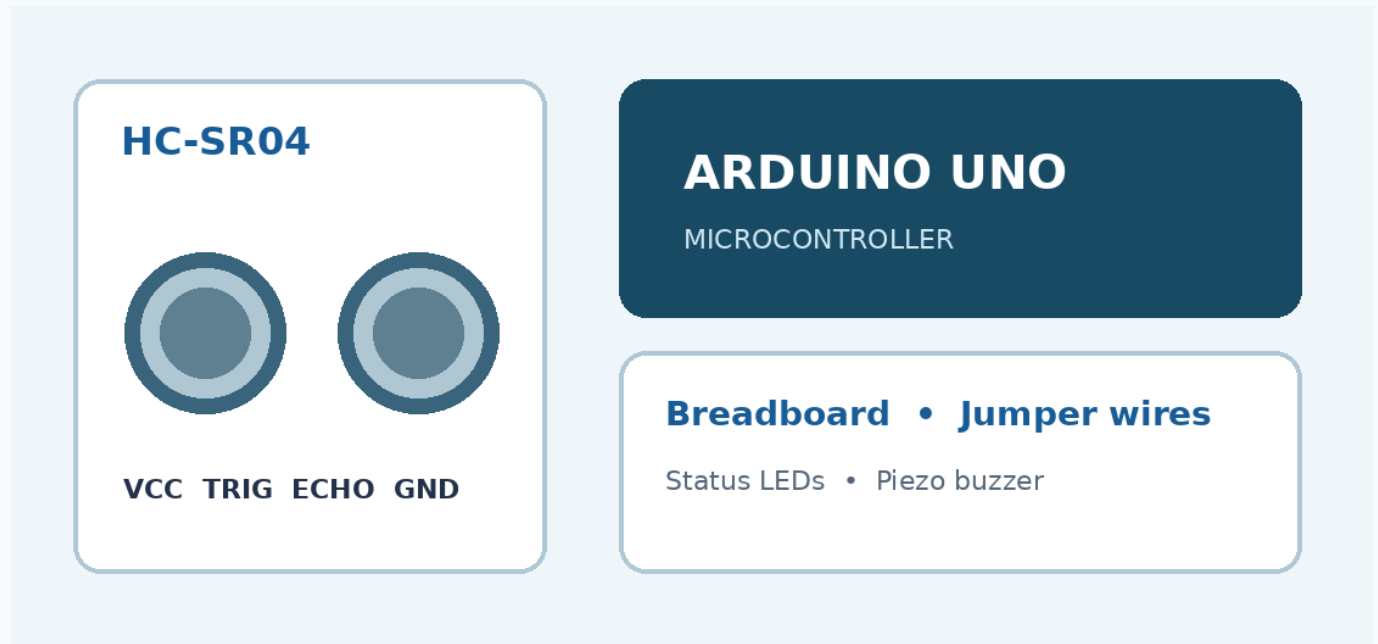


Touchless Musical Synthesizer

01 / COVER

An illustrated, beginner-first guide to building a real Arduino embedded system in a 90-minute IEEE lab.



YOU WILL BUILD

A working hand-controlled electronic instrument with a laptop companion dashboard.

YOU WILL LEARN

Wiring, firmware, sensing, state machines, safe electronics, validation, and serial communication.

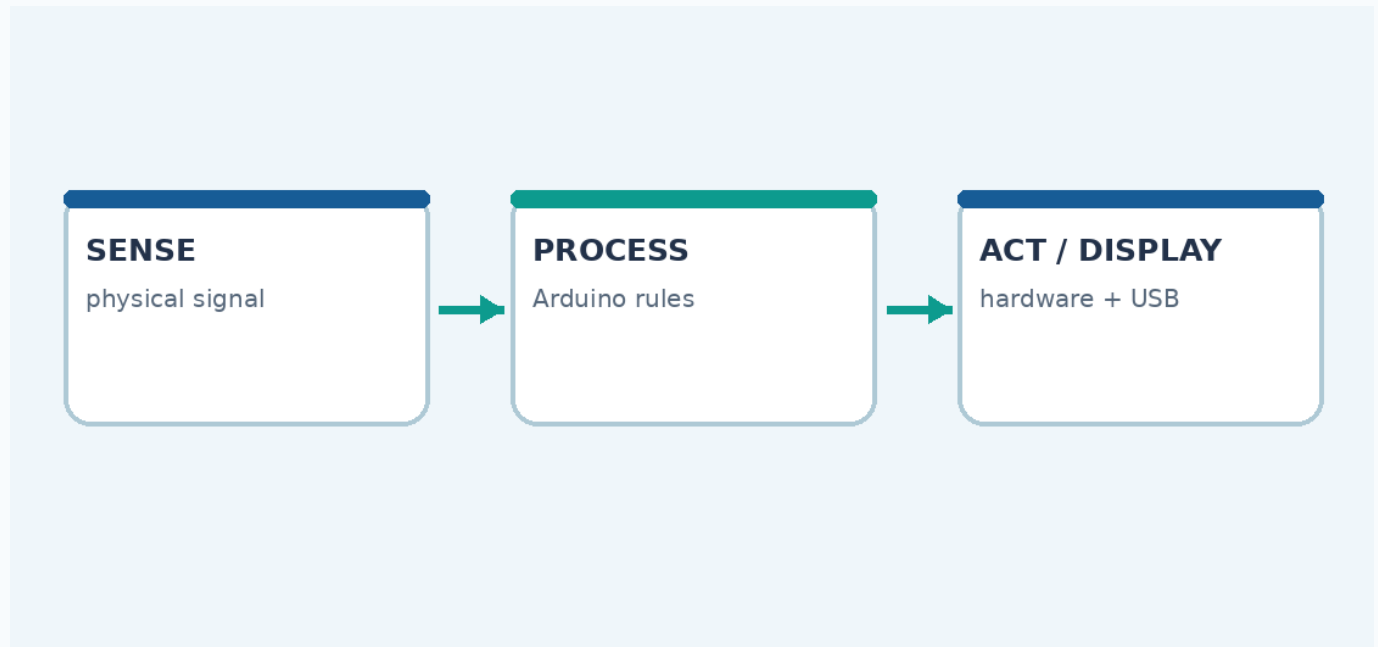
NO PYTHON LISTINGS

The dashboard is provided separately through the workshop GitHub repository.

The build at a glance

02 / YOUR MISSION

Before touching a wire, understand the real-world problem your controller is solving.



SCENARIO

An instrument should translate a musician's hand position into sound without physical contact.

SUCCESS

A measurable, repeatable response and a useful status display.

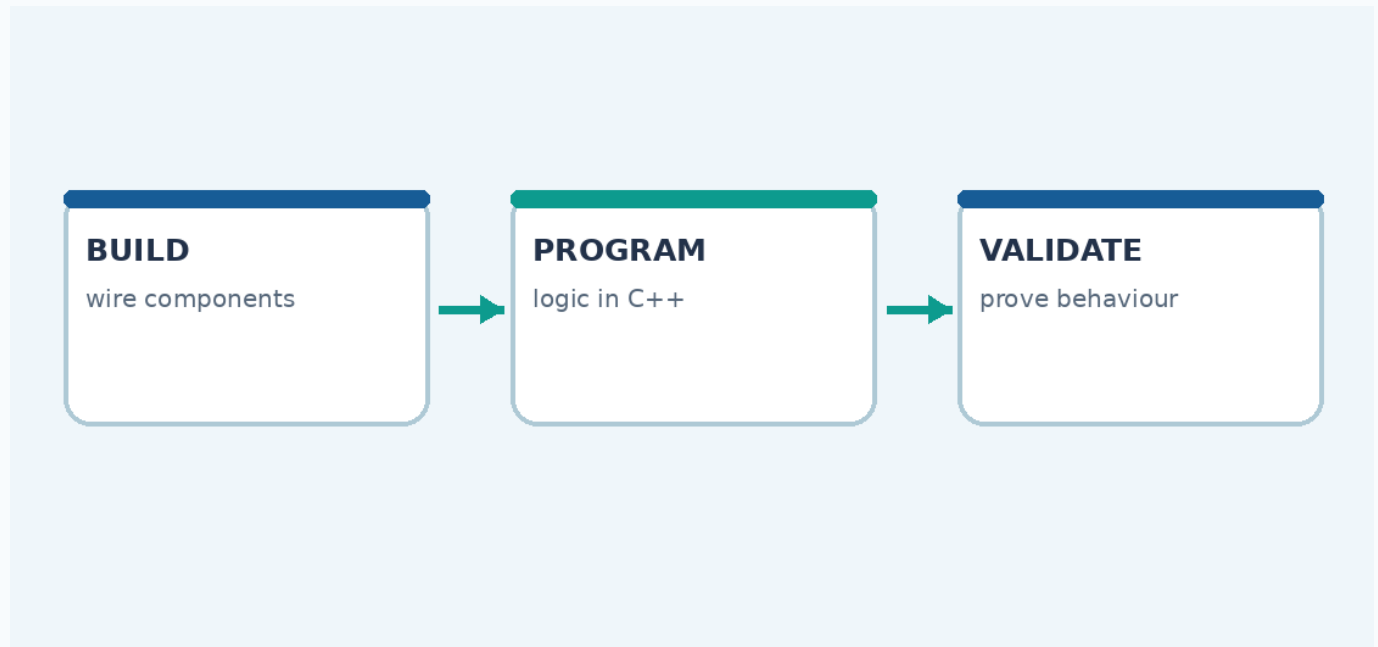
FINISH LINE

The Arduino works independently; the laptop visualizes telemetry, not the other way around.

Your 90-minute route

03 / WORKSHOP MAP

The core activity has three checkpoints and an optional extension for faster groups.



00-15 MIN

Identify components, wire power safely, and confirm the first output.

15-35 MIN

Measure distance and interpret a sensor reading.

35-60 MIN

Create the distance-to-pitch controller.

60-75 MIN

Add LED feedback and test edge cases.

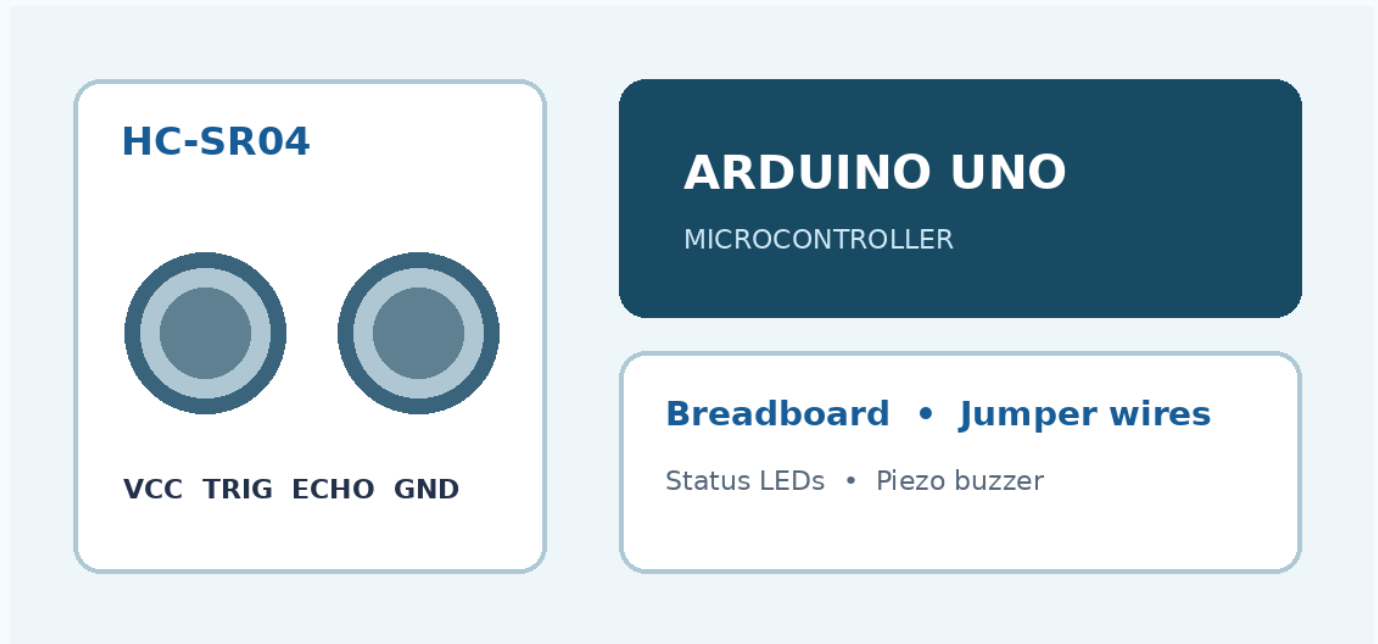
75-90 MIN

Connect the laptop dashboard; demonstrate and explain your design.

Get to know your components

04 / KIT INVENTORY

Lay everything out. Match the pictures and names before connecting the Arduino.



ARDUINO UNO

The microcontroller that reads sensors and runs your program.

HC-SR04

An ultrasonic ranging sensor; it needs four connections.

OUTPUT

A passive piezo buzzer plus two status LEDs and resistors.

BREADBOARD

Lets you assemble circuits without soldering.

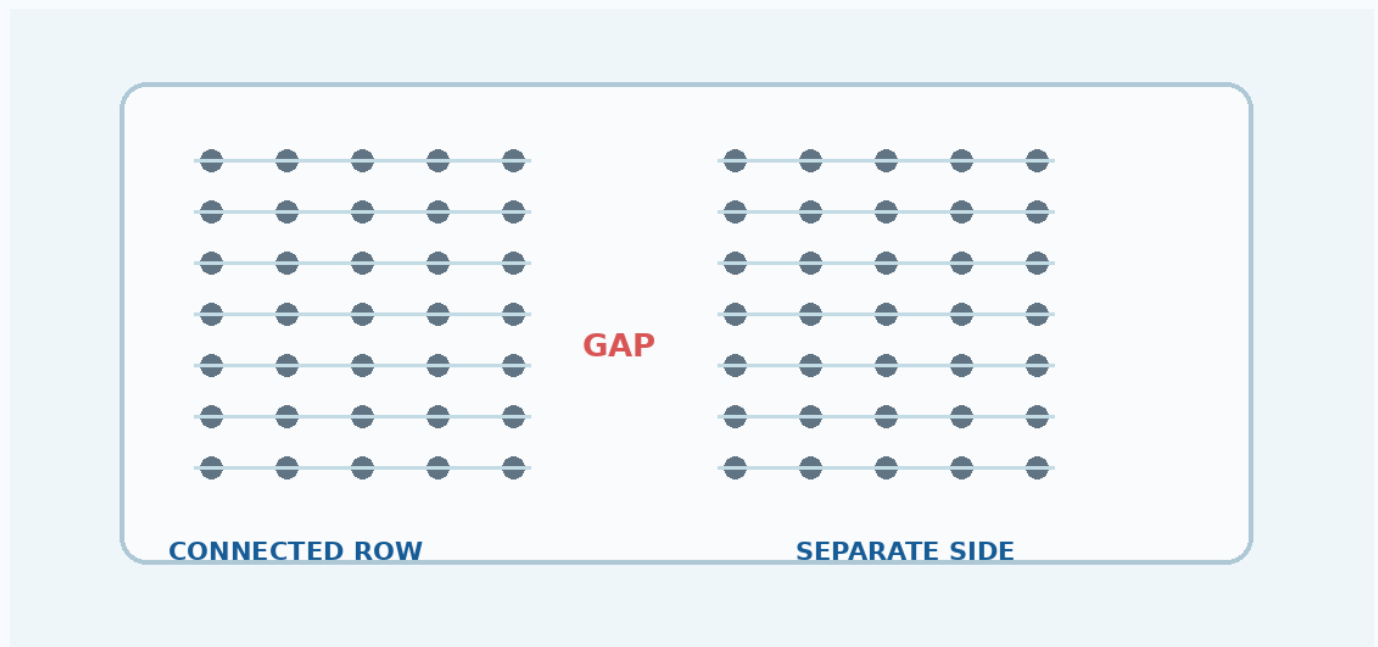
JUMPERS + USB

Male-to-male wires connect pins and breadboard rows; USB powers/programs the Uno.

Breadboard anatomy

05 / HOW CONNECTIONS WORK

Most beginner wiring failures are actually breadboard layout mistakes.



MAIN TERMINALS

On a typical breadboard, five holes in one letter group and numbered row are connected.

CENTRE GAP

The two sides of the centre gap are not connected to one another.

POWER RAILS

Long + and - strips run separately; some boards split the rails halfway.

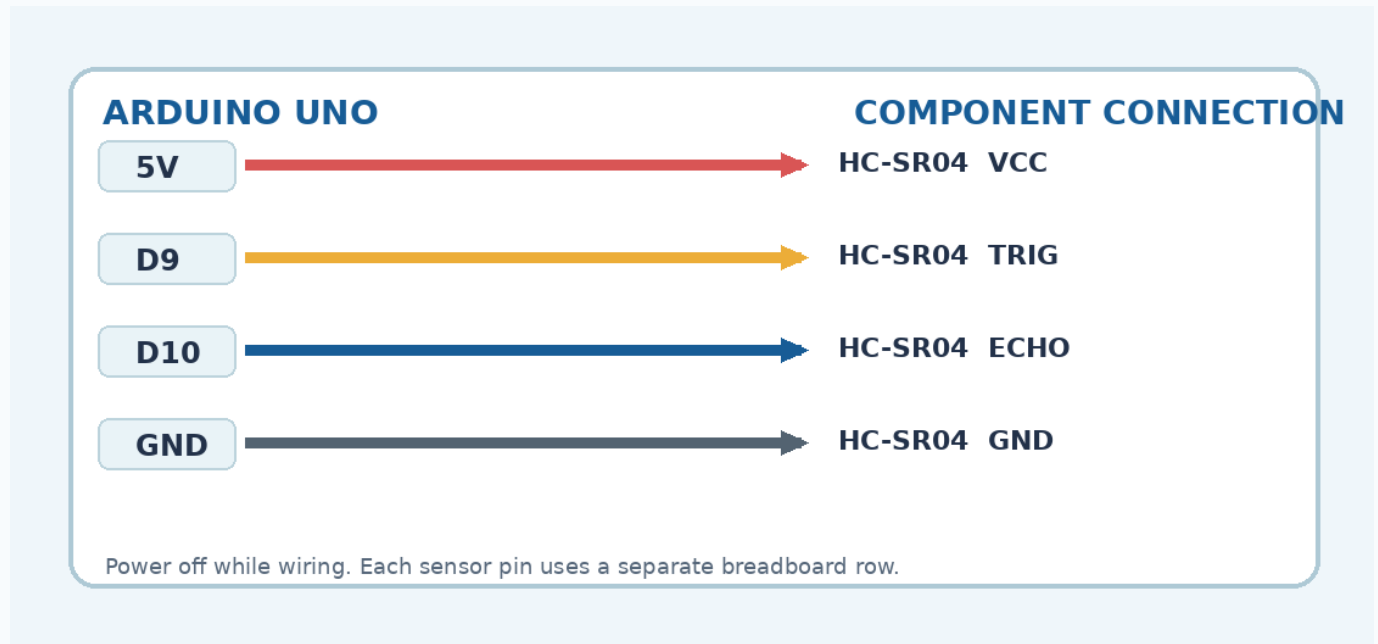
VERIFY

Look for the printed row numbers and rail markings on your board.

Understand the electrical path

06 / SAFE POWER

Disconnect USB before changing wires. Never let 5V and GND touch.



5V

Supplies regulated power to the HC-SR04.

GND

Completes the electrical circuit; all components must share a common ground.

SIGNAL

Arduino digital I/O pins send or receive logic signals.

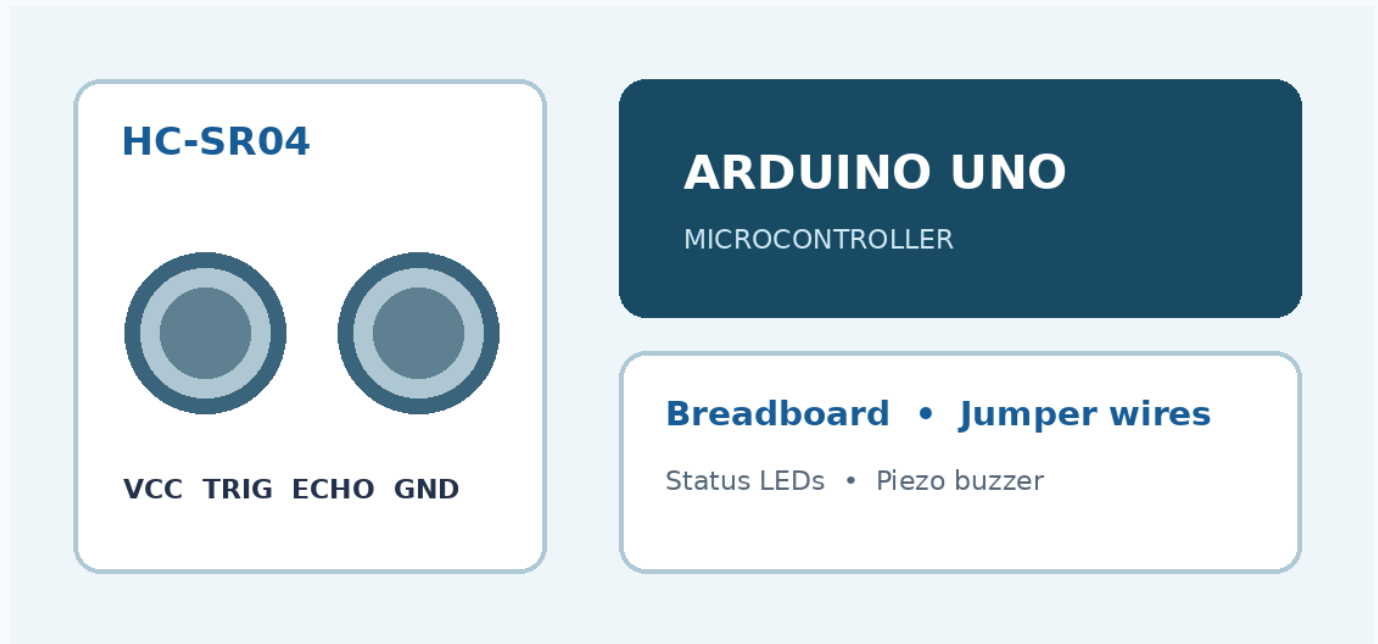
LED SAFETY

Every bare LED needs its own series resistor (220–330 ohms suggested).

Meet the HC-SR04

07 / SENSOR BASICS

The ranging sensor has two round ultrasonic transducers and four pins.



VCC

Connect to Arduino 5V.

TRIG

The Arduino emits a 10-microsecond trigger pulse.

ECHO

A HIGH pulse encodes the time of flight.

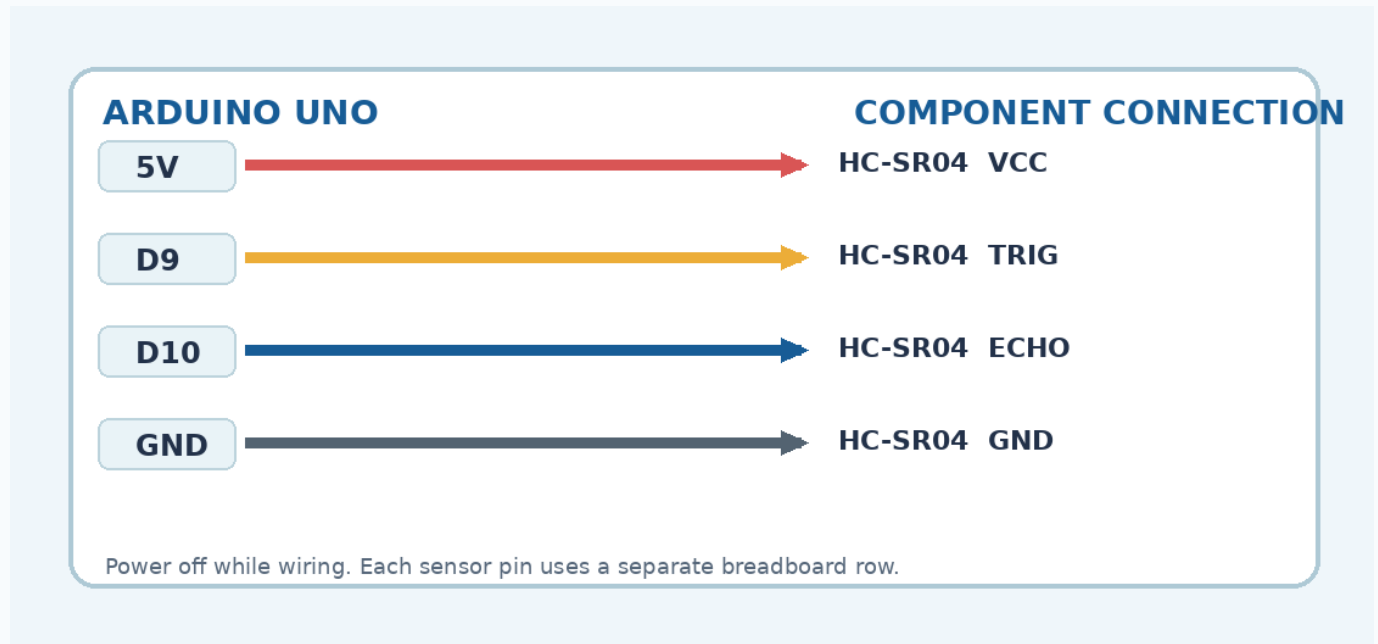
GND

Connect to common ground.

Wire the distance sensor

08 / HARDWARE CHECKPOINT

Position all four sensor pins in electrically separate breadboard rows.



1

Unplug the Arduino and insert the sensor into the breadboard.

2

Connect HC-SR04 VCC → Arduino 5V.

3

Connect HC-SR04 GND → Arduino GND.

4

Connect TRIG → Arduino D9 and ECHO → Arduino D10.

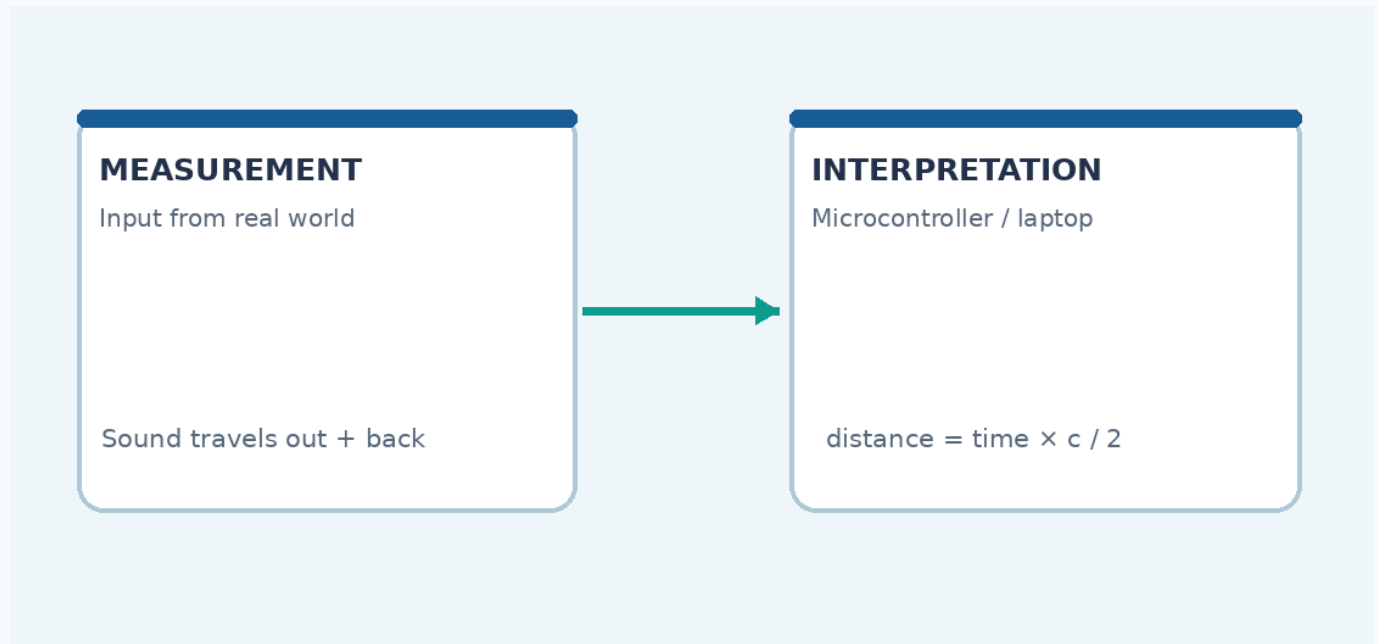
5

Inspect for shorts. Plug in USB only after checking the wiring.

Why ultrasonic distance works

09 / MEASUREMENT PHYSICS

The Arduino times the outward and returning ultrasonic pulse.



ONE TRIP OUT + ONE BACK

The measured echo time covers twice the object distance.

SPEED OF SOUND

Approximately 0.0343 centimetres per microsecond at room temperature.

EQUATION

distance (cm) = echo duration (microseconds) × 0.0343 ÷ 2.

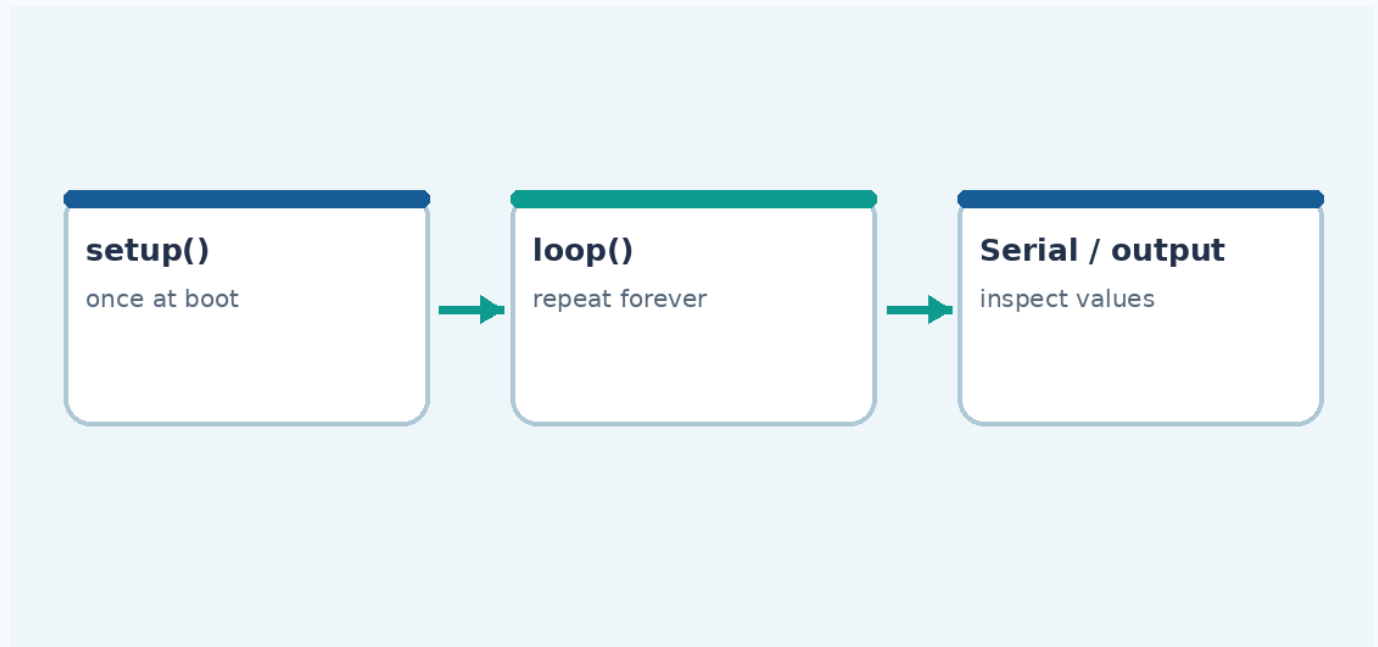
EXAMPLE

A 1,000 microsecond echo gives about 17.15 cm.

Your first Arduino sketch

10 / CODE BUILD 01

Practice the three building blocks of Arduino programming.



CONSTANTS

Assign pin numbers once, at the top of the sketch.

setup()

Runs once when the Arduino powers on or resets.

ARDUINO C++ / COPY OR DISCUSS

```
const int TRIG_PIN = 9;
const int ECHO_PIN = 10;

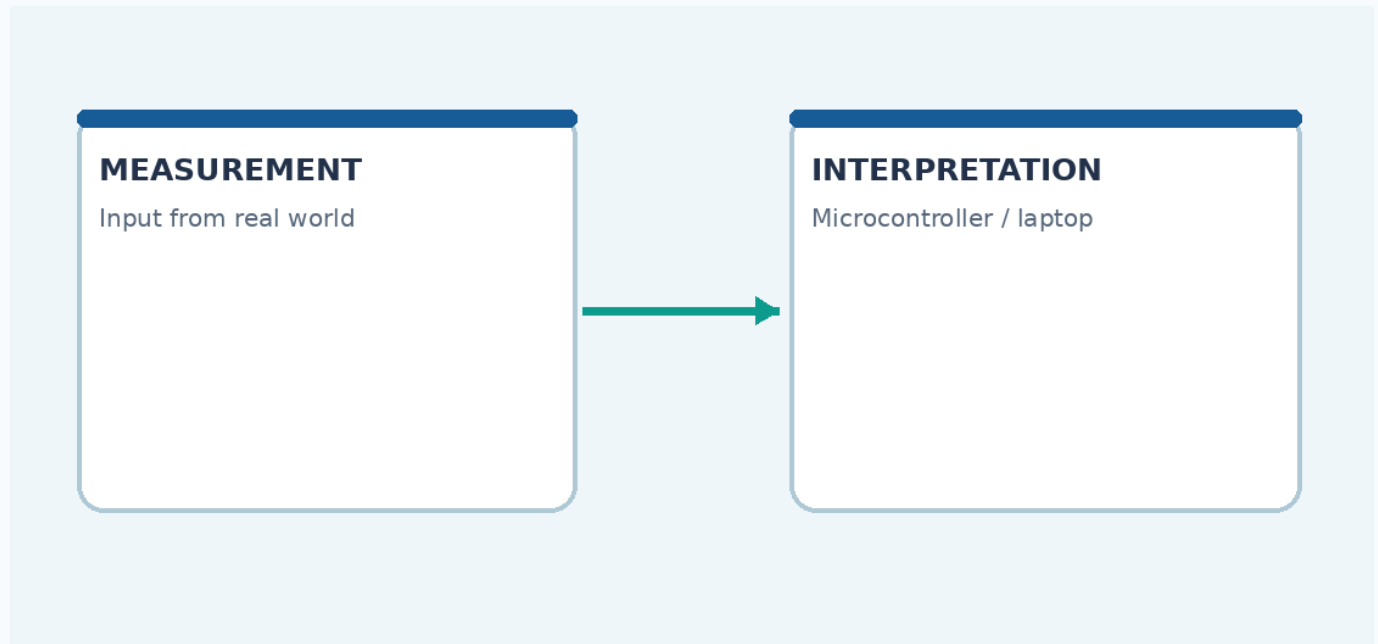
void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

void loop() {
  // Your repeated sensing code goes here.
}
```

Send and time the pulse

11 / CODE BUILD 02

Create a short TRIG pulse, then time the ECHO signal.



PULSE

Drive TRIG low, then high for 10 microseconds, then low again.

TIMEOUT

Use `pulseIn(..., 25000UL)` so a missing echo cannot block forever.

ARDUINO C++ / COPY OR DISCUSS

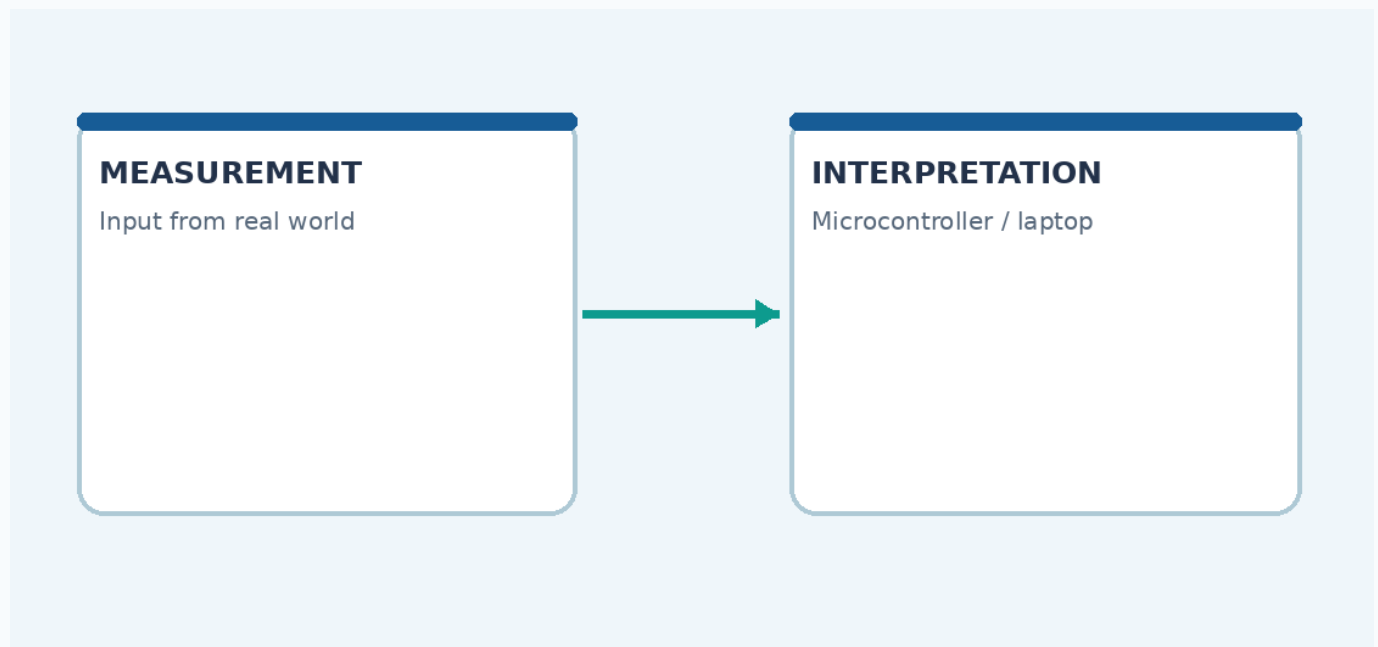
```
digitalWrite(TRIG_PIN, LOW);
delayMicroseconds(2);
digitalWrite(TRIG_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);

unsigned long duration =
  pulseIn(ECHO_PIN, HIGH, 25000UL);
float distance = duration * 0.0343 / 2.0;
```

Read live measurements

12 / FIRST DEMO

The Serial Monitor is your first engineering instrument.



UPLOAD

Select Arduino Uno and the correct serial port.

MONITOR

Open Serial Monitor at 9600 baud.

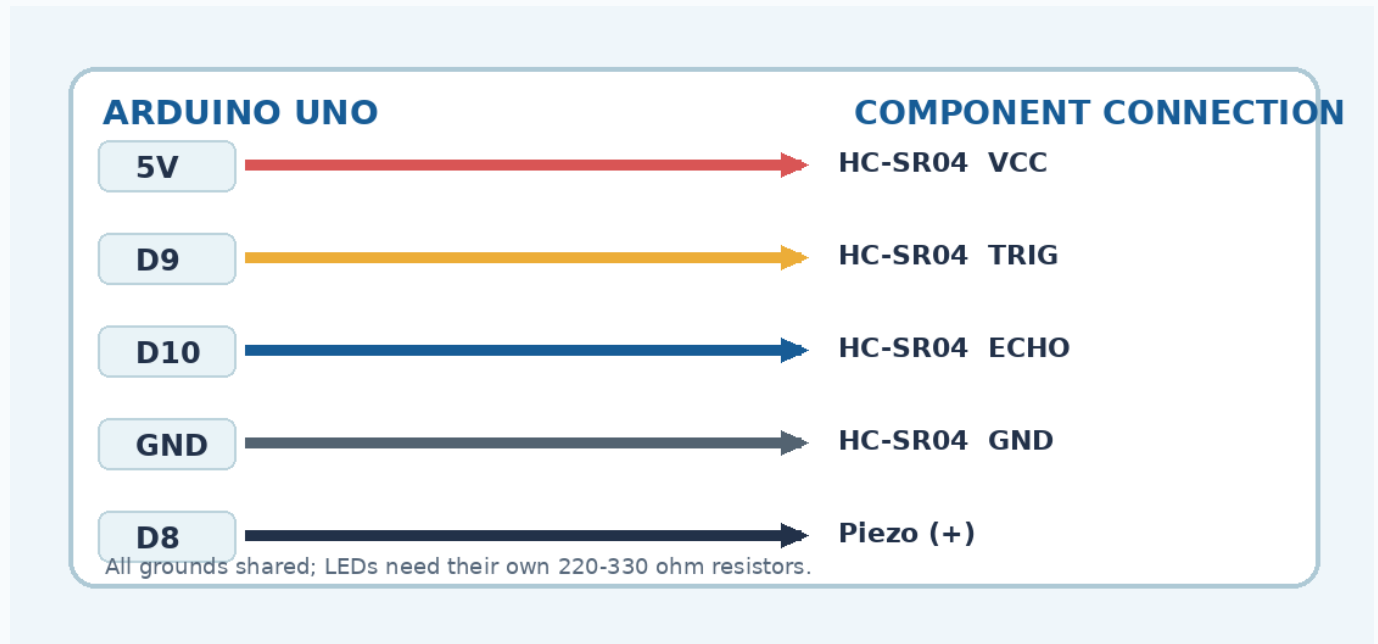
ARDUINO C++ / COPY OR DISCUSS

```
if (duration == 0) {  
  Serial.println("No valid echo");  
} else {  
  Serial.print("Distance: ");  
  Serial.println(distance);  
}
```

Wire and test the piezo

13 / HARDWARE CHECKPOINT

Keep the sensor wiring; add a small passive piezo on the breadboard.



POSITIVE

Piezo (+) → D8.

GROUND

Piezo (-) → common Arduino GND.

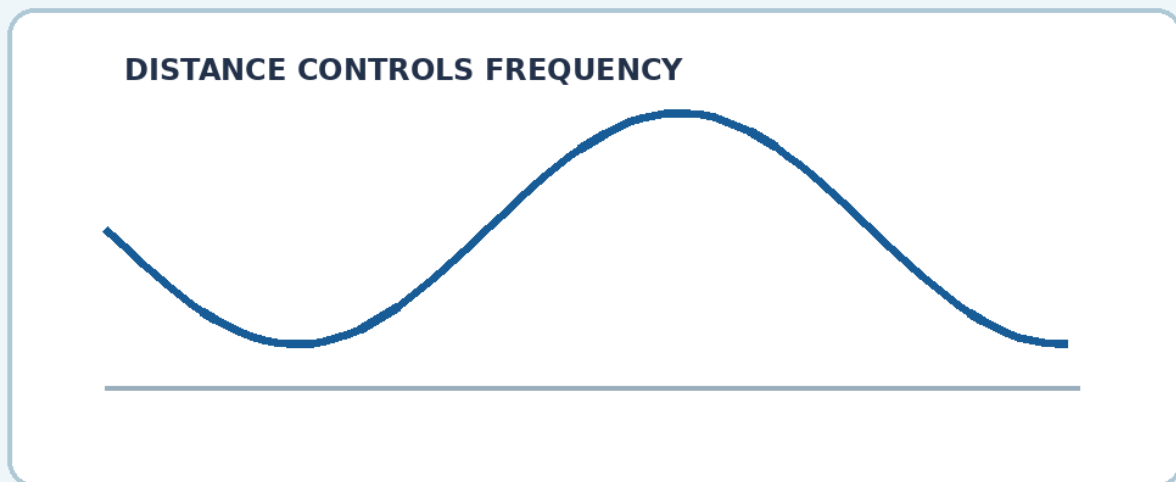
ARDUINO C++ / COPY OR DISCUSS

```
const int BUZZER = 8;
void setup() { pinMode(BUZZER, OUTPUT); }
void loop() {
  tone(BUZZER, 440); delay(800);
  tone(BUZZER, 523); delay(800);
  noTone(BUZZER); delay(300);
}
```

Map distance to pitch

14 / SENSOR-TO-SOUND

Transform a physical measurement into an audible control signal.



RANGE

Use distances from 2 cm to 40 cm.

NEAR

Closer hand means higher pitch, up to 1,000 Hz.

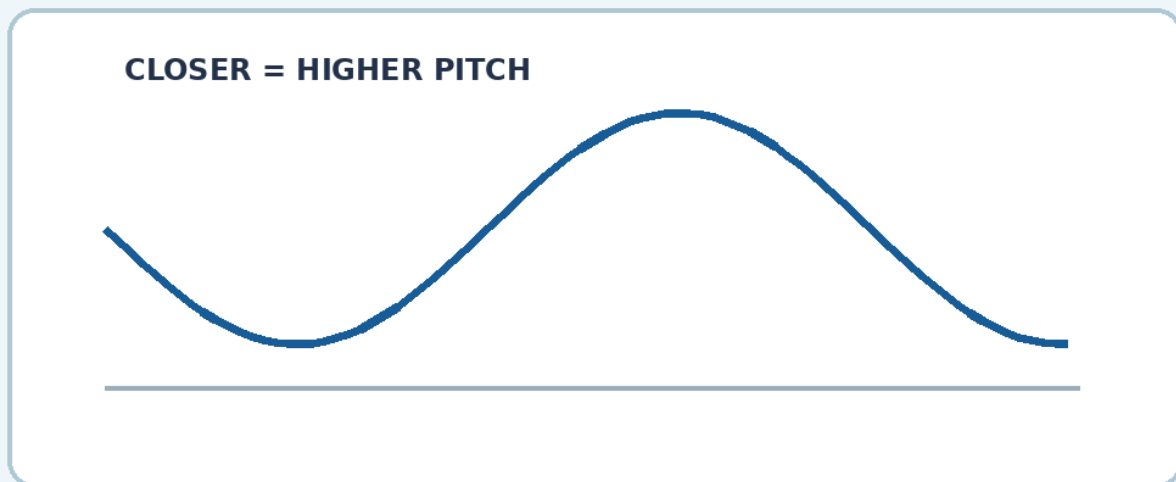
ARDUINO C++ / COPY OR DISCUSS

```
float f = (distance - 2.0) / (40.0 - 2.0);  
f = constrain(f, 0.0, 1.0);  
int hz = 1000 - f * (1000 - 200);  
tone(8, hz);
```

Generate continuous pitches

15 / MODE 1

Continuous mode lets your hand slide through a large range of frequencies.



OUTPUT

`tone(BUZZER_PIN, frequency)` maintains the current sound.

UPDATES

The code recomputes pitch as measurements change.

TIMING

Use a short measurement interval so the instrument remains responsive.

TEST

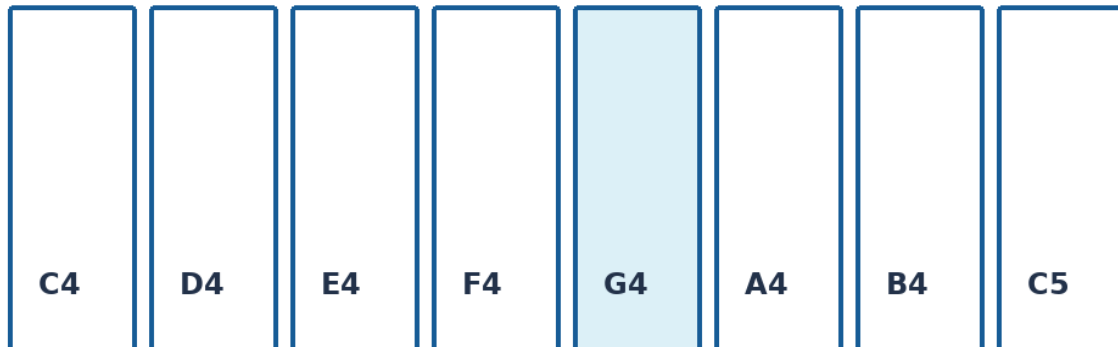
Sweep your hand slowly from 40 cm to 2 cm and listen for smooth pitch movement.

Quantize into notes

16 / MODE 2

A musical scale has distinct notes rather than every possible frequency.

8-NOTE C MAJOR SCALE



SCALE

C4 D4 E4 F4 G4 A4 B4 C5.

ORDER

The closest distance maps to C5, the farthest to C4.

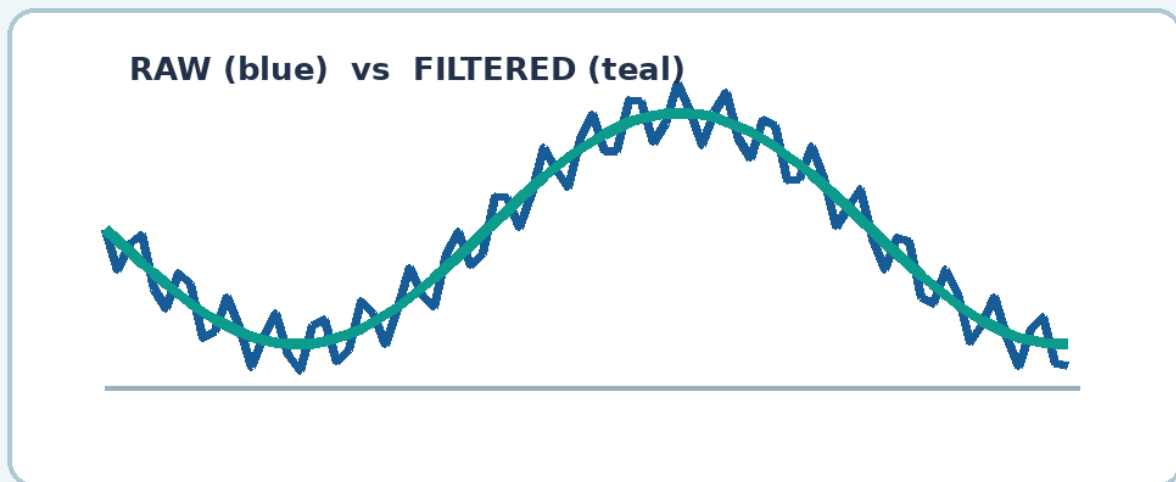
ARDUINO C++ / COPY OR DISCUSS

```
const int NOTES[8] = {  
  523, 494, 440, 392, 349, 330, 294, 262  
};  
int index = constrain((int)(fraction * 8), 0, 7);  
tone(BUZZER_PIN, NOTES[index]);
```

Fix unstable readings

17 / FILTERING

Hand motion and ultrasonic noise make pitch jitter. Smooth the readings.



EMA

$\text{filtered} = 0.25 \times \text{raw} + 0.75 \times \text{previous_filtered}$.

SMOOTH

Small alpha reduces jitter but increases response delay.

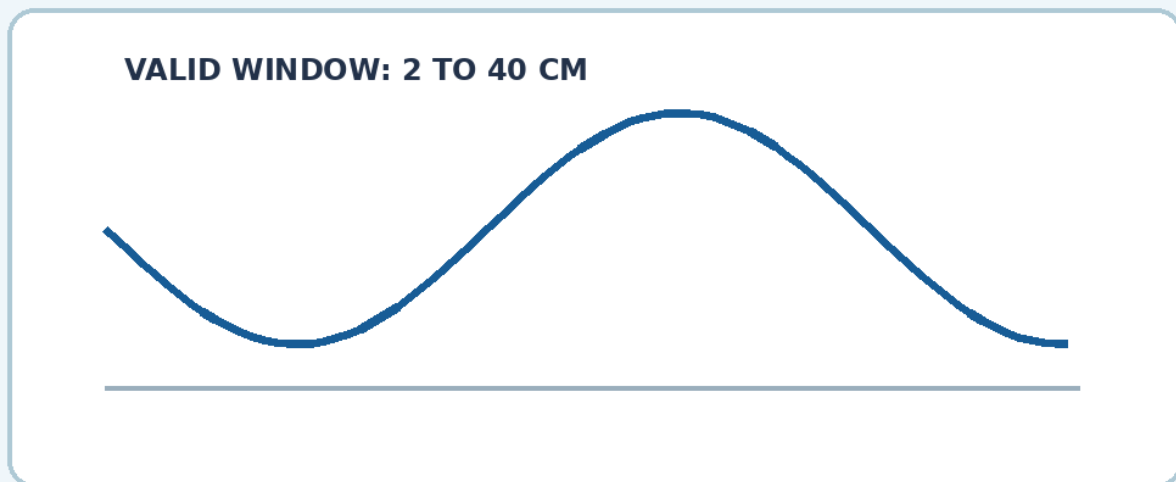
ARDUINO C++ / COPY OR DISCUSS

```
const float ALPHA = 0.25;
if (!filterInitialized) {
  filteredDistance = distance;
  filterInitialized = true;
} else {
  filteredDistance = ALPHA * distance
    + (1.0 - ALPHA) * filteredDistance;
}
```

Decide when to play

18 / VALID RANGE

The instrument should be silent if no valid hand is within the operating range.



SILENCE

Mute measurements outside 2–40 cm or with no valid echo.

CONFIRM

Two consecutive invalid samples trigger mute.

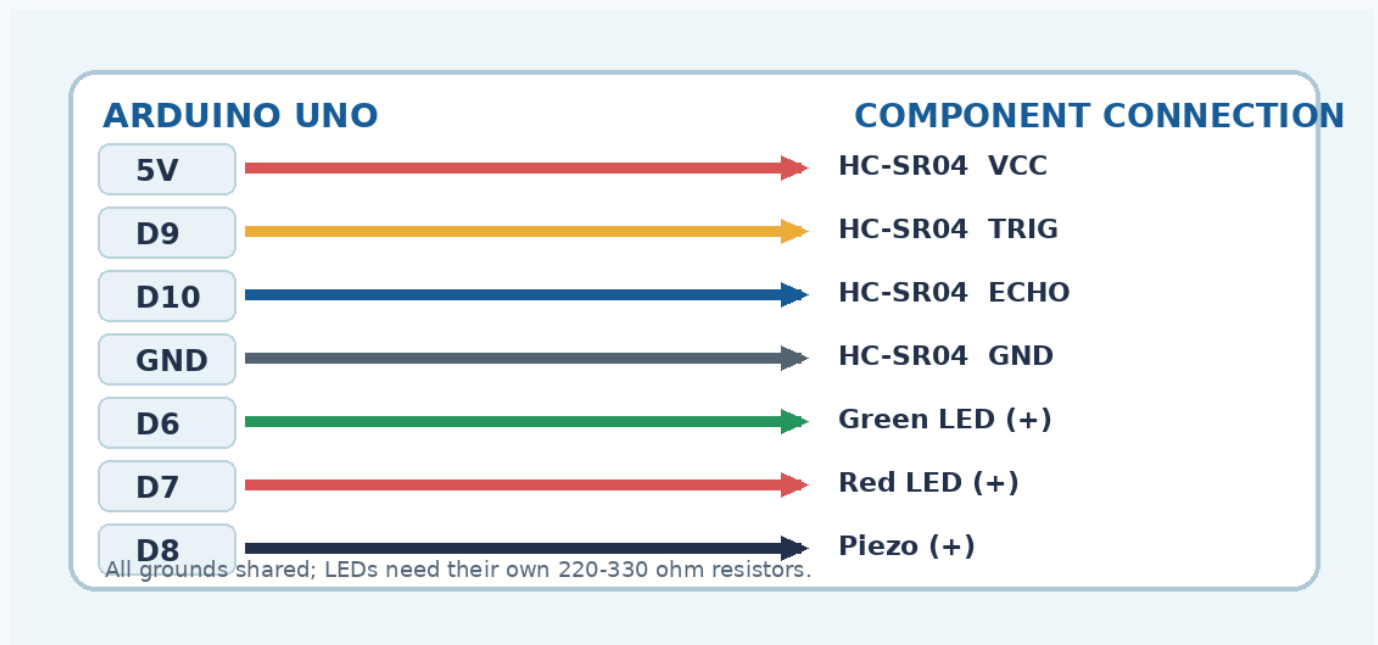
ARDUINO C++ / COPY OR DISCUSS

```
if (distance < 2.0 || distance > 40.0) {  
  if (++invalidReadings >= 2) {  
    noTone(BUZZER_PIN);  
    filterInitialized = false;  
  }  
  delay(70); return;  
}  
invalidReadings = 0;
```

Wire both feedback LEDs

19 / HARDWARE CHECKPOINT

Give participants an immediate visual indicator of whether the instrument is active.



GREEN

Anode → D6; cathode → resistor → common GND.

RED

Anode → D7; cathode → a separate resistor → common GND.

PLAYING

Green on, red off.

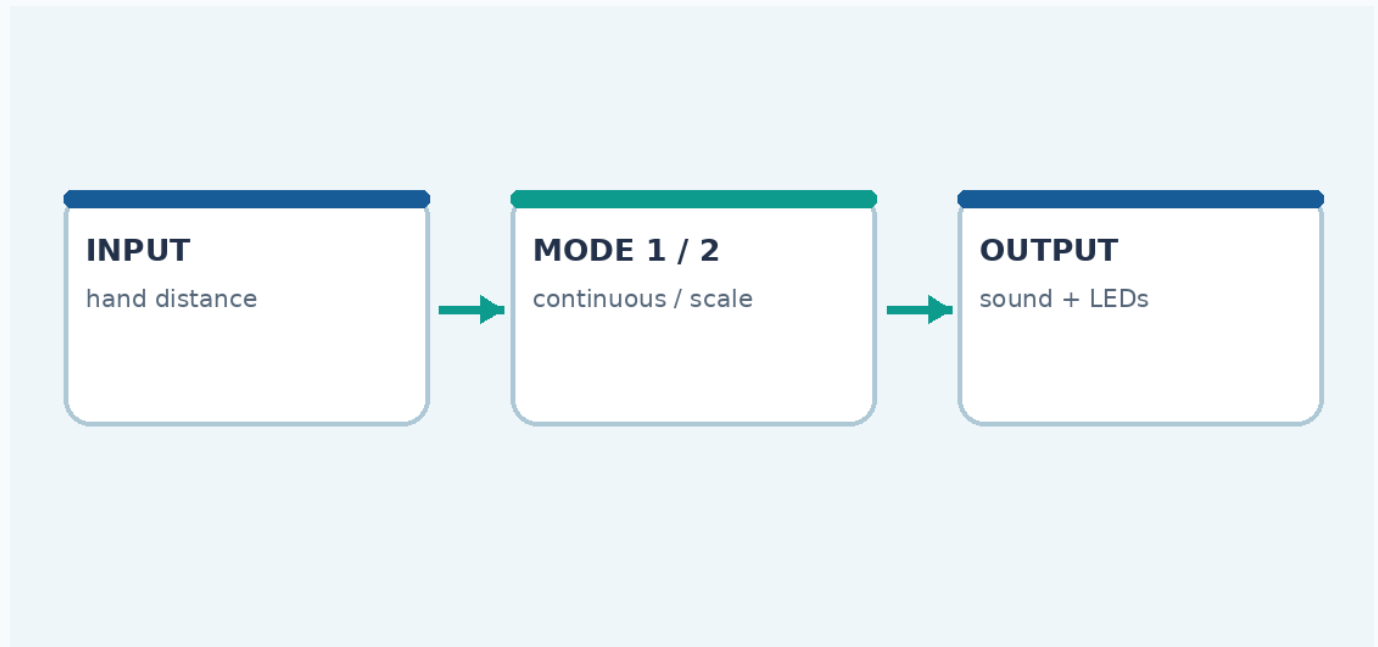
SILENT

Red on, green off.

Switch modes without flashing

20 / SERIAL COMMANDS

A single character from the computer chooses the sound-generation algorithm.



SEND 1

Continuous pitch mode.

SEND 2

Eight-note C major scale mode.

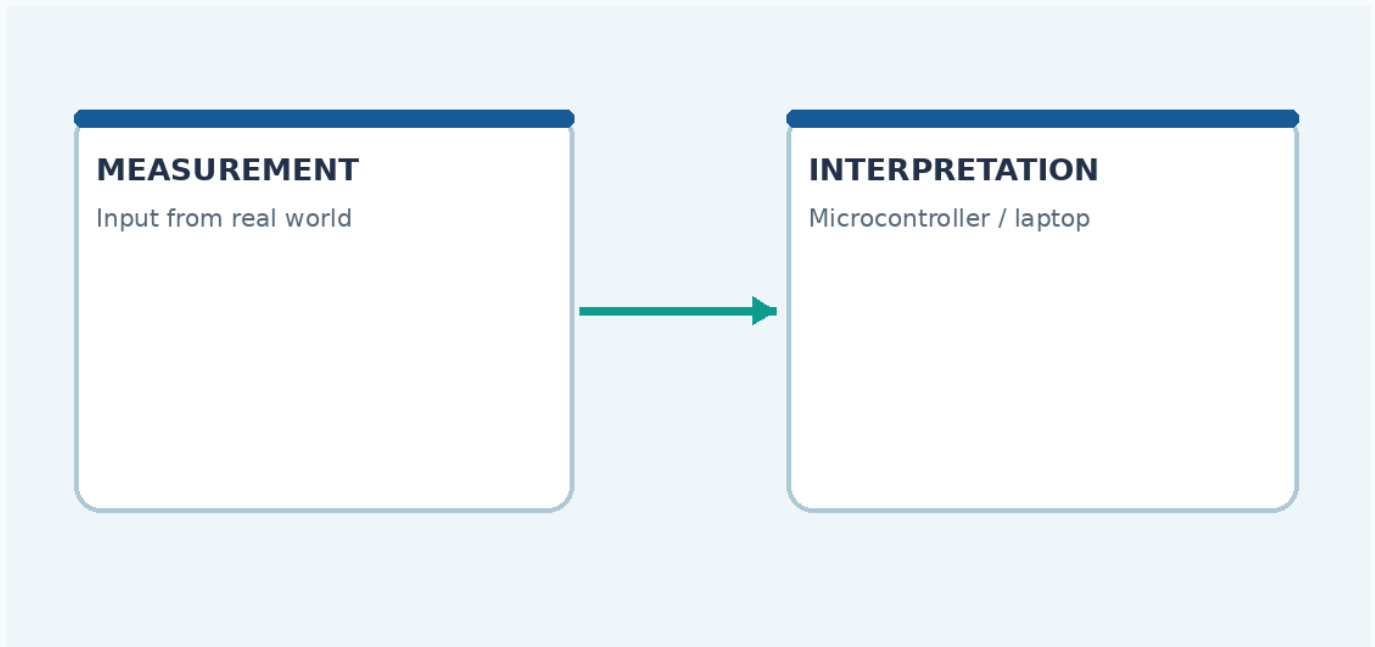
ARDUINO C++ / COPY OR DISCUSS

```
if (Serial.available() > 0) {  
  char c = Serial.read();  
  if (c == '1' || c == '2') {  
    instrumentMode = c - '0';  
    filterInitialized = false;  
    invalidReadings = 0;  
  }  
}
```

Serial data for the Mac app

21 / EMBEDDED TELEMETRY

The Python dashboard reads the descriptive status messages from this exact sketch.



RAW

Unfiltered distance in centimetres.

FILTERED

Smoothed control distance.

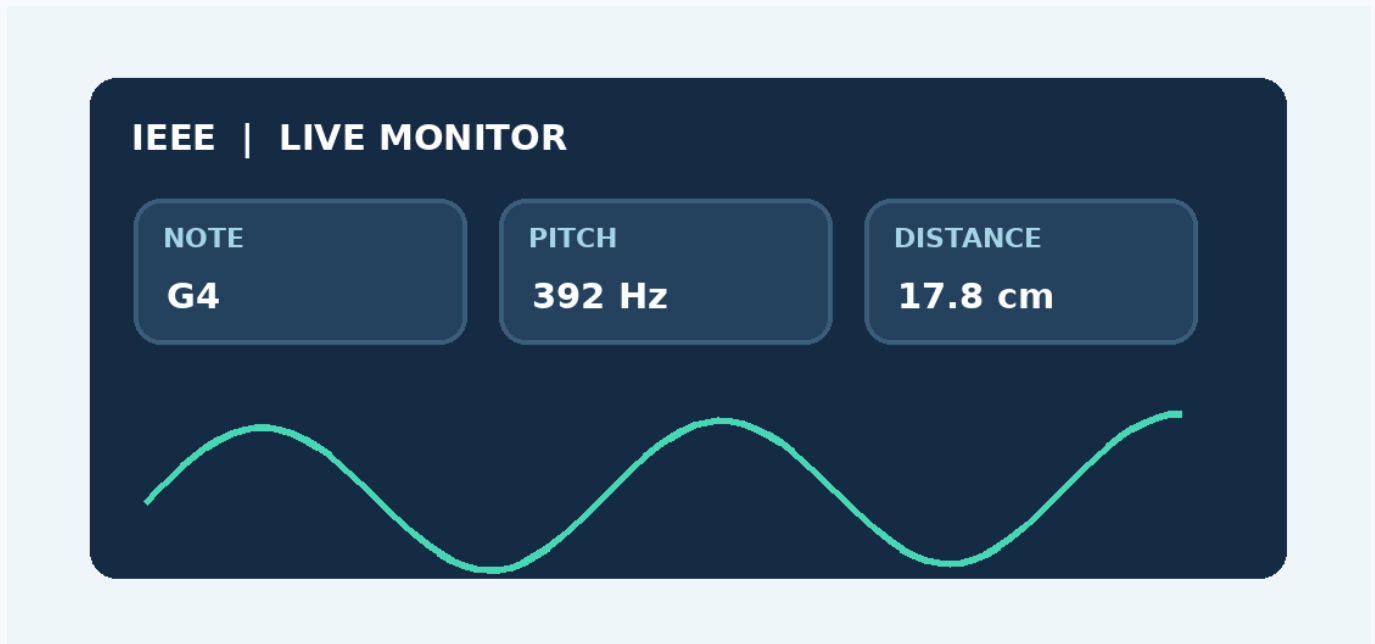
ARDUINO C++ / COPY OR DISCUSS

```
Serial.print("Raw: "); Serial.print(distance);  
Serial.print(" cm | Filtered: ");  
Serial.print(filteredDistance);  
Serial.print(" cm | Mode: ");  
Serial.print(instrumentMode);  
Serial.print(" | Note: "); Serial.print(noteName);  
Serial.print(" | Frequency: ");  
Serial.print(frequency); Serial.println(" Hz");
```

The live music dashboard

22 / LAPTOP INTEGRATION

The Arduino makes the sound; the laptop makes the operation visible.



DISPLAY

Current note, frequency, filtered distance, and sensor state.

KEYBOARD

A visual C-major keyboard highlights the active note.

GRAPHS

Compare raw and filtered distance against frequency changes.

CONTROL

Switch modes through the app over USB.

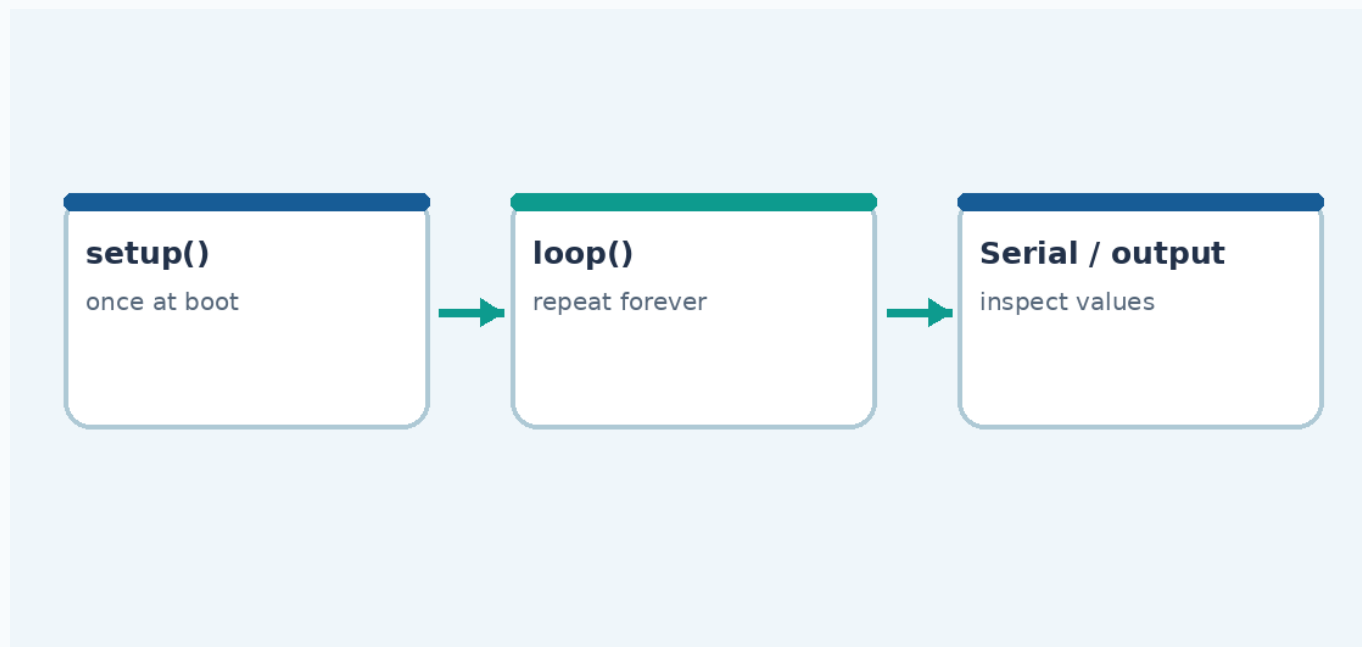
GITHUB

Get the completed Python file from your instructor's GitHub repository.

Full firmware: configuration

23 / REFERENCE CODE A

Combine all three reference-code pages into one sketch, in order.



BEGIN

Define pins, thresholds, notes and persistent state.

CONSTANTS

Pin assignments match the wiring in this guide.

ARDUINO C++ / COPY OR DISCUSS

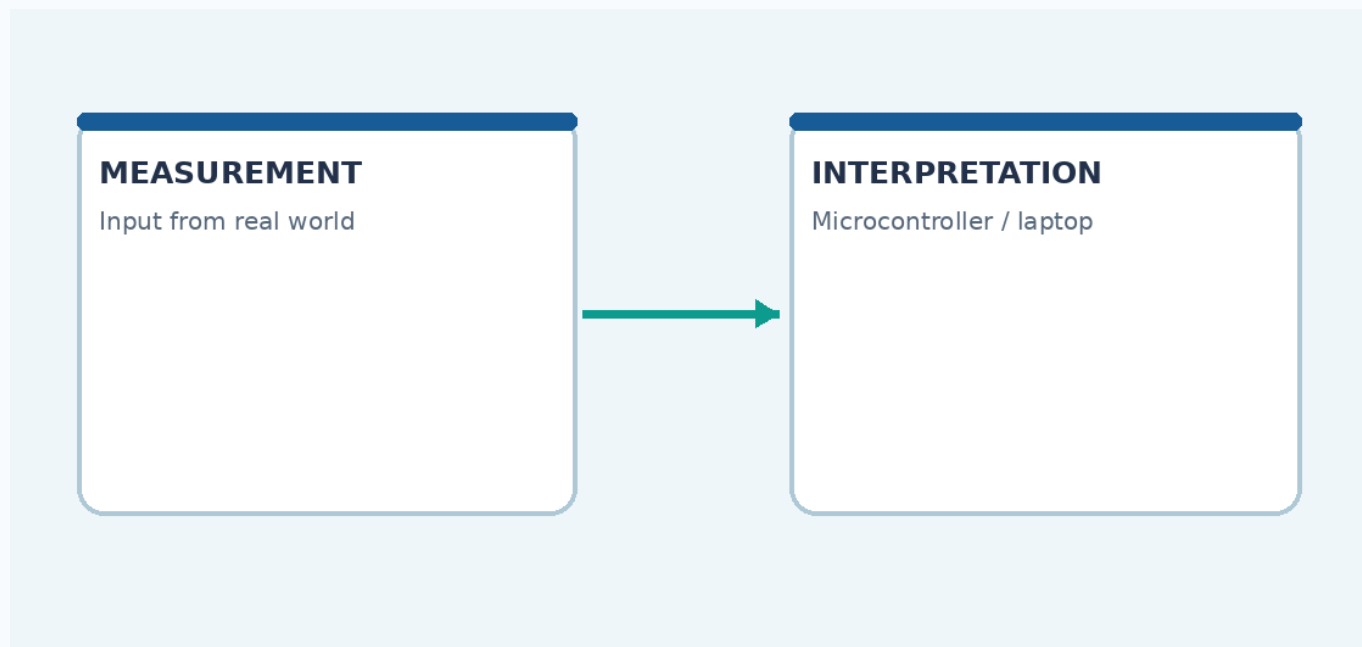
```
const int TRIG=9, ECHO=10, BUZZER=8;
const int GREEN=6, RED=7;
const float MIN_D=2.0, MAX_D=40.0;
const float ALPHA=0.25;
const int NOTES[8]={523,494,440,392,349,330,294,262};
const char* NAMES[8]={"C5","B4","A4","G4",
  "F4","E4","D4","C4"};
int mode=2, invalid=0;
float filtered=0;
bool initialized=false;

void setup() {
  Serial.begin(9600);
  pinMode(TRIG,OUTPUT); pinMode(ECHO,INPUT);
  pinMode(BUZZER,OUTPUT);
  pinMode(GREEN,OUTPUT); pinMode(RED,OUTPUT);
  digitalWrite(TRIG,LOW);
  digitalWrite(GREEN,LOW); digitalWrite(RED,HIGH);
}
```

Full firmware: sensing + modes

24 / REFERENCE CODE B

Measure range and decide whether there is a playable target.



INVALID

ARDUINO C++ / COPY OR DISCUSS

```

float readCm() {
  digitalWrite(TRIG,LOW); delayMicroseconds(2);
  digitalWrite(TRIG,HIGH); delayMicroseconds(10);
  digitalWrite(TRIG,LOW);
  unsigned long t=pulseIn(ECHO,HIGH,25000UL);
  return t ? t*0.0343/2.0 : -1.0;
}

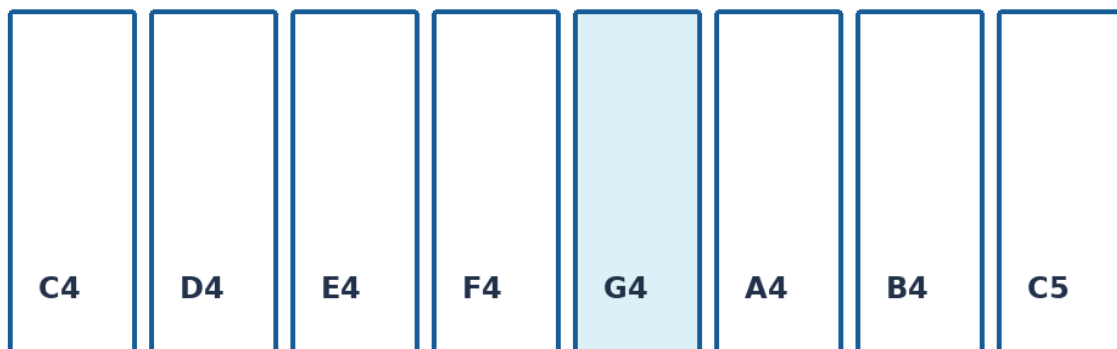
void loop() {
  if (Serial.available()) {
    char c=Serial.read();
    if(c=='1' || c=='2') {
      mode=c-'0'; initialized=false; invalid=0;
    }
  }
  float d=readCm();
  if (d<MIN_D || d>MAX_D) {
    if(++invalid>=2) {
      noTone(BUZZER); initialized=false;
      digitalWrite(GREEN,LOW);
      digitalWrite(RED,HIGH);
      Serial.println("OUT OF RANGE - SILENT");
    }
    delay(70); return;
  }
  invalid=0;
  if(!initialized) {filtered=d; initialized=true;}
}
  
```

Full firmware: audio + output

25 / REFERENCE CODE C

Finish loop(): map filtered distance, play the tone, update LEDs and send telemetry.

8-NOTE C MAJOR SCALE



MAP

Normalized 0–1 distance value maps into continuous pitch or eight notes.

REPORT

The telemetry text matches the Python dashboard.

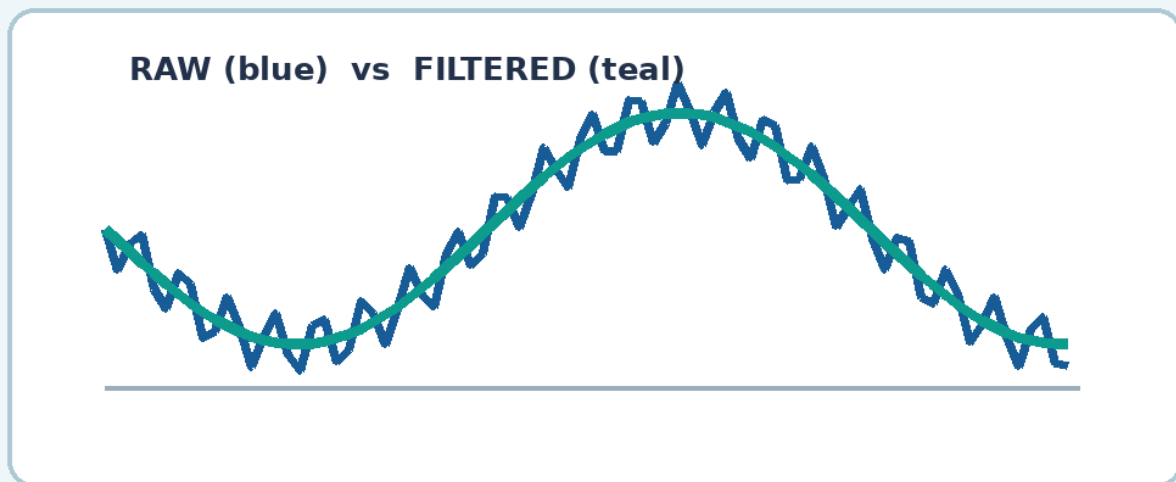
ARDUINO C++ / COPY OR DISCUSS

```
float f=constrain((filtered-MIN_D)/(MAX_D-MIN_D),0.0,1.0);
int hz; const char* name="CONTINUOUS";
if(mode==1) hz=1000-(int)(f*800);
else {
  int i=constrain((int)(f*8),0,7);
  hz=NOTES[i]; name=NAMES[i];
}
tone(BUZZER,hz);
digitalWrite(GREEN,HIGH); digitalWrite(RED,LOW);
Serial.print("Raw: "); Serial.print(d);
Serial.print(" cm | Filtered: "); Serial.print(filtered);
Serial.print(" cm | Mode: "); Serial.print(mode);
Serial.print(" | Note: "); Serial.print(name);
Serial.print(" | Frequency: ");
Serial.print(hz); Serial.println(" Hz");
delay(70);
}
```

Experiment: tune the filter

26 / ENGINEERING LAB

Change one parameter at a time and record how the instrument feels.



ALPHA 0.10

More stable but noticeably slower.

ALPHA 0.25

Recommended starting point.

ALPHA 0.60

More responsive, but may jump between notes.

METHOD

Test the same hand movement for each setting and document your observation.

Reliability and edge cases

27 / VERIFICATION

Test the instrument as a system rather than trusting a successful upload.

TEST CASE

TEST A

TEST B

TEST C

TEST D

TEST E

RESULT

☐

PASS / FAIL

☐

PASS / FAIL

☐

PASS / FAIL

☐

PASS / FAIL

☐

PASS / FAIL

TEST A

Near 2-5 cm: pitch should rise without invalid readings.

TEST B

At 20 cm: stable tone when the hand is still.

TEST C

Beyond 40 cm: mute after two invalid samples.

TEST D

Switch between modes using 1 and 2.

TEST E

Move around scale boundaries and observe note changes.

Debugging field guide

28 / TROUBLESHOOTING

Start from power and simple outputs before debugging more complex software.

1 CHECK POWER

2 CHECK PINS

3 CHECK SERIAL

4 CHECK LOGIC

NO SOUND

Verify passive piezo, D8, GND, and the 440/523 Hz test sketch.

NO DISTANCE

Verify HC-SR04 connections to 5V, GND, D9 and D10.

NOTES JITTER

Check object geometry and adjust EMA alpha.

NO MUTE

Aim away from walls or nearby objects within 40 cm.

DASHBOARD FAILS

Close Serial Monitor and check the correct port and baud rate.

Take it further

29 / CREATIVE CHALLENGE

A 90-minute build can become an independently developed embedded audio project.



LEVEL 1

Add a third mode with a pentatonic scale.

LEVEL 2

Add note-boundary hysteresis to avoid rapidly switching keys.

LEVEL 3

Record melody sessions and compare raw/filtered data.

LEVEL 4

Use a different sensor or a real audio-output stage for better expressiveness.

Showcase your synthesizer

30 / COMPLETION

You have built an ultrasonic controller that converts hand motion into electronic music.



HARDWARE

Arduino Uno, HC-SR04, passive piezo and two LEDs.

FIRMWARE

Ranging, filtered distance, continuous pitch, C-major scale, and serial mode controls.

DASHBOARD

Download the prebuilt Python studio from the instructor-provided GitHub repository.

PORTFOLIO

Capture a photo, 20-second demo, a wiring diagram, and your test notes.