Arduino Programming

1. Setting up the Arduino board. Installation of IDE in PC/ laptop for Arduino programming (Sketch).

2. Program structure: Data types, variables, constants, operators, control statements, loops, functions, string. Conditional like if elseif; for and while loop. Idea about global variable and local variable. Use of serial monitor for input/output and serial plotter for observation of variation of data.

3. Some Basic Operations: (i) Binary operation through HIGH/LOW status of digital pin. Operation on inbuilt LED/ LED connected externally in series with a resistance e.g., blinking. (ii) Sending analog voltage. Use of analog pins. Changing brightness of an LED. (iii) Measurement of voltage through appropriate pins.

Arduino Programming

Date Types: Arduino supports several data types, which are used to declare variables and store different types of data.

A. Basic Data Types:

1. `int` (Integer): Used for whole numbers, Range: –32,768 to 32,767 (16-bit).

```
int count = 10;  // Declare an integer variable and initialize it to 10
```

2. `float` (Floating point): Used for decimal numbers (precision upto 6-7 decimal places)

```
float temperature = 36.5;  // Declare a float variable for a temperature value
```

3. `char` (Character): Used to store a single character or a small integer (ASCII value), occupies 1 byte

```
char letter = 'A';  // Stores the character 'A'
```

Generated in https://chatgpt.com with the prompt ‘Data types in Arduino with examples’

Arduino Programming

A. Basic Data Types (contd.):

4. `byte`: Used for unsigned 8-bit data (0 to 255), used for raw data storage

```
byte data = 255;  // Maximum value a byte can hold
```

5. `boolean`: Used for true/false values

```
boolean isLedOn = true;  // Boolean variable for LED state
```

B. Derived Data Types:

1. `string`: Used for strings of text

```
string message = "Hello, Arduino!";  // String variable for a message
```

2. Array: Used to store multiple values of the same data type

```
int numbers[] = {1, 2, 3, 4, 5};  // Array of integers
```

Generated in https://chatgpt.com with the prompt ‘Data types in Arduino with examples’

Arduino Programming

C. Advanced Data Types:

1. `unsigned int`: Integers without negative values, Range: 0 to 65,535 (16-bit)

```
unsigned int distance = 500;  // Distance in millimeters
```

2. `long`: Used for large integers, Range: -2,147,483,648 to 2,147,483,647 (32-bit)

```
long population = 8000000000;  // Earth's population
```

3. `unsigned long`: Used for large unsigned integers, Range: 0 to 4,294,967,295 (32-bit).

```
unsigned long uptime = 1000000;  // Time in milliseconds
```

4. `double`: Similar to `float` (on most Arduino boards, it has the same precision as `float`).

```
double pi = 3.14159265359;  // Double variable
```

Generated in https://chatgpt.com with the prompt ‘Data types in Arduino with examples’

Arduino Programming

Variables:

- Variables are essential for programming as they allow your code to manipulate and use data dynamically.
- Variables are declared with a data type, a name, and optionally initialized with a value.
- Variable names must start with a letter or underscore; can include letters, numbers, and underscores but no spaces or special characters; cannot use reserved keywords.
- General syntax: `dataType variableName = value;`

Generated in https://chatgpt.com with the prompt ‘Variables in Arduino with examples’

Arduino Programming

Global Variables:

- Declared outside any function.
- Accessible from any function in the program.
- Typically used to store values needed throughout the program.

```
int count = 0;  // Global variable

void setup() {
  Serial.begin(9600);
}

void loop() {
  count++;  // Increment the global variable
  Serial.println(count);
  delay(1000);
}
```

Generated in https://chatgpt.com with the prompt ‘Variables in Arduino with examples’

Arduino Programming

Local Variables:

- Declared inside any function or bloc.
- Accessible only within that function or block.
- Used for temporary or function-specific data.

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int localCount = 0;  // Local variable
  localCount++;        // Only accessible within this loop()
  Serial.println(localCount);
  delay(1000);
}
```

Generated in https://chatgpt.com with the prompt ‘Variables in Arduino with examples’

Arduino Programming

Static Variables:

- Declared inside a function with the `static` keyword.
- Retains its value between function calls.

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  static int persistentCount = 0;  // Retains its value
  persistentCount++;
  Serial.println(persistentCount);
  delay(1000);
}
```

Generated in https://chatgpt.com with the prompt ‘Variables in Arduino with examples’

Arduino Programming

Constants:

- Declared with the `const` keyword.
- Immutable values that do not change during program execution.

```
const int ledPin = 13;  // Constant variable

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

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

Generated in https://chatgpt.com with the prompt ‘Variables in Arduino with examples’

Arduino Programming

Constants:

- Declared with `#define` preprocessor directive as well.

```
#define LED_PIN 13  // Preprocessor constant

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

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

Generated in https://chatgpt.com with the prompt ‘Variables in Arduino with examples’

Arduino Programming

Variable Initialization and Assignment:

Declare and Initialize Separately:

```
int value;  // Declaration

value = 5;  // Assignment
```

Declare and Initialize Together:

```
int value = 5;  // Declaration and initialization
```

Modify Variable Values:

```
value = value + 10;  // Modify variable
```

Generated in https://chatgpt.com with the prompt ‘Variables in Arduino with examples’

Arduino Programming

Operators:

1. Arithmetic Operators: Perform basic mathematical operations.

Operator	Description	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus (remainder)	x % y

Generated in https://chatgpt.com with the prompt ‘Operators in Arduino with examples’

Arduino Programming

1. Arithmetic Operators (contd.)

```
void setup() {
  Serial.begin(9600);
  int a = 10, b = 3;
  Serial.println(a + b);  // Output: 13
  Serial.println(a - b);  // Output: 7
  Serial.println(a * b);  // Output: 30
  Serial.println(a / b);  // Output: 3
(integer division)
  Serial.println(a % b);  // Output: 1
}
void loop() {}
```

Generated in https://chatgpt.com with the prompt ‘Operators in Arduino with examples’

Arduino Programming

2. Assignment Operators: Assign or update the value of variables.

Operator	Description	Example
=	Assign value	x = 5
+=	Add and assign	x += 5
-=	Subtract and assign	x -= 5
*=	Multiply and assign	x *= 5
/=	Divide and assign	x /= 5
%=	Modulus and assign	x %= 5

Generated in https://chatgpt.com with the prompt ‘Operators in Arduino with examples’

Arduino Programming

2. Assignment Operators (contd.)

```
void setup() {
  Serial.begin(9600);
  int x = 10;
  x += 5;  // x = x + 5
  Serial.println(x);  // Output: 15
}

void loop() {}
```

Generated in https://chatgpt.com with the prompt ‘Operators in Arduino with examples’

Arduino Programming

3. Relational Operators: Compare two values. They return `true` (1) or `false` (0).

Operator	Description	Example
==	Equal to	x == y
!=	Not equal to	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater or equal	x >= y
<=	Less or equal	x <= y

Generated in https://chatgpt.com with the prompt ‘Operators in Arduino with examples’

Arduino Programming

3. Relational Operators (contd.)

```
void setup() {
  Serial.begin(9600);
  int a = 10, b = 20;
  Serial.println(a > b);  // Output: 0 (false)
  Serial.println(a < b);  // Output: 1 (true)
}

void loop() {}
```

Generated in https://chatgpt.com with the prompt ‘Operators in Arduino with examples’

Arduino Programming

4. Logical Operators: perform logical operations.

Operator	Description
&&	Logical AND (both true)
||	Logical OR (either one is true)
!	Logical NOT (invert truth value)

```
void setup() {
  Serial.begin(9600);
  int a = 10, b = 20;
  Serial.println((a < b) && (a > 5));  // Output: 1 (true)
  Serial.println((a > b) || (a > 5));  // Output: 1 (true)
  Serial.println(!(a < b));            // Output: 0 (false)
}

void loop() {}
```

Generated in https://chatgpt.com with the prompt ‘Operators in Arduino with examples’

Arduino Programming

5. Increment and Decrement Operators: Used to increase or decrease a variable's value by 1.

Operator	Description
++	Increment by 1 (++x or x++)
--	Decrement by 1 (--x or x--)

```
void setup() {
  Serial.begin(9600);
  int x = 10;
  Serial.println(x++);  // Output: 10 (post-increment)
  Serial.println(++x);  // Output: 12 (pre-increment)
}
void loop() {}
```

Generated in https://chatgpt.com with the prompt ‘Operators in Arduino with examples’

Arduino Programming

6. Compound Operators: Combine arithmetic and assignment.

Operator	Description	Example
+=	Add and assign	x += y
-=	Subtract and assign	x -= y
*=	Multiply and assign	x *= y
/=	Divide and assign	x /= y
%=	Modulus and assign	x %= y

```
void setup() {
  Serial.begin(9600);
  int x = 10;
  x += 5;  // Equivalent to x = x + 5
  Serial.println(x);  // Output: 15
}
void loop() {}
```

Generated in https://chatgpt.com with the prompt ‘Operators in Arduino with examples’

Arduino Programming

Functions: Functions in Arduino are blocks of reusable code designed to perform a specific task. They help organize code, make it more readable, and avoid repetition.

1. **Built-in Functions:** These are pre-defined functions provided by the Arduino environment, such as `digitalWrite()`, `delay()`, `pinMode()` etc.
2. **Used-defined Functions:** Users create to implement custom behavior.

Syntax of a Function

```
returnType functionName(parameters) {
    // Function body
    return value;  // Optional, if returnType is not void
}
```

Generated in https://chatgpt.com with the prompt ‘Functions in Arduino with examples’

Arduino Programming

Key Components of Functions

1. **Return Type:** Specifies the type of data the function will return (`int`, `float`, `void`, etc.).
2. **Function Name:** Used to call the function.
3. **Parameters:** Optional inputs the function accepts.
4. **Function Body:** The code that runs when the function is called.
5. **Return Statement:** Optional, used to return a value if the return type is not void.

Generated in https://chatgpt.com with the prompt ‘Functions in Arduino with examples’

Arduino Programming

1. Basic Functions

Function	Use
`setup()`	Runs once at the start of the program. Used to initialize settings.
`loop()`	Runs repeatedly after `setup()`. Contains the main program logic.

---

2. Digital I/O Functions

Function	Use
`pinMode(pin, mode)`	Sets a pin as `INPUT`, `INPUT_PULLUP`, or `OUTPUT`.
`digitalWrite(pin, value)`	Sets a digital pin to `HIGH` or `LOW`.
`digitalRead(pin)`	Reads the state of a digital pin (`HIGH` or `LOW`).

Generated in https://chatgpt.com with the prompt ‘various functions name and use in arduino’

Arduino Programming

3. Analog I/O Functions

Function	Use
`analogRead(pin)`	Reads an analog value (0–1023) from an analog input pin.
`analogWrite(pin, value)`	Writes a PWM signal (0–255) to a pin (only for PWM-enabled pins).
`analogReference(type)`	Sets the reference voltage for analog inputs. Default is `DEFAULT` (5V or 3.3V).

4. Time Functions

Function	Use
`delay(ms)`	Pauses the program for a specified number of milliseconds.
`millis()`	Returns the number of milliseconds since the program started.
`micros()`	Returns the number of microseconds since the program started.
`delayMicroseconds(us)`	Pauses the program for a specified number of microseconds.

Generated in https://chatgpt.com with the prompt ‘various functions name and use in arduino’

Arduino Programming

5. Communication Functions

Serial Communication

Function	Use
`Serial.begin(baudRate)`	Starts serial communication at a specified baud rate (e.g., 9600).
`Serial.print(value)`	Prints data to the serial monitor without a newline.
`Serial.println(value)`	Prints data to the serial monitor followed by a newline.
`Serial.read()`	Reads incoming serial data.
`Serial.available()`	Returns the number of bytes available to read.

Generated in https://chatgpt.com with the prompt ‘various functions name and use in arduino’

Arduino Programming

6. Math Functions

Function	Use
`min(a, b)`	Returns the smaller of two numbers.
`max(a, b)`	Returns the larger of two numbers.
`abs(x)`	Returns the absolute value of a number.
`constrain(x, a, b)`	Constrains a value between a minimum (`a`) and a maximum (`b`).
`map(value, fromLow, fromHigh, toLow, toHigh)`	Maps a number from one range to another.
`pow(base, exponent)`	Returns the result of raising a base to an exponent.
`sqrt(x)`	Returns the square root of a number.
`randomSeed(seed)`	Initializes the random number generator with a seed.
`random(min, max)`	Returns a random number between `min` and `max` .

Generated in https://chatgpt.com with the prompt ‘various functions name and use in arduino’

Arduino Programming

Example: Mapping Analog Input

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);  // Read analog input
  int mappedValue = map(sensorValue, 0, 1023, 0, 255);  // Map to 0–255
  Serial.println(mappedValue);
  delay(500);
}
```

Generated in https://chatgpt.com with the prompt ‘various functions name and use in arduino’

Arduino Programming

7. String Functions

Function	Use
`strlen(string)`	Returns the length of a string.
`strcmp(a, b)`	Compares two strings. Returns 0 if they are equal.
`strcat(dest, src)`	Appends one string to another.
`strcpy(dest, src)`	Copies one string to another.

Generated in https://chatgpt.com with the prompt ‘various functions name and use in arduino’

Arduino Programming

Control Statements: Used to manage the flow of execution in a program. They help in decision-making, looping, and controlling the overall structure of the code.

1. Decision-Making Statements

- `if`
- `if-else`
- `else if`
- `switch-case`

2. Looping Statements

- `for`
- `while`
- `do-while`

3. Jump Statements

- `break`
- `continue`
- `return`

Generated in https://chatgpt.com with the prompt ‘control statements in Arduino with examples’

Arduino Programming

`if` Statement: Executes a block of code if a condition is `true`

```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(7, INPUT);
}

void loop() {
  if (digitalRead(7) == HIGH) {  // If button is pressed
    digitalWrite(13, HIGH);      // Turn on LED
  }
}
```

Generated in https://chatgpt.com with the prompt ‘control statements in Arduino with examples’

Arduino Programming

`If-else` Statement: Executes one block if the condition is `true`, and another block if the condition is `false`

```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(7, INPUT);
}

void loop() {
  if (digitalRead(7) == HIGH) {  // If button is pressed
    digitalWrite(13, HIGH);      // Turn on LED
  } else {
    digitalWrite(13, LOW);       // Turn off LED
  }
}
```

Generated in https://chatgpt.com with the prompt ‘control statements in Arduino with examples’

Arduino Programming

`else if` Statement: Checks multiple conditions

```
void setup() {
  pinMode(9, OUTPUT);  // LED connected to PWM pin
}

void loop() {
  int sensorValue = analogRead(A0);  // Read sensor value
  if (sensorValue < 300) {
    analogWrite(9, 50);  // Low brightness
  } else if (sensorValue < 700) {
    analogWrite(9, 150);  // Medium brightness
  } else {
    analogWrite(9, 255);  // High brightness
  }
}
```

Generated in https://chatgpt.com with the prompt ‘control statements in Arduino with examples’

Arduino Programming

`Switch-case` Statement: Selects one of many blocks of code to execute based on a variable's value

```
//LED colour selection
void setup() {
  pinMode(3, OUTPUT);  // Red
  pinMode(5, OUTPUT);  // Green
  pinMode(6, OUTPUT);  // Blue
}

void loop() {
  int mode = analogRead(A0) / 341;  // Map sensor value to 0-2
  switch (mode) {
    case 0:
      digitalWrite(3, HIGH);  // Red ON
      digitalWrite(5, LOW);
      digitalWrite(6, LOW);
      break;
    case 1:
      digitalWrite(3, LOW);
      digitalWrite(5, HIGH);  // Green ON
      digitalWrite(6, LOW);
      break;
    case 2:
      digitalWrite(3, LOW);
      digitalWrite(5, LOW);
      digitalWrite(6, HIGH);  // Blue ON
      break;
  }
}
```

Generated in https://chatgpt.com with the prompt ‘control statements in Arduino with examples’

Arduino Programming

`for` Loop Statement: Repeats a block of code a specific number of times.

```
//LED Blink Sequence
void setup() {
  for (int i = 2; i <= 7; i++) {  // Set
pins 2-7 as outputs
    pinMode(i, OUTPUT);
  }
}

void loop() {
  for (int i = 2; i <= 7; i++) {  //
Iterate over pins
    digitalWrite(i, HIGH);
    delay(500);
    digitalWrite(i, LOW);
    delay(500);
  }
}
```

Generated in https://chatgpt.com with the prompt ‘control statements in Arduino with examples’

Arduino Programming

`while` Loop Statement: Repeats a block of code as long as a condition is `true`.

```
//Wait for Button Press
void setup() {
  pinMode(7, INPUT);
}

void loop() {
  while (digitalRead(7) == LOW) {
    // Do nothing, wait for button press
  }
  Serial.println("Button Pressed!");
}
```

Generated in https://chatgpt.com with the prompt ‘control statements in Arduino with examples’

Arduino Programming

`do-while` Loop Statement: Repeats a block of code as long as a condition is `true`.

```
//LED Blinking Until Button Pressed
void setup() {
  pinMode(13, OUTPUT);
  pinMode(7, INPUT);
}

void loop() {
  do {
    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
    delay(500);
  } while (digitalRead(7) == LOW);  // Stop when button is pressed
}
```

Generated in https://chatgpt.com with the prompt ‘control statements in Arduino with examples’

Arduino Programming

`break` Statement: Exits a loop or `switch-case`.

```
void setup() {
  for (int i = 0; i < 10; i++) {
    if (i == 5) {
      break;  // Exit loop when i equals 5
    }
    Serial.println(i);
  }
}

void loop() {}
```

Generated in https://chatgpt.com with the prompt ‘control statements in Arduino with examples’

Arduino Programming

`continue` Statement: Skips the rest of the current iteration and moves to the next iteration.

```
//Skip Even Numbers
void setup() {
  for (int i = 0; i < 10; i++) {
    if (i % 2 == 0) {
      continue;  // Skip even numbers
    }
    Serial.println(i);
  }
}

void loop() {}
```

Generated in https://chatgpt.com with the prompt ‘control statements in Arduino with examples’

Arduino Programming

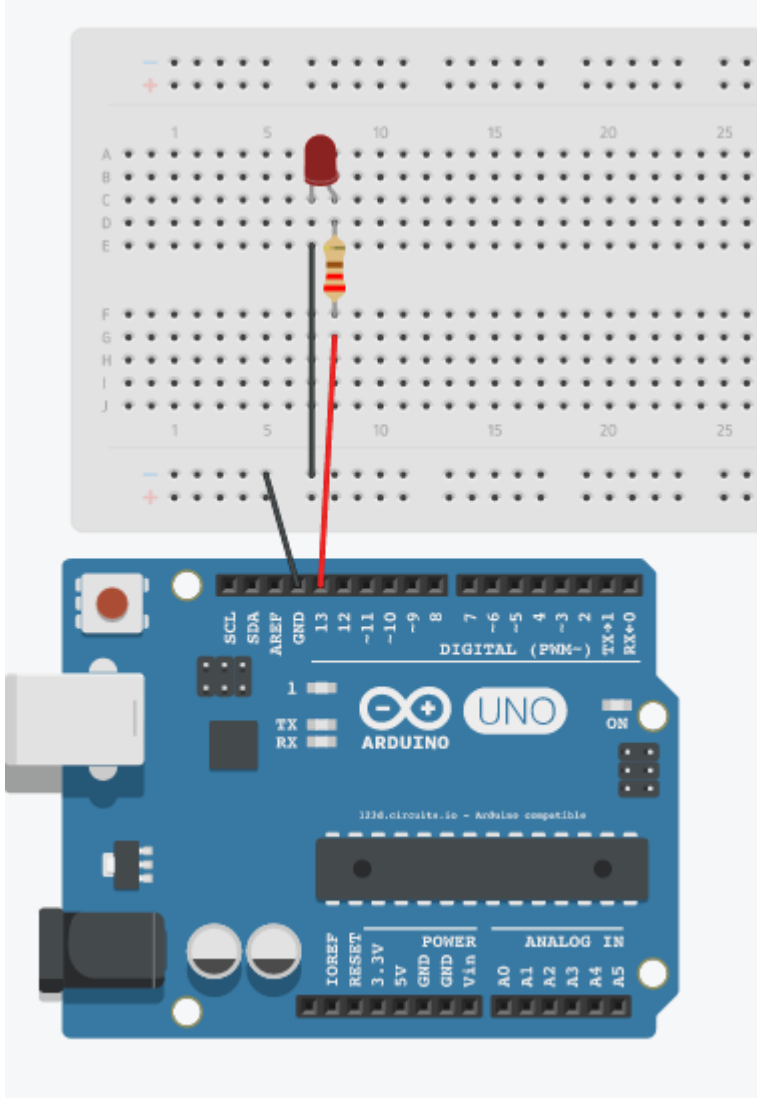
`return` Statement: Exits a function and optionally returns a value.

```
int sum(int a, int b) {
  return a + b;
}

void setup() {
  Serial.begin(9600);
  int result = sum(5, 10);  // Call
function and get result
  Serial.println(result);   // Output: 15
}

void loop() {}
```

Generated in https://chatgpt.com with the prompt ‘control statements in Arduino with examples’

Arduino Programming

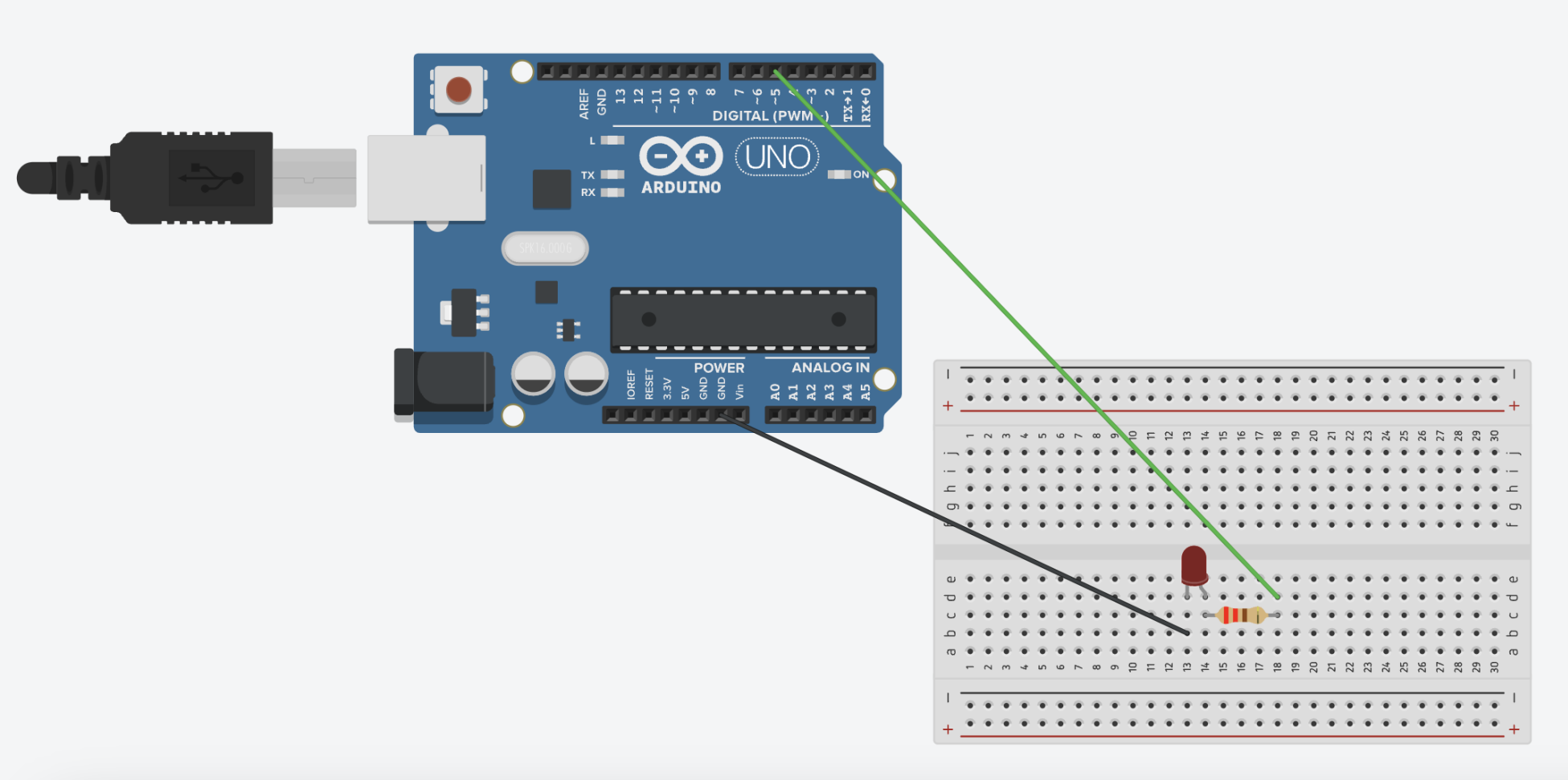
Some Basic Operations: Blinking of LED




```
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
}
```

Arduino Programming

Some Basic Operations: Changing brightness of an LED



Arduino Programming

Some Basic Operations: Changing brightness of an LED

```
// Define the pin where the LED is connected
const int ledPin = 5;  // PWM-enabled pin

void setup() {
  pinMode(ledPin, OUTPUT);  // Set the pin as output
}

void loop() {
  // Gradually increase the brightness
  for (int i = 0; i <= 255; i++) {
    analogWrite(ledPin, i);  // Set LED brightness
    delay(100);              // Wait 100 ms for a smooth fade
  }
  // Gradually decrease the brightness
  for (int i = 255; i >= 0; i--) {
    analogWrite(ledPin, i);  // Set LED brightness
    delay(10);               // Wait 10ms for a smooth fade
  }
}
```

Arduino Programming

Some Basic Operations: Measurement of voltage through appropriate pins

```
const int aP = A0;   // Analog pin to read voltage
const float rV = 5.0; // Reference voltage of Arduino (3.3V or
5V)
const float rR = 1.0; // Resistor ratio, adjust if using a
voltage divider

void setup() {
  Serial.begin(9600); // Initialize serial communication
}

void loop() {
  int value = analogRead(aP); // Read raw ADC value (0-1023)
  float voltage = (value / 1023.0) * rV * rR;
  Serial.print("Voltage: ");
  Serial.print(voltage);
  Serial.println(" V");
  delay(1000); // Delay for readability
}
```