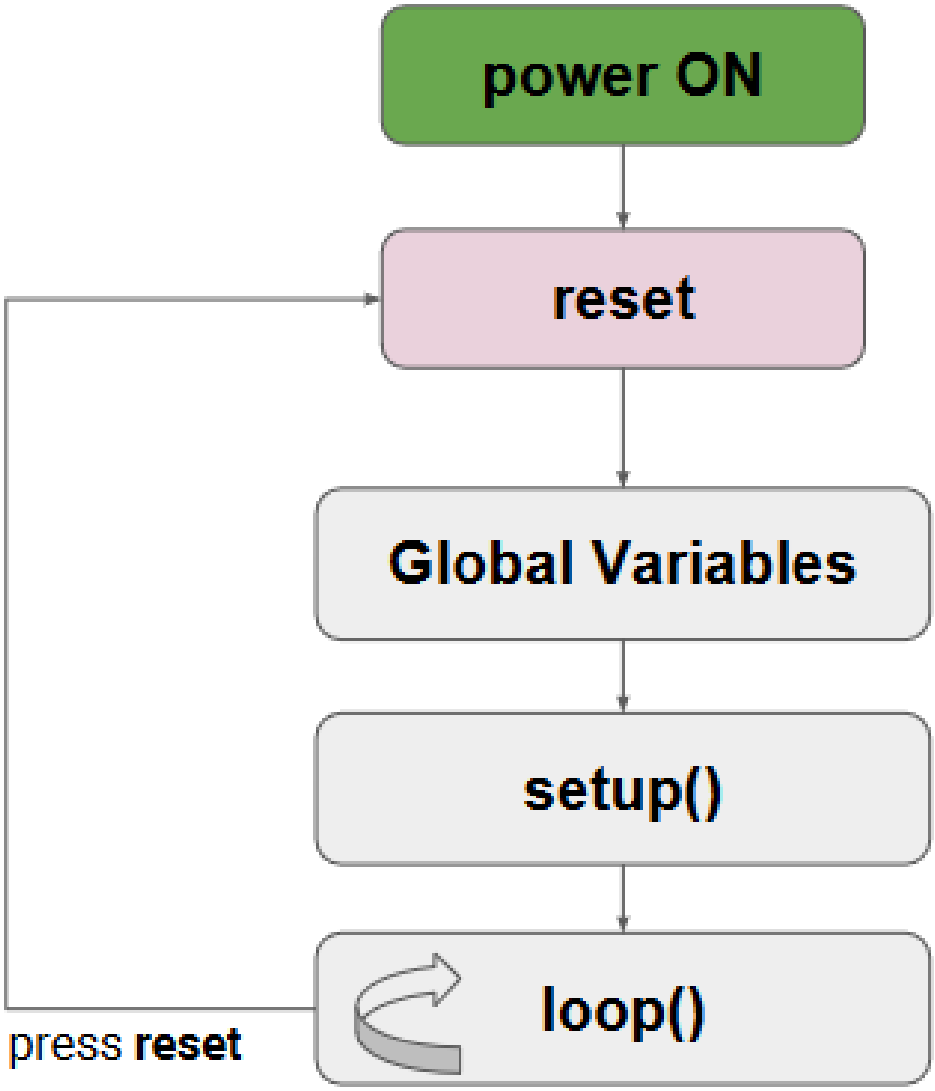
Arduino UNO (ATmega 328)



Technical characteristics

- Clock speed: 16 MHz (*Intel 286*: 12.5 MHz) - 8-bit
- Flash program memory: 32 KBytes (0.5 used by bootloader)
- SRAM: 2 KBytes
- Input / Output
  - 14 digital input/output pins
  - 6 analog input pins
  - 6 analog output pins (PWM)

Programming in Arduino

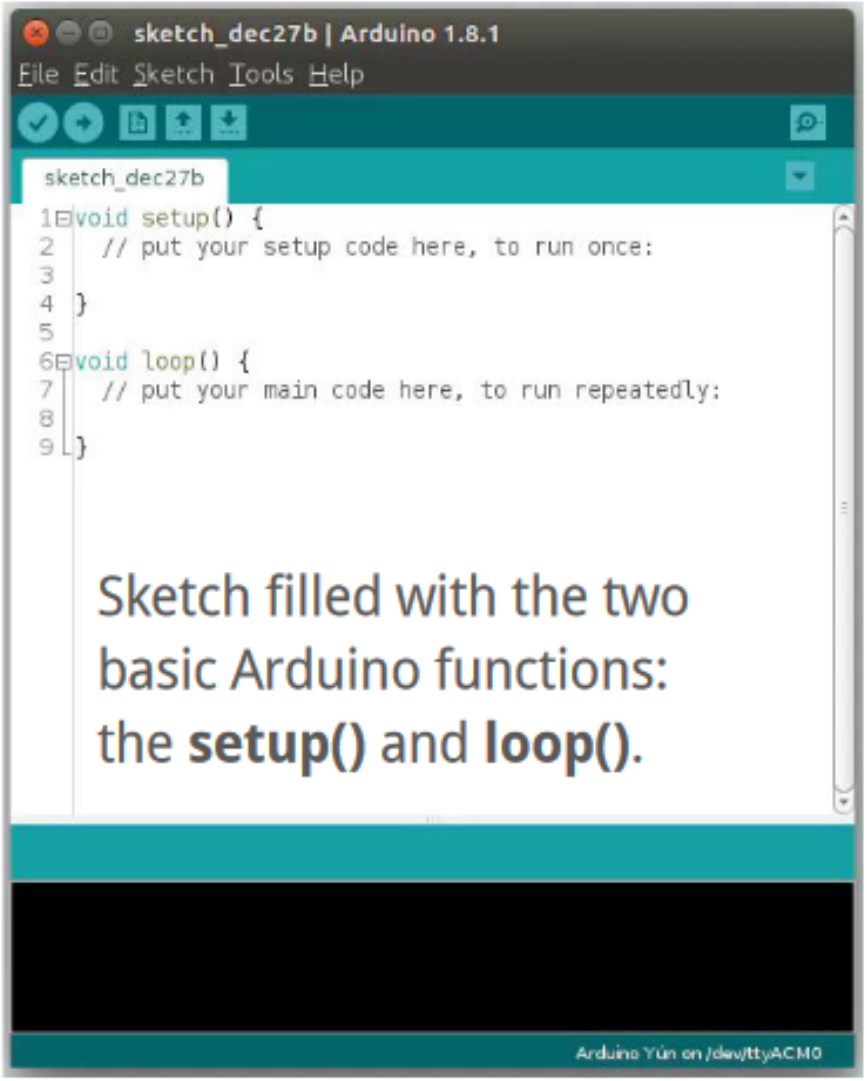


- Declare variables
- Initialize variables

- Run once at beginning
- Set pins

- Run the main program
- Repeatedly, after the setup

Arduino IDE Software

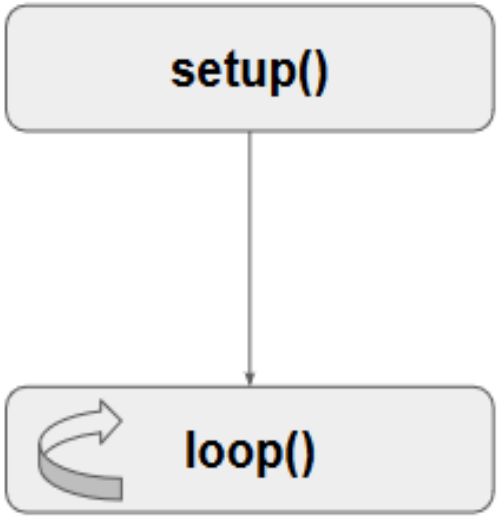


- **Text editor** containing syntax highlighting and automatic indentation
- **Toolbar**
- **Text Console**
- **Compiler**
- **Serial monitor** to debug
  - Allows you to read the data that Arduino communicates through the COM serial port

Writing Based Language

- Open Source computer programming language
- Derived by C/C++ language
  - With some slight simplifications and modifications
  - Includes classical libraries and functions
    - Data types (Integer, float, long, character, ...)
    - Operators (Mathematical, logical, comparison, ...)
    - Control statements (If, switch/case, while, for, ...)
- Offer to the programmer simple access to I/O devices
- Wiring programs are called ***sketch***

Some Functions



- `pinMode(pin, Input|Output)` set pin ledPin as an input or output
- `Serial.begin(9600)` talk to the computer at 9600 baud rate
  - Some values: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200

- `Serial.print(" ... ")` write text on Serial Monitor
- `digitalWrite(pin, HIGH|LOW)` set a digital pin high/low
- `digitalRead(pin)` read a digital pin's state
- `analogRead(pin)` read an a analog pin
- `analogWrite(pin, intValue)` write an "analog" PWM value
- `delay(milliseconds)` wait an amount of time

Bare Minimum Code

- The bare minimum of code needed to start an Arduino sketch.

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

}

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

}
```

Bare Minimum Code

- 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.

- After creating a setup() function, **the loop()** function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond as it runs. Code in the loop() section of your sketch is used to actively control the Arduino board.

Variables

Variables are essential for storing data that your programs can manipulate.
Key types:

1. Integers:

- int: Stores whole numbers (e.g., -32,768 to 32,767).
- long: For larger integers (e.g., -2,147,483,648 to 2,147,483,647).
- short: A smaller integer type (e.g., -32,768 to 32,767).
- byte: Stores an 8-bit unsigned integer (0 to 255).

2. Floating-Point

- float: For decimal numbers with single precision (approximately 6-7 decimal places).
- double: For decimal numbers with double precision (approximately 15 decimal places), but on some Arduino boards, it may be the same as `float`.

3. Character:

- char : Stores a single character (e.g., 'A', '1', or '#'). It uses 1 byte of memory.

Variables

4. Boolean:

- bool: Represents a truth value, either `true` or `false`.

5. String

- String: A class that allows manipulation of text strings. It dynamically allocates memory.

6. Arrays

- Arrays allow you to store multiple values of the same type in a single variable.
  - Example: `int numbers[5];` creates an array of five integers.

7. User-Defined Types

- You can define your own data types using `struct` or `class` for more complex data structures.

Variables Examples


```
int sensorValue;      // Integer variable
float temperature;    // Float variable
char letter = 'A';    // Character variable
bool isActive = true; // Boolean variable
String message = "Hello, World!"; // String variable
```

Serial Monitor

- The Serial Monitor in Arduino is a tool that allows you to communicate with your Arduino board from your computer. It is useful for debugging, monitoring data, and sending commands to your Arduino.

- In the Arduino IDE, Serial Monitor displays data sent from the Arduino board to your computer via the serial port. We can also send data from your computer to the Arduino.

- The basic function includes setting the Baud Rate. Common rates include 9600, 115200, etc.

- Sending Data: You can type text into the input field at the top of the Serial Monitor and send it to the Arduino by clicking the “Send” button or pressing Enter.

Common Serial commands

1. Serial.begin(baudRate): Initializes serial communication at the specified baud rate.
2. Serial.print(data): Sends data to the Serial Monitor without a newline.
3. Serial.println(data): Sends data followed by a newline character.
4. Serial.read(): Reads incoming data from the serial buffer.
5. Serial.available(): Checks how many bytes are available to read from the serial buffer.

```
void setup() {
  Serial.begin(9600); // Start serial communication at 9600 baud
}

void loop() {
  int sensorValue = analogRead(A0); // Read from an analog sensor
  Serial.print("Sensor Value: "); // Print a label
  Serial.println(sensorValue); // Print the sensor value
  delay(1000); // Wait for a second
}
```

Types of Sensors

Arduino supports various sensors that allow you to gather data from the environment and interact with physical systems. Here are some common types of sensors used with Arduino:

1) Temperature Sensors:

- DHT11/DHT22: Measures temperature and humidity. Commonly used in weather stations.
- LM35: Analog temperature sensor that provides a voltage output proportional to the temperature.

2) Light Sensors:

- Photoresistor (LDR): Changes resistance based on light levels. Used in projects like automatic lighting.
- TSL2561: Digital light sensor that provides precise readings of light intensity.

3) Distance Sensors:

- Ultrasonic Sensor (HC-SR04): Measures distance using ultrasonic waves. Useful in obstacle detection and range finding.
- Infrared (IR) Sensors: Measure distance using infrared light. Common in robotics.

Types of Sensors

4) Motion Sensors:

- PIR Sensor: Detects motion based on changes in infrared radiation. Used in security systems.
- Accelerometer (e.g., ADXL345): Measures acceleration and tilt, useful in mobile applications and robotics.

5) Pressure Sensors:

- BMP180/BMP280: Measures atmospheric pressure and temperature. Useful in altitude detection and weather applications.

6) Gas Sensors:

- MQ Series (e.g., MQ-2, MQ-7): Detects various gases (smoke, CO, etc.). Useful in air quality monitoring.
- CCS811: Measures CO2 and total volatile organic compounds (TVOCs).

7) Humidity Sensors:

- DHT11/DHT22: Besides temperature, they also measure humidity.

8) Soil Moisture Sensors:

- Measures the moisture level in soil, commonly used in automated irrigation systems.

Motors

There are three different type of motors –

- DC motor
- Servo motor
- Stepper motor

1) A **DC motor** (Direct Current motor) is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.

Motors

- Speed direction control:

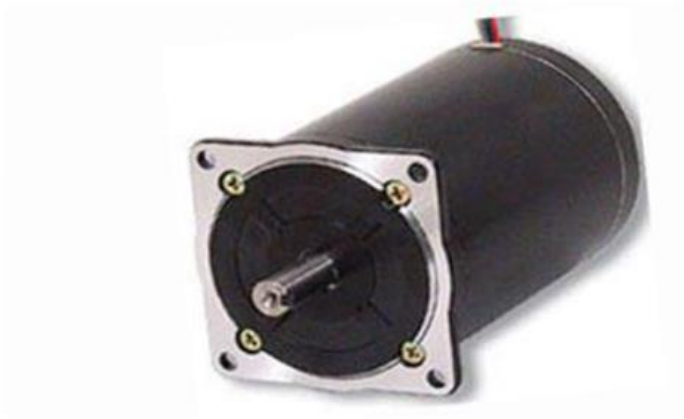
To control the direction of the spin of DC motor, without interchanging the leads, you can use a circuit called an H-Bridge. An H-bridge is an electronic circuit that can drive the motor in both directions. H-bridges are used in many different applications. One of the most common application is to control motors in robots. It is called an H-bridge because it uses four transistors connected in such a way that the schematic diagram looks like an "H." Ex: L298 H-Bridge IC. The L298 can control the speed and direction of DC motors and stepper motors, and can control two motors simultaneously. Its current rating is 2A for each motor. At these currents, however, you will need to use heat sinks.

Motors

2) A **Servo Motor** is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. If the coded signal changes, the angular position of the shaft changes. In practice, servos are used in radio-controlled airplanes to position control surfaces like the elevators and rudders. They are also used in radio-controlled cars, puppets, and robots.

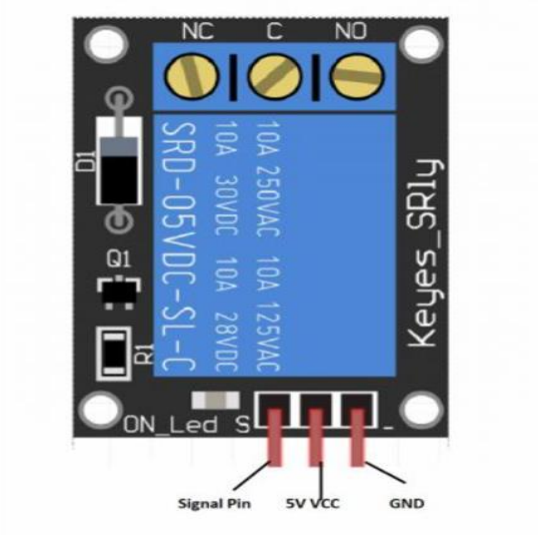
Motors

3) A **Stepper Motor** or a step motor is a brushless, synchronous motor, which divides a full rotation into a number of steps. Unlike a brushless DC motor, which rotates continuously when a fixed DC voltage is applied to it, a step motor rotates in discrete step angles. The Stepper Motors therefore are manufactured with steps per revolution of 12, 24, 72, 144, 180, and 200, resulting in stepping angles of 30, 15, 5, 2.5, 2, and 1.8 degrees per step. The stepper motor can be controlled with or without feedback.



Relays

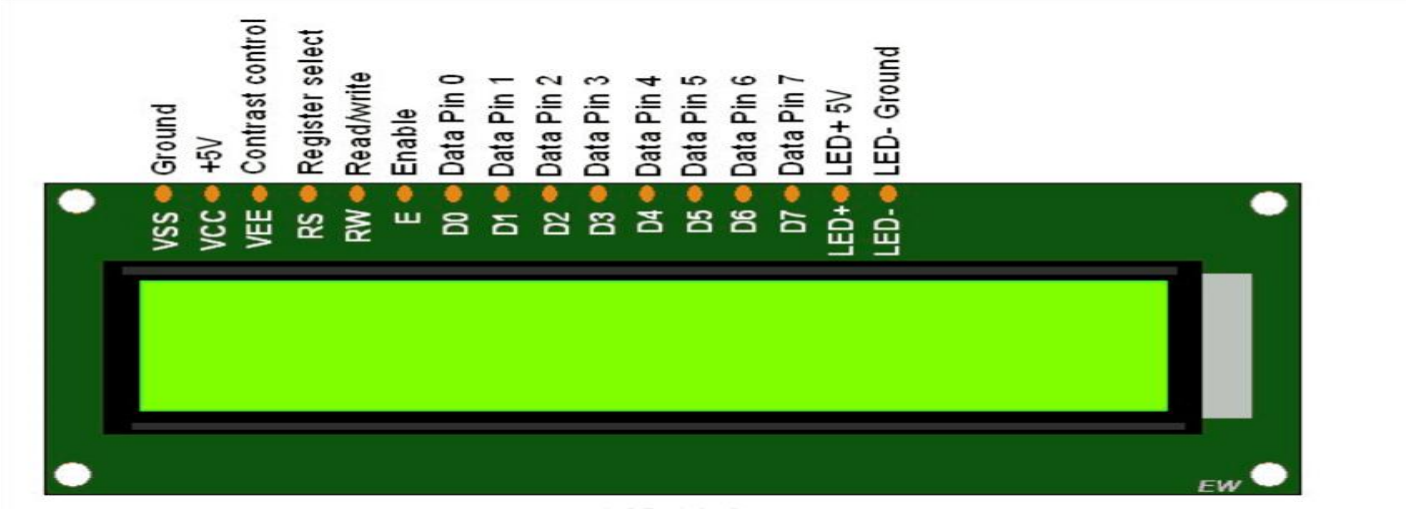
- Arduino Uno work on 5V and the maximum current a digital pin can drive is less than 40mA. So technically we cannot drive higher power devices like home appliances directly with arduino. That is the need of electromechanical RELAY.
- A relay is generally an electrically operated switch. The principle used by the relays is an electromagnet to mechanically operate the switch. So basically it operates a switch using an electromagnet which needs only less power like 5V, 12V or 24V. Different kinds of relays are available in the market like SPDT, DPDT, SPST, 5V, 12V, 24V and with various high current/voltage driving capacity.



Relay Module

Displays

- LCDs (Liquid Crystal Displays) are used in embedded system applications for displaying various parameters and status of the system.
- LCD 16x2 is a 16-pin device that has 2 rows that can accommodate 16 characters each.
- LCD 16x2 can be used in 4-bit mode or 8-bit mode.
- It is also possible to create custom characters.
- It has 8 data lines and 3 control lines that can be used for control purposes.

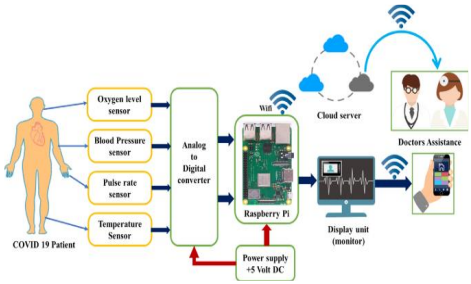


CHITKARA UNIVERSITY

Arduino Case Studies

Case Studies and Applications

- Example 1: Smart Cities
  - Smart Traffic Management System
  - Smart Waste Management System
- Example 2: Healthcare
  - Heart rate monitoring system
  - Glucose Monitoring system
- Example 3: Smart Home Automation
  - Smart Lighting System
  - Security system
- Example 4: Wearable Devices
  - Posture Monitoring Device
  - Smart Fitness Tracker



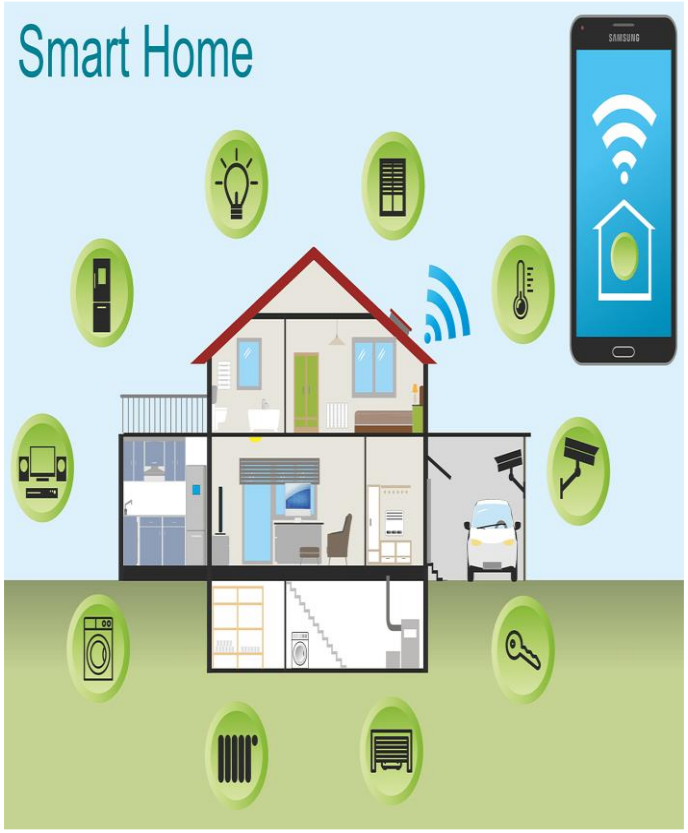
Smart Home Automation Overview

What is Smart Home Automation?

- Smart home automation involves using technology to control home appliances, lighting, heating, and security remotely or automatically.
- Common examples include controlling lights with a smartphone app, using voice assistants like Alexa to manage home devices, and using sensors to adjust home conditions based on occupancy or temperature.

Why Arduino?

- Arduino provides an accessible, low-cost platform for building custom smart home systems.
- Unlike expensive, proprietary smart home systems, Arduino projects can be tailored to individual needs and connected to a wide variety of components and devices.
- With its ability to connect to the internet (via Wi-Fi modules) and interact with sensors, Arduino makes it easy to build smart home projects from scratch.



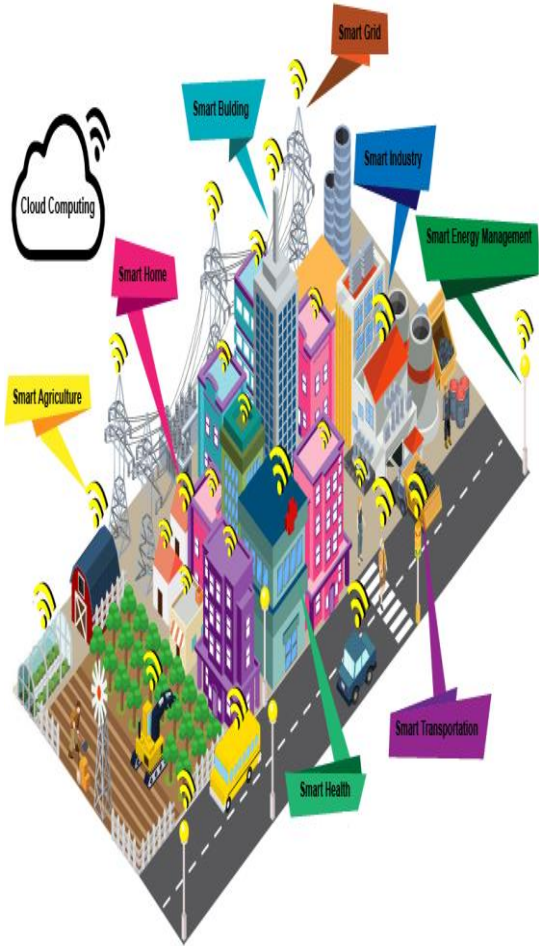
Arduino in Smart Home Automation

- **Project Example: Smart Lighting System**
  - **Arduino Components Used**: Arduino Uno (the main microcontroller board), motion sensors (to detect movement), relays (to switch high-voltage devices like light bulbs), and the light bulbs themselves.
  - **Functionality**: The motion sensors detect when someone enters or leaves a room, automatically turning the lights on or off accordingly. This system can be extended to dim lights, change colors, or set timers.
  - **Benefits**: This saves energy by ensuring lights are only on when needed, and it's also a cost-effective solution compared to commercial smart lighting systems.
- **Project Example: Security System**
  - **Components**: Motion sensor to detect intruders, camera module to capture images or video, buzzer for alerts, and a Wi-Fi module for sending notifications.
  - **Functionality**: When the motion sensor detects movement, the Arduino triggers the buzzer and captures images via the camera module. An alert is then sent to the homeowner’s phone.
  - **Impact**: Such a system improves home security and is relatively inexpensive compared to professional home security systems.

Smart Cities Overview

What are Smart Cities?

- Smart cities integrate advanced technologies (IoT, sensors, data analysis) into infrastructure to improve quality of life.
- Examples include efficient public transportation, smart grids, environmental monitoring, and waste management systems.
- The goal of a smart city is to make urban living more sustainable, efficient, and responsive to the needs of its inhabitants.



Arduino in Smart Cities

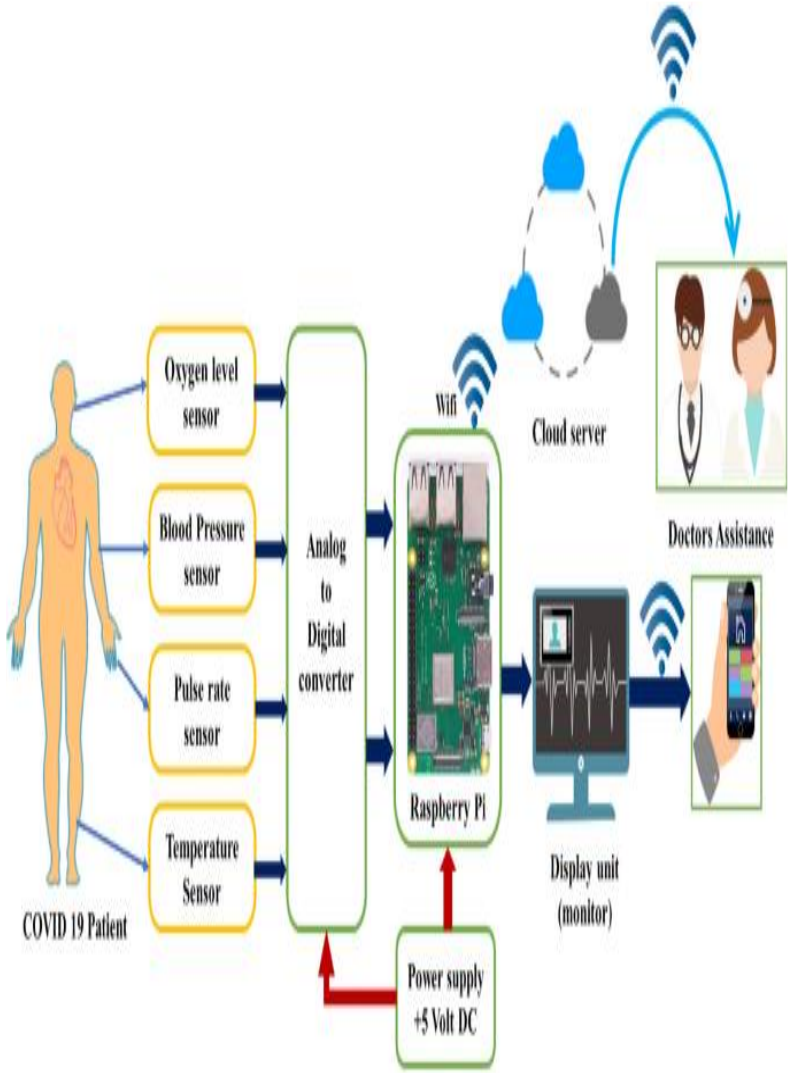
CHITKARA UNIVERSITY

- **Project Example: Smart Traffic Management**
  - **Arduino Components**: Ultrasonic sensors (to measure the number of vehicles), RF modules (for wireless communication between traffic lights), and LEDs (to signal traffic).
  - **Functionality**: Ultrasonic sensors detect traffic volume at intersections. The Arduino adjusts traffic light timings dynamically based on the data, allowing for more efficient flow of vehicles during peak hours.
  - **Impact**: This reduces traffic congestion and travel time, leading to a smoother flow of traffic, reduced pollution, and fewer accidents.
- **Project Example: Smart Waste Management**
  - **Components**: Ultrasonic sensors (to measure waste levels in bins), Arduino Mega (to process data), GSM module (to send alerts to waste management services).
  - **Functionality**: The sensors track the level of waste in public bins, and when full, an alert is sent to city waste services, ensuring timely emptying of bins.
  - **Impact**: This minimizes the overflow of trash, reduces the need for constant monitoring by city personnel, and leads to a cleaner city environment.

Healthcare Overview

How is Arduino Revolutionizing Healthcare?

- Arduino enables rapid development of low-cost medical devices that can help in monitoring and managing health conditions.
- For communities or countries with limited access to expensive healthcare equipment, Arduino-based devices can be a more affordable alternative.
- It also encourages DIY medical innovation, with tools that allow individuals to monitor their own health or create custom health solutions.



Arduino in Healthcare

Project Example: Heart Rate Monitoring System

- **Arduino Components**: Pulse sensor (to measure heartbeats), OLED display (to show data), Bluetooth module (to send data to a smartphone).
- **Functionality**: The pulse sensor monitors the user's heart rate, displaying real-time data on the OLED screen. The Bluetooth module allows for wireless transmission of the data to a smartphone or computer for further analysis.
- **Impact**: This device is portable and affordable compared to commercial heart rate monitors, making it accessible for home use or in remote areas.

Project Example: Glucose Monitoring Device

- **Components**: Arduino Nano (a small and compact board), glucose sensor (to measure blood sugar levels), and a buzzer (for alerts).
- **Functionality**: The sensor monitors blood glucose levels and sends an alert when readings are outside the normal range. The device can log the data for future reference or transmit it to a doctor.
- **Impact**: This low-cost device can help diabetic patients monitor their condition at home, reducing the need for constant doctor visits and improving self-care.

Arduino in Wearable Devices

- **Project Example: Smart Fitness Tracker**
  - **Arduino Components**: Arduino Pro Mini (small and low-power), accelerometer (to measure movement), pulse sensor (to measure heart rate).
  - **Functionality**: The fitness tracker monitors daily activity like steps taken and heart rate. It provides feedback to the user via a smartphone or a small display.
  - **Impact**: This makes fitness tracking more affordable and customizable. Unlike commercial fitness trackers, you can build and tweak the system based on personal preferences.
- **Project Example: Posture Monitoring Device**
  - **Components**: Gyroscope sensor (to measure body orientation), Arduino Nano (small and compact), vibration motor (for feedback).
  - **Functionality**: The gyroscope sensor tracks the user's posture. If it detects slouching or poor posture, the vibration motor activates, gently reminding the user to adjust their posture.
  - **Impact**: This wearable device helps users develop better posture habits, reducing the risk of back pain or long-term posture issues, and is a low-cost alternative to commercial posture correction devices.

Conclusion

Key Takeaways:

- Arduino is a versatile and powerful tool, driving innovation across multiple industries.
- In smart homes, it enables low-cost, energy-efficient automation solutions.
- For smart cities, Arduino improves resource management and urban efficiency.
- In healthcare, it offers affordable, customizable devices for health monitoring.
- The wearable industry benefits from Arduino’s compactness and ease of prototyping, allowing anyone to create fitness, health, and lifestyle trackers.

The Future of Arduino: With continued advancements, Arduino's role in these fields will only grow, offering more sophisticated and integrated solutions.