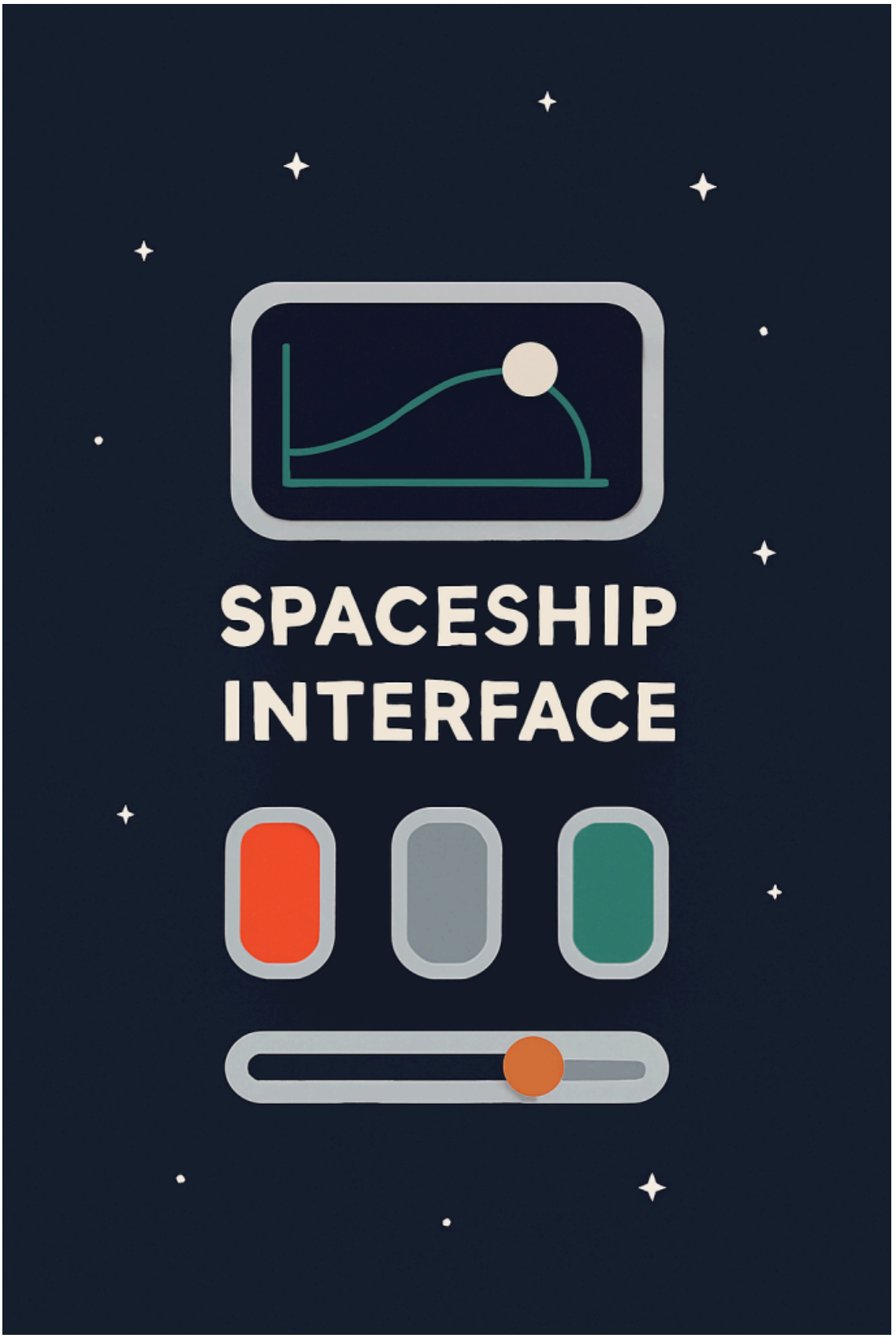
🚀 واجهة المركبة الفضائية

/schwila.com/arduino-led-switch-resistor-guide



✨ أردوينو الخاص بك سيؤدي دور البطولة في فيلم خيال علمي

🔌 **التحكم الكهربائي يبدأ الآن** بعد أن أتقنت أساسيات الكهرباء، حان الوقت للانتقال إلى التحكم بالأشياء باستخدام أردوينو. في هذا المشروع، ستقوم ببناء واجهة تحكم قد تبدو وكأنها جزء من مركبة فضائية في فيلم خيال علمي من سبعينيات القرن الماضي.

🛠️ **لوحة التحكم الفضائية** ستصمم لوحة تحكم أنيقة تحتوي على زر ومصابيح تضيء عند الضغط عليه. ويمكنك أن تقرر بنفسك: هل تعني هذه الأضواء "تشغيل المحرك الفائق" أم "إطلاق الليزر"؟

🟢 في البداية، سيكون مصباح LED أخضر مضاءً. وعندما يلتقط الأردوينو إشارة من الزر، ينطفئ الضوء الأخضر وتبدأ مصابيح أخرى في الوميض.

⚙️ **منطق الأردوينو الرقمي** المنافذ الرقمية في الأردوينو تقرأ حالتين فقط: وجود جهد كهربائي أو عدمه. يُطلق على هذا النوع من الإدخال اسم "رقمي" أو "ثنائي" لأنه يحتوي على حالتين فقط.

- **HIGH** تعني: "يوجد جهد هنا!"
- **LOW** تعني: "لا يوجد جهد على هذا المنفذ!"

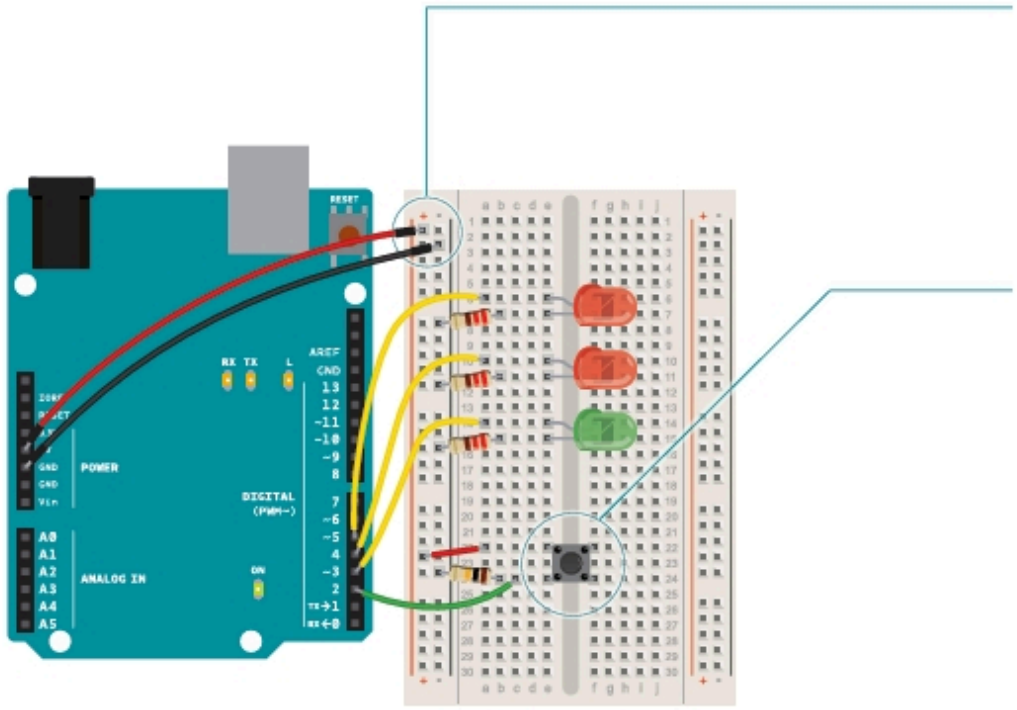
🧠 عندما تستخدم الأمر `digitalWrite()` لجعل منفذ إخراج في حالة HIGH، فأنت تقوم بتشغيله. وإذا قمت بقياس الجهد بين المنفذ والأرضي، ستحصل على 5 فولت. أما إذا جعلته LOW، فأنت تطفئه.

🔁 منافذ الأردوينو الرقمية يمكن أن تعمل كمدخلات أو كمخرجات. في الكود، ستقوم بتحديد وظيفتها حسب الحاجة:

- كمخرجات: لتشغيل مكونات مثل المصابيح
- كمدخلات: للتحقق مما إذا تم الضغط على الزر أم لا

💡 ملاحظة تقنية: بما أن المنفذين 0 و1 مخصصان للتواصل مع الحاسوب، من الأفضل أن تبدأ باستخدام المنفذ رقم 2.

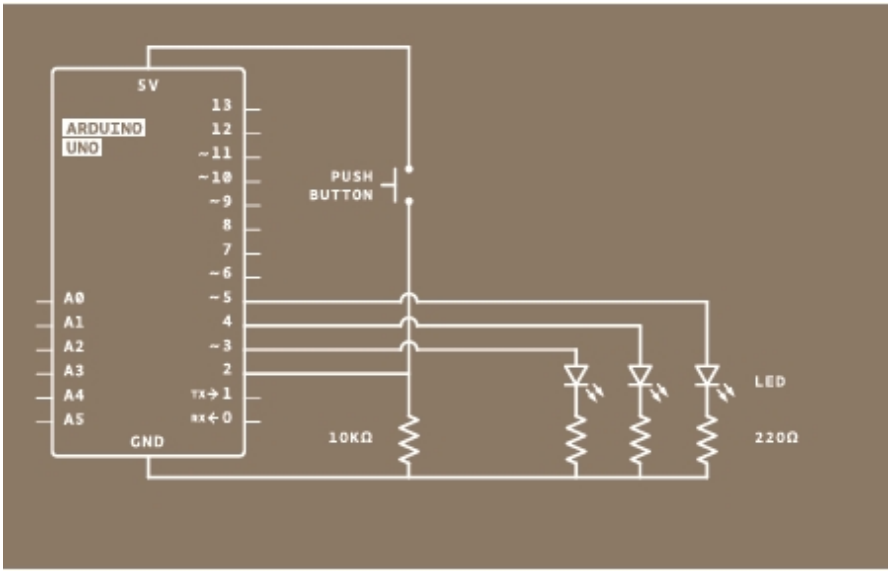
ابنِ الدارة الكهربائية



1 ـــ قم بتوصيل لوحة التجارب (Breadboard) بمنفذي 5 فولت والأرضي (GND) في الأردوينو، تمامًا كما في المشروع السابق. ضع مصباحي LED أحمر ومصباح LED أخضر على لوحة التجارب. اربط الكاثود (الساق القصيرة) لكل مصباح LED بالأرضي عبر مقاومة بقيمة 220 أوم. وصل الأنود (الساق الطويلة) للمصباح الأخضر بالمنفذ رقم 3. وصل الأنودين للمصابيح الحمراء بالمنفذين رقم 4 و5 على التوالي.

2 ـــ ضع الزر (Switch) على لوحة التجارب تمامًا كما فعلت في المشروع السابق. قم بتوصيل أحد طرفي الزر بالطاقة (5 فولت)، والطرف الآخر بالمنفذ الرقمي رقم 2 في الأردوينو. ستحتاج أيضًا إلى إضافة مقاومة بقيمة 10 كيلوأوم بين الأرضي والطرف الموصول بالمنفذ رقم 2.

هذه المقاومة تُعرف باسم "مقاومة السحب إلى الأسفل" (Pull-down resistor)، وهي تربط المنفذ بالأرضي عندما يكون الزر غير مضغوط، بحيث يقرأ الأردوينو الحالة LOW عندما لا يكون هناك جهد كهربائي قادم عبر الزر.



الكود البرمجي

```
int switchState = ;
int lastSwitchState = ;
bool isRunning = fals ;

void setu () {
  pinMode(13, INPUT);    // الزر
  pinMode(2, OUTPUT);    // LED الأول
  pinMode(3, OUTPUT);    // LED الثاني
  pinMode(4, OUTPUT);    // LED الثالث
}

void loo () {
  switchState = digitalRead(13);

  // التبديل عند الضغط (من LOW إلى HIGH)
  if (switchState == HIGH && lastSwitchState == LOW) {
    isRunning = !isRunning;
    delay(50); // منع الاهتزاز
  }

  lastSwitchState = switchState;

  // إذا كانت الحالة تشغيل، نفّذ التناوب
  if (isRunning) {
    digitalWrite(2, HIGH);
    delay(300);
    digitalWrite(2, LOW);

    digitalWrite(3, HIGH);
    delay(300);
    digitalWrite(3, LOW);

    digitalWrite(4, HIGH);
    delay(300);
    digitalWrite(4, LOW);
  }
}
```

🎯 ما وظيفة هذا الكود؟

عند الضغط على الزر:

- تبدأ المصابيح الثلاثة في الإضاءة **واحدة تلو الأخرى** بشكل متكرر.
- عند الضغط مرة أخرى: تتوقف المصابيح عن التناوب وتظل مطفأة.
- كل ضغطة تغيّر الحالة: تشغيل ↔ إيقاف.

🧱 شرح الكود خطوة بخطوة

1. تعريف المتغيرات

cpp

```
;int switchState =
;int lastSwitchState =
;bool isRunning = fals
```

- `switchState`: يقرأ حالة الزر الآن (مضغوط أو لا).
- `lastSwitchState`: يتذكر حالة الزر في الدورة السابقة.
- `isRunning`: هل المصابيح تعمل الآن؟ (true = نعم، false = لا).

2. إعداد المكونات

cpp

```
} ()void setu
;pinMode(13, INPUT)    // الزر موصول بالمسمار 13
pinMode(2, OUTPUT);    // LED الأول
pinMode(3, OUTPUT);    // LED الثاني
pinMode(4, OUTPUT);    // LED الثالث
```

نخبر Arduino أن الزر هو مدخل (INPUT)، والمصابيح هي مخرجات (OUTPUT).

3. التحكم في التشغيل والإيقاف

cpp

```
;switchState = digitalRead(13
```

نقرأ هل الزر مضغوط الآن.

cpp

```
} if (switchState == HIGH && lastSwitchState == LOW
;isRunning = !isRunning
;delay(50)
```

- إذا كان الزر **قد تم ضغطه الآن** ولم يكن مضغوطًا قبل لحظة، نغيّر حالة التشغيل.
- `!isRunning` يعني: إذا كانت تعمل، أوقفها؛ وإذا كانت متوقفة، شغّلها.
- `delay(50)` يمنع الاهتزازات غير المرغوبة (debounce).

4. تشغيل المصابيح بالتناوب

cpp

```
        } if (isRunning
;digitalWrite(2, HIGH)
          ;delay(300)
 ;digitalWrite(2, LOW)

;digitalWrite(3, HIGH)
          ;delay(300)
 ;digitalWrite(3, LOW)

;digitalWrite(4, HIGH)
          ;delay(300)
 ;digitalWrite(4, LOW)
```

- إذا كانت `isRunning` = true، تبدأ المصابيح في التناوب.
- كل مصباح يضيء لمدة 300 ميلي ثانية ثم ينطفئ، ويبدأ التالي.

5. تحديث حالة الزر

cpp

```
;lastSwitchState = switchStat
```

نحفظ حالة الزر الحالية لنقارنها في الدورة القادمة.

🧪 كيف تختبر الكود؟

1. اضغط على الزر مرة واحدة ← تبدأ المصابيح في التناوب.
2. اضغط مرة أخرى ← تتوقف المصابيح.
3. كرر الضغط لتبديل الحالة.

```
          } ()void setu
الأول pinMode(2, OUTPUT); // LED
الثاني pinMode(3, OUTPUT); // LED
الثالث pinMode(4, OUTPUT); // LED


          } ()void loo
;digitalWrite(2, HIGH)
          ;delay(300)
 ;digitalWrite(2, LOW)

;digitalWrite(3, HIGH)
          ;delay(300)
 ;digitalWrite(3, LOW)

;digitalWrite(4, HIGH)
          ;delay(300)
 ;digitalWrite(4, LOW)
```

🎯 ما وظيفة هذا الكود؟

- يشعل المصابيح الثلاثة (LEDs) بالتتابع.
- كل مصباح يضيء لمدة 300 ميلي ثانية ثم ينطفئ.
- الدورة تتكرر إلى ما لا نهاية.

🧱 شرح الكود خطوة بخطوة

1. `setup()`: إعداد المكونات

cpp

```
} ()void setu
الأول pinMode(2, OUTPUT); // LED
الثاني pinMode(3, OUTPUT); // LED
الثالث pinMode(4, OUTPUT); // LED
```

- نخبر Arduino أن المسامير 2، 3، و4 ستُستخدم لإخراج الكهرباء إلى المصابيح.
- هذا يعني أن Arduino يستطيع تشغيل أو إطفاء هذه المصابيح.

2. `loop()`: تشغيل المصابيح بالتتابع

cpp

```
الأول LED تشغيل // ;digitalWrite(2, HIGH
300 ميلي ثانية انتظر //           ;delay(300
الأول LED إطفاء //  ;digitalWrite(2, LOW
```

يشعل المصباح الأول، ينتظر قليلاً، ثم يطفئه.

cpp

```
الثاني LED تشغيل // ;digitalWrite(3, HIGH
                                ;delay(300
الثاني LED إطفاء //  ;digitalWrite(3, LOW
```

نفس الشيء للمصباح الثاني.

cpp

```
الثالث LED تشغيل // ;digitalWrite(4, HIGH
                                ;delay(300
الثالث LED إطفاء //  ;digitalWrite(4, LOW
```

ثم المصباح الثالث.

🔁 ماذا يحدث بعد ذلك؟

- بعد تشغيل المصابيح الثلاثة بالتتابع، تبدأ الدورة من جديد.
- هذا يحدث بشكل مستمر، لأن `loop()` تعني "كرّر إلى الأبد".

🧪 كيف تختبر الكود؟

1. صِل ثلاثة مصابيح إلى المسامير 2، 3، و4 مع مقاومات (220Ω).
2. حمّل الكود إلى لوحة Arduino.

```
int switchState =;
int lastSwitchState =;
bool ledsOn = fals;

void setu() {
  pinMode(13, INPUT);    // الزر
  pinMode(2, OUTPUT);    // LED الأول
  pinMode(3, OUTPUT);    // LED الثاني
  pinMode(4, OUTPUT);    // LED الثالث
}

void loo() {
  switchState = digitalRead(13);

  // التحقق من تغير الحالة (من LOW إلى HIGH)
  if (switchState == HIGH && lastSwitchState == LOW) {
    ledsOn = !ledsOn; // تبديل الحالة
    delay(50);        // تأخير لمنع الاهتزاز (debounce)
  }

  // تحديث حالة المصابيح حسب الحالة الحالية
  if (ledsOn) {
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(4, HIGH);
  } else {
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
  }

  lastSwitchState = switchState; // حفظ الحالة السابقة
```

🎯 ما وظيفة الكود؟

- عند الضغط على الزر: تشتعل المصابيح الثلاثة وتبقى مضيئة.
- عند الضغط مرة أخرى: تنطفئ المصابيح وتبقى مطفأة.
- كل ضغطة تغيّر الحالة: تشغيل ↔ إيقاف.

🧱 شرح الكود خطوة بخطوة

1. تعريف المتغيرات

cpp

```
;int switchState =
;int lastSwitchState =
;bool ledsOn = fals
```

- `switchState`: يقرأ حالة الزر الآن (هل هو مضغوط؟).
- `lastSwitchState`: يتذكر حالة الزر في الدورة السابقة.
- `ledsOn`: هل المصابيح تعمل الآن؟ (true = نعم، false = لا).

2. إعداد المكونات

cpp

```
} ()void setu
    // الزر    ;pinMode(13, INPUT)
الأول pinMode(2, OUTPUT);    // LED
الثاني pinMode(3, OUTPUT);    // LED
الثالث pinMode(4, OUTPUT);    // LED
```

نخبر Arduino أن الزر هو مدخل (INPUT)، والمصابيح هي مخرجات (OUTPUT).

3. قراءة حالة الزر وتبديل التشغيل

cpp

```
;switchState = digitalRead(13
```

نقرأ هل الزر مضغوط الآن.

cpp

```
} if (switchState == HIGH && lastSwitchState == LOW
        ;ledsOn = !ledsOn
        ;delay(50)
```

- إذا كان الزر **قد تم ضغطه الآن** ولم يكن مضغوطًا قبل لحظة، نغيّر حالة التشغيل.
- `!ledsOn` يعني: إذا كانت تعمل، أوقفها؛ وإذا كانت متوقفة، شغّلها.
- `delay(50)` يمنع الاهتزازات غير المرغوبة (debounce).

4. تشغيل أو إطفاء المصابيح حسب الحالة

cpp

```
    } if (ledsOn
;digitalWrite(2, HIGH)
;digitalWrite(3, HIGH)
;digitalWrite(4, HIGH)
    } else
;digitalWrite(2, LOW)
;digitalWrite(3, LOW)
;digitalWrite(4, LOW)
```

- إذا كانت `ledsOn = true`، نشغّل المصابيح.
- إذا كانت `ledsOn = false`، نطفئها.

5. تحديث حالة الزر

cpp

```
;lastSwitchState = switchStat
```

نحفظ حالة الزر الحالية لنقارنها في الدورة القادمة.

🧪 كيف تختبر الكود؟

1. صِل الزر إلى المسمار 13 مع مقاومة سحب (10kΩ إلى الأرض).
2. صِل المصابيح إلى المسامير 2، 3، و4 مع مقاومات (220Ω).
3. حمّل الكود إلى لوحة Arduino.
4. اضغط على الزر: المصابيح تشتعل.
5. اضغط مرة أخرى: المصابيح تنطفئ.

```
;int switchState =

} ()void setu
    ;pinMode(13, INPUT)    // الزر
pinMode(2, OUTPUT);    // LED الأول
pinMode(3, OUTPUT);    // LED الثاني
pinMode(4, OUTPUT);    // LED الثالث


} ()void loo
;switchState = digitalRead(13)

} if (switchState == HIGH)
// تتابع الإضاءة إلى ما لا نهاية طالما الزر مضغوط
;digitalWrite(2, HIGH)
;delay(300)
;digitalWrite(2, LOW)

;digitalWrite(3, HIGH)
;delay(300)
;digitalWrite(3, LOW)

;digitalWrite(4, HIGH)
;delay(300)
;digitalWrite(4, LOW)
} else {
// إذا لم يتم الضغط على الزر، أطفئ كل شيء
;digitalWrite(2, LOW)
;digitalWrite(3, LOW)
;digitalWrite(4, LOW)
{
```

🎯 ما وظيفة هذا الكود؟

- عند الضغط على الزر: تبدأ المصابيح الثلاثة في الإضاءة **واحدة تلو الأخرى**.
- تستمر الدورة **طالما الزر مضغوط**.
- عند رفع اليد عن الزر: تنطفئ المصابيح فورًا.

🧱 شرح الكود خطوة بخطوة

1. تعريف المتغير

cpp

```
;int switchState =
```

هذا المتغير سيحفظ حالة الزر: هل هو مضغوط (`HIGH`) أم غير مضغوط (`LOW`)؟

2. إعداد المكونات في `setup()`

cpp

```
                  الزر //    ;pinMode(13, INPUT
             الأول pinMode(2, OUTPUT);    // LE
            الثاني pinMode(3, OUTPUT);    // LE
            الثالث pinMode(4, OUTPUT);    // LE
```

- نخبر Arduino أن المسمار 13 هو مدخل (لزر الضغط).
- والمسامير 2، 3، و4 هي مخارج (للمصابيح).

3. قراءة حالة الزر في `loop()`

cpp

```
;switchState = digitalRead(13
```

نقرأ هل الزر مضغوط الآن أم لا.

4. إذا كان الزر مضغوطًا (`HIGH`)

cpp

```
} if (switchState == HIGH
// تشغيل المصابيح بالتتابع
```

إذا تم الضغط على الزر، تبدأ المصابيح في الإضاءة واحدة تلو الأخرى.

cpp

```
;digitalWrite(2, HIGH  // تشغيل LED الأول
;delay(300             // انتظر 300 ميلي ثانية
;digitalWrite(2, LOW   // إطفاء LED الأول

;digitalWrite(3, HIGH  // تشغيل LED الثاني
;delay(300
;digitalWrite(3, LOW   // إطفاء LED الثاني

;digitalWrite(4, HIGH  // تشغيل LED الثالث
;delay(300
;digitalWrite(4, LOW   // إطفاء LED الثالث
```

كل مصباح يضيء لمدة قصيرة ثم ينطفئ، ويبدأ التالي.

5. إذا لم يتم الضغط على الزر (`LOW`)

cpp

```
;digitalWrite(2, LOW
;digitalWrite(3, LOW
;digitalWrite(4, LOW
```

نطفئ جميع المصابيح فورًا.

🧪 كيف تختبر الكود؟

1. صِل الزر إلى المسمار 13 مع مقاومة سحب (10kΩ إلى الأرض).
2. صِل المصابيح إلى المسامير 2، 3، و4 مع مقاومات (220Ω).
3. حمّل الكود إلى لوحة Arduino.
4. اضغط على الزر: راقب المصابيح وهي تتناوب.

بعض الملاحظات قبل أن تبدأ

> كل برنامج أردوينو يحتوي على وظيفتين رئيسيتين. **الوظائف (Functions)** هي أجزاء من البرنامج تقوم بتنفيذ أوامر محددة. لكل وظيفة اسم فريد، ويتم "استدعاؤها" عند الحاجة.
>
> الوظيفتان الضروريتان في برنامج الأردوينو هما:
>
> - `setup()`
> - `loop()`
>
> يجب **تصريح** هذه الوظائف، أي إخبار الأردوينو بما ستقوم به كل وظيفة. يتم التصريح بوظيفتي `setup()` و `loop()` كما هو موضح في الجانب الأيمن.
>
> في هذا البرنامج، ستقوم بإنشاء **متغير (Variable)** قبل الدخول إلى الجزء الرئيسي من البرنامج. المتغيرات هي أسماء تُعطى لمواقع في ذاكرة الأردوينو، لتتمكن من تتبع ما يحدث. يمكن أن تتغير هذه القيم حسب تعليمات البرنامج.
>
> يُفضل أن تكون أسماء المتغيرات **معبرة عن القيمة التي تخزنها**. على سبيل المثال، المتغير المسمى `switchState` يوضح أنه يخزن حالة الزر. أما المتغير المسمى `x` فلا يوضح الكثير عن محتواه.

لنبدأ البرمجة

> لإنشاء متغير، تحتاج إلى تحديد نوعه. نوع البيانات **int** يُستخدم لتخزين عدد صحيح (ويُسمى أيضًا عددًا صحيحًا)، أي رقم بدون فاصلة عشرية. عند التصريح عن متغير، غالبًا ما تعطيه قيمة ابتدائية أيضًا. يجب أن تنتهي كل عبارة تصريح بفاصلة منقوطة `;`

تهيئة وظيفة المنافذ

> تعمل الدالة `setup()` مرة واحدة فقط، عند تشغيل الأردوينو لأول مرة. في هذه الدالة، تقوم بتهيئة المنافذ الرقمية لتكون إما **مدخلات (inputs)** أو **مخرجات (outputs)**، وذلك باستخدام دالة تُسمى `pinMode()`. المنافذ المتصلة بمصابيح LED ستكون **مخرجات (OUTPUTs)**، بينما المنفذ المتصل بالزر سيكون **مدخلاً (INPUT)**.

أنشئ دالة التكرار (loop)

> تعمل الدالة `loop()` بشكل مستمر بعد انتهاء تنفيذ الدالة `setup()`. في `loop()`، تقوم بفحص وجود الجهد الكهربائي على المنافذ المدخلة، وتشغيل أو إيقاف المنافذ المخرجة.
>
> لفحص مستوى الجهد على منفذ رقمي مدخل، تستخدم الدالة `digitalRead()`، والتي تتحقق من وجود الجهد على المنفذ المحدد.
>
> لكي تعرف أي منفذ يجب فحصه، تحتاج دالة `digitalRead()` إلى **وسيط (argument)**. الوسيط هو معلومة تمررها إلى الدالة لتخبرها كيف تؤدي وظيفتها.
>
> على سبيل المثال، `digitalRead()` تحتاج إلى وسيط واحد: **رقم المنفذ الذي يجب فحصه**. في برنامجك، ستقوم `digitalRead()` بفحص حالة... *(ويُفترض أن يكمل النص بذكر رقم المنفذ، مثل pin 2)*.

{ الأقواس المعقوفة } أي كود تكتبه داخل الأقواس المعقوفة سيتم تنفيذه عندما يتم استدعاء الدالة.

void setup(){ // هنا تضع التعليمات التي تُنفذ مرة واحدة عند تشغيل الأردوينو } void loop(){ // هنا تضع التعليمات التي تُنفذ بشكل متكرر طالما أن الأردوينو يعمل }

انتبه لحساسية حالة الأحرف (Case Sensitivity) في الكود الخاص بك. على سبيل المثال، `pinMode` هو اسم أمر صحيح، لكن كتابة `pinmode` ستؤدي إلى ظهور خطأ.

```
} ()void loo
// قراءة حالة الزر
;switchState = digitalRead(2)

} if (switchState == HIGH)
// إذا تم الضغط على الزر، شغّل المصابيح
الأخضر digitalWrite(3, HIGH); // LED
الأحمر الأول digitalWrite(4, HIGH); // LED
الأحمر الثاني digitalWrite(5, HIGH); // LED
} else {
// إذا لم يتم الضغط على الزر، أطفئ المصابيح
;digitalWrite(3, LOW)
;digitalWrite(4, LOW)
;digitalWrite(5, LOW)
{
```

التعليقات (Comments) إذا أردت يومًا أن تُدرج لغة طبيعية داخل برنامجك، يمكنك إضافة تعليق. التعليقات هي ملاحظات تكتبها لنفسك، ويتجاهلها الحاسوب تمامًا. لكتابة تعليق، أضف علامتي // سيتجاهل الحاسوب كل ما يأتي بعد هاتين العلامتين على نفس السطر.

```
}()void loo
// هذا تعليق ;switchState = digitalRead(2)
```

The if statement عبارة if الشرطية

> المنفذ 2 (pin 2) وتخزين القيمة في المتغير `switchState`. إذا كان هناك جهد كهربائي على المنفذ عند استدعاء `()digitalRead`، فإن المتغير `switchState` سيأخذ القيمة **HIGH** (أو 1). وإذا لم يكن هناك جهد على المنفذ، فسيأخذ `switchState` القيمة **LOW** (أو 0).
>
> في الأعلى، استخدمت الكلمة **if** للتحقق من حالة شيء ما (وهو موضع الزر). تُستخدم عبارة `()if` في البرمجة لمقارنة شيئين، وتحديد ما إذا كانت المقارنة **صحيحة (true)** أو **خاطئة (false)**. ثم تقوم بتنفيذ الأوامر التي تطلبها منها.
>
> عند مقارنة شيئين في البرمجة، تستخدم علامتي التساوي `==`. إذا استخدمت علامة تساوي واحدة فقط `=`, فستقوم **بإعطاء قيمة**، وليس **بالمقارنة**.

ابنِ مركبتك الفضائية Build up your spaceship

> `()digitalWrite` هو الأمر الذي يسمح لك بإرسال **5 فولت** أو **0 فولت** إلى منفذ إخراج. تأخذ `()digitalWrite` وسيطين (arguments):
>
> - أي منفذ تريد التحكم فيه
> - والقيمة التي تريد تعيينها لذلك المنفذ: **HIGH** أو **LOW**
>
> إذا أردت تشغيل مصابيح LED الحمراء وإطفاء المصباح الأخضر داخل عبارة `()if`، فسيبدو الكود كالتالي.
>
> إذا قمت بتشغيل برنامجك الآن، ستتغير الأضواء عندما تضغط على الزر. هذا أمر رائع، لكن يمكنك إضافة بعض التعقيد إلى البرنامج للحصول على مخرجات أكثر إثارة للاهتمام.

لقد أخبرت الأردوينو بما يجب فعله عندما يكون الزر مفتوحًا. الآن، عرّف ما الذي يحدث عندما يكون الزر مغلقًا. عبارة `()if` تحتوي على جزء اختياري يُسمى `else`، ويسمح بحدوث شيء ما إذا لم يتحقق الشرط الأصلي.

في هذه الحالة، بما أنك تحققت من أن الزر في حالة **LOW**، اكتب الكود الخاص بالحالة **HIGH** بعد عبارة `else`.

لكي تجعل مصابيح LED الحمراء تومض عند الضغط على الزر، ستحتاج إلى تشغيل وإطفاء الأضواء داخل عبارة `else` التي كتبتها للتو. للقيام بذلك، غيّر الكود ليبدو كما يلي.

الآن سيقوم برنامجك بوميض مصابيح LED الحمراء عندما يتم الضغط على زر التبديل.

بعد ضبط حالة مصابيح LED، سترغب في أن يتوقف الأردوينو للحظة قبل تغييرها مرة أخرى. إذا لم تنتظر، ستتغير الأضواء بسرعة كبيرة لدرجة أنها ستبدو وكأنها خافتة قليلاً، وليس مضاءة ومطفأة. وذلك لأن الأردوينو يمر عبر دالة `()loop` آلاف المرات في كل ثانية، وسيتم تشغيل وإطفاء الـ LED بسرعة تفوق قدرتنا على الإدراك.

تسمح لك دالة `()delay` بإيقاف تنفيذ الأردوينو لأي شيء لفترة زمنية محددة. تأخذ `()delay` وسيطًا يحدد عدد **الميلي ثانية** قبل تنفيذ مجموعة التعليمات التالية. يوجد 1000 ميلي ثانية في الثانية الواحدة. `delay(250)` سيؤدي إلى توقف لمدة **ربع ثانية**.

```
} if (switchState == LOW // الزر غير مضغوط
  ;digitalWrite(3, HIGH) // تشغيل مصباح LED الأخضر
  ;digitalWrite(4, LOW)  // إطفاء مصباح LED الأحمر الأول
  ;digitalWrite(5, LOW)  // إطفاء مصباح LED الأحمر الثاني
```

> قد يكون من المفيد كتابة تدفق برنامجك باستخدام **الكود الكاذب (pseudocode)**: وهو طريقة لوصف ما تريد أن يفعله البرنامج بلغة بسيطة، لكنها منظمة بطريقة تجعل من السهل تحويلها إلى برنامج حقيقي.
>
> في هذه الحالة، ستقوم بتحديد ما إذا كانت قيمة `switchState` هي **HIGH** (أي أن الزر مضغوط) أم لا. إذا تم الضغط على الزر، ستقوم بإطفاء مصباح LED الأخضر وتشغيل المصابيح الحمراء.

```
} else { // الزر مضغوط
  ;digitalWrite(3, LOW)  // إطفاء مصباح LED الأخضر
  ;digitalWrite(4, LOW)  // إطفاء مصباح LED الأحمر الأول
  ;digitalWrite(5, HIGH) // تشغيل مصباح LED الأحمر الثاني

  ;delay(250) // الانتظار لمدة ربع ثانية

  // تبديل حالة المصابيح
  ;digitalWrite(4, HIGH) // تشغيل مصباح LED الأحمر الأول
  ;digitalWrite(5, LOW)  // إطفاء مصباح LED الأحمر الثاني

  ;delay(250) // الانتظار لمدة ربع ثانية

  // العودة إلى بداية الحلقة loop
```

بمجرد برمجة الأردوينو، يجب أن ترى المصباح الأخضر يضيء. وعندما تضغط على الزر، ستبدأ المصابيح الحمراء في الوميض، وسينطفئ المصباح الأخضر.

جرّب تغيير مدة دالتي `delay()`، ولاحظ ما يحدث للأضواء، وكيف يتغير تفاعل النظام حسب سرعة الوميض.

عند استدعاء دالة `delay()` في برنامجك، فإنها توقف تنفيذ أي وظائف أخرى. لن يتم قراءة أي مستشعرات حتى تنقضي تلك الفترة الزمنية.

وعلى الرغم من أن التأخيرات غالبًا ما تكون مفيدة، إلا أنه عند تصميم مشاريعك الخاصة، تأكد من أنها لا تتداخل بشكل غير ضروري مع واجهة الاستخدام.

فكر

كيف تجعل مصابيح LED الحمراء تومض عند بدء تشغيل البرنامج؟

وكيف يمكنك إنشاء واجهة أكبر أو أكثر تعقيدًا لمغامراتك بين النجوم باستخدام مصابيح LED والمفاتيح؟

عندما تبدأ في إنشاء واجهة لمشروعك، فكّر في توقعات الأشخاص أثناء استخدامها. عندما يضغطون على زر، هل سيرغبون في الحصول على استجابة فورية؟ هل يجب أن يكون هناك تأخير بين فعلهم وما يفعله الأردوينو؟

حاول أن تضع نفسك مكان مستخدم مختلف أثناء التصميم، وانظر ما إذا كانت توقعاتك تتطابق مع واقع مشروعك.

طريقة قراءة **رموز ألوان المقاومات** (Resistor Color Codes)

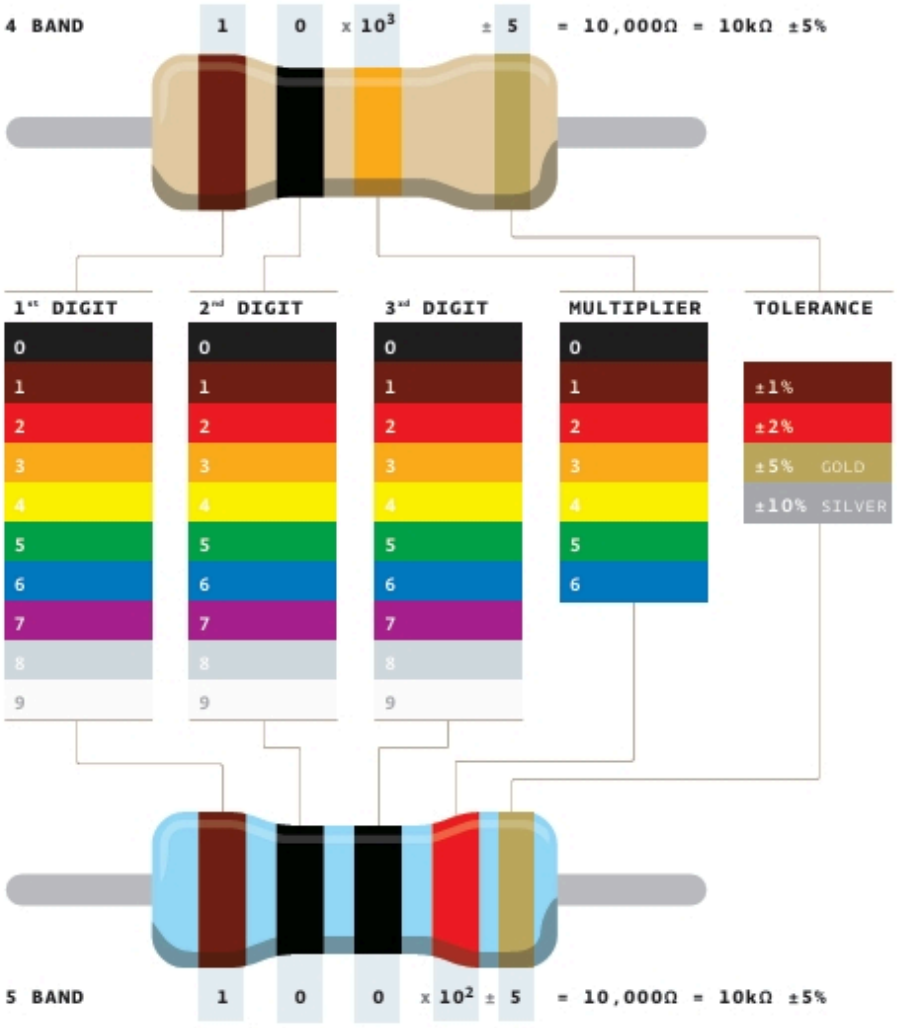
يتم تحديد قيم المقاومات باستخدام شرائط ملونة، وفقًا لرمز تم تطويره في عشرينيات القرن الماضي، حين كان من الصعب كتابة الأرقام على أجسام صغيرة جدًا.

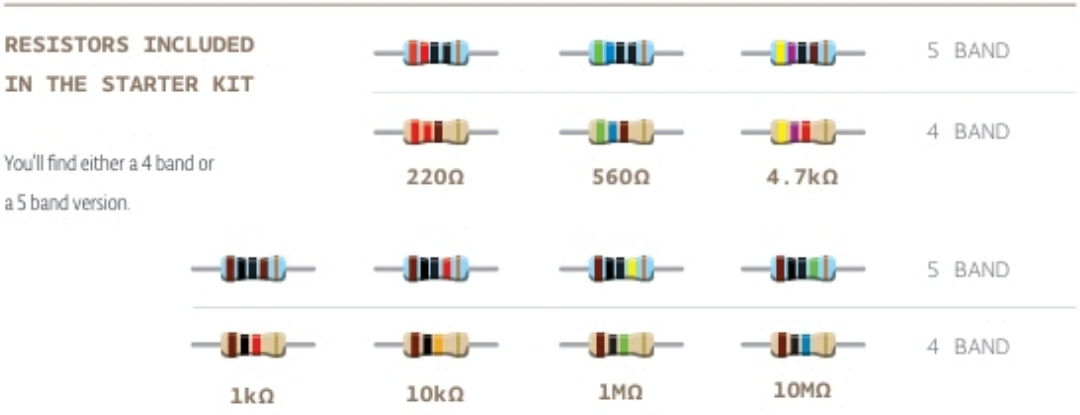
كل لون يُقابل رقمًا معينًا، كما هو موضح في الجدول أدناه. كل مقاومة تحتوي إما على 4 أو 5 شرائط.

في النوع المكوّن من 4 شرائط:

- الشريطان الأولان يُشيران إلى أول رقمين من قيمة المقاومة
- الشريط الثالث يُشير إلى عدد الأصفار التي تلي (من الناحية التقنية، يُمثل قوة الرقم عشرة)
- الشريط الأخير يُحدد نسبة التفاوت (الدقة)

في المثال أدناه، يشير اللون الذهبي إلى أن قيمة المقاومة يمكن أن تكون **10 كيلو أوم ±5%**.


4 BAND
1
0
x 10³
± 5
= 10,000Ω = 10kΩ ±5%
1ST DIGIT
2ND DIGIT
3RD DIGIT
MULTIPLIER
TOLERANCE
0
1
2
3
4
5
6
7
8
9
±1%
±2%
±5% GOLD
±10% SILVER
5 BAND
1
0
0
x 10² ± 5
= 10,000Ω = 10kΩ ±5%


RESISTORS INCLUDED IN THE STARTER KIT
You'll find either a 4 band or a 5 band version.
5 BAND
4 BAND
220Ω
560Ω
4.7kΩ
5 BAND
4 BAND
1kΩ
10kΩ
1MΩ
10MΩ

schwila