Парктронік зі звуковим сигналом

Наближаємо предмет — світлодіод і пищалка подають дедалі частіші сигнали. Зовсім близько — світло та звук безперервні.

- Схема з уроку 06-01: Arduino Uno, HC-SR04, макетна плата, червоний LED, резистор 220 Ом, USB і дроти.
- У Wokwi — пасивний п’єзозумер; на платі можна використати наш активний модуль зумера.
- Дроти, макетна плата, лінійка та плаский картон.

Використаємо вимірювання з уроку 06-01 й додамо пищалку. Світлодіод та його підключення залишаються.

Перед зміною схеми від’єднай живлення. У Wokwi зупини симуляцію. Справжню схему перевір разом з учителем перед підключенням USB.

Відлуння і відстань

Датчик посилає короткий ультразвуковий сигнал і чекає відбиття від предмета. Людина цей звук зазвичай не чує.

1. Arduino тримає `TRIG` у HIGH протягом 10 мікросекунд.
2. `pulseIn` вимірює, скільки мікросекунд `ECHO` перебуває у HIGH.
3. Ділимо тривалість на 58 і отримуємо приблизну відстань у сантиметрах.

Звук проходить шлях до предмета й назад. Коефіцієнт 58 уже враховує обидва напрямки. Наприклад, 580 мкс відповідає приблизно 10 см.

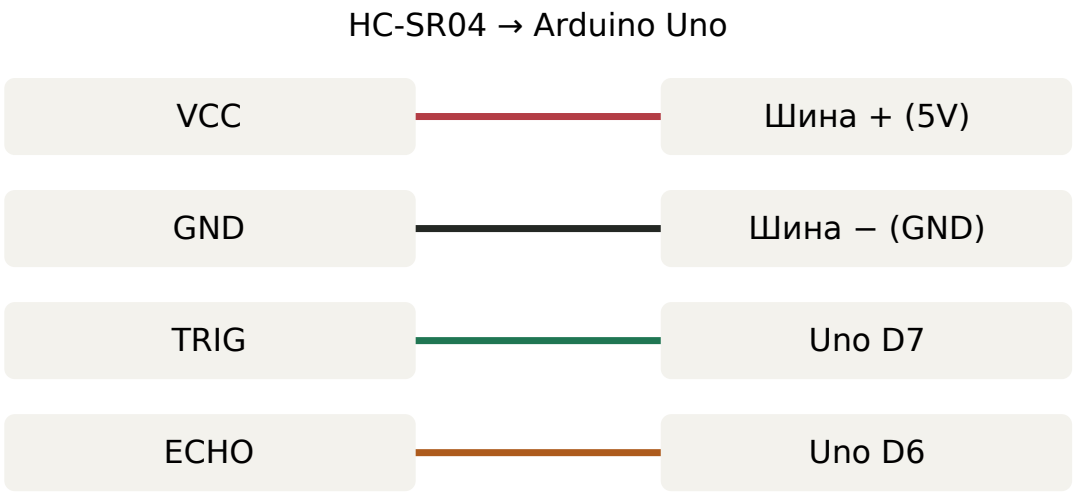
Чекаємо відповідь максимум 25 мс. Якщо її немає або результат поза 2–400 см, функція повертає `-1`. Це позначка невдалого вимірювання.

Датчик на макетній платі

1. Від’єднай живлення. Встав HC-SR04 у ряд `g`: `VCC → 12g`, `TRIG → 13g`, `ECHO → 14g`, `GND → 15g`. Ніжки мають бути в різних колонках. У межах кожної колонки отвори f–j з’єднані всередині плати.
2. З’єднай `5V` Uno з верхньою шиною `+`, а `GND` Uno — з верхньою шиною `−` макетної плати.
3. Проведи від `12h` (`VCC`) червоний дріт до шини `+`, а від `15h` (`GND`) — чорний до шини `−`.
4. З’єднай `13j` (TRIG) з `D7`. Цим виходом Arduino запускає вимірювання.
5. З’єднай `14i` (ECHO) з `D6`. На цьому вході Arduino вимірює тривалість відповіді.

Шини `+` і `−` мають з’єднані між собою отвори вздовж плати. Через них підводимо живлення до датчика. На фізичній платі перевір, чи немає розриву посередині шини; за потреби з’єднай її половини перемичкою.

Звіряй саме написи: напрямок роз’єму може відрізнятися від малюнка. Якщо ніжки твого модуля не дозволяють вставити його в макетну плату, під’єднай ті самі контакти дротами «мама–тато».



Електричні з’єднання · не масштаб плати

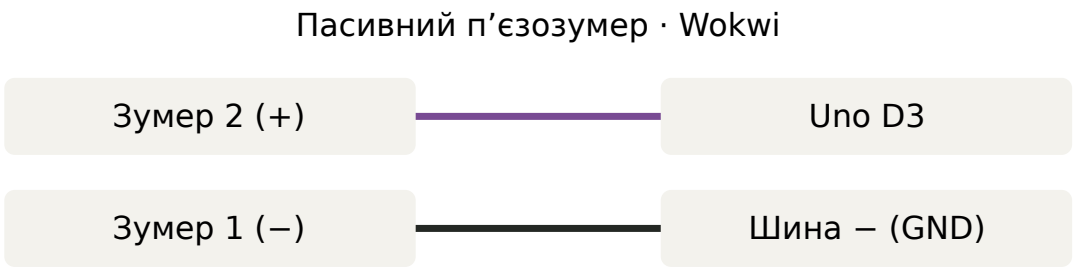
Додаємо звук до світла

Залиш LED із 06-01: A → `24a`, C → `23a`, резистор `24d–30d`, `D8` → `30e`, `23b` → шина −.

1. Плюсовий контакт зумера `2` → `D3`.
2. Мінусовий контакт `1` → верхня шина `–`, спільна з датчиком.

Пасивний п’єзозумер потребує коливань. Команда `tone(BUZZER_PIN, 2000)` створює звук із частотою 2000 Гц; `noTone` його зупиняє.

У програмі ці команди розміщені у функціях `startSound()` — увімкнути світло й звук і `stopSound()` — вимкнути обидва.



Електричні з’єднання · не масштаб плати

Три кроки програми

```
void loop() {
  float distanceCm = readDistanceCm(); // 1. Вимірюємо відстань.
  Serial.println(distanceCm); // 2. Показуємо результат у моніторі порту.
  playDistanceSignal(distanceCm); // 3. Відтворюємо сигнал для цієї відстані.
}
```

1. `readDistanceCm()` вимірює відстань. Зберігаємо її у змінній `distanceCm`.
2. `Serial.println(distanceCm)` показує це число в моніторі порту.
3. `playDistanceSignal(distanceCm)` вибирає та відтворює звук.

Після завершення третього кроку `loop()` починається знову. Функції вище описують, як виконати кожну дію.

Як підключити активний модуль зумера

Для модуля з керувальним входом і живленням 5 В: **`VCC`** → шина +, **`GND`** → шина −, S/IN → **`D3`**. Звір позначення та напругу з документацією модуля.

Якщо модуль вмикається рівнем HIGH, заміни тільки дві функції:

```
void startSound() {
  digitalWrite(LED_PIN, HIGH); // Увімкнути світлодіод.
  digitalWrite(BUZZER_PIN, HIGH); // Увімкнути активний зумер.
}

void stopSound() {
  digitalWrite(LED_PIN, LOW); // Вимкнути світлодіод.
  digitalWrite(BUZZER_PIN, LOW); // Вимкнути активний зумер.
}
```

Для модуля, який вмикається рівнем LOW, поміняй HIGH і LOW місцями тільки в командах для BUZZER_PIN, а також заміни LOW на HIGH у команді `digitalWrite(BUZZER_PIN, LOW)` у `setup()`. Попроси вчителя перевірити підключення. Двоконтактний електромагнітний зумер може потребувати окремого драйвера.

Перевір: на 120 см — тиша, на 8 см — безперервний звук.

Від імпульсу до сантиметрів

```
float readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW); // Готуємо вихід до нового імпульсу.
  delayMicroseconds(2); // Чекаємо 2 мікросекунди.
  digitalWrite(TRIG_PIN, HIGH); // Починаємо імпульс запуску.
  delayMicroseconds(10); // Тримаємо його 10 мікросекунд.
  digitalWrite(TRIG_PIN, LOW); // Завершуємо імпульс.

  unsigned long echoTime = pulseIn(ECHO_PIN, HIGH, 25000); // Чекаємо
відповідь до 25 мс.
  float distanceCm = echoTime / 58.0; // Перетворюємо тривалість відповіді на
сантиметри.

  if (echoTime == 0 || distanceCm < 2 || distanceCm > 400) {
    return -1; // Повідомляємо, що вимірювання не вдалося.
  }

  return distanceCm; // Повертаємо виміряну відстань.
}
```

1. Надсилаємо короткий імпульс через `TRIG`.
2. `pulseIn()` вимірює тривалість відповіді на `ECHO`.
3. Ділимо час на 58 й отримуємо сантиметри.
4. `return` повертає результат у `loop()`.

Якщо відповіді немає або відстань поза межами 2–400 см, повертаємо `-1`. Це позначення помилки вимірювання.

Вибираємо звук за відстанню

```
void playDistanceSignal(float distanceCm) {

  if (distanceCm < 0) { // Датчик не зміг виміряти відстань.
    stopSound(); // Вимикаємо звук.
    delay(100); // Чекаємо перед повторною спробою.

  } else if (distanceCm > 100) { // Перешкода далі ніж 100 см.
    stopSound(); // Залишаємо тишу.
    delay(100); // Чекаємо перед наступним вимірюванням.

  } else if (distanceCm > 30) { // Перешкода в межах понад 30–100 см.
    beep(600); // Короткий сигнал, потім пауза 600 мс.

  } else if (distanceCm > 10) { // Перешкода в межах понад 10–30 см.
    beep(150); // Короткий сигнал, потім пауза 150 мс.

  } else { // Перешкода в межах 2–10 см.
    startSound(); // Звук залишається ввімкненим.
    delay(100); // Через 100 мс знову виміряємо відстань.
  }

}
```

Читаємо умови зверху вниз. Виконується лише перша гілка, умова якої справдилася.

- `-1` — вимірювання не вдалося: тиша.
- Понад 100 см — тиша.
- Понад 30 і до 100 см — короткий сигнал, пауза 600 мс.
- Понад 10 і до 30 см — короткий сигнал, пауза 150 мс.
- Від 2 до 10 см — безперервний звук.

Наприклад, для 20 см перші три умови хибні. Умова `distanceCm > 10` істинна, тому виконується `beep(150)`.

Один короткий сигнал

```
void beep(int pauseMs) {
  startSound(); // Вмикаємо зумер.
  delay(80); // Сигнал звучить 80 мс.
  stopSound(); // Вимикаємо зумер.
  delay(pauseMs); // Чекаємо перед наступним вимірюванням.
}
```

`pauseMs` — тривалість тиші, яку передаємо в дужках: 600 або 150 мс. Сам звук завжди триває 80 мс.

Під час `delay()` нових вимірювань немає. Для рідких сигналів наступне вимірювання почнеться після 80 + 600 = 680 мс; для частих — після 230 мс. У тиші та під час безперервного звуку пауза становить 100 мс.

Порівняй чотири відстані

Запуск у Wokwi

Натисни кнопку запуску. Для самостійного завдання збережи власну копію проєкту.

Готовий проєкт: https://wokwi.com/projects/475804740918654977. Схему можна зібрати вручну за кроками уроку. Повний код — на останній сторінці.

У Wokwi натисни на HC-SR04 та змінюй відстань повзунком. На справжній платі тримай перед датчиком плаский картон на відомій відстані, перевіряючи її лінійкою.

Не притискай предмет до датчика: ближче 2 см показ ненадійний. М’яка тканина, нахилена поверхня або порожній простір можуть не дати відлуння.

1. Відкрий монітор порту 115200.
2. Установи 120 см — тиша, LED вимкнений.
3. Установи 60 см — рідкі сигнали й спалахи разом.
4. Установи 20 см — часті сигнали й спалахи разом.
5. Установи 8 см — постійне світло та безперервний звук.

Почни з показу відстані

Постійно бачиш -1

Перевір живлення, TRIG на `D7` і ECHO на `D6`. Постав плаский картон на відстані 20–50 см перпендикулярно до датчика.

Є правильні сантиметри, але немає звуку

Звір `D3` і `GND`, тип зумера та команди у startSound() і stopSound(). У браузері перевір гучність вкладки. Для активного модуля звір рівень увімкнення з його документацією.

Чому на 30 см сигнал уже частий?

Для 30 см умова > 30 хибна, а > 10 істинна: виконується beep(150). Для 10 см обидві умови хибні, тому виконується остання гілка з безперервним звуком.

Попереджай раніше

> Зміни межу частих сигналів із 30 на 40 см. Межі тиші й безперервного звуку залиш незмінними.

1. Знайди умову `distanceCm > 30`.
2. Зміни число, запусти й перевір 35 см.
3. Потім перевір 50 см і 8 см.

Очікуваний результат

В умові distanceCm > 30 заміни 30 на 40. На 35 см сигнали часті, на 50 см рідкі, на 8 см звук безперервний.

УРОК 06-02 · ПОВНИЙ КОД

Парктронік зі звуком

До світлодіода додаємо пищалку: світло й звук працюють разом.

```
#include <Arduino.h> // Підключаємо команди Arduino.

const byte TRIG_PIN = 7; // Вихід для запуску вимірювання.
const byte ECHO_PIN = 6; // Вхід для відповіді датчика.
const byte LED_PIN = 8; // Вихід світлодіода.
const byte BUZZER_PIN = 3; // Вихід для зумера.

float readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW); // Готуємо вихід до нового імпульсу.
  delayMicroseconds(2); // Чекаємо 2 мікросекунди.
  digitalWrite(TRIG_PIN, HIGH); // Починаємо імпульс запуску.
  delayMicroseconds(10); // Тримаємо його 10 мікросекунд.
  digitalWrite(TRIG_PIN, LOW); // Завершуємо імпульс.

  unsigned long echoTime = pulseIn(ECHO_PIN, HIGH, 25000); // Чекаємо відповідь до 25
мс.
  float distanceCm = echoTime / 58.0; // Перетворюємо тривалість відповіді на
сантиметри.

  if (echoTime == 0 || distanceCm < 2 || distanceCm > 400) {
    return -1; // Повідомляємо, що вимірювання не вдалося.
  }

  return distanceCm; // Повертаємо виміряну відстань.
}

void startSound() {
  digitalWrite(LED_PIN, HIGH); // Світлодіод світиться разом зі звуком.
  tone(BUZZER_PIN, 2000); // Вмикаємо звук із частотою 2000 Гц.
}

void stopSound() {
  digitalWrite(LED_PIN, LOW); // Світлодіод гасне разом зі звуком.
  noTone(BUZZER_PIN); // Вимикаємо звук.
}

void beep(int pauseMs) {
  startSound(); // Вмикаємо зумер.
  delay(80); // Сигнал звучить 80 мс.
  stopSound(); // Вимикаємо зумер.
  delay(pauseMs); // Чекаємо перед наступним вимірюванням.
}

void playDistanceSignal(float distanceCm) {

  if (distanceCm < 0) { // Датчик не зміг виміряти відстань.
```

```
    stopSound(); // Вимикаємо звук.
    delay(100); // Чекаємо перед повторною спробою.

  } else if (distanceCm > 100) { // Перешкода далі ніж 100 см.
    stopSound(); // Залишаємо тишу.
    delay(100); // Чекаємо перед наступним вимірюванням.

  } else if (distanceCm > 30) { // Перешкода в межах понад 30–100 см.
    beep(600); // Короткий сигнал, потім пауза 600 мс.

  } else if (distanceCm > 10) { // Перешкода в межах понад 10–30 см.
    beep(150); // Короткий сигнал, потім пауза 150 мс.

  } else { // Перешкода в межах 2–10 см.
    startSound(); // Звук залишається ввімкненим.
    delay(100); // Через 100 мс знову виміряємо відстань.
  }

}

void setup() {
  pinMode(LED_PIN, OUTPUT); // Керуємо світлодіодом.
  pinMode(TRIG_PIN, OUTPUT); // Надсилаємо команду датчику через TRIG.
  pinMode(ECHO_PIN, INPUT); // Отримуємо відповідь через ECHO.
  digitalWrite(BUZZER_PIN, LOW); // Готуємо вимкнений вихід зумера.
  pinMode(BUZZER_PIN, OUTPUT); // Керуємо зумером через цей вихід.
  stopSound(); // Починаємо з тиші.
  Serial.begin(115200); // Відкриваємо зв’язок із монітором порту.
}

void loop() {
  float distanceCm = readDistanceCm(); // 1. Вимірюємо відстань.
  Serial.println(distanceCm); // 2. Показуємо результат у моніторі порту.
  playDistanceSignal(distanceCm); // 3. Відтворюємо сигнал для цієї відстані.
}
```