Відстань на LCD 1602

На першому рядку екрана — напис Distance:, на другому — відстань у сантиметрах.

- Arduino Uno, USB-кабель, макетна плата й дроти.
- HC-SR04, червоний LED, резистор 220 Ом і зумер зі схеми 06-02.
- LCD 1602 та I²C-перехідник PCF8574.
- Лінійка й плаский картон.

Залишаємо світлодіод і пищалку з 06-02. Додаємо екран, щоб бачити відстань числом.

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

Екран і перехідник

LCD 1602 має два рядки по 16 символів. Перехідник I²C дає змогу під’єднати його до Uno чотирма дротами.

1. Перевір із учителем, чи перехідник уже припаяний до 16 контактів LCD.
2. Якщо деталі окремі, учитель звіряє контакт 1 перехідника з VSS LCD й припаює весь ряд. Просте прикладання контактів не забезпечує надійного з’єднання.
3. Знайди `GND`, `VCC`, SDA та SCL на перехіднику.

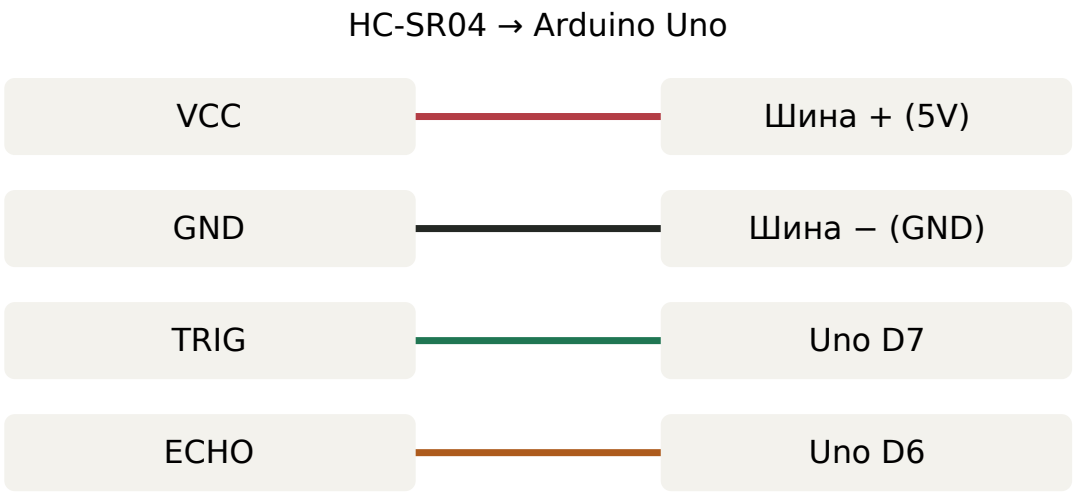
У Wokwi обери LCD1602 з атрибутом `pins: "i2c"`: це вже модель екрана з перехідником.

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».



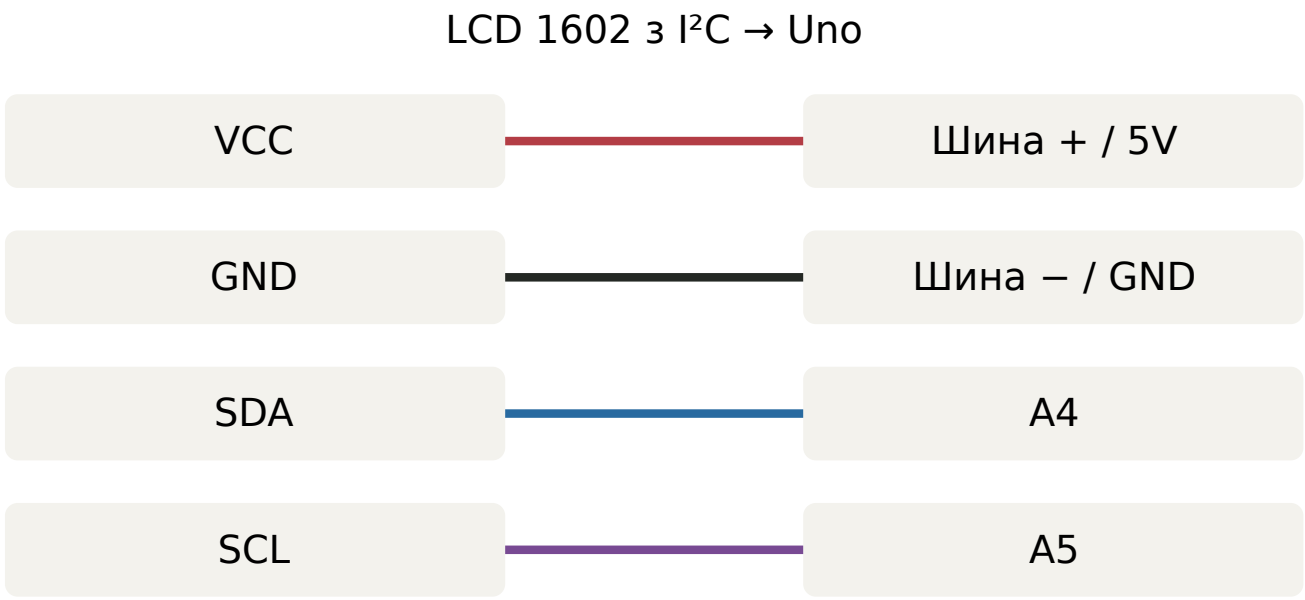
Електричні з’єднання · не масштаб плати

Залиш підключення з 06-02: LED A → `24a`, C → `23a`; резистор `24d–30d`; `D8` → `30e`; `23b` → шина −. Зумер: плюс → `D3`, мінус → шина −. Для активного модуля використай налаштування з кроку 5 уроку 06-02.

Живлення та дані

1. LCD `VCC` → верхня шина +, під’єднана до `5V` Uno.
2. LCD `GND` → верхня шина −, під’єднана до `GND` Uno.
3. LCD SDA → `A4` Uno.
4. LCD SCL → `A5` Uno.

Датчик і LCD отримують живлення паралельно від спільних шин. SDA передає дані, SCL задає такт обміну.



Електричні з’єднання · не масштаб плати

Під’єднуй дроти при вимкненому живленні. Після запуску повільно поверни синій регулятор контрасту на перехіднику, щоб символи стали видимими.

Налаштування LCD

1. У Wokwi додай `LiquidCrystal I2C` через Library Manager або перенеси підготовлений `libraries.txt`.
2. В Arduino IDE встанови `LiquidCrystal I2C` від Frank de Brabander.

```
#include <Arduino.h> // Команди Arduino.
#include <LiquidCrystal_I2C.h> // Керування LCD через I²C.

const byte TRIG_PIN = 7; // Вихід запуску датчика.
const byte ECHO_PIN = 6; // Вхід відповіді датчика.
const byte LED_PIN = 8; // Вихід світлодіода.
const byte BUZZER_PIN = 3; // Вихід зумера.
const byte LCD_ADDRESS = 0x27; // Адреса екрана у Wokwi; звір її для свого модуля.
LiquidCrystal_I2C lcd(LCD_ADDRESS, 16, 2); // Екран: 16 символів у кожному з 2 рядків.
```

`0x27` — адреса екрана у схемі Wokwi. Числа 16 і 2 задають кількість символів та рядків.

Адреса фізичного перехідника може відрізнятися. Перед заняттям учитель перевіряє її I²C-сканером та за потреби змінює `LCD_ADDRESS`. Положення контактів **`A0`**–**`A2`** на перехіднику впливає на адресу.

Підсвічування та курсор

```
void setup() {
  pinMode(LED_PIN, OUTPUT); // Керуємо світлодіодом.
  digitalWrite(BUZZER_PIN, LOW); // Готуємо вимкнений вихід зумера.
  pinMode(BUZZER_PIN, OUTPUT); // Керуємо зумером.
  stopSound(); // Починаємо без світла й звуку.
  pinMode(TRIG_PIN, OUTPUT); // Надсилаємо імпульс датчику.
  pinMode(ECHO_PIN, INPUT); // Приймаємо відповідь датчика.
  Serial.begin(115200); // Виводимо число для перевірки в монітор порту.
  lcd.init(); // Готуємо екран до роботи.
  lcd.backlight(); // Вмикаємо підсвічування.
  lcd.setCursor(0, 0); // Ставимо курсор на початок першого рядка.
  lcd.print("Distance:"); // Пишемо заголовок «Відстань».
}
```

`lcd.init()` готує екран, `backlight()` вмикає підсвічування. `setCursor(0, 0)` ставить курсор на перший символ першого рядка.

Нумерація починається з нуля: рядки 0 і 1, позиції 0–15. `print()` пише текст від поточного положення курсора. Використовуємо латиницю, яку підтримує шрифт цього екрана.

Оновлюємо другий рядок

```
void showDistance(float distanceCm) {
  lcd.setCursor(0, 1); // Ставимо курсор на початок другого рядка.
  lcd.print("                "); // Стираємо попередній показ 16 пробілами.
  lcd.setCursor(0, 1); // Повертаємо курсор на початок рядка.

  if (distanceCm < 0) { // Вимірювання не вдалося.
    lcd.print("No echo"); // Повідомляємо, що немає надійного відлуння.

  } else { // Датчик повернув відстань.
    lcd.print(distanceCm, 0); // Показуємо сантиметри без десяткових знаків.
    lcd.print(" cm"); // Додаємо одиницю вимірювання.
  }

}
```

1. Переміщуємо курсор у другий рядок.
2. Стираємо старий показ 16 пробілами.
3. Повертаємо курсор на початок.
4. Пишемо відстань або `No echo` — «немає відлуння».

Очищення потрібне, щоб після зміни 120 на 8 не залишалися старі цифри. Аргумент 0 у `lcd.print(distanceCm, 0)` задає показ без десяткових знаків.

```
void loop() {
  float distanceCm = readDistanceCm(); // 1. Вимірюємо відстань.
  Serial.println(distanceCm); // 2. Показуємо число в моніторі порту.
  showDistance(distanceCm); // 3. Оновлюємо другий рядок LCD.
  playDistanceSignal(distanceCm); // 4. Подаємо світловий і звуковий сигнал.
}
```

Правила світла й звуку залишаються з 06-02. Екран оновлюється перед кожним сигналом; під час його паузи нових вимірювань немає.

Від 120 до 8 см

Запуск у Wokwi

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

Готовий проєкт: https://wokwi.com/projects/475804951929936897. 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. Установи 120 см — другий рядок показує `120 cm`.
2. Зміни на 8 см — бачиш `8 cm`, без залишків попереднього числа.
3. Перевір 60 см та 20 см: спалахи й звук стають частішими. На 8 см світло і звук безперервні.
4. На фізичній платі порівняй екран із монітором порту на 115200.
5. Коли надійного відлуння немає, очікуй `No echo`.

Підсвічування є, тексту немає

Екран світиться, але символів немає

Повільно відрегулюй контраст синім регулятором. Якщо не допомогло, вимкни живлення та перевір пайку, SDA–`A4`, SCL–`A5` і адресу модуля.

Немає підсвічування

Вимкни живлення. Перевір `VCC`, `GND` і перемичку підсвічування на перехіднику. У коді має бути lcd.backlight().

На екрані No echo

LCD працює. Перевір датчик і постав перед ним плаский картон на 20–50 см.

Помилка LiquidCrystal_I2C.h

Встанови бібліотеку з кроку 5 й запусти компіляцію знову.

Власний заголовок

> Заміни `Distance:` на `Vidstan:`. Покажи відстань з одним знаком після крапки.

1. Зміни текст у setup().
2. У `lcd.print(distanceCm, 0)` заміни 0 на 1.
3. Перевір покази на 120 і 8 см.

Очікуваний результат

Угорі Vidstan:, внизу, наприклад, 8.0 cm. Десятковий знак змінює формат показу, але не точність датчика.

УРОК 06-03 · ПОВНИЙ КОД

Відстань на LCD 1602

До світлодіода й пищалки додаємо LCD 1602 із показом відстані.

```
#include <Arduino.h> // Команди Arduino.
#include <LiquidCrystal_I2C.h> // Керування LCD через I²C.

const byte TRIG_PIN = 7; // Вихід запуску датчика.
const byte ECHO_PIN = 6; // Вхід відповіді датчика.
const byte LED_PIN = 8; // Вихід світлодіода.
const byte BUZZER_PIN = 3; // Вихід зумера.
const byte LCD_ADDRESS = 0x27; // Адреса екрана у Wokwi; звір її для свого модуля.
LiquidCrystal_I2C lcd(LCD_ADDRESS, 16, 2); // Екран: 16 символів у кожному з 2 рядків.

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 showDistance(float distanceCm) {
  lcd.setCursor(0, 1); // Ставимо курсор на початок другого рядка.
  lcd.print("                "); // Стираємо попередній показ 16 пробілами.
  lcd.setCursor(0, 1); // Повертаємо курсор на початок рядка.

  if (distanceCm < 0) { // Вимірювання не вдалося.
    lcd.print("No echo"); // Повідомляємо, що немає надійного відлуння.

  } else { // Датчик повернув відстань.
    lcd.print(distanceCm, 0); // Показуємо сантиметри без десяткових знаків.
    lcd.print(" cm"); // Додаємо одиницю вимірювання.
  }

}

void setup() {
  pinMode(LED_PIN, OUTPUT); // Керуємо світлодіодом.
  digitalWrite(BUZZER_PIN, LOW); // Готуємо вимкнений вихід зумера.
  pinMode(BUZZER_PIN, OUTPUT); // Керуємо зумером.
  stopSound(); // Починаємо без світла й звуку.
  pinMode(TRIG_PIN, OUTPUT); // Надсилаємо імпульс датчику.
  pinMode(ECHO_PIN, INPUT); // Приймаємо відповідь датчика.
  Serial.begin(115200); // Виводимо число для перевірки в монітор порту.
  lcd.init(); // Готуємо екран до роботи.
  lcd.backlight(); // Вмикаємо підсвічування.
  lcd.setCursor(0, 0); // Ставимо курсор на початок першого рядка.
  lcd.print("Distance:"); // Пишемо заголовок «Відстань».
}

void loop() {
```

```
  float distanceCm = readDistanceCm(); // 1. Вимірюємо відстань.
  Serial.println(distanceCm); // 2. Показуємо число в моніторі порту.
  showDistance(distanceCm); // 3. Оновлюємо другий рядок LCD.
  playDistanceSignal(distanceCm); // 4. Подаємо світловий і звуковий сигнал.
}
```