Week 1 Assignment

Introduction to Arduino Platform and Programming

The Arduino platform is an open-source electronics platform based on easy-to-use hardware and software. It is widely used by beginners, hobbyists, and professionals to create interactive electronic projects.

1. What is Arduino?

Arduino refers to both an open-source electronics platform and the development environment used to program it. It consists of a microcontroller (like the ATmega328 on the Arduino Uno) and a software IDE to write, compile, and upload code.

2. Key Features of Arduino:

- Open-source and cost-effective
- Cross-platform (works on Windows, macOS, and Linux)
- Simple and accessible IDE (Integrated Development Environment)
- A large community with lots of tutorials and forums
- Various models for different applications (Uno, Nano, Mega, Leonardo, etc.)

3. Arduino Hardware:

The most commonly used board is the Arduino Uno. It features:

- 14 digital input/output pins (of which 6 can be used as PWM outputs)
- 6 analog inputs
- A 16 MHz quartz crystal
- USB connection
- Power jack
- ICSP header
- Reset button

4. Arduino Software (IDE):

The Arduino IDE is where users write code (called sketches), verify it, and upload it to the Arduino board. The programming language is based on a simplified version of C/C++.

5. Basic Arduino Programming Structure:

Arduino code typically consists of two main functions:

- `setup()`: This function runs once when the board is powered up or reset. It's used to initialize variables, pin modes, start using libraries, etc.
- `loop()`: This function runs continuously in a loop. It is the core of the Arduino program where the main logic is implemented.

6. Example Code: Blinking an LED

```cpp
void setup() {
  pinMode(13, OUTPUT); // Set pin 13 as output
}

void loop() {
  digitalWrite(13, HIGH); // Turn on the LED
  delay(1000);            // Wait for one second
  digitalWrite(13, LOW);  // Turn off the LED
  delay(1000);            // Wait for one second
}
```

This simple code turns an LED connected to pin 13 on and off with a 1-second delay, demonstrating the basics of digital output and timing.

Conclusion:
The Arduino platform is a great way to get started with electronics and embedded systems. Understanding its components, software, and basic programming structure is essential for future projects.

Week 2 Assignment

In Week 3, we explore the practical implementation of basic Arduino programming concepts using the Tinkercad website. Tinkercad is an online simulator that allows us to build and test Arduino circuits virtually without needing physical hardware. The activities covered in this assignment are:

i. LED Blinking

This is the simplest example of Arduino programming. It involves connecting an LED to a digital pin and using code to turn it on and off with a delay in between.

```cpp
void setup() {
  pinMode(13, OUTPUT); // Set pin 13 as output
}

void loop() {
  digitalWrite(13, HIGH); // LED on
  delay(1000);          // Wait for 1 second
  digitalWrite(13, LOW);  // LED off
  delay(1000);          // Wait for 1 second
}
```

ii. Delay Effect

Using the `delay()` function in Arduino allows us to create time gaps between actions. This is commonly used to control timing, such as blinking rate or sensor reading intervals.

Example of rapid blink with shorter delay:

```cpp
void loop() {
  digitalWrite(13, HIGH);
  delay(200);          // 200 milliseconds delay
  digitalWrite(13, LOW);
  delay(200);
}
```

iii. Serial Transmission

The `Serial` library enables communication between Arduino and the computer. This is useful for debugging or displaying sensor values.

```cpp
void setup() {
  Serial.begin(9600);     // Initialize serial communication at 9600 bps
}

void loop() {
  Serial.println("Hello from Arduino!");
  delay(1000);
}
```

iv. LCD Display Interface

An LCD (Liquid Crystal Display) can be used to show text output directly on the screen connected to the Arduino. In Tinkercad, a 16x2 LCD is typically used.

```cpp
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  lcd.begin(16, 2);               // Initialize the LCD with 16 columns and 2 rows
  lcd.print("Hello, Tinkercad!");   // Print message to LCD
}

void loop() {
  // Nothing needed here for static text
}
```

Conclusion:
This week's tasks provide a hands-on introduction to the Arduino platform using Tinkercad. These foundational skills—blinking LEDs, using delays, transmitting data via serial, and interfacing with an LCD—are crucial for developing more complex projects in future weeks.

Assignment - 3

Simple Program Read/Write using LED and Switch

In this assignment, we focus on performing basic read/write operations using digital and analog pins on the Arduino platform. The tasks will demonstrate how to control LEDs using both software timing and physical inputs like push buttons.

i. Digital Read/Write
Digital pins are used to read the state of buttons or switches (HIGH or LOW) and to control outputs like LEDs or relays.

ii. Analog Read/Write
Analog read allows us to read sensor values (0-1023), while analog write (PWM) allows us to control outputs like LED brightness (0-255).

a. To light the LED connected to pin number 13:
We want the LED to remain ON for 4 seconds and OFF for 1.5 seconds repeatedly.

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

void loop() {
  digitalWrite(13, HIGH); // LED ON
  delay(4000);          // Wait for 4 seconds
  digitalWrite(13, LOW);  // LED OFF
  delay(1500);          // Wait for 1.5 seconds
}
```

b. Controlling the Light Emitting Diode (LED) with a push button:
When the button is pressed, the LED should turn ON. When released, the LED should turn OFF.

```cpp
const int buttonPin = 2;
const int ledPin = 13;
int buttonState = 0;

void setup() {
  pinMode(buttonPin, INPUT);
```

```
  pinMode(ledPin, OUTPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

c. Fade-in and fade-out the LED using switch:
We control LED brightness gradually increasing and decreasing when a switch is ON.

```cpp
const int ledPin = 9;    // PWM pin for LED
const int switchPin = 2;
int switchState = 0;

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

void loop() {
  switchState = digitalRead(switchPin);

  if (switchState == HIGH) {
    for (int brightness = 0; brightness <= 255; brightness++) {
      analogWrite(ledPin, brightness);
      delay(10);
    }
    for (int brightness = 255; brightness >= 0; brightness--) {
      analogWrite(ledPin, brightness);
      delay(10);
    }
  } else {
    analogWrite(ledPin, 0); // Turn off LED if switch not pressed
```

Assignment - 4

Interfacing Sensors

In Assignment 4, we explore how to interface sensors and communication modules with Arduino. Specifically, we focus on temperature/humidity sensors and Bluetooth modules.

---

i. Temperature Sensor (LM35) and/or Humidity Sensor (e.g., DHT11)

LM35 Temperature Sensor:

The LM35 outputs an analog voltage proportional to the temperature. It does not require external calibration.

Connections:

- VCC → 5V
- GND → GND
- Output → A0

Code:

```
int sensorPin = A0;
float temperature;

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

void loop() {
  int reading = analogRead(sensorPin);
  temperature = reading * 0.488; // Convert analog value to temperature
in °C
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");
  delay(1000);
}
```

DHT11 Temperature & Humidity Sensor:

This digital sensor provides both temperature and humidity readings. It requires the `DHT` library.

Connections:

- VCC → 5V
- GND → GND
- Data → Digital Pin 2

Code:

```
#include <DHT.h>

#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

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

void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C");
  delay(2000);
}
```

---

ii. Bluetooth Module (HC-05/HC-06)

The HC-05 or HC-06 module allows wireless serial communication between the Arduino and other Bluetooth-enabled devices such as smartphones.

Connections:

- VCC → 5V
- GND → GND
- TX of HC-05 → RX of Arduino (through voltage divider)
- RX of HC-05 → TX of Arduino

Code:

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

void loop() {
  if (Serial.available()) {
```

```
    char data = Serial.read();
    Serial.print("Received: ");
    Serial.println(data);
  }
}
```

Note: You can use any Bluetooth terminal app on your smartphone to send data to the Arduino. The Arduino will display the received data on the Serial Monitor.

Conclusion:

This assignment introduces essential interfacing techniques for real-time data acquisition and wireless communication using Arduino. Mastering these components opens the door to building advanced IoT and sensor-based systems.