CamJam EduKit Robotics – Distance Sensor

Project Ultrasonic distance measurement

Description In this worksheet you will use an HR-SC04 sensor to measure real world distances.

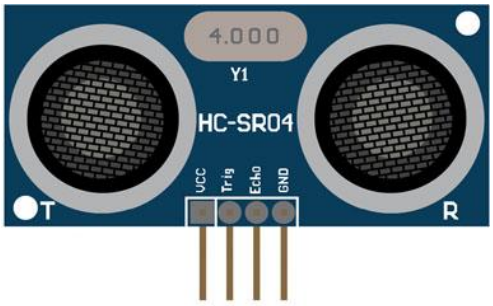
Equipment Required

For this worksheet you will require:

- Your robot, including the line follower installed in the last worksheet
- The HR-SC04 ultrasonic module
- Jumper leads (4 Male-Male)
- 330Ω resistor
- 470Ω resistor

You will be adding the distance sensor to the breadboard used in the previous worksheet, and connecting it to the EduKit Motor Controller Board.

HR-SC04 Ultrasonic Module



The HR-SC04 Ultrasonic Sensor module is used to detect the distance from the sensor to a surface.

The Trigger: When the sensor is triggered by a voltage being applied to the Trig pin, it sends a high pitched (ultrasonic) sound from the speaker marked T (transmit). The voltage is only applied for 1 microsecond (0.0001 seconds).

The Echo: When the sound is heard by the receiver (marked R), the Echo pin is 'taken high', which means that it is made to provide 5 volts.

By timing how long it is between the sound being produced and the sound being detected, and making a calculation, it is possible to work out the distance between the sensor and an object in front of it.

The echo pin should stay high however long it takes the pulse to return. You work out the distance from how long the pulse is using the following formulae:

distance = elapsed * 34326

where:

elapsed is the length of the pulse (the time between trigger and echo) in seconds

34326 is the speed of sound in cm/s

Since the ultrasound has to travel both to the object and echoed back from the object, it has covered twice the distance. Therefore, you need to halve the distance:

distance = distance / 2.0

The HR-SC04 requires 5 volts to work. This is fed from the Raspberry Pi via the EduKit Motor Controller Board. It also outputs 5 volts on the Echo pin. However, the Raspberry Pi GPIO input pins should only be supplied with 3.3 volts. Therefore, you are going to use what is known as a 'voltage divider' to split the output voltage between the GPIO input pin and the ground pin. This is made using resistors on the ultrasonic sensor's output pin, diverting some voltage to the GPIO input pin with the rest going to 'ground'.

Resistors

Resistors are a way of limiting the amount of electricity going through a circuit; specifically, they limit the amount of 'current' that is allowed to flow. The measure of resistance is called the Ohm (Ω), and the larger the resistance, the more it limits the current. The value of a resistor is marked with coloured bands along the length of the resistor body.

The EduKit is supplied with two sets of resistors. There are two 330Ω resistors and two 470Ω resistors. Only one of each is required; extras are supplied as spares. You can identify the resistors by the colour bands along the body. The colour coding will depend on how many bands are on the resistors supplied:

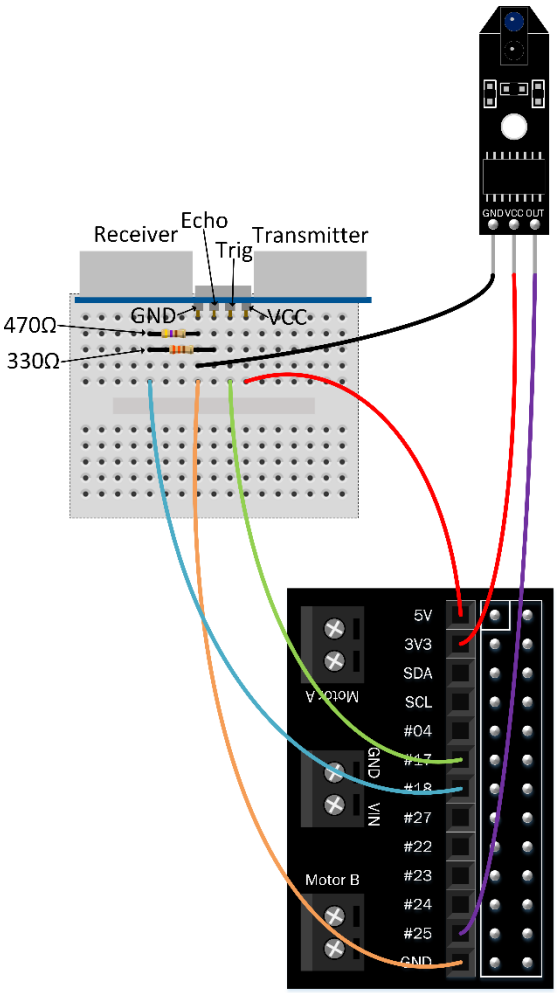
- If there are four colour bands:
  - The 330Ω resistor will be Orange, Orange, Brown, and then Gold.
  - The 470Ω resistor will be Yellow, Violet, Brown, and then Gold.
- If there are five colour bands:
  - The 330Ω resistor will be Orange, Orange, Black, Black, Brown.
  - The 470Ω resistor will be Yellow, Violet, Black, Black, Brown.

The resistors in this circuit will be acting as a 'voltage divider', reducing the voltage of the output from the ultrasonic sensor (5 volts) down to a level that the Raspberry Pi can handle (3.3 volts).

It does not matter which way round you connect the resistors. Current flows in both ways through them.

Note: You may find that your kit has been supplied with THREE sets of resistors. One set is marked Yellow, Violet, Black Brown, Brown (4.7kΩ), and is not required.

Building the Circuit



Push the ultrasonic sensor into the holes on the breadboard, with the pin marked GND in the same column as the jumper wire that goes to the ground of the EduKit Controller Board.

Bend the legs of the two resistors and place them in the breadboard as on the diagram. Ensure that the correct resistors are placed in the right position. The 330Ω resistor (orange-orange-brown) goes between the Echo pin of the sensor and an unused column of the breadboard. The 470Ω resistor (yellow-purple-brown) goes between that same column and the ground (GND) pin. Then connect the breadboard column to socket 18 of the EduKit Controller Board.

Connect the row with the sensors' VCC connection (red wire in the diagram) to the EduKit Controller Board's 5v socket. The sonar module requires 5v to run.

Connect the column with the sensors' trigger (green wire in the diagram) to the EduKit Controller Board's socket marked #17 (The Pi's GPIO pin 17).

Remember: the echo pin of the sensor module is connected to the Raspberry Pi GPIO with resistors and ground because the module uses a +5V level for a "high", but this is too high for the inputs on the GPIO header, which only likes 3.3V. In order to ensure the Pi only gets hit with 3.3V you use a basic "voltage divider" formed with the two resistors.

Code

In a terminal window, change to the EduKitRobotics directory using:

```
cd ~/EduKitRobotics/
```

Create a new text file "5-distance.py" by typing the following:

```
nano 6-distance.py
```

Type in the following code:

```
# CamJam EduKit 3 - Robotics
# Worksheet 6 – Measuring Distance

import RPi.GPIO as GPIO # Import the GPIO Library
import time # Import the Time library

# Set the GPIO modes
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Define GPIO pins to use on the Pi
pinTrigger = 17
pinEcho = 18

print("Ultrasonic Measurement")
```

```
# Set pins as output and input
GPIO.setup(pinTrigger, GPIO.OUT)  # Trigger
GPIO.setup(pinEcho, GPIO.IN)      # Echo

try:
    # Repeat the next indented block forever
    while True:
        # Set trigger to False (Low)
        GPIO.output(pinTrigger, False)

        # Allow module to settle
        time.sleep(0.5)

        # Send 10us pulse to trigger
        GPIO.output(pinTrigger, True)
        time.sleep(0.00001)
        GPIO.output(pinTrigger, False)

        # Start the timer
        StartTime = time.time()

        # The start time is reset until the Echo pin is taken high (==1)
        while GPIO.input(pinEcho)==0:
            StartTime = time.time()

        # Stop when the Echo pin is no longer high - the end time
        while GPIO.input(pinEcho)==1:
            StopTime = time.time()
            # If the sensor is too close to an object, the Pi cannot
            # see the echo quickly enough, so it has to detect that
            # problem and say what has happened
            if StopTime-StartTime >= 0.04:
                print("Hold on there!  You're too close for me to see.")
                StopTime = StartTime
                break

        # Calculate pulse length
        ElapsedTime = StopTime - StartTime

        # Distance pulse travelled in that time is
        # time multiplied by the speed of sound (cm/s)
        Distance = ElapsedTime * 34326

        # That was the distance there and back so halve the value
        Distance = Distance / 2

        print("Distance : %.1f" % Distance)

        time.sleep(0.5)

# If you press CTRL+C, cleanup and stop
except KeyboardInterrupt:
    # Reset GPIO settings
    GPIO.cleanup()
```

Once you have typed all the code and checked it, save and exit the text editor with “Ctrl + x” then “y” then “enter”.

Running the Code

To start the program, type the following into the terminal window:

```
python3 6-distance.py
```

Move an object (e.g. your hand) in front of the sensor and watch the distance change.

If the code does not run correctly there may be an error in the code. Re-edit the code by using the nano editor, typing `nano 6-distance.py`.

Next Steps

The breadboard has double sided tape on the bottom. Remove the wax paper to stick it to the front of your robot so that the ultrasonic sensor points forward.