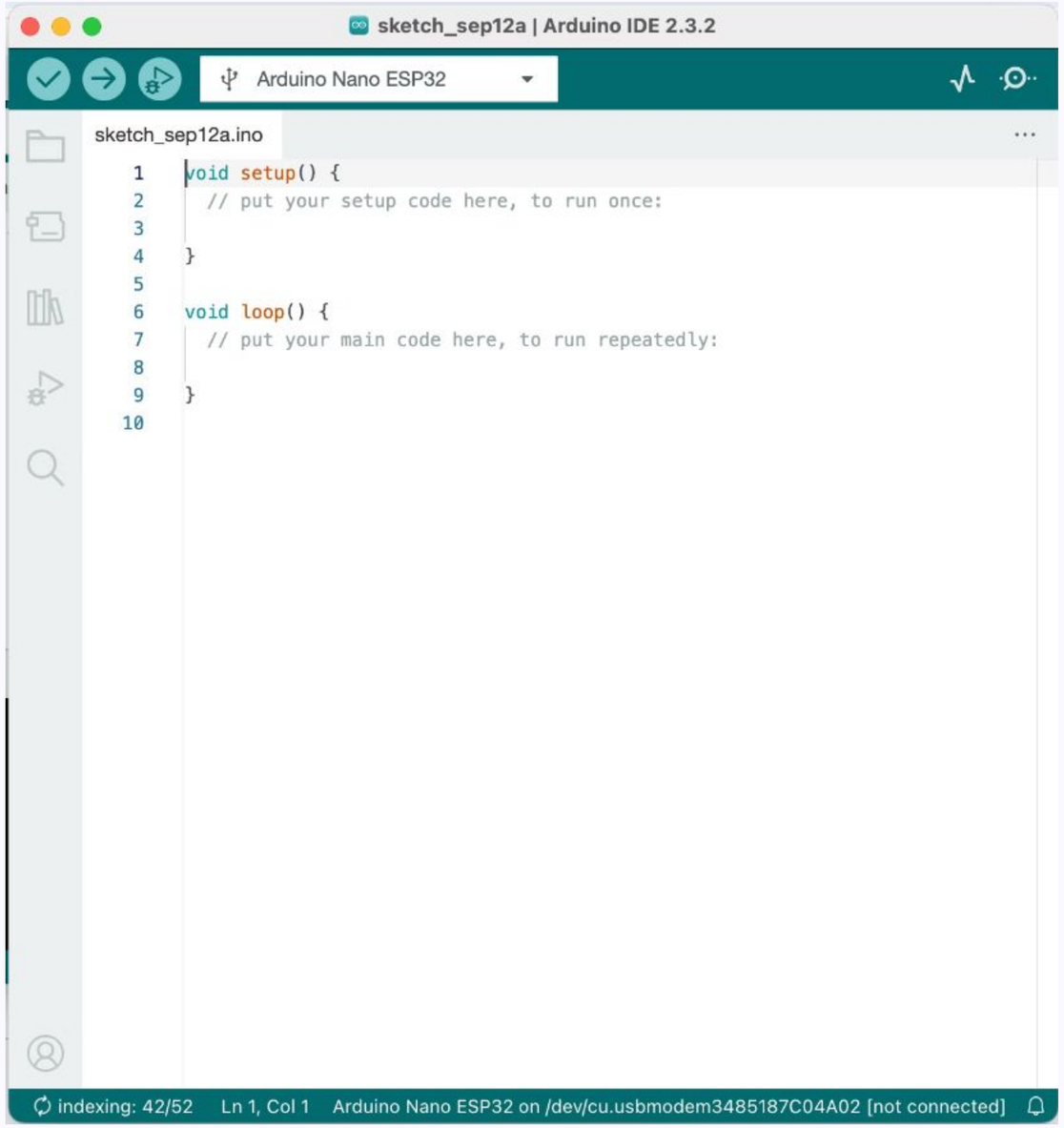
第5課:Arduino聲音程式庫



Arduino IDE (整合開發環境)

提供所有撰寫、編譯和上傳代碼到Arduino主板所需的軟體應用程式。

- 具有語法突顯(syntax highlighting)的程式碼編輯器
- 包含將您的代碼轉換為機器語言的編譯器
- 提供用於除錯的序列監視器
- 管理用於擴展功能的函式庫

為什麼在人工智能時代,創意編碼技能仍然很重要?

了解基礎知識

人工智能可以生成代碼,但了解電路、感測器和程式設計的核心原理,可以讓您自訂超越範本的樂器,並排除導電墨水專案中出現的意外錯誤。

創意解決問題

在創作像"墨奏"這樣的新型樂器時,您將遇到現有代碼庫或人工智能訓練資料中沒有的獨特挑戰,需要原創性的解決方案和調適。

實體運算整合

將代碼與Arduino等硬體融合需要親身瞭解數位訊號如何與導電墨水等類比材料互動,這是人工智能無法完全取代的。

藝術表達

創意編碼允許您在手勢如何轉換為聲音方面做出決定,為您的樂器賦予個性化特色,這是一般人工智能生成的代碼無法做到的。

平衡基礎知識與擴增智慧(Augmented Intelligence)工具, 以創意為介入

故障排除能力

了解電路基礎知識,可以讓學生診斷”墨奏”專案中出現的問題,尤其是在使用新型導電材料時,這是人工智慧可能無法識別的。

超越範本的自訂

掌握Arduino程式設計原理,可以讓學生修改人工智慧生成的代碼, 專門針對獨特的手勢控制聲音。

材料-數位整合

了解導電墨水如何與感測器互動, 讓學生創造出樂器, 同時使用人工智慧來加速重複性的編碼任務。

Arduino基本程式設計

1

編碼結構 (Structure)

編碼包含setup()和loop()。

2

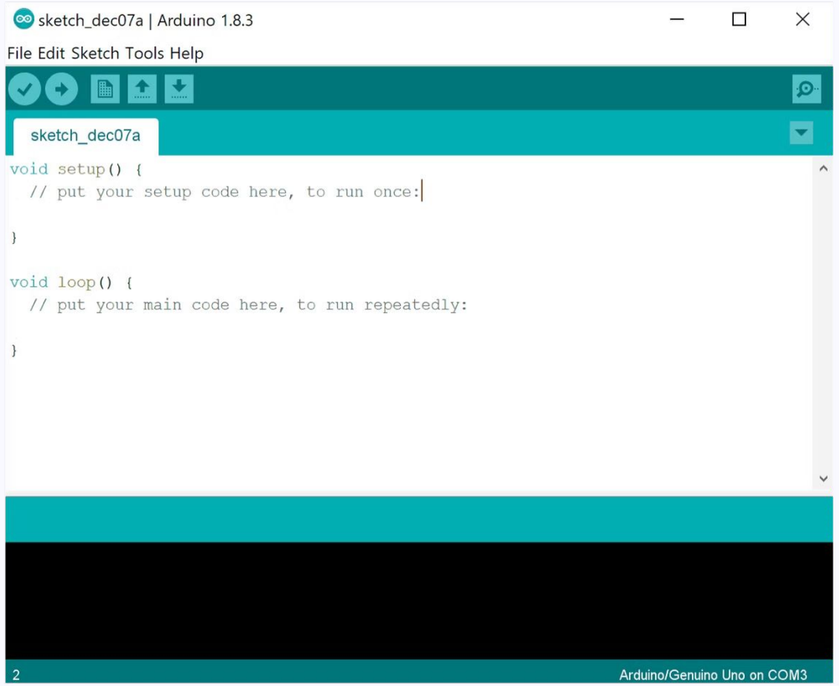
Setup

初始化變量和引腳。

3

Loop

重複執行主要程式碼。



setup() 和 loop()

setup()

在Arduino上電或重置時只會執行一次。

- 使用pinMode(pin, INPUT/OUTPUT)初始化引腳模式
- 使用Serial.begin(9600)開始串列通訊
- 設定變數的初始值

```
void setup() {
  pinMode(13, OUTPUT);  // LED引腳
  Serial.begin(9600);   // 開始串列監控
  digitalWrite(13, LOW); // 初始狀態
}
```

loop()

在setup()完成後持續重複執行。

- 包含主要程式邏輯
- 讀取感測器和控制輸出
- 一直執行直到斷電

```
void loop() {
  digitalWrite(13, HIGH); // 打開LED
  delay(1000);            // 等待1秒
  digitalWrite(13, LOW);  // 關閉LED
  delay(1000);            // 等待1秒
}
```

例子: LED閃爍 (Blinking LED)

連接LED

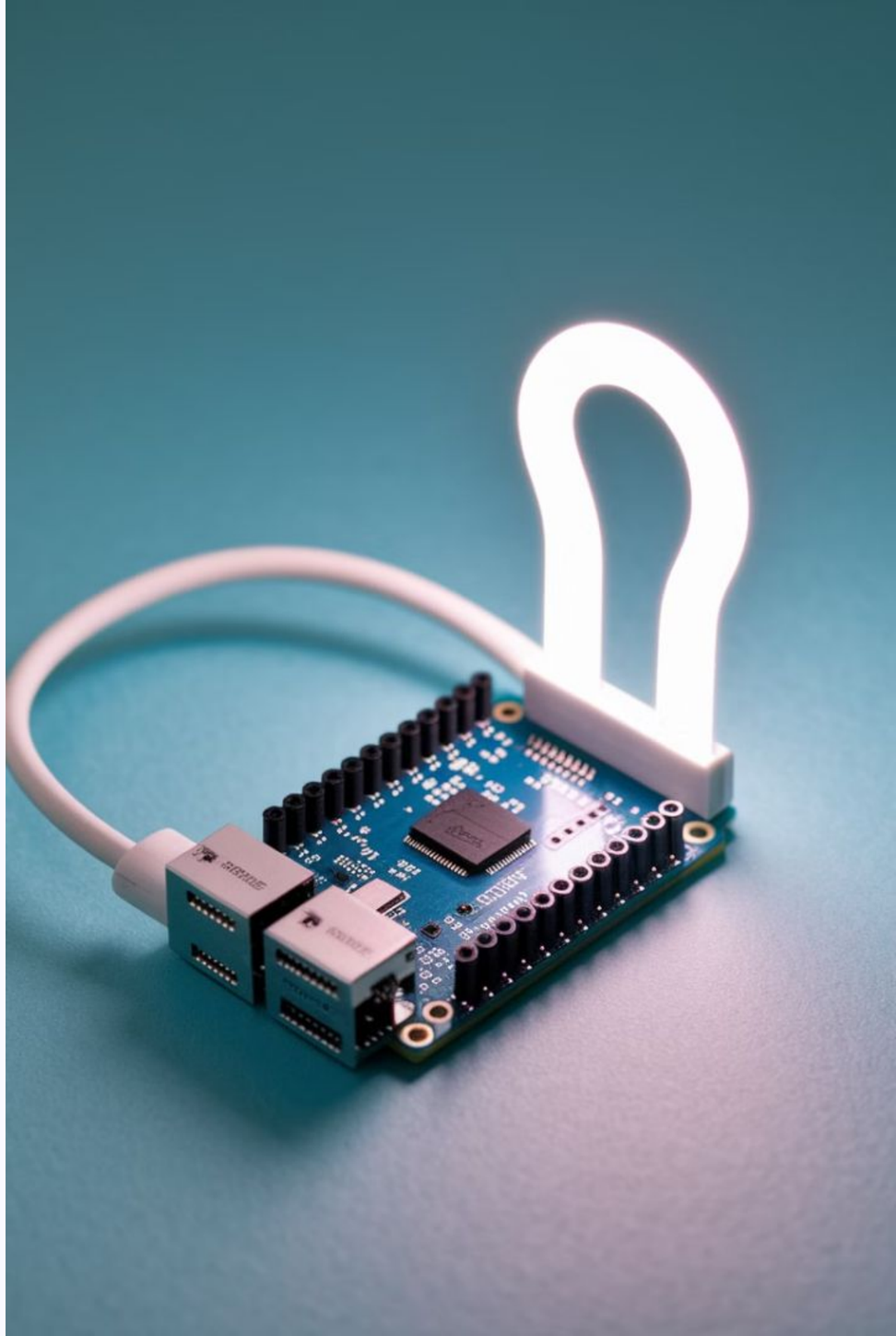
將LED連接到Arduino引腳上。

編寫程式

建立編碼來令LED閃爍。

上傳

將程式上傳到Arduino板上。



LED閃爍 (Blinking LED)

```
void setup() {
  // 將數位腳位LED_BUILTIN初始化為輸出模式。
  pinMode(LED_BUILTIN, OUTPUT);
}


// loop()函式會一直重複執行
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // 打開LED(HIGH表示高電位)
  delay(1000);                      // 等待1秒
  digitalWrite(LED_BUILTIN, LOW);   // 關閉LED(LOW表示低電位)
  delay(1000);                      // 等待1秒
}
```

什麼是Arduino的序列監視器(Serial Monitor)?

一個調試工具,允許開發人員了解Arduino與電腦之間進行通信。

目的

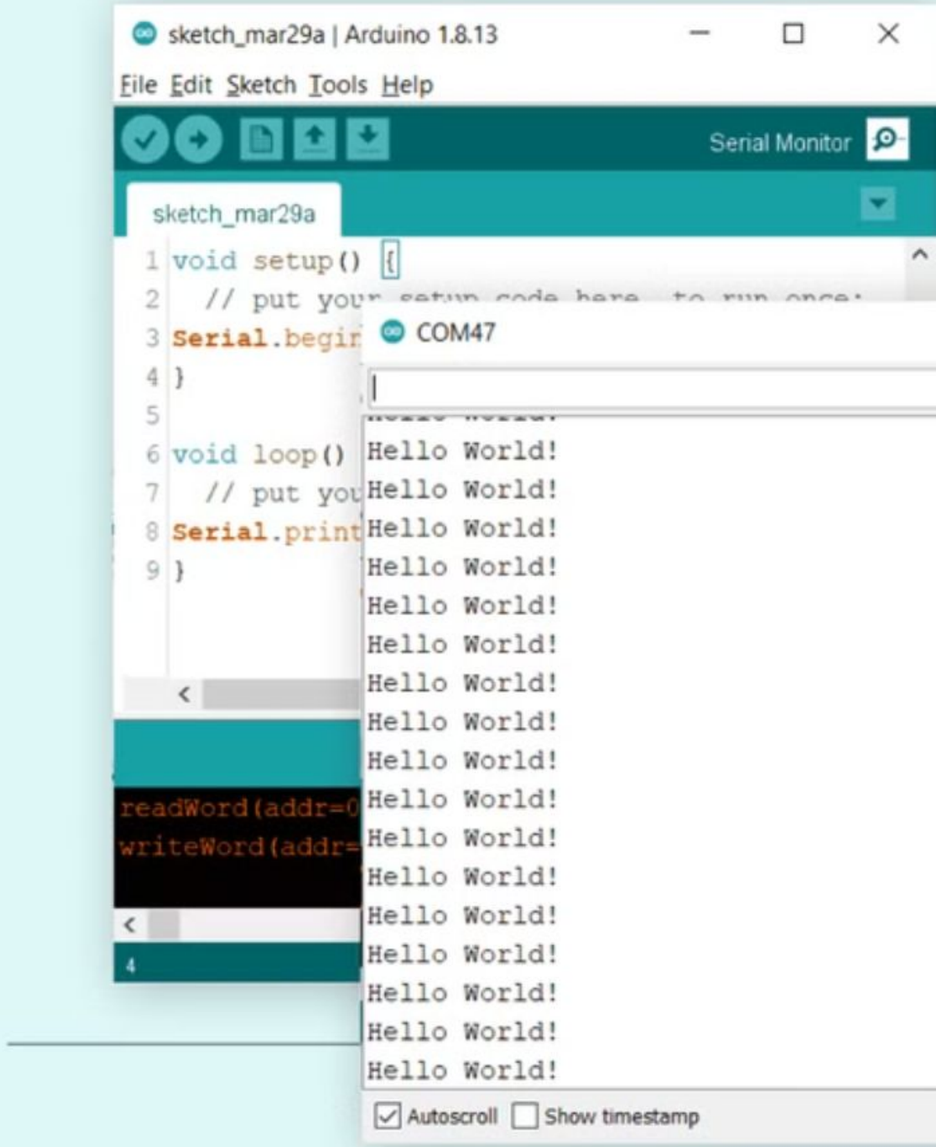
顯示從Arduino發送到電腦的數據,對於故障排除和監控感測器值至關重要。

使用

可從Arduino IDE通過點擊放大鏡圖標或使用Ctrl+Shift+M來訪問。

配置

必須與代碼中的波特率(baud rate)(例如, Serial.begin(9600))匹配,才能正確顯示數據。



Arduino函式庫簡介

1 什麼是函式庫?

預先編寫好的程式碼集合。

2 為什麼要使用函式庫?

簡化複雜任務,節省時間。

3 在哪裡找到函式庫?

可在線上和Arduino IDE中找到。

什麼是程式庫?

可重複使用的程式碼套件

預先編寫好能執行特定功能的程式碼集合,可節省開發者開發時間。

Arduino程式庫的結構

Arduino程式庫通常由C++頭文件(.h)和實現文件(.cpp)組成, 以提供易於使用的函數。

我們使用的Arduino函式庫

Tone Library

在任何Arduino針腳上產生方波音調,用於墨奏專案中創造音樂音符。提供tone()和noTone()等函數來控制音調輸出。

Capacitive Sensor Library

使Arduino能夠通過導電材料檢測觸摸。此函式庫用於把導電墨水繪圖轉換為互動式觸摸感測器。

SFEMP3Shield Library

與VS1053 MP3解碼芯片進行接口,以播放存儲在SD卡上的音頻文件。允許墨奏產生超越簡單音調的聲音。

SdFat Library

提供對FAT16或FAT32格式SD卡的高效解碼。用於存儲和檢索樂器的音頻樣本和配置文件。

Arduino程式設計基礎

我們需要了解Arduino中的程式設計概念:

變數和資料類型

宣告適當類型的變數(int(整數)、float(點數)、boolean(布林值(是/否)))來儲存從我們的導電墨水介面讀取的感測器數據

全域變數

使用貫穿整個程式的全域變數來管理我們的觸摸感測器和音效播放功能的狀態

函式庫導入

包含外部.h函式庫文件, 以獲取觸摸檢測和音訊播放所需的功能,例如我們下載的電容式感測器和SFEMP3Shield函式庫

touch_to_play_mp3

```
// 引入所需的SD卡、工具函數和MP3播放器盾板的函式庫。
#include <SdFat.h>
#include <SdFatUtil.h>
#include <SFE_MPHAT_Shield.h>

// 建立SdFat(SD卡)和SFEMP3Shield(MP3播放器)類的實例。
SdFat sd;
SFEMP3Shield MP3player;

// 儲存從A0到A5腳位讀取的模擬值的變數。
int InData0 = 0; // 從A0腳位讀取的模擬值
int InData1 = 0; // 從A1腳位讀取的模擬值
int InData2 = 0; // 從A2腳位讀取的模擬值
int InData3 = 0; // 從A3腳位讀取的模擬值
int InData4 = 0; // 從A4腳位讀取的模擬值
int InData5 = 0; // 從A5腳位讀取的模擬值
int InData0 = 0;

// 檢測觸摸輸入的靈敏度閾值。
int TouchSensitivity = 280;
```

.h C++ 函式庫在Arduino中

Arduino中的標頭(.h)檔案是C++函式庫,包含預先編寫的代碼、函式定義和變數聲明,以擴展Arduino的功能。

對於"墨奏"專案,我們使用了兩個關鍵的函式庫:

CapacitiveSensor 函式庫(.h)使我們的Arduino能夠通過導電墨水圖案檢測觸摸

SFEMP3Shield 函式庫(.h)提供了從VS1053 MP3解碼器播放音頻文件的函數

touch_to_play_mp3

```
void setup() {
    // 初始化串列通訊,設置波特率為115200,用於除錯。
    Serial.begin(115200);

    // 使用9號腳位作為晶片選擇(CS)來初始化SD卡。
    // 如果SD卡初始化失敗,停止程式並顯示錯誤。
    if (!sd.begin(9, SPI_HALF_SPEED)) sd.initErrorHalt();

    // 切換目錄到SD卡的根目錄("/")。
    // 如果目錄切換失敗,停止程式並顯示錯誤。
    if (!sd.chdir("/")) sd.errorHalt("sd.chdir(\"/\")");

    // 初始化MP3播放器,並將音量設為最大(10, 10)。
    MP3player.begin();
    MP3player.setVolume(10, 10);

    // 將A0到A5腳位設定為輸入模式,用於觸摸感測。
    for (int i = A0; i <= A5; i++) {
        pinMode(i, INPUT);
    }

    // 禁用Timer0溢位中斷,避免與其他操作衝突。
    TIMSK0 &= !(1 << TOIE0);
}
```

Serial.begin 115200 vs 9600 ?

Arduino中的串行通信需要設定一個波特率 - 即Arduino與電腦之間傳輸數據的速度。

115200 波特率

- 更快的數據傳輸(比9600快12倍)
- 適用於大量數據,如感測器讀數
- 非常適合”墨奏”的實時響應
- 我們在touch_to_play_mp3代碼中使用 115200

9600 波特率

- Arduino的傳統預設速率
- 對於基本項目更加穩定
- 足夠用於簡單的除錯
- 較慢,但與舊硬件兼容

MP3player.begin() 和 MP3player.setVolume()

MP3player.begin()

初始化VS1053 MP3解碼器屏蔽,以準備音頻播放。此功能:

- 設置Arduino和VS1053晶片之間的通信
- 配置內部設置以獲得最佳音頻性能
- 必須在調用任何其他MP3player函數之前調用

MP3player.setVolume(10, 10)

調整左右聲道的輸出音量級別:

- 參數表示左右聲道的音量

touch_to_play_mp3

```
void loop() {
    // 讀取模擬腳位A0到A5的值,並反轉(較高的值表示檢測到觸摸)。
    InData0 = 1024 - analogRead(A0); // 反轉A0讀數
    InData1 = 1024 - analogRead(A1); // 反轉A1讀數
    InData2 = 1024 - analogRead(A2); // 反轉A2讀數
    InData3 = 1024 - analogRead(A3); // 反轉A3讀數
    InData4 = 1024 - analogRead(A4); // 反轉A4讀數
    InData5 = 1024 - analogRead(A5); // 反轉A5讀數


    ...
}
```

什麼是analogRead()?

analogRead()函數是Arduino命令,它可以讀取指定模擬引腳的值,將電壓(0-5V)轉換為整數值(0-1023)。

讀取感測器輸入

在"墨奏"項目中,我們使用analogRead()來測量來自A0-A5引腳的導電墨水感測器值。

值的解釋

較高的值(接近1023)表示導電性或觸摸強度更大,我們將其反轉(1024 - analogRead())以使觸摸檢測更直觀。

觸摸靈敏度閾值

我們將這些模擬讀數與我們的TouchSensitivity值(280)進行比較,以確定何時正在觸摸導電墨水墊。

此函數對於將我們導電墨水感測器的模擬電信號轉換為Arduino可用的數字值以觸發聲音非常重要。

為什麼要使用1024 - analogRead()?

在我們的Inkstrument項目中,我們使用公式**1024 - analogRead()**來反轉來自我們導電墨水觸摸墊的感測器讀數。這種反轉有三個關鍵目的:

直觀的觸摸反應

如果不進行反轉,由於我們的導電墨水在被觸摸時會創造一條通往接地的路徑,因此更強的觸摸會產生較低的值(接近0)。通過使用1024 - analogRead()進行反轉,較高的值(接近1024)現在代表更強的觸摸。

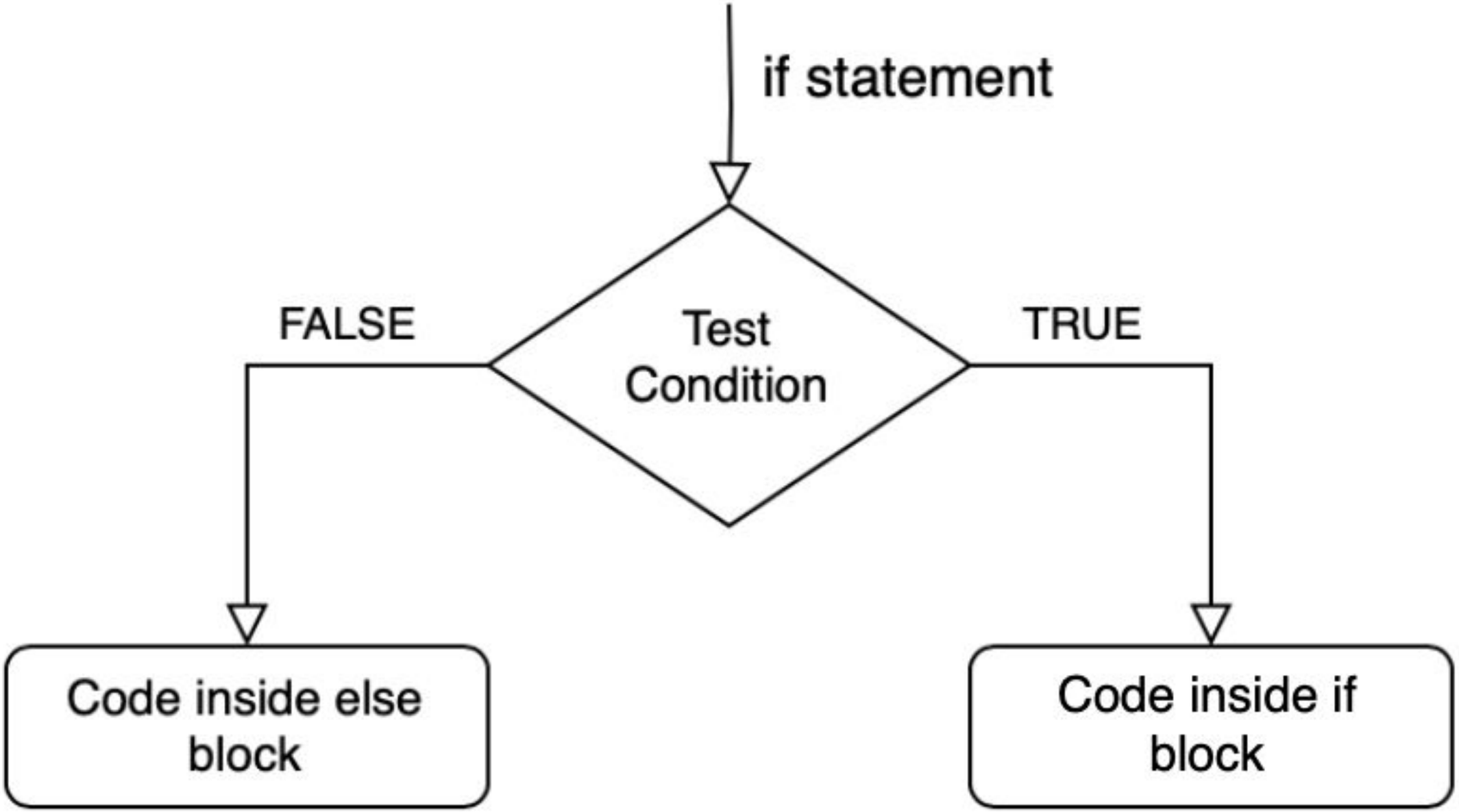
touch_to_play_mp3

```
void loop() {
    ...
    // 檢查哪個腳位已被觸摸,根據靈敏度閾值來判斷。
    // 如果檢測到觸摸,就播放對應的MP3音軌。
    if (InData0 >= TouchSensitivity) {
        MP3player.playTrack(0); // 如果觸摸A0,播放音軌0
    } else if (InData1 >= TouchSensitivity) {
        MP3player.playTrack(1); // 如果觸摸A1,播放音軌1
    } else if (InData2 >= TouchSensitivity) {
        MP3player.playTrack(2); // 如果觸摸A2,播放音軌2
    } else if (InData3 >= TouchSensitivity) {
        MP3player.playTrack(3); // 如果觸摸A3,播放音軌3
    } else if (InData4 >= TouchSensitivity) {
        MP3player.playTrack(4); // 如果觸摸A4,播放音軌4
    } else if (InData5 >= TouchSensitivity) {
        MP3player.playTrack(5); // 如果觸摸A5,播放音軌5
    } else {
        MP3player.stopTrack(); // 如果未檢測到觸摸,停止播放
    }
    ...
}
```

條件語句if(){}else{}

條件語句對於”墨奏”的互動性至關重要。它們允許Arduino根據感測器輸入值做出決策。

```
if (InData0 >= TouchSensitivity) {
    MP3player.playTrack(0);  // 如果觸摸則播放第一個音效
} else if (InData1 >= TouchSensitivity) {
    MP3player.playTrack(1);  // 如果觸摸則播放第二個音效
} else {
    MP3player.stopTrack();   // 否則停止所有音效
}
```


if statement
Test Condition
FALSE
TRUE
Code inside else block
Code inside if block

touch_to_play_mp3

```
void loop() {
    ...
    ...
    // 將MP3播放器的當前狀態和模擬讀數輸出到序列監視器。
    Serial.print(MP3player.isPlaying()); // 輸出MP3播放器是否正在播放
    Serial.print(" ");
    Serial.print(InData0); // 輸出A0的反向值
    Serial.print(" ");
    Serial.print(InData1); // 輸出A1的反向值
    Serial.print(" ");
    Serial.print(InData2); // 輸出A2的反向值
    Serial.print(" ");
    Serial.print(InData3); // 輸出A3的反向值
    Serial.print(" ");
    Serial.print(InData4); // 輸出A4的反向值
    Serial.print(" ");
    Serial.println(InData5); // 輸出A5的反向值,後接換行

    // 增加延遲以防止過於頻繁地讀取觸摸 腳位。
    delay(100);
}
```

Serial.print()

Serial.print()對於除錯我們的Inkstrument至關重要。它可以將數據從Arduino傳送到電腦,讓您實時觀察感測器讀數。

```
// 將觸摸感測器值輸出到Serial Monitor
Serial.print(InData0);  // 第一個墨水感測器讀數
Serial.print(" ");      // 在值之間添加空格
Serial.print(InData1);  // 第二個墨水感測器讀數
Serial.println(InData2); // 在最後一個值後添加換行
```

在設計您的Inkstrument時,請使用Serial.print()來:

- 校準觸摸靈敏度閾值
- 驗證您的導電墨水路徑是否正常工作
- 調試聲音無法按預期觸發的問題

打開Serial Monitor(工具→Serial Monitor),將波特率設置為9600即可查看輸出。