In this worksheet, we are going to be looking at how we can use the distance sensor that came with your robotics kits. Don't want to be crashing into stuff now do we?

Ultrasonic Sensing

The sensor you have works by using echolocation. If this sounds familiar, it is because both bats and dolphins can also navigate their environments in the same way! Sound outside of the realm of human hearing is emitted from the sensor. If an obstacle is present, the sound will bounce back towards the sensor as shown in Figure 1.

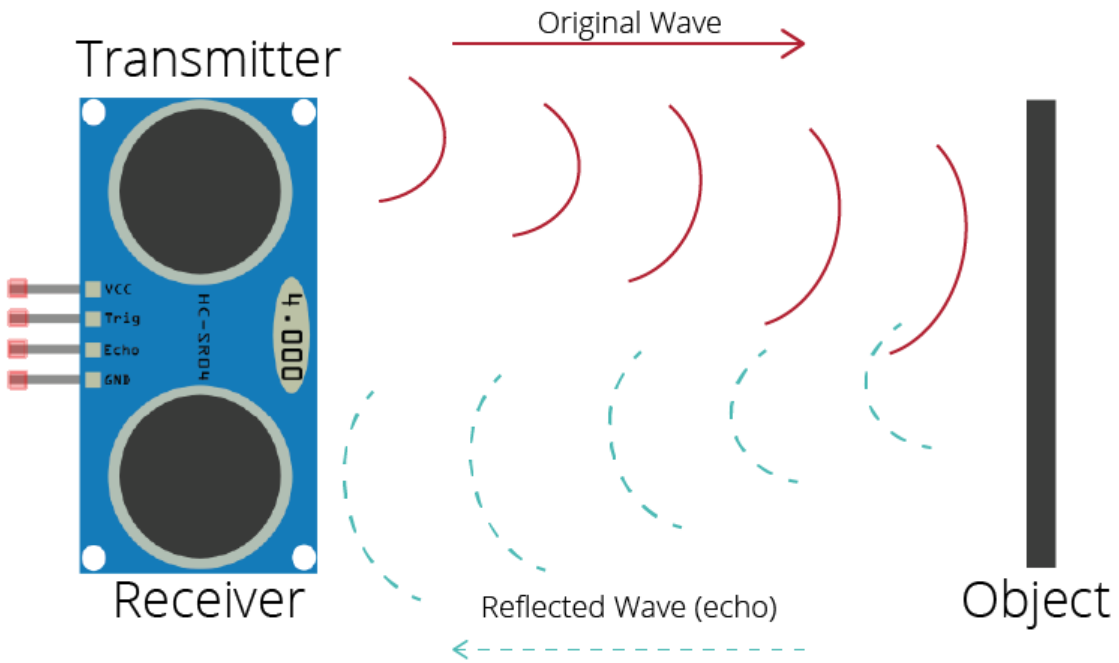

Figure 1 Sound waves reflecting off obstacle (Random Nerd Tutorials, n.d.)

As the speed of sound in air is a constant value, the distance to the object can be calculated using the following sum (Random Nerd Tutorials, n.d.):

$$Distance = \frac{speed\ of\ sound * time\ taken\ for\ sound\ to\ return}{2}$$

$$speed\ of\ sound = 343m/s$$

In the equation above, distance is measured in metres, speed in metres per second and time in seconds. We do not have to use this equation in our code, the distance is calculated for us automatically thanks to the distance function in the GPIO Zero library: 13. API - Input Devices —

GPIO Zero 1.6.2 Documentation (Nuttall & Jones, n.d.). This is another example of how libraries can streamline your code.

Coding

We begin our code just like the test program by importing the necessary libraries, setting our constants and making our objects (Figure 2).

```
# Import necessary Python libraries
from gpiozero import CamJamKitRobot, DistanceSensor

# Motor constants (must be in the range 0 < x <= 1)
FORWARD_SPEED = 0.25
BACKWARD_SPEED = 0.25
TURN_SPEED = 0.4

# Component constants
# These could change depending on where you plug your components onto the GPIO pins of your Pi
TRIGGER_PIN = 17 # Triggers transmission of sound waves
ECHO_PIN = 18 # Listens for sound wave echo.
THRESHOLD = 0.3 # Sets the sensor threshold to trigger at 0.3m (30cm). Objects at and below 30cm treated as obstacles.

# Setting objects so their associated functions can be called
robot = CamJamKitRobot()
distanceSensor = DistanceSensor(echo = ECHO_PIN, trigger = TRIGGER_PIN, threshold_distance = THRESHOLD)
```

Figure 2 Setting up program

We then create the functions to run the motors, just like in our test program from the last worksheet (Figure 3).

```
# Functions to control the motors of the robot
def forward():
    robot.forward(FORWARD_SPEED)

def backward():
    robot.backward(BACKWARD_SPEED)

def left():
    robot.left(TURN_SPEED)

def stop():
    robot.stop()
```

Figure 3 Motor functions

Figure 4 shows where the program begins to differ from what you have written before. We create a "distance" function to output the distance in cm by multiplying the distance returned by the sensor (in metres) by 100. Then, we create a function to avoid any obstacles at or below the threshold we

set in Figure 2. The robot will turn on the spot until the distance returned by the sensor is above the set threshold. Finally, we create a function that moves the robot forward to be used when no obstacles are in range.

```
32 # Prints the distance calculated by the sensor in cm
33 def distance():
34     distance = distanceSensor.distance * 100
35     print(distance)
36
37 # Turn left to avoid obstacle and move in direction where no obstacle is within range
38 def avoid():
39     distance()
40     left()
41     distanceSensor.wait_for_out_of_range()
42     move()
43
44 def move():
45     distance()
46     forward()
```

Figure 4 Distance functions

As shown in Figure 5, “distanceSensor” has two functions: “wait_for_out_of_range()” and “wait_for_in_range()”. These functions are useful as they will prevent the code from advancing until they are triggered. This means that until you place your robot out of range of an obstacle, it will not move. After it is happy there is no obstacle in range, it will begin and continue to move until “wait_for_in_range()” is triggered. The “avoid” function will then be called. If you want to stop the robot from moving and end the program, use ctrl + c.

```
48 while True:
49     # Using "try" with "except" helps handle any errors that occur
50     try:
51         # Wait until robot is out of range of obstacle to begin
52         distanceSensor.wait_for_out_of_range()
53         move()
54
55         # If robot moves within range of obstacle, avoid obstacle
56         distanceSensor.wait_for_in_range()
57         avoid()
58
59     # Press ctrl + c to stop the motors and terminate the program
60     except KeyboardInterrupt:
61         stop()
62         quit()
```

Figure 5 Main event loop

Challenge

- Using what you have learned in this worksheet and the previous worksheets, try building a little maze and programming your robot to drive around it.

END OF WORKSHEET 3

Reference List

Nuttall, B. & Jones, D., n.d. *gpiozero.* [Online]
Available at: https://gpiozero.readthedocs.io/en/stable/#
[Accessed 20 04 2022].

Random Nerd Tutorials, n.d. *Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino.* [Online]
Available at: https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/
[Accessed 20 04 2022].