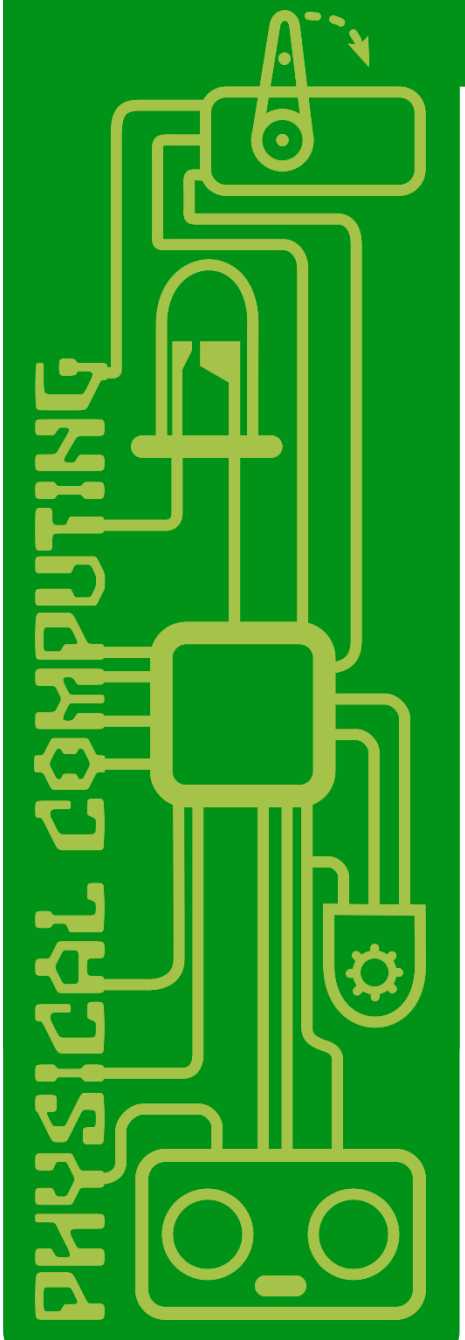
ROBOTICS

Ultrasonic radar



Author: Ivan Novosel

Institution: V. gymnasium, Zagreb

Co-funded by the Erasmus+ Programme of the European Union

Introduction

In this exercise we will learn how to connect several components to make an ultrasonic “radar”. Ultrasonic sensor is used to measure distance, and we will turn the sensor around using a servo motor. The data will be sent from Arduino to the computer using the serial interface. In the end we will adapt the custom serial monitor from the last exercise to make a program that can visualize the data using Python.

You will learn how to:

1. Read the documentation for the HC-SR04 ultrasonic sensor and use it to measure the distance.
2. How to connect multiple components into a working part.
3. How to use Python to visualize data that we read from the sensor.

Ultrasonic sensor

There are multiple ways to measure distance but one of the simplest is using an ultrasonic sensor. All these sensors work in a similar way: **they let out a short sound of high frequency and wait for the echo to return back.** The sound propagates through the air at constant speed of approximately 340 m/s and from that we can calculate the distance travelled by sound.

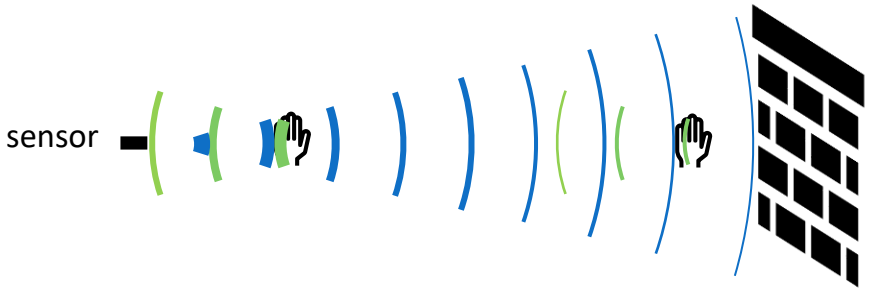
Picture 1 – HC-SR04 ultrasonic sensor has a speaker (T) and receiver (R) on the same board and uses only 4 pins.

The sensor that we are working with is the HC-SR04 that has a ultrasonic speaker and receiver on a single board, there are other sensors that separate these components.

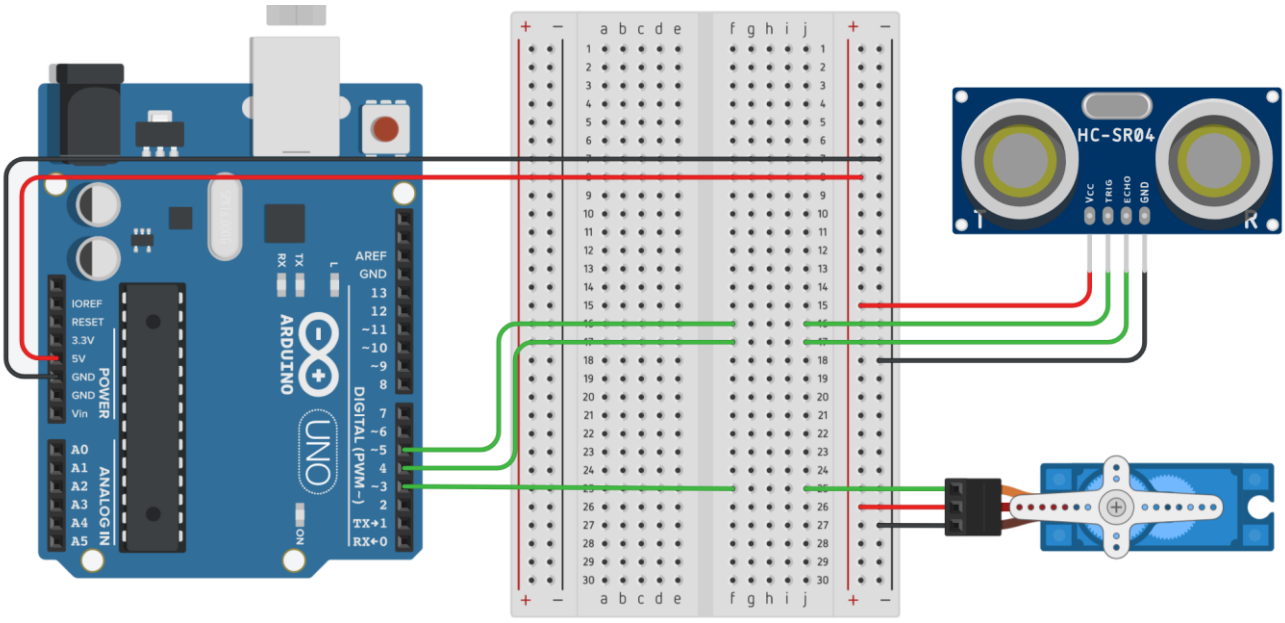
If the object is too far or too small sensor will report the maximal distance.

Ultrasonic sensors are cheap and easy to use but have a few limitations that we must account for when working with them. The reflected signal must be strong enough to detect, so ideally, we would like all of the original sound to reflect back into the receiver. The sound propagates in a conus shaped wave front from the speaker.

That means that at a distance of a few centimeters the sound isn't spread much, and you can get a full reflection from a small object, but if the object is further away too little of the sound is reflected back.



Picture 2 – on the left the small black square marks the position of the sensor. As the ultrasound spreads from the sensor its intensity gets smaller. A hand sized object can reflect most of the wave when it is close, but when it is further away it is just not large enough and will not be detected. At maximal distance you may need a wall sized object to reflect enough of the sound back.



Picture 3 – how to wire up the HC-SR04 ultrasonic sensor and the servo motor together to an Arduino. The servo motor must be connected to a pin with PWM, and the Trig and Echo pins of the sensor can be connected to any I/O pin.

Connect the sensor and the servo motor to the Arduino as shown. After that your task is to read the documentation for the HC-SR04 sensor which can be accessed on the link in the sidebar. **Try to figure out the algorithm on how to calculate the distance by yourself.** Write the ideas down.

HC-SR04 ultrasonic sensor datasheet, URL: https://bit.ly/3NAuibd

Calculating the distance

Functions are parts of code that are given a name that we can call when we need it. All the commands in C++ are functions!

This is a function definition – we give it a name, parameters and we define the return data type.

This part triggers the sensor to send the sound.

pulseIn(pin,state) is used to measure how long a pulse on a certain pin is in milliseconds.

Distance is calculated with the $s = vt$ formula, but we divide with 2 because the sound travels back and forth.

One thing we learned before is that when we are building a larger project it is good to split it into several smaller problems and build up to the final solution. **Another smart idea is to organize the code in functions.** The main advantage of using functions is that we can isolate parts of the code that do a particular thing – a function should be doing only one thing. This lets us stay organized, and it is easier to test the code for errors.

A programming function should have only one task.

This is an example of a function that would return a distance measured with the ultrasound sensor in cm, if we give it the echo and trig pin numbers.

```
float readDistance(int echo, int trig)
{
  long duration = 0;
  float distance = 0;

  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  duration = pulseIn(echo, HIGH);

  distance = duration * 0.034 / 2;
  return distance;
}
```

Now we can call the function by stating its name in the code and giving it the necessary parameters. A function definition should exist before its name is called!

Next step would be to add the code that would read the state of the ultrasonic sensor and send it to the computer over the serial port.

After that we can add the code that will also turn the servo motor back and forth (something like the Sweep example in Arduino IDE).

Reminder: When building a larger program split the work in smaller steps!

If you have problems in any part feel free to check out the code in the simulation but try it by yourself first.

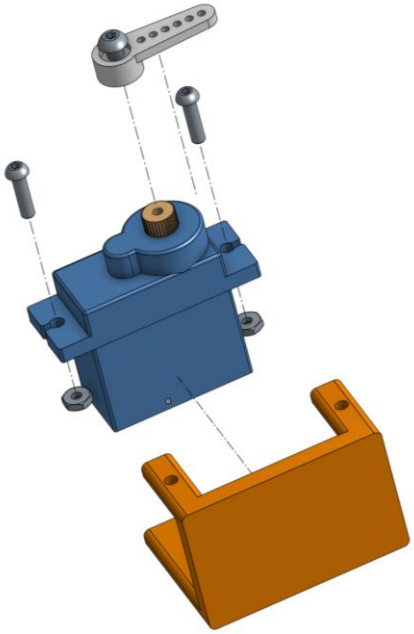
Simulation of **Ultrasonic radar** on Tinkercad, URL: https://bit.ly/3tpsqLB

Connecting the physical parts

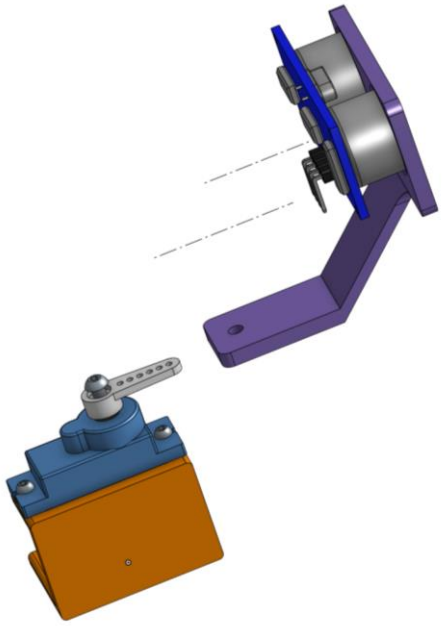
Now that we have a functional program on the Arduino, we need to connect everything so it moves together. You have two 3D printed parts: one is a holder for the ultrasonic sensor, and the other is a holder for the servo.

In the picture below you can see how it fits together but remember that you need to calibrate the servo before you connect everything. This design was made so the 90° position is pointing to the front of the servo.

Reminder: Check the exercise on servo motors for calibration procedure!

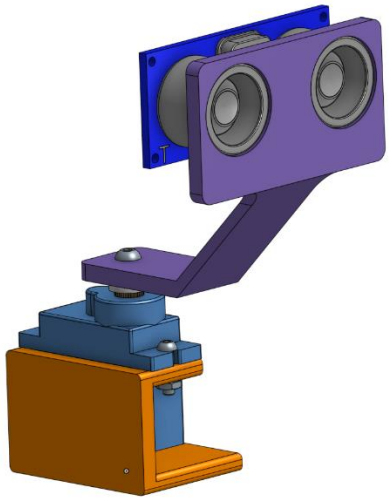


Picture 4 A – first attach the servo holder (orange part) to the servo and attach the horn of the servo in the pictured orientation and do the calibration.



Picture 4 B – insert the ultrasonic servo through the holes on the sensor holder (purple part). Unscrew the horn, insert it in the holder and then screw it back.

The wires are not pictured here, but they need to stay connected to the right places on the breadboard. If you disconnected anything while you were assembling the physical parts, check if everything is working as intended.



Picture 5 – fully assembled parts for the ultrasonic radar. The breadboard, wires and Arduino are not shown here.

Visualization of the data

Code for **Ultrasonic radar visualisation** on Github, URL: https://bit.ly/41sVWgf

Last time you saw an example of how to make a custom serial monitor using pySerial module. We will use that program as a framework to visualize the data coming from the ultrasonic radar. We are sending pairs of data: angle and distance.

We could use any of the modules for drawing in Python, even the simple Turtle module will do nicely. With Tkinter we can make a whole app with a window that can show the data.

It is outside of the scope of this document to explain how these modules for Python work, but we will discuss the idea in class. There is also a finished example of the code that you can access over the link in the sidebar!

- **Ultrasonic sensors use echo to measure distance like bats.**
- **If you measure outside the range of the sensor it will show the maximum distance.**
- **Use functions to make the code manageable.**
- **Function definitions look like:**

```
type_returned name(type parameter)
```