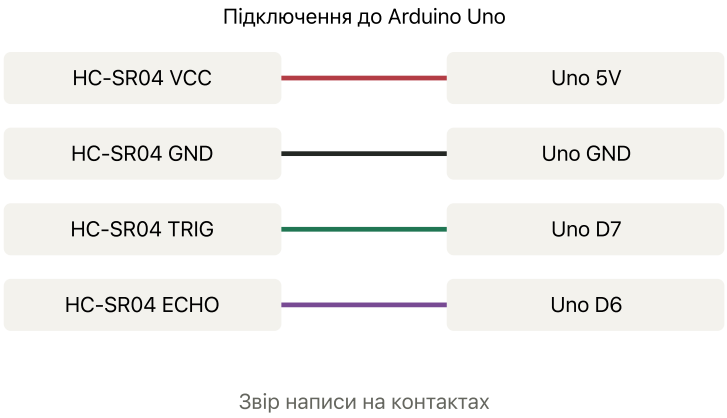
УРОК 07 · КРОК 1 / 8

Вимірюємо відстань звуком

1. Візьми Uno, USB, HC-SR04, 4 дроти «мама–тато», лінійку й плаский картон.
2. Знайди два круглі елементи датчика: передавач та приймач.

Передавач надсилає ультразвук, приймач ловить відбиття. Датчик вимірює час до повернення звуку; більший час означає більшу відстань.

Іди по черзі. На кроках із кодом подано повну програму: замінюй нею весь скетч. Не додавай новий код під старим. Змінюй один параметр, запускай і порівнюй результат.



Звір написи на контактах

TRIG запускає, ECHO відповідає

1. Знайди на датчику контакти `VCC`, TRIG, ECHO та `GND`.

HC-SR04 — назва моделі датчика відстані. Пін — металевий контакт для підключення дроту. Напис «HC-SR04 TRIG» на схемі означає контакт TRIG саме цього датчика.

TRIG — команда почати вимірювання

TRIG — скорочення від *trigger*, «запуск». Це вхід датчика: Arduino через `D7` подає на нього короткий електричний імпульс. Після цього датчик надсилає ультразвук.

Напрямок сигналу: Uno `D7` → TRIG датчика.

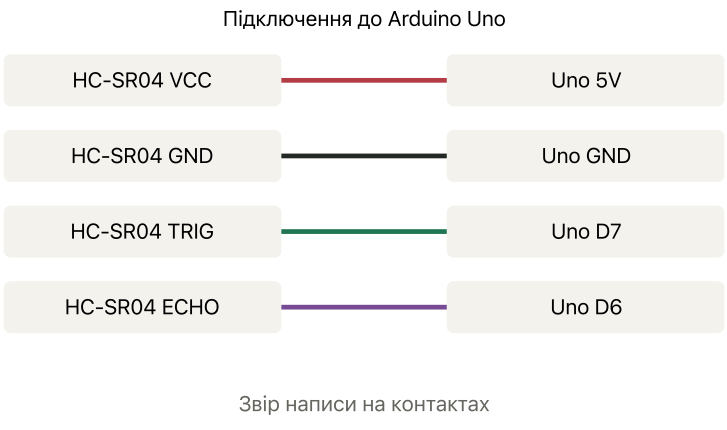
ECHO — відповідь датчика

ECHO означає «відлуння». Це вихід датчика: він подає імпульс на `D6` Arduino. Його тривалість відповідає часу проходження звуку до перешкоди й назад. Arduino вимірює тривалість і обчислює відстань.

Напрямок сигналу: ECHO датчика → Uno `D6`.

`VCC` та `GND` — живлення

`VCC` → `5V` **Uno** подає живлення. `GND` → `GND` **Uno** з'єднує спільний нуль — точку відліку напруги та зворотний шлях струму. Через ці два контакти датчик живиться; команди й відповідь проходять через TRIG та ECHO.



Звір написи на контактах

УРОК 07 · КРОК 3 / 8

Чотири дроти без екрана

1. Від'єднай USB.
2. Підключи `VCC` → `5V`, `GND` → `GND`, TRIG → `D7`, ECHO → `D6`.
3. Звір підключення зі схемою й під'єднай USB.
4. Заміни весь скетч кодом цього кроку та завантаж.

Датчик живиться від Uno через USB. На цьому етапі на самій платі немає показу відстані — відкриємо монітор.

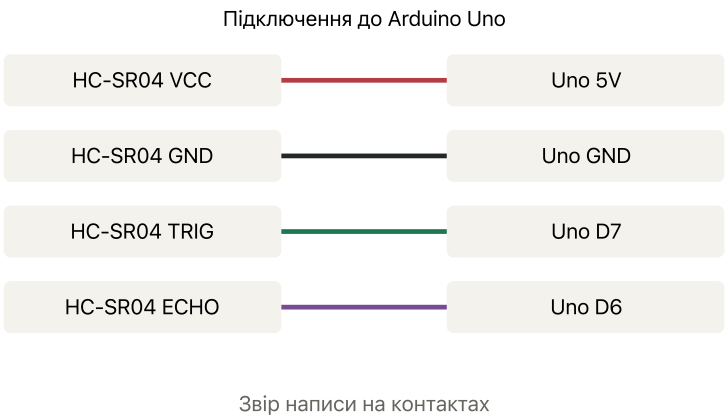
Що можна змінити

READ_MS — пауза між вимірюваннями; поки 500 мс. Окремих бібліотек цей код не потребує.

Чому один пін OUTPUT, а інший INPUT

`TRIG_PIN = 7` — це номер контакту `D7` на Arduino. `pinMode(TRIG_PIN, OUTPUT)` налаштовує його на надсилання сигналу. Для датчика TRIG є входом, а для Arduino підключений до нього `D7` — виходом.

`ECHO_PIN = 6` відповідає `D6`. `pinMode(ECHO_PIN, INPUT)` дозволяє Arduino читати відповідь датчика. INPUT і OUTPUT у коді описують напрямок сигналу саме для Arduino.



Крок 3: код · 1/1

Це одна програма. Частини коду читаємо по черзі.

```
#include <Arduino.h>
const byte TRIG_PIN = 7;
const byte ECHO_PIN = 6;
const unsigned long READ_MS = 500; // Пауза між показами

float readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW); // Готуємо короткий імпульс
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH); // Запускаємо вимірювання
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000UL);
  float cm = echoUs / 58.0; // Шлях туди й назад уже врахований

  if (echoUs == 0 || cm < 2 || cm > 400) {
    return -1; // Немає надійного вимірювання
  }

  return cm;
}

void setup() {
  pinMode(TRIG_PIN, OUTPUT); // TRIG — команда датчику
  pinMode(ECHO_PIN, INPUT); // ECHO — відповідь датчика
  digitalWrite(TRIG_PIN, LOW);
  Serial.begin(115200); // Така сама швидкість у моніторі
}

void loop() {
  float cm = readDistanceCm();

  if (cm < 0) {
    Serial.println("No echo");
  } else {
    Serial.print("Distance: ");
    Serial.print(cm, 1);
    Serial.println(" cm");
  }

  delay(READ_MS);
}
```

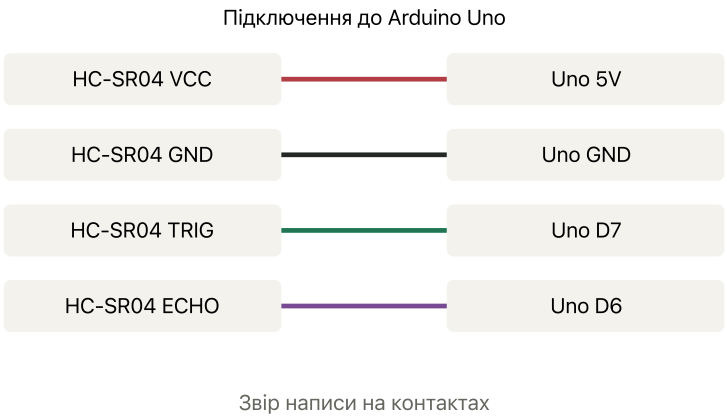
Де побачити число

1. В Arduino IDE вибери плату Arduino Uno та її порт.
2. Завантаж код.
3. Відкрий «Інструменти → Монітор порту» (Tools → Serial Monitor).
4. Унизу вибери 115200 baud — як у Serial.begin(115200).
5. Поле надсилання залиш порожнім.
6. У Wokwi запусти симуляцію: внизу з'явиться Serial Monitor.

При картоні перед датчиком з'являються рядки на кшталт Distance: 20.0 cm. Serial.print додає текст у той самий рядок; Serial.println завершує рядок.

Що можна змінити

READ_MS = 500 дає приблизно два рядки за секунду. Швидкість монітора має збігатися з 115200 у коді.



Крок 4: код · 1/1

Це одна програма. Частини коду читаємо по черзі.

```
#include <Arduino.h>
const byte TRIG_PIN = 7;
const byte ECHO_PIN = 6;
const unsigned long READ_MS = 500; // Пауза між показами

float readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW); // Готуємо короткий імпульс
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH); // Запускаємо вимірювання
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000UL);
  float cm = echoUs / 58.0; // Шлях туди й назад уже врахований

  if (echoUs == 0 || cm < 2 || cm > 400) {
    return -1; // Немає надійного вимірювання
  }

  return cm;
}

void setup() {
  pinMode(TRIG_PIN, OUTPUT); // TRIG — команда датчику
  pinMode(ECHO_PIN, INPUT); // ECHO — відповідь датчика
  digitalWrite(TRIG_PIN, LOW);
  Serial.begin(115200); // Така сама швидкість у моніторі
}

void loop() {
  float cm = readDistanceCm();

  if (cm < 0) {
    Serial.println("No echo");
  } else {
    Serial.print("Distance: ");
    Serial.print(cm, 1);
    Serial.println(" cm");
  }

  delay(READ_MS);
}
```

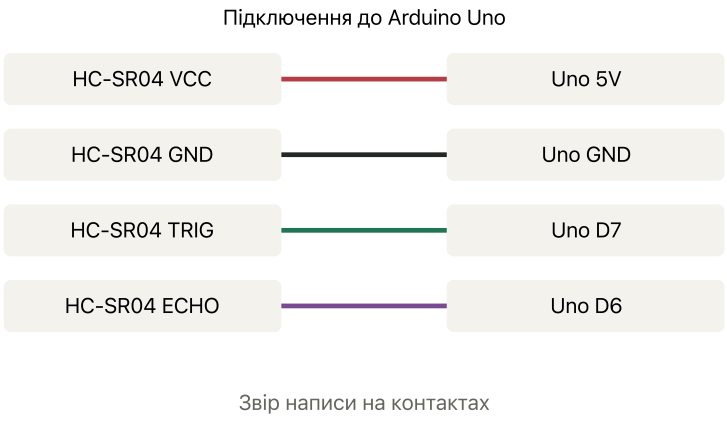
Порівняй 10, 20 та 40 см

1. Постав картон паралельно передній поверхні датчика на 10 см.
2. Повтори на 20 і 40 см.
3. Порівняй число в моніторі з лінійкою; не очікуй ідеальної точності.

Наближаєш картон — число меншає. У Wokwi натисни HC-SR04 і змінюй Distance. Обидва круглі елементи мають дивитися на перешкоду.

Що можна змінити

READ_MS: порівняй 200 та 1000 мс. Для цих дослідів тримай паузу щонайменше 100 мс.



Крок 5: код · 1/1

Це одна програма. Частини коду читаємо по черзі.

```
#include <Arduino.h>
const byte TRIG_PIN = 7;
const byte ECHO_PIN = 6;
const unsigned long READ_MS = 500; // Пауза між показами

float readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW); // Готуємо короткий імпульс
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH); // Запускаємо вимірювання
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000UL);
  float cm = echoUs / 58.0; // Шлях туди й назад уже врахований

  if (echoUs == 0 || cm < 2 || cm > 400) {
    return -1; // Немає надійного вимірювання
  }

  return cm;
}

void setup() {
  pinMode(TRIG_PIN, OUTPUT); // TRIG — команда датчику
  pinMode(ECHO_PIN, INPUT); // ECHO — відповідь датчика
  digitalWrite(TRIG_PIN, LOW);
  Serial.begin(115200); // Така сама швидкість у моніторі
}

void loop() {
  float cm = readDistanceCm();

  if (cm < 0) {
    Serial.println("No echo");
  } else {
    Serial.print("Distance: ");
    Serial.print(cm, 1);
    Serial.println(" cm");
  }

  delay(READ_MS);
}
```

Як час стає сантиметрами

1. Знайди delayMicroseconds(10), pulseIn та echoUs / 58.0.
2. Простеж: імпульс TRIG запускає датчик; pulseIn вимірює тривалість HIGH на ECHO; ділення перетворює її на сантиметри.

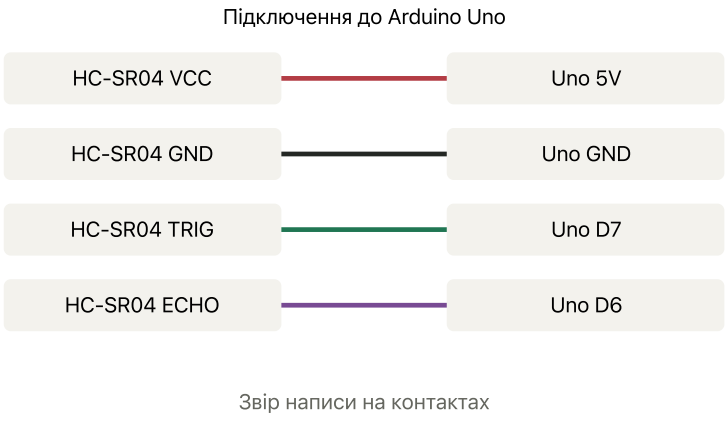
580 мкс / 58 ≈ 10 см. Звук іде до предмета й назад; коефіцієнт 58 це враховує. 25000UL — максимум очікування у мікросекундах, а не відстань.

Що можна змінити

Не змінюй 58.0 для «підгонки» показів. Змінюй READ_MS, щоб дослідити частоту оновлення.

Які команди працюють із пінами

1. `digitalWrite(TRIG_PIN, LOW)` спочатку встановлює на **`D7`** низький рівень — приблизно 0 V.
2. `digitalWrite(TRIG_PIN, HIGH)` встановлює високий рівень — приблизно 5 V на Uno.
3. `delayMicroseconds(10)` утримує його 10 мікросекунд, а наступний LOW завершує імпульс запуску.
4. `pulseIn(ECHO_PIN, HIGH, 25000UL)` вимірює тривалість високого рівня на **`D6`** у мікросекундах. Якщо імпульс не завершився за відведений час, повертається 0.
5. `echoUs / 58.0` перетворює цю тривалість на сантиметри.



Звір написи на контактах

Крок 6: код · 1/1

Це одна програма. Частини коду читаємо по черзі.

```
#include <Arduino.h>
const byte TRIG_PIN = 7;
const byte ECHO_PIN = 6;
const unsigned long READ_MS = 500; // Пауза між показами

float readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW); // Готуємо короткий імпульс
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH); // Запускаємо вимірювання
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000UL);
  float cm = echoUs / 58.0; // Шлях туди й назад уже врахований

  if (echoUs == 0 || cm < 2 || cm > 400) {
    return -1; // Немає надійного вимірювання
  }

  return cm;
}

void setup() {
  pinMode(TRIG_PIN, OUTPUT); // TRIG — команда датчику
  pinMode(ECHO_PIN, INPUT); // ECHO — відповідь датчика
  digitalWrite(TRIG_PIN, LOW);
  Serial.begin(115200); // Така сама швидкість у моніторі
}

void loop() {
  float cm = readDistanceCm();

  if (cm < 0) {
    Serial.println("No echo");
  } else {
    Serial.print("Distance: ");
    Serial.print(cm, 1);
    Serial.println(" cm");
  }

  delay(READ_MS);
}
```

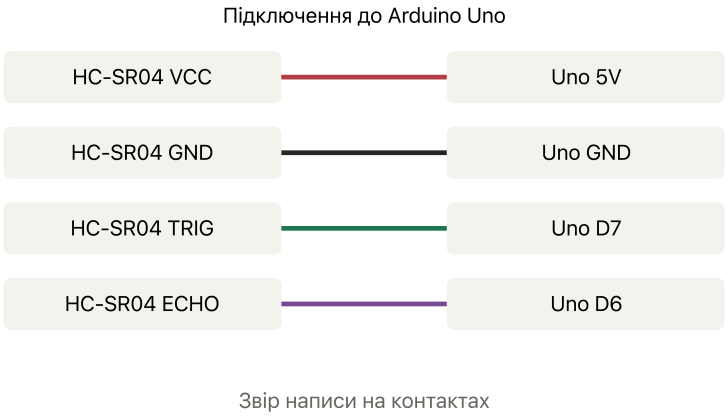
No echo — це не нуль

1. На справжній схемі поверни картон убік або прибери його.
2. Простеж покази.
3. Якщо звук усе ще відбивається від стіни — це інше вимірювання, а не помилка.

Коли відлуння не отримано вчасно або показ поза 2–400 см, бачиш No echo. -1 — внутрішня позначка помилки. Wokwi-повзунок 2–400 см не відтворює всі фізичні причини втрати відлуння.

Що можна змінити

READ_MS лиш 500. Не замінюй No echo на 0 cm — це змінило б зміст показу.



Звір написи на контактах

Крок 7: код · 1/1

Це одна програма. Частини коду читаємо по черзі.

```
#include <Arduino.h>
const byte TRIG_PIN = 7;
const byte ECHO_PIN = 6;
const unsigned long READ_MS = 500; // Пауза між показами

float readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW); // Готуємо короткий імпульс
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH); // Запускаємо вимірювання
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000UL);
  float cm = echoUs / 58.0; // Шлях туди й назад уже врахований

  if (echoUs == 0 || cm < 2 || cm > 400) {
    return -1; // Немає надійного вимірювання
  }

  return cm;
}

void setup() {
  pinMode(TRIG_PIN, OUTPUT); // TRIG — команда датчику
  pinMode(ECHO_PIN, INPUT); // ECHO — відповідь датчика
  digitalWrite(TRIG_PIN, LOW);
  Serial.begin(115200); // Така сама швидкість у моніторі
}

void loop() {
  float cm = readDistanceCm();

  if (cm < 0) {
    Serial.println("No echo");
  } else {
    Serial.print("Distance: ");
    Serial.print(cm, 1);
    Serial.println(" cm");
  }

  delay(READ_MS);
}
```

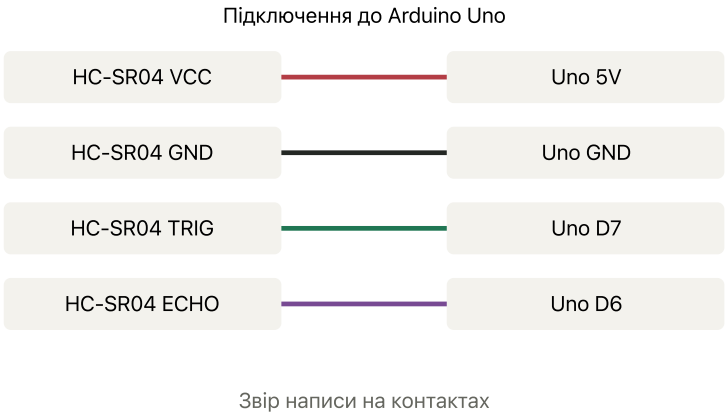
Вимірювання раз на секунду

1. Задай READ_MS = 1000.
2. Виміряй відстань до картону на 10, 20 і 40 см.
3. Збережи скетч Distance_Serial.

Приблизно раз на секунду з’являється новий підписаний рядок. Якщо символи незрозумілі — звір 115200; якщо порожньо — плату, порт, кабель даних і завантаження.

Що можна змінити

READ_MS — 1000. Порівняй, що змінюється: частота рядків, а не сама одиниця вимірювання.



Звір написи на контактах

Крок 8: код · 1/1

Це одна програма. Частини коду читаємо по черзі.

```
#include <Arduino.h>
const byte TRIG_PIN = 7;
const byte ECHO_PIN = 6;
const unsigned long READ_MS = 500; // Пауза між показами

float readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW); // Готуємо короткий імпульс
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH); // Запускаємо вимірювання
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long echoUs = pulseIn(ECHO_PIN, HIGH, 25000UL);
  float cm = echoUs / 58.0; // Шлях туди й назад уже врахований

  if (echoUs == 0 || cm < 2 || cm > 400) {
    return -1; // Немає надійного вимірювання
  }

  return cm;
}

void setup() {
  pinMode(TRIG_PIN, OUTPUT); // TRIG — команда датчику
  pinMode(ECHO_PIN, INPUT); // ECHO — відповідь датчика
  digitalWrite(TRIG_PIN, LOW);
  Serial.begin(115200); // Така сама швидкість у моніторі
}

void loop() {
  float cm = readDistanceCm();

  if (cm < 0) {
    Serial.println("No echo");
  } else {
    Serial.print("Distance: ");
    Serial.print(cm, 1);
    Serial.println(" cm");
  }

  delay(READ_MS);
}
```

Наступний урок: Відстань на власному екрані →