*Background*

A radar determines the distance between its sensor and an object by sending out a wave, awaiting a response and performing signal processing technique on the feedback. In this lab, we will use a component called HC-SR04, as seen in *Figure 1*.

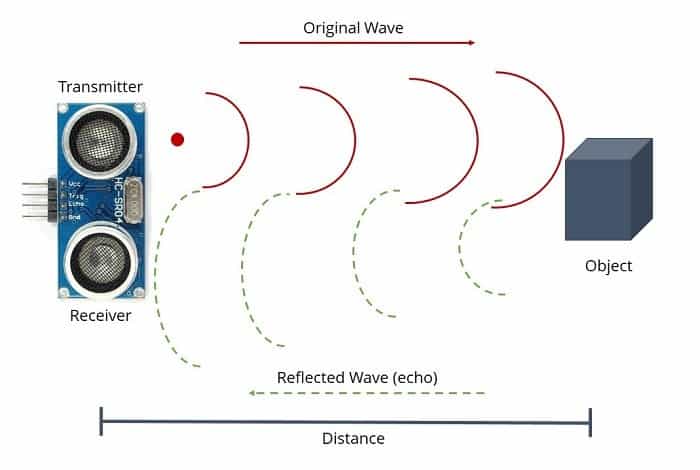


Figure : The HC-SR04 sensor to be used in this lab

The component calculates the distance between its sensor and an object in three steps:

1. The component emits an ultrasonic wave from its transmitter (**Tx**)
2. An object reflects the ultrasonic wave back in the direction of emission
3. The component collects the reflected ultrasonic wave via its receiver (**Rx**)

This is depicted more clearly in *Figure 2.*



Rx

Tx

Figure : The behaviour of a wave emitted by the sensor and reflected by an object

The figure shows that the wave has to make two trips between the sensor and the object: one out (red wave) and one back (green wave). This is taken into consideration during processing, where we use the speed / distance / time equation, seen in *Figure 3*, to calculate the distance between the two objects of interest. We have to modify the equation slightly by swapping *d* with *2d*, which will then produce *Equation 1.*

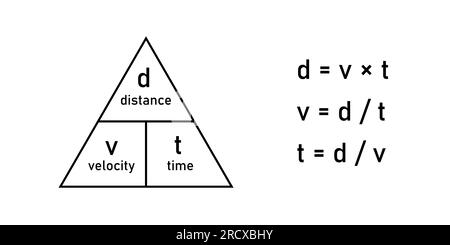


Figure : The velocity / distance / time equation

Equation

It is interesting to consider the consequences of the shape of the target on its ability to be detected by radar. See *Figure 4* for an example of how the behaviour of the waves can vary depending on these factors. They can be supported by mathematical equations (don’t worry we won’t be looking at these; they’re just here for fun!)

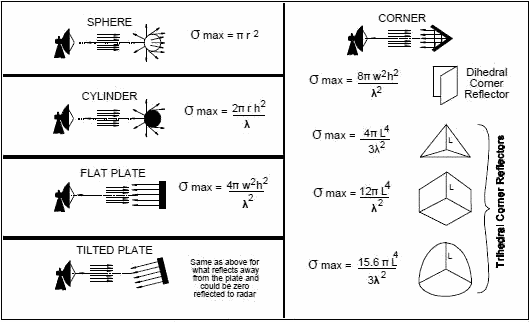


Figure : Wave behaviour on different target shapes

This can be linked to what we do at Leonardo to the concept of radar cross section (**RCS**). It is important to consider the shape of the aircraft we want to design (or find!) and how its shape can make it more detectable by radar, as seen in *Figure 5.* For a stealthy aircraft, you want to have a very small RCS so that as few reflections are produced as possible so that the radar has little to no information to work with.

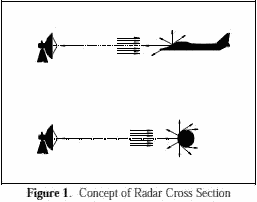


Figure : Aircraft shape can change its ability to be detected

Discussion Points:

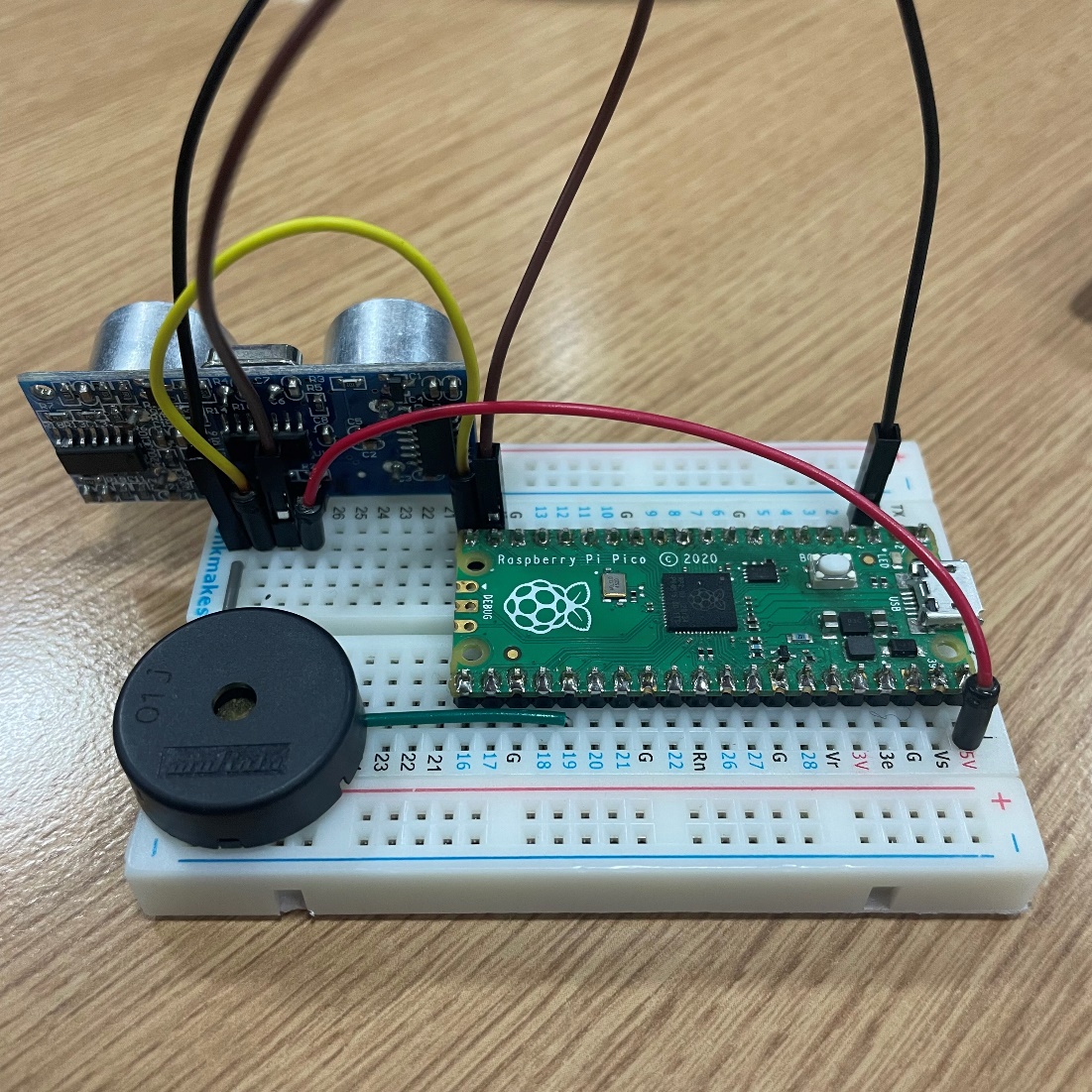
* Edge cases: what happens right on the cusp of 30cm? Why?
* “Principle of Least Astonishment”

*Electrical Connections*

1. Gather the following equipment:

|  |  |  |
| --- | --- | --- |
| Item | Quantity | Notes |
| Raspberry Pi Pico board | 1 | Green card fitted to a breadboard |
| M-M jumper wires | 6 | Pins on both ends of the wire, choose all different colours |
| HC-SR04 ultrasonic sensor | 1 | Blue card with 2 silver barrel-shaped transducers on the front |
| Power cable | 1 | USB to Micro-USB |
| Piezo buzzer | 1 | Black disc with 2 pins |

1. Using the pin-out diagram of *Figure 6* and breadboard guide of *Figure 8,* configure the board according to the schematics shown in *Figure 7.*
2. The final board configuration should look something like this!



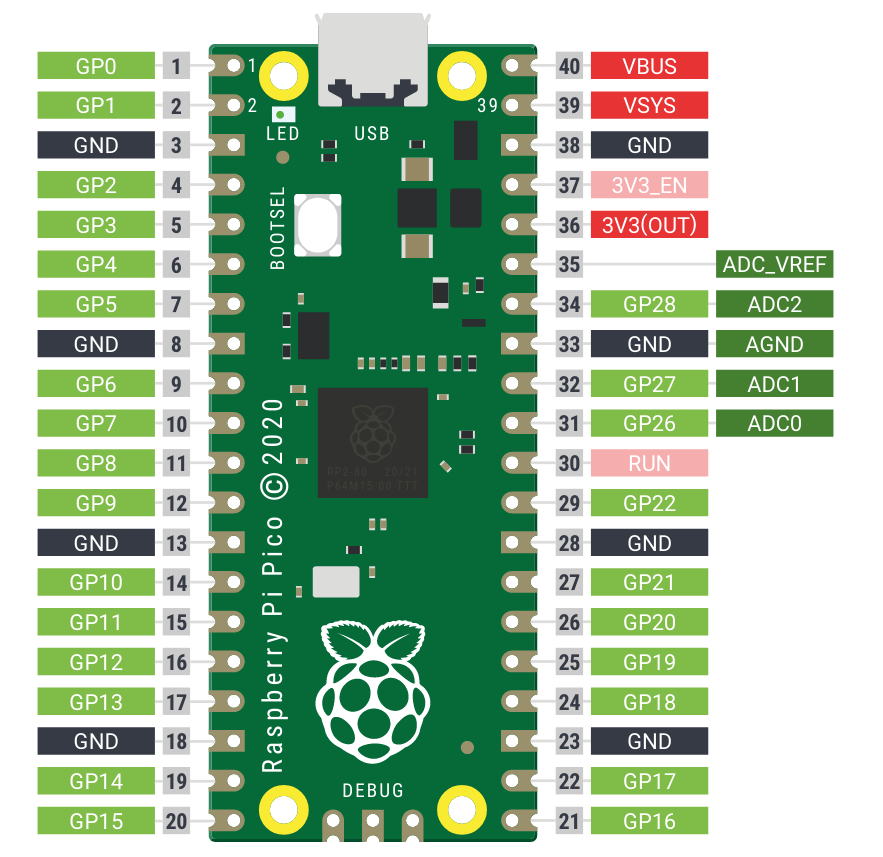


Figure : Pins on the Raspberry Pi Pico

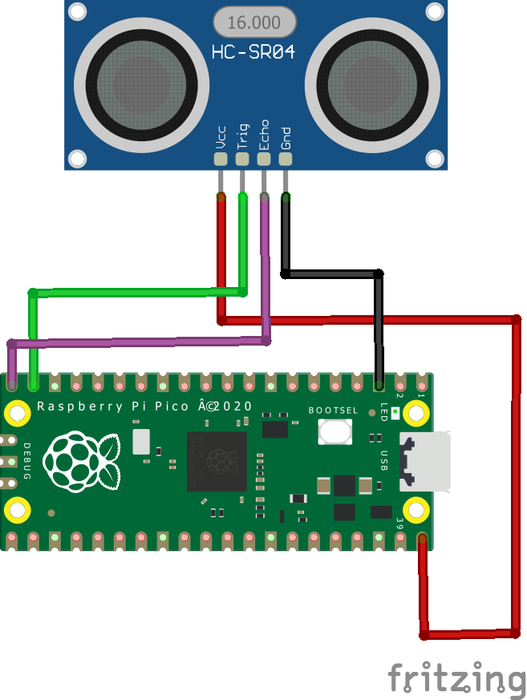


Figure 7: Connections required to perform distance measurement

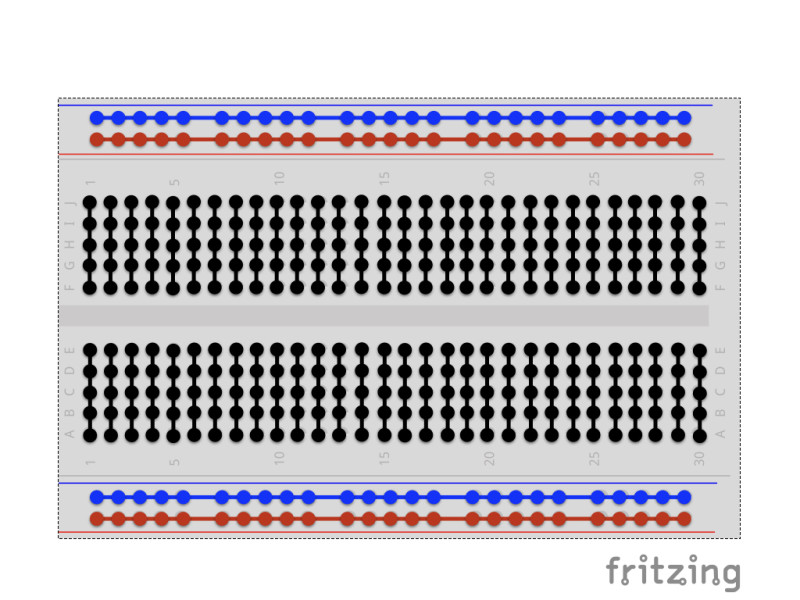


Figure 8: Breadboard connection guidance

*Writing the Software*

|  |  |  |
| --- | --- | --- |
| Step | Instruction | Notes |
| 1 | Open the *Thonny* IDE (integrated development environment). | Transparent App icon on macos is difficult to hit · Issue ... |
| 2 | Press and hold the BOOTSEL button on the RPi Pico, then connect it to the computer via the micro USB cable. | ../../_images/bootsel_onboard1.png |
| 3 | Install MicroPython by clicking the bottom right corner of the Thonny window. | ../../_images/set_pico2.png |
| 4 | In the Target volume, the volume of the Pico you just plugged in will automatically appear, and in the Micropython variant, select Raspberry Pi.Pico/Pico H. | ../../_images/set_pico3.png |
| 5 | Click the **Install** button, wait for the installation to complete and then close this page. | ../../_images/set_pico4.png |
| 6 | Copy the code from the last page of this worksheet into a new Thonny file. |  |
| 7 | Click run, then when prompted select MicroPython device as the save destination. Give your file a name- but ensure you add the .py extension at the end!  Your code should now be running on the board. | Option to save the file on **This computer** or the **MicroPython device** |
| 8 | Switch on the plotter tool by clicking View > Plotter, it will open up a new blank window in the right hand side of the Shell. | Implementing PWM with Raspberry Pi Pico using MicroPython, pwm rpi pico, rpi pico pwm, pwm with raspberry pi pico, raspberry pi pico pwm, how to use pwm in raspberry pi pico, pwm raspberry pi pico, pulse width modulation in rpi4 |
| 9 | Save and run your code, as in *Step 7.* |  |

#### EXERCISE ####

from machine import Pin

import utime

''' Pins for sensor '''

trigger = Pin(14, Pin.OUT) # GPIO 14, pin 19

echo = Pin(15, Pin.IN) # GPIO 15, pin 20

''' Pin for Piezo'''

piezo = Pin(19, Pin.OUT) # GPIO 19, pin 25

''' Pin for Piezo '''

led = Pin(25, Pin.OUT)

def main\_function():

trigger.low() # set low

utime.sleep\_us(2) # stay at the low setting for 2 micro seconds

trigger.high() # set high

utime.sleep\_us(5) # stay at the high setting for 5 micro seconds

trigger.low() # set low

''' Check to see if an echo has been received or not'''

while echo.value() == 0:

signaloff = utime.ticks\_us()

while echo.value() == 1:

signalon = utime.ticks\_us()

''' Find out how much time has passed since we sent out the signal '''

timepassed = signalon - signaloff

''' Find out the distance that is between the sensor and the reflected surface '''

distance = (timepassed \* 0.0343) / 2 # Speed of sound in air = 342 m/s

''' Print the distance to the console for us to see'''

print (0, "Distance (cm) :", distance, 30 )

if (distance > 30):

led.high()

piezo.high()

else:

led.low()

piezo.low()

''' Cycle thorugh these instructions forever '''

while True:

main\_function() # go to the function we just wrote and execute that behaviour

utime.sleep(0.02) # how frequently we want to check the distance and report back to the console