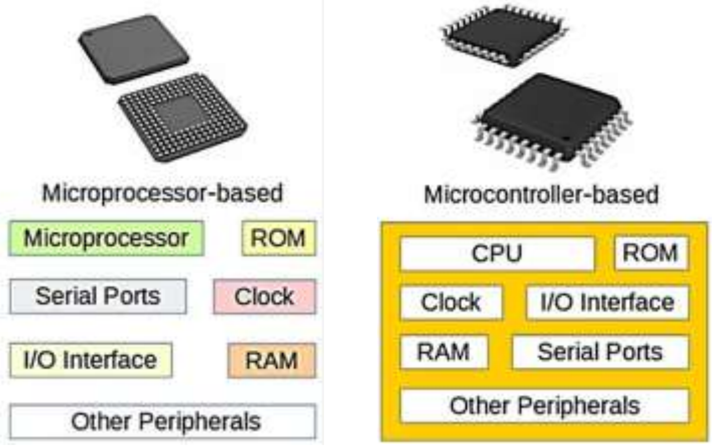
Internet Of Things - Short Note

Embedded System

Embedded system is a computer system embedded into some other system such as a refrigerator, washing machine, car, etc. It also follows Input, Process and Output (IPO) model.
Input → Sensors capture the state
Process → Processor processes
Output → Performs through actuators

The **microcontroller** is a single chip containing a CPU, memory, I/O ports, and other peripherals.
Most embedded systems are microcontroller based because they **do not require expensive, powerful microprocessors** to implement their basic functionalities.

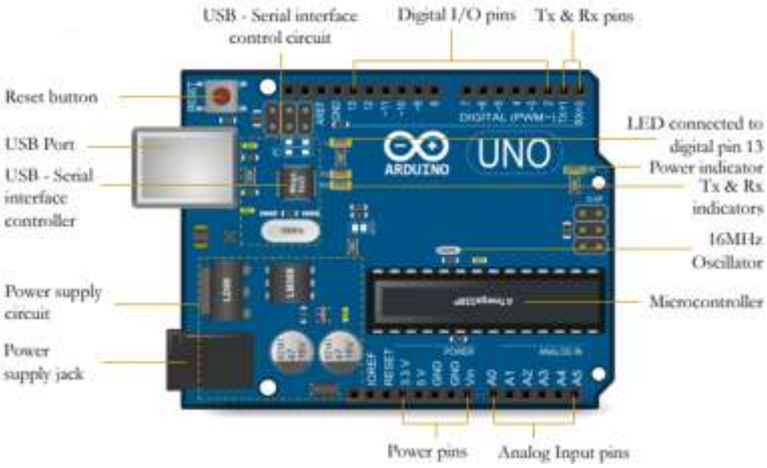


Examples for Microcontrollers

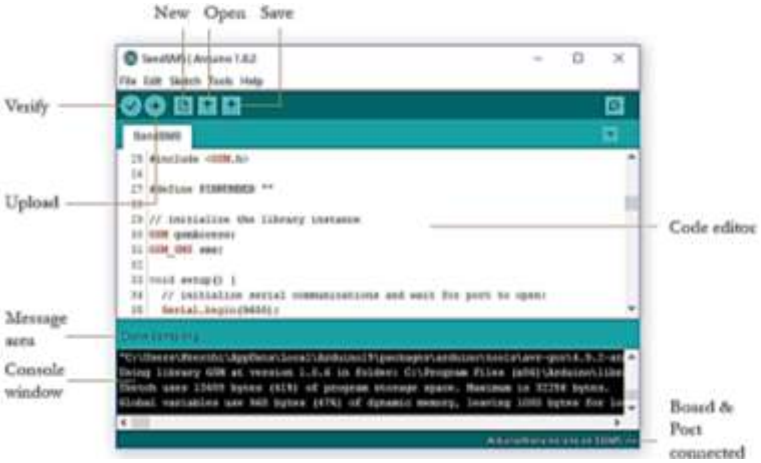
- micro:bit
- EasyPIC
- Arduino board
- Raspberry Pi board

Arduino - open-source, low cost, easy-to-use hardware, and software platform with Cross platform support. Offers extensive official and community support and Extensive availability of software libraries.

Arduino Uno Board



Arduino IDE components



```
#include <GSM.h>

#define [illegible]

// initialize the library instance
[illegible]
[illegible]

void setup() {
  // initialize serial communications and wait for port to open:
  Serial.begin(9600);
```

Microcontroller based Development Systems vs. microprocessor-based systems

Microprocessor	Microcontroller
Used in general-purpose systems like PCs	Used in embedded systems like appliances
Only CPU; external memory & I/O required	CPU, memory, and I/O are built-in
Larger circuit, not suitable for compact systems	Compact and ideal for small devices
Higher system cost and power consumption	Lower cost and low power usage
Lacks power-saving features	Has power-saving modes
Based on Von Neumann architecture	Based on Harvard architecture
Requires external bus for peripherals	Uses internal bus
High-speed operation	Operates at lower speeds (up to 200MHz)

Key Differences between Microprocessor and Microcontroller:

- A **microprocessor** contains only the CPU, while a **microcontroller** has a CPU, memory, and I/O ports all on a single chip.
- **Microprocessors** are mainly used in personal computers, whereas **microcontrollers** are designed for embedded systems.
- **Microprocessors** connect memory and peripherals via an external bus; **microcontrollers** use an internal bus for control.
- **Microprocessors** typically follow the **Von Neumann** architecture, while **microcontrollers** use the **Harvard** architecture.
- Microprocessors are more complex, costly, and handle many instructions; microcontrollers are simpler, cheaper, and handle fewer instructions.

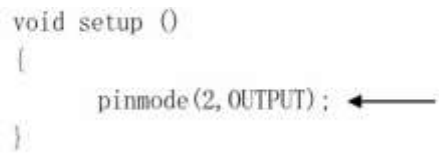
Basic Arduino Programming Guidelines

1. Every instruction must end with a **semicolon (;)**.
2. Conditional (if) and loop structures (for, while) as well as functions must be enclosed in **curly braces {}**.
3. **Single-line**: // comment here
   Multi-line: /* comment block */

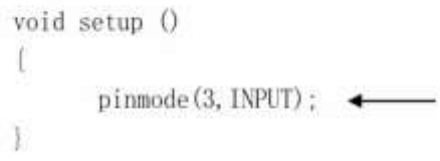
4. void setup() runs once when the program starts.
   void loop() runs repeatedly.
5. Use pinMode(pin, mode) in setup() to set a pin as INPUT, OUTPUT
6. Use digitalRead(pin) for digital input.
   Use analogRead(pin) for analog input.
7. digitalWrite() is used to send output signals to a digital pin.

- Arduino code is **case-sensitive**

Pinmode: This identifies the pin on which the inputs and output must be given.



```
void setup ()
{
      pinmode(2, OUTPUT); ←
}
```



```
void setup ()
{
      pinmode(3, INPUT); ←
}
```

DigitalWrite: This command is used to change the voltage of I/O pins of Arduino board.
Ex: digitalWrite(2,HIGH)
   digitalWrite(2,LOW)

Begin: define the number of bits should be transmitted when communicating with a communication device
Ex : Serial.begin (9600)

Delay: to change the me frame on which a task will be performed.

```
Ex: digitalWrite(2,HIGH);
    delay(1000);
    digitalwrite(2,LOW)
```

If:

```
 if (condition)
 {
          statement(s)
 }
```

For

```
 for (initialization;condition;increment)
 {
          statement(s);
 }
```

While

```
 while(condition)
 {
          statement(s)
 }
```

IoT (Internet of Things)
A network of interconnected smart systems where everyday objects communicate and act autonomously to improve convenience and comfort in life.

Smart World
A vision where **autonomous, interconnected smart systems** work together seamlessly—for example, a smart alarm clock communicating with a smart kettle or refrigerator.

Things
Everyday physical objects, ranging from **wristwatches to cars and buildings**.

When embedded systems are connected to the **Internet**, they can:

- **Interact** with each other, **Communicate** with users
  Form a large network called the **Internet of Things**

IoT is enabled by technologies like IPv6 for large address space, cheaper and faster networking, compact and affordable sensors, and efficient, low-power processors and storage. These advancements make connecting and controlling smart devices easier and more practical.

Major components of IOT

1) Smart devices & sensors
2) IOT gateway
3) Cloud
4) Analytics
5) User Interface

Examples for IOT

Smart Watch
Google Home devices
Smart door locks
Smart Gardening
Video doorbells
Personal Assistant

Future Systems Using IoT

1. **Smart transportation** – Automated control based on vehicle size.
2. **Environmental management** – Automated control and improvement.
3. **Healthcare** – Monitoring patients, automatic medicine dispensing.
4. **Construction** – Observing and managing activities remotely.
5. **Machine communication** – Transmitting data between machines.
6. **Vehicle tracking** – Real-time location tracking.
7. **Agriculture** – Automatic water supply based on need.
8. **City monitoring** – Smart city surveillance and management.
9. **Remote home control** – Operate lights, motors from a distance.
10. **Smart infrastructure** – Homes, outlets, schools, cities.

Challenges of IoT

- Data confidentiality and copyright concerns.
- Insufficient research and updates.
- Complex hardware requirements.
- Dependence on stable power supply.

Disadvantages of IoT

- **Privacy issues** – User data may be exposed.
- **Security issues** – Risk of unauthorized access.

To construct an embedded system, it is essential to follow some steps as given below:

- Construct the schematic diagram and assemble hardware
- Design firmware
- Develop firmware
- Compile firmware and Uploade machine code

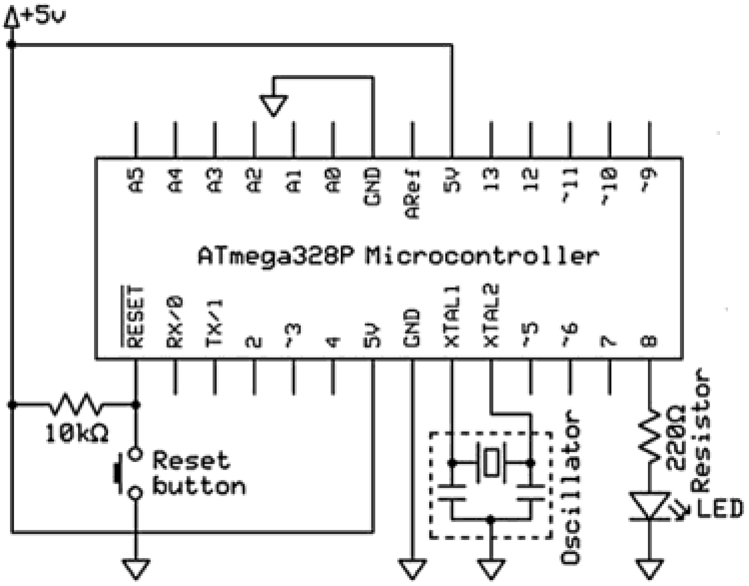
Examples for Embedded systems using Arduino

SYSTEM 1: Blinker

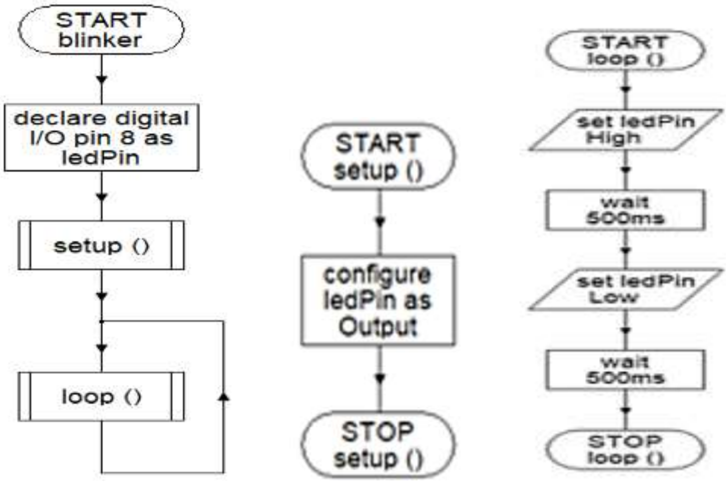
Required components

1 × Arduino Uno microcontroller-based development board
1 × LED
1 × 220Ω Resistor

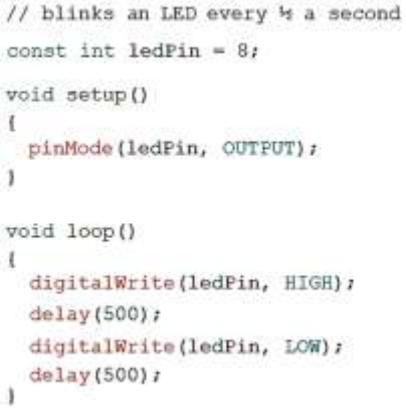
Schematic Diagram



Flowchart



Develop Firmware



```
// blinks an LED every ½ a second
const int ledPin = 8;

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

void loop()
{
  digitalWrite(ledPin, HIGH);
  delay(500);
  digitalWrite(ledPin, LOW);
  delay(500);
}
```

SYSTEM2: AutoLight

Required components

1 × Arduino Uno microcontroller-based development board

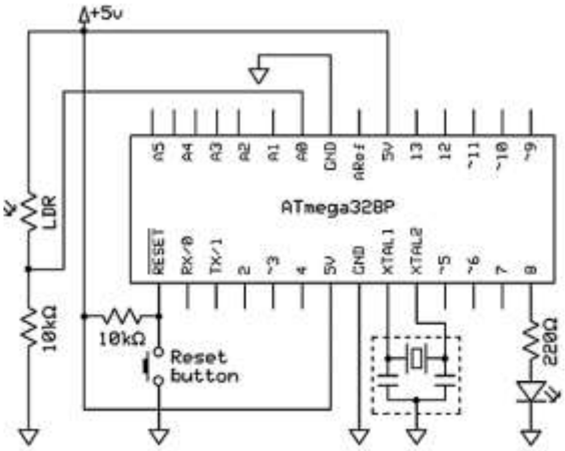
1 × LED

1 × 220Ω Resistor

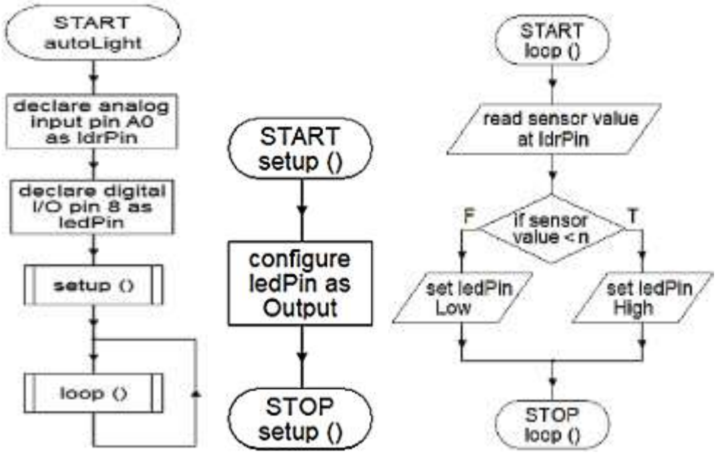
1 × Light Dependent Resistor (LDR)

1 × 10Ω Resistor

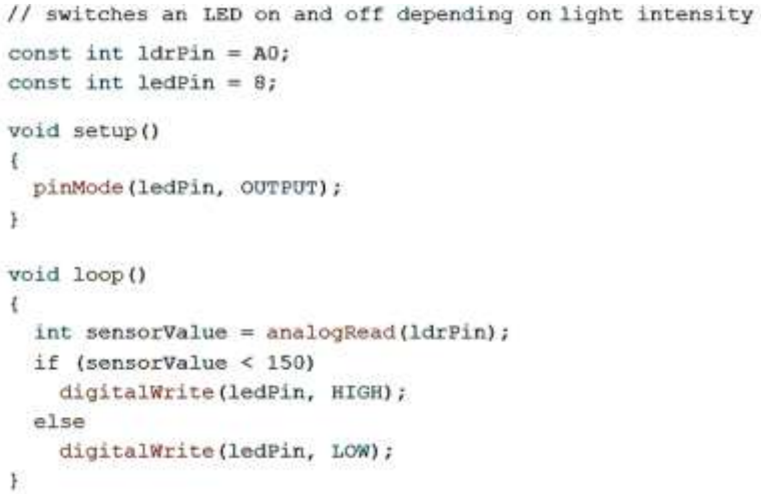
Schematic Diagram



Flowchart



Develop Firmware



```
// switches an LED on and off depending on light intensity
const int ldrPin = A0;
const int ledPin = 8;

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

void loop()
{
  int sensorValue = analogRead(ldrPin);
  if (sensorValue < 150)
    digitalWrite(ledPin, HIGH);
  else
    digitalWrite(ledPin, LOW);
}
```

SYSTEM3: AutoFan

Required components

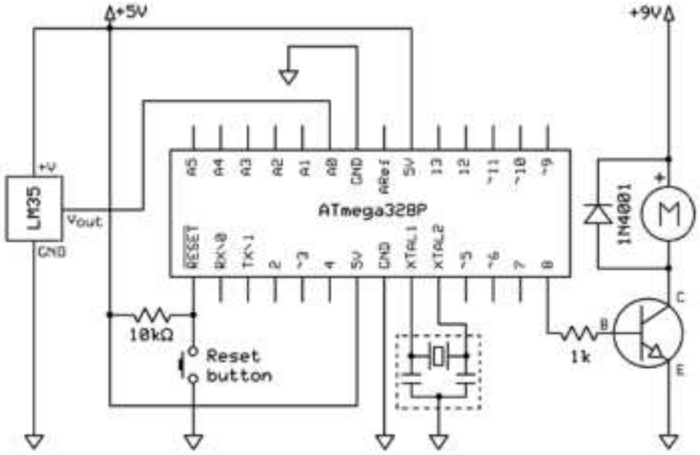
1 × Arduino Uno microcontroller-based development

1 × 9 Volts DC Motor & 1 × LM35 Temperature Sensor

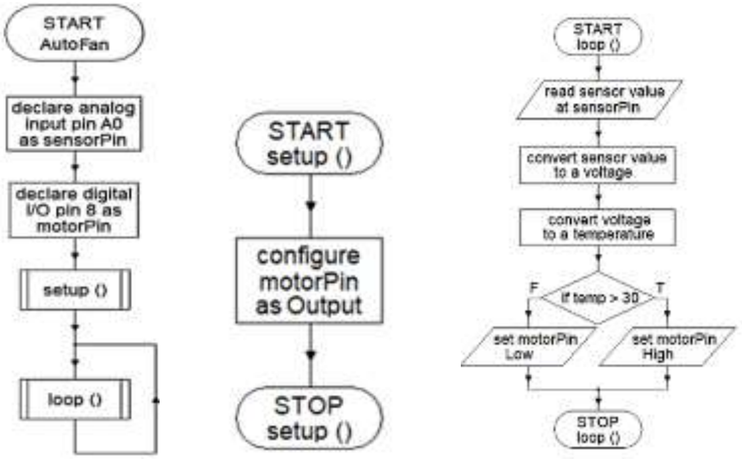
1 × BC547 Transistor & 1 × 1kΩ Resistor

1 × 1N4001Rectifier Diode

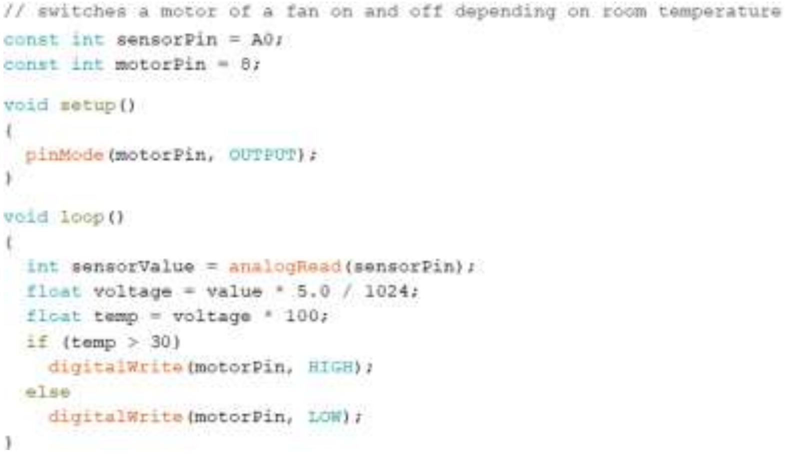
Schematic Diagram



Flowchart


START
AutoFan
declare analog input pin A0 as sensorPin
declare digital I/O pin 8 as motorPin
setup ()
loop ()
START
setup ()
configure motorPin as Output
STOP
setup ()
START
loop ()
read sensor value at sensorPin
convert sensor value to a voltage
convert voltage to a temperature
F
if temp > 30
T
set motorPin Low
set motorPin High
STOP
loop ()

Develop Firmware



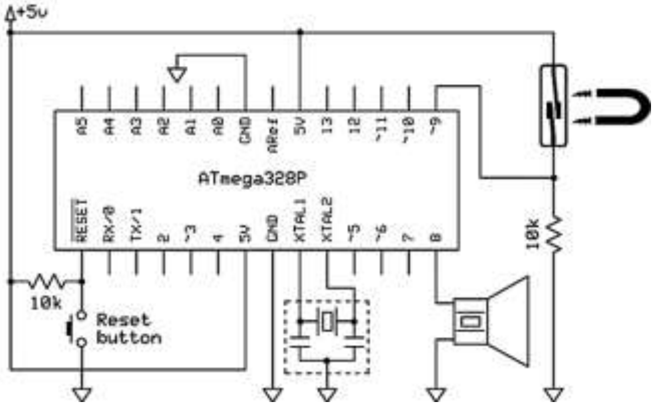
```
// switches a motor of a fan on and off depending on room temperature
const int sensorPin = A0;
const int motorPin = 8;

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

void loop()
{
  int sensorValue = analogRead(sensorPin);
  float voltage = value * 5.0 / 1024;
  float temp = voltage * 100;
  if (temp > 30)
    digitalWrite(motorPin, HIGH);
  else
    digitalWrite(motorPin, LOW);
}
```

SYSTEM4: Door-Alarm

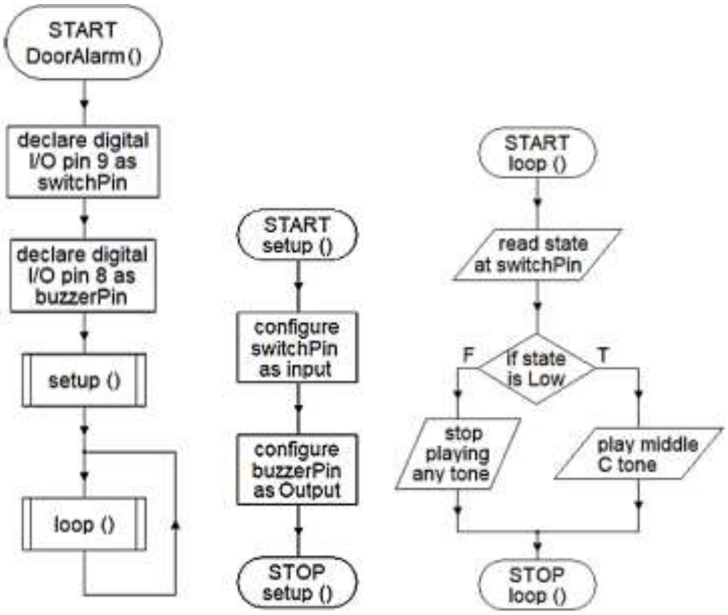
Required components

1 × Arduino Uno microcontroller-based development Board , 1 × Piezo Buzzer , 1 × Reed Switch
1 × 10kΩ Resistor

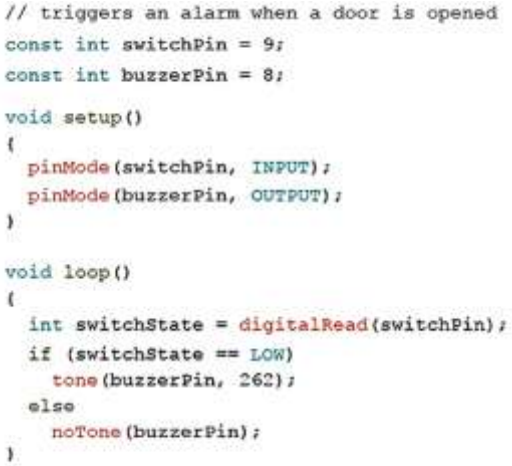
Schematic Diagram



Flowchart



Develop Firmware



```
// triggers an alarm when a door is opened
const int switchPin = 9;
const int buzzerPin = 8;

void setup()
{
  pinMode(switchPin, INPUT);
  pinMode(buzzerPin, OUTPUT);
}

void loop()
{
  int switchState = digitalRead(switchPin);
  if (switchState == LOW)
    tone(buzzerPin, 262);
  else
    noTone(buzzerPin);
}
```