Ultrasonic Range Finder in TinkerCAD

Building a Distance Detection System in Tinkercad

This project will guide you through setting up an Arduino Uno with a Ping ultrasonic range finder to detect object distances, which will then be displayed in Tinkercad's Serial Monitor.

1. Getting Started with Tinkercad Circuits

Tinkercad Circuits is a web-based simulator that allows you to build and test electronic circuits, including those with Arduino.

• Access Tinkercad: Navigate to www.tinkercad.com and log in.

• Create a New Circuit: From the main Tinkercad page, click on "Circuits" in the left-hand menu, then select "Create new Circuit".

• Rename Your Circuit: Tinkercad assigns random names to new circuits. To give it a more meaningful name, click on the generated name in the upper left corner and change it (e.g., "Ultrasonic Distance Detector").

Basic Navigation:

◦ Zoom and Pan: Use your scroll wheel to zoom in and out, and click and drag a blank area to pan across the screen.

◦ Rotate: Select a component and click the rotate tool (top left) to rotate it.

◦ Delete/Undo/Redo: Use the delete button to remove components, or the undo/redo buttons for immediate corrections.

2. Adding Components to Your Workplane

You will drag and drop components from the right-side menu onto your editing area.

• Arduino Uno R3: In the component selector (scroll box on the right), type "Arduino Uno R3" into the search bar. Click on "Arduino Uno R3" and then click in the empty space to the left to place the board. The Arduino Uno R3 is a versatile development board equipped with ATmega328P and ATmega16U2 processors, making it ideal for electronics and coding introductions.

• Breadboard: Search for "breadboard" and place a "Breadboard Small" next to your Arduino Uno. Breadboards are rectangular plastic boards with tiny holes that allow you to easily insert and connect electronic components for prototyping circuits without soldering.

• Ultrasonic Range Finder (Ping Sensor): Search for "ultrasonic" and locate the "Ping Ultrasonic Range Finder" or "SEN136B5B". Drag and drop it onto your workplane.

3. Wiring the Circuit

Making correct electrical connections is crucial for your circuit to function.

• Powering the Breadboard:

◦ Locate the "5V" pin on your Arduino board (bottom row of pins, next to "GND") and click on it. This starts a wire.

◦ Drag the wire to the red (+) rail on your breadboard and click to connect. Change the wire colour to red using the inspector that appears after clicking the wire. This rail is typically used for positive power connections.

◦ Locate any "GND" pin on your Arduino (there are several, e.g., next to 5V, or above digital pins). Click on it to start another wire.

◦ Drag this wire to the blue or black (-) rail on your breadboard and click to connect. Change the wire colour to black. This rail is used for ground connections.

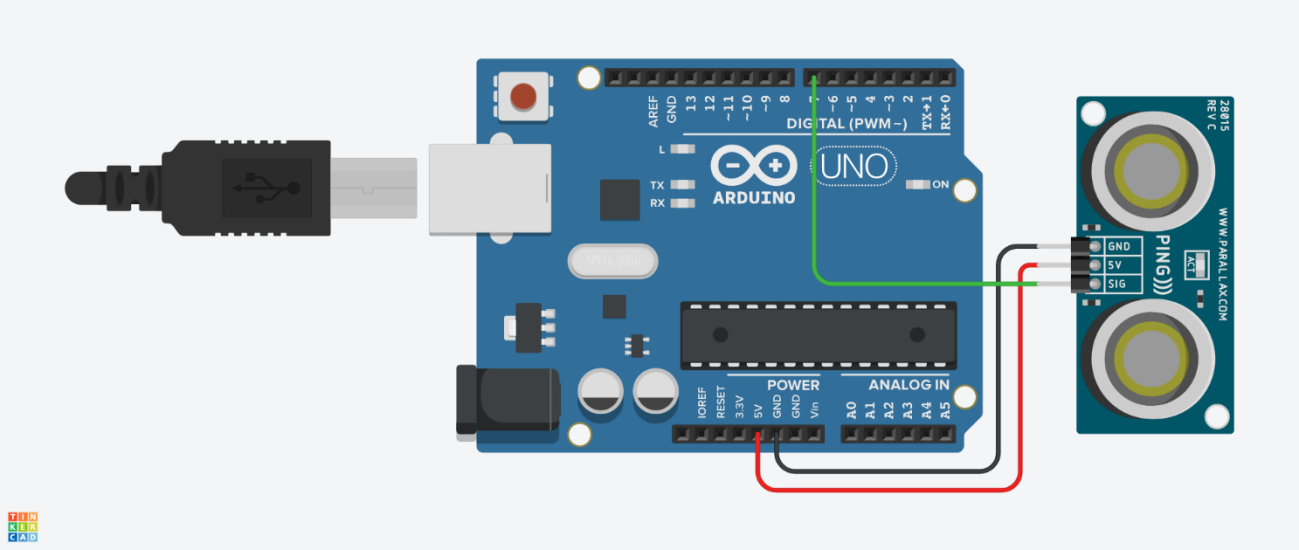
◦ Important: The holes along the (+) and (-) rails of the breadboard are connected vertically. The middle rows of the breadboard are connected horizontally in sets of five holes, with a gap in the middle.

• Connecting the Ultrasonic Sensor:

◦ Sensor's 5V pin: Connect this pin to any hole on the red (+) rail of your breadboard.

◦ Sensor's GND pin: Connect this pin to any hole on the black (-) rail of your breadboard.

◦ Sensor's SIG pin: Connect this pin to Digital Pin 7 on your Arduino board. This pin will be used for both sending the ultrasound pulse and listening for the echo.



4. Coding the Arduino

The Arduino is programmed using a C++ based language.

• Open the Code Editor: Click the "Code" button in the upper right corner of the Tinkercad interface. You can choose between "Blocks" or "Text" code. For this project, we'll use "Text". If prompted, press "Continue".

• Delete Existing Code: Highlight all default code and delete it.

• Enter the Code: Type (or copy-paste) the following code into the editor. This code is adapted from the Ping Sensor guide:

```
/*

  Ping))) Sensor
```

```
 This sketch reads a PING))) ultrasonic rangefinder and returns the distance
 to the closest object in range. To do this, it sends a pulse to the sensor to
 initiate a reading, then listens for a pulse to return. The length of the
 returning pulse is proportional to the distance of the object from the sensor.

 The circuit:
 - +V connection of the PING))) attached to +5V
 - GND connection of the PING))) attached to ground
 - SIG connection of the PING))) attached to digital pin 7

 created 3 Nov 2008
 by David A. Mellis
 modified 30 Aug 2011
 by Tom Igoe

 This example code is in the public domain.

 https://www.arduino.cc/en/Tutorial/BuiltInExamples/Ping
*/

// this constant won't change. It's the pin number of the sensor's output:
const int pingPin = 7; [34]

void setup() {
 // initialize serial communication:
 Serial.begin(9600); [34, 37, 38]
}

void loop() {
 // establish variables for duration of the ping, and the distance result
 // in inches and centimeters:
```

```
  long duration, inches, cm; [34]

  // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  pinMode(pingPin, OUTPUT); [34, 39]
  digitalWrite(pingPin, LOW); [34, 39]
  delayMicroseconds(2); [34, 39]
  digitalWrite(pingPin, HIGH); [34, 39]
  delayMicroseconds(5); [5, 39]
  digitalWrite(pingPin, LOW); [5, 39]

  // The same pin is used to read the signal from the PING))): a HIGH pulse
  // whose duration is the time (in microseconds) from the sending of the ping
  // to the reception of its echo off of an object.
  pinMode(pingPin, INPUT); [5, 39]
  duration = pulseIn(pingPin, HIGH); [5, 40]

  // convert the time into a distance
  inches = microsecondsToInches(duration); [5]
  cm = microsecondsToCentimeters(duration); [5]

  Serial.print(inches); [5, 41]
  Serial.print("in, "); [5]
  Serial.print(cm); [5]
  Serial.print("cm"); [5]
  Serial.println(); [5, 41]

  delay(100); [35]
}

long microsecondsToInches(long microseconds) {
```

```
  // According to Parallax's datasheet for the PING))), there are 73.746
  // microseconds per inch (i.e. sound travels at 1130 feet per second).
  // This gives the distance travelled by the ping, outbound and return,
  // so we divide by 2 to get the distance of the obstacle.
  // See: https://www.parallax.com/package/ping-ultrasonic-distance-sensor-downloads/
  return microseconds / 74 / 2; [35]
}

long microsecondsToCentimeters(long microseconds) {
  // The speed of sound is 340 m/s or 29 microseconds per centimeter.
  // The ping travels out and back, so to find the distance of the object we
  // take half of the distance travelled.
  return microseconds / 29 / 2; [36]
}
```

Code Explanation:

• const int pingPin = 7;: This line defines that the ultrasonic sensor's signal pin is connected to digital pin 7 on the Arduino.

• void setup(): This function runs once when the Arduino starts. Serial.begin(9600); initializes serial communication, allowing the Arduino to send data (like distance readings) back to your computer (Tinkercad's Serial Monitor) at a baud rate of 9600.

• void loop(): This function runs repeatedly (infinitely) after setup() is complete.

◦ Triggering the Ping Sensor: The lines using pinMode(pingPin, OUTPUT);, digitalWrite(pingPin, LOW/HIGH);, and delayMicroseconds(); send a short pulse from pin 7 to trigger the ultrasonic sensor to emit a sound wave.

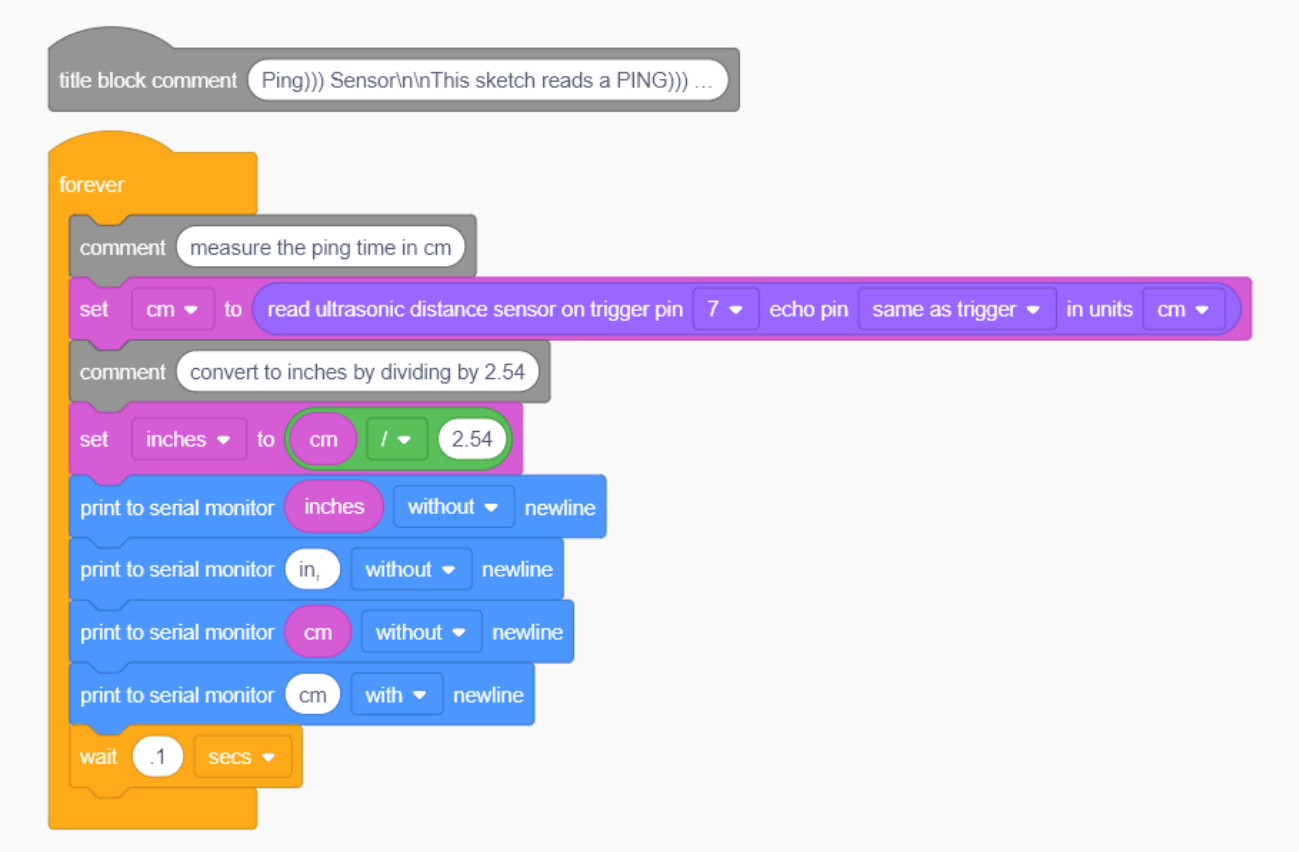
◦ Reading the Echo: pinMode(pingPin, INPUT); changes pin 7 to an input. duration = pulseIn(pingPin, HIGH); then measures the length of the high pulse received on the same pin. This duration represents the time it took for the sound wave to travel to an object and bounce back.

◦ Calculating Distance: The microsecondsToInches() and microsecondsToCentimeters() functions convert this measured time (duration) into distance. These calculations account for the speed of sound and divide the total travel time by two because the sound travels to the object and back.

◦ Displaying Results: Serial.print() and Serial.println() commands send the calculated distance values (in inches and centimetres) to the Serial Monitor. Serial.println() adds a new line after each reading.

◦ delay(100);: This command introduces a small pause between readings.

Block Code:



5. Simulating Your Circuit

Once your circuit is wired and coded, you can run the simulation.

• Start Simulation: Click the "Start Simulation" button located in the top right of the Tinkercad toolbar.

• Observe Results:

◦ You should see the virtual USB cable connect to your Arduino board, providing power.

◦ To view the distance readings, click on the "Serial Monitor" tab located at the bottom left of the programming screen (when the code editor is open). Here, you will see the calculated distances being continuously printed.

◦ In the simulation, you can click and drag the object in front of the ultrasonic sensor to change the simulated distance and observe how the readings in the Serial Monitor update [Source information not explicitly provided for "dragging object to change distance in Tinkercad simulation," but it's an expected feature of such a simulator].

• Stop Simulation: Click the "Stop Simulation" button when you are finished.

Important Considerations

• Breadboard Connections: Remember that while the power rails run vertically, the main rows (a-e and f-j) are connected horizontally in sets of five holes.

• Troubleshooting: If your circuit doesn't work as expected, double-check all your connections against the diagram and ensure wires are pushed in all the way. Small errors like placing a wire in the wrong row can prevent the entire circuit from functioning.

• Serial Monitor Baud Rate: Ensure the baud rate set in Serial.begin(9600) in your code matches the baud rate selected in the Serial Monitor window if you were using a physical Arduino (Tinkercad usually handles this automatically).

By following these steps, you should be able to successfully build and simulate your distance detection system in Tinkercad.