C Arduino - Les conditions

Dans la programmation Arduino en C, les conditions permettent de contrôler le comportement de votre programme en fonction des événements ou des valeurs mesurées. Elles permettent de prendre des décisions et de déclencher des actions différentes selon les situations. Dans cet article, nous allons explorer les instructions conditionnelles `if`, `else if`, et `else`, ainsi que les opérateurs logiques `&&` (ET) et `||` (OU), pour vous aider à mieux gérer les flux d'exécution dans vos projets Arduino.

Découverte

Les Bases des Conditions : if, else if, et else

1. Utilisation de `if`

L'instruction `if` permet de tester une condition et d'exécuter un bloc de code seulement si cette condition est vraie. Si la condition est fausse, le programme ignore le bloc de code.

Syntaxe de base :

```
if (condition) {
    // Code à exécuter si la condition est vraie
}
```

Exemple : Allumer une LED si un bouton est appuyé.

```
int boutonPin = 2;    // Pin où le bouton est connecté
int ledPin = 13;      // Pin où la LED est connectée

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

void loop() {
    int etatBouton = digitalRead(boutonPin);

    if (etatBouton == HIGH) { // Si le bouton est appuyé
        digitalWrite(ledPin, HIGH); // Allume la LED
    }
}
```

Dans cet exemple, la LED s'allume uniquement lorsque le bouton est appuyé.

2. Utilisation de `else if`

L'instruction `else if` permet de tester une condition supplémentaire si la première condition (`if`) est fausse. Vous pouvez chaîner plusieurs `else if` pour tester différentes conditions.

Syntaxe de base :

```
if (condition1) {
    // Code si condition1 est vraie
} else if (condition2) {
    // Code si condition2 est vraie
}
```

Exemple : Contrôler la luminosité d'une LED selon la valeur d'un capteur de lumière.

```
int capteurLumierePin = A0; // Pin du capteur de lumière
int ledPin = 13;

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

void loop() {
    int valeurLumiere = analogRead(capteurLumierePin);

    if (valeurLumiere < 200) {
        analogWrite(ledPin, 255); // Luminosité maximale si faible lumière
    } else if (valeurLumiere < 500) {
        analogWrite(ledPin, 128); // Luminosité moyenne si lumière modérée
    } else {
        analogWrite(ledPin, 0);   // Éteint la LED si lumière forte
    }
}
```

3. Utilisation de `else`

L'instruction `else` est un cas de repli qui s'exécute si aucune des conditions précédentes (`if` ou `else if`) n'est vraie. Elle est souvent utilisée pour définir une action par défaut.

Syntaxe de base :

```
if (condition1) {
    // Code si condition1 est vraie
} else if (condition2) {
    // Code si condition2 est vraie
} else {
    // Code si aucune condition n'est vraie
}
```

Exemple : Allumer, réduire ou éteindre une LED selon la température mesurée.

```arduino
int capteurTemperaturePin = A1;
int ledPin = 13;

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

void loop() {
    int temperature = analogRead(capteurTemperaturePin);

    if (temperature < 300) {
        digitalWrite(ledPin, HIGH); // Allume la LED pour température basse
    } else if (temperature < 600) {
        analogWrite(ledPin, 128);   // Réduit la LED pour température modérée
    } else {
        digitalWrite(ledPin, LOW);  // Éteint la LED pour température élevée
    }
}
```

Opérateurs Logiques : `&&` (ET) et `||` (OU)

Pour combiner plusieurs conditions dans une même instruction `if`, vous pouvez utiliser les opérateurs logiques `&&` (ET) et `||` (OU).

Utilisation de `&&` (ET)

L'opérateur `&&` est utilisé lorsque vous voulez que **plusieurs conditions soient vraies en même temps** pour exécuter le code.

Exemple : Allumer une LED uniquement si deux boutons sont appuyés en même temps.

```arduino
int boutonPin1 = 2;
int boutonPin2 = 3;
int ledPin = 13;

void setup() {
    pinMode(boutonPin1, INPUT);
    pinMode(boutonPin2, INPUT);
    pinMode(ledPin, OUTPUT);
}

void loop() {
    int etatBouton1 = digitalRead(boutonPin1);
    int etatBouton2 = digitalRead(boutonPin2);

    if (etatBouton1 == HIGH && etatBouton2 == HIGH) { // Si les deux boutons sont appuyés
        digitalWrite(ledPin, HIGH); // Allume la LED
    } else {
        digitalWrite(ledPin, LOW); // Éteint la LED
```

```
    }
}
```

Dans cet exemple, la LED s'allume uniquement si **les deux boutons** sont appuyés simultanément.

Utilisation de `||` (OU)

L'opérateur `||` est utilisé lorsque vous voulez que **au moins une des conditions soit vraie** pour exécuter le code.

Exemple : Allumer une LED si au moins un des deux boutons est appuyé.

```
int boutonPin1 = 2;
int boutonPin2 = 3;
int ledPin = 13;

void setup() {
    pinMode(boutonPin1, INPUT);
    pinMode(boutonPin2, INPUT);
    pinMode(ledPin, OUTPUT);
}

void loop() {
    int etatBouton1 = digitalRead(boutonPin1);
    int etatBouton2 = digitalRead(boutonPin2);

    if (etatBouton1 == HIGH || etatBouton2 == HIGH) { // Si un des deux boutons est appuyé
        digitalWrite(ledPin, HIGH); // Allume la LED
    } else {
        digitalWrite(ledPin, LOW); // Éteint la LED
    }
}
```

Dans cet exemple, la LED s'allume si **au moins un des boutons** est appuyé.

Conclusion

Les conditions `if`, `else if`, `else`, ainsi que les opérateurs logiques `&&` et `||`, sont essentiels pour prendre des décisions dans vos programmes Arduino. En combinant ces structures de contrôle, vous pouvez créer des comportements complexes et adaptés aux situations de vos projets, qu'il s'agisse de vérifier plusieurs capteurs ou de déclencher des actions en fonction d'événements spécifiques. Prendre le temps de bien comprendre ces concepts est crucial pour développer des programmes robustes et fonctionnels.