Introduction to Arduino and Coding

14 June 2019

UNIVERSITY OF CAMBRIDGE
DEPARTMENT OF ENGINEERING

INTRODUCTION - (Just Read for Now!)

What will you be doing today?

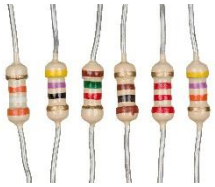
- Coding (‘programming’)
- Measuring distances via ultrasound
- Displaying your measurements using screens and coloured LEDs

What items will you need?

- Arduino

- Ultrasonic distance sensor

- LEDs and resistors



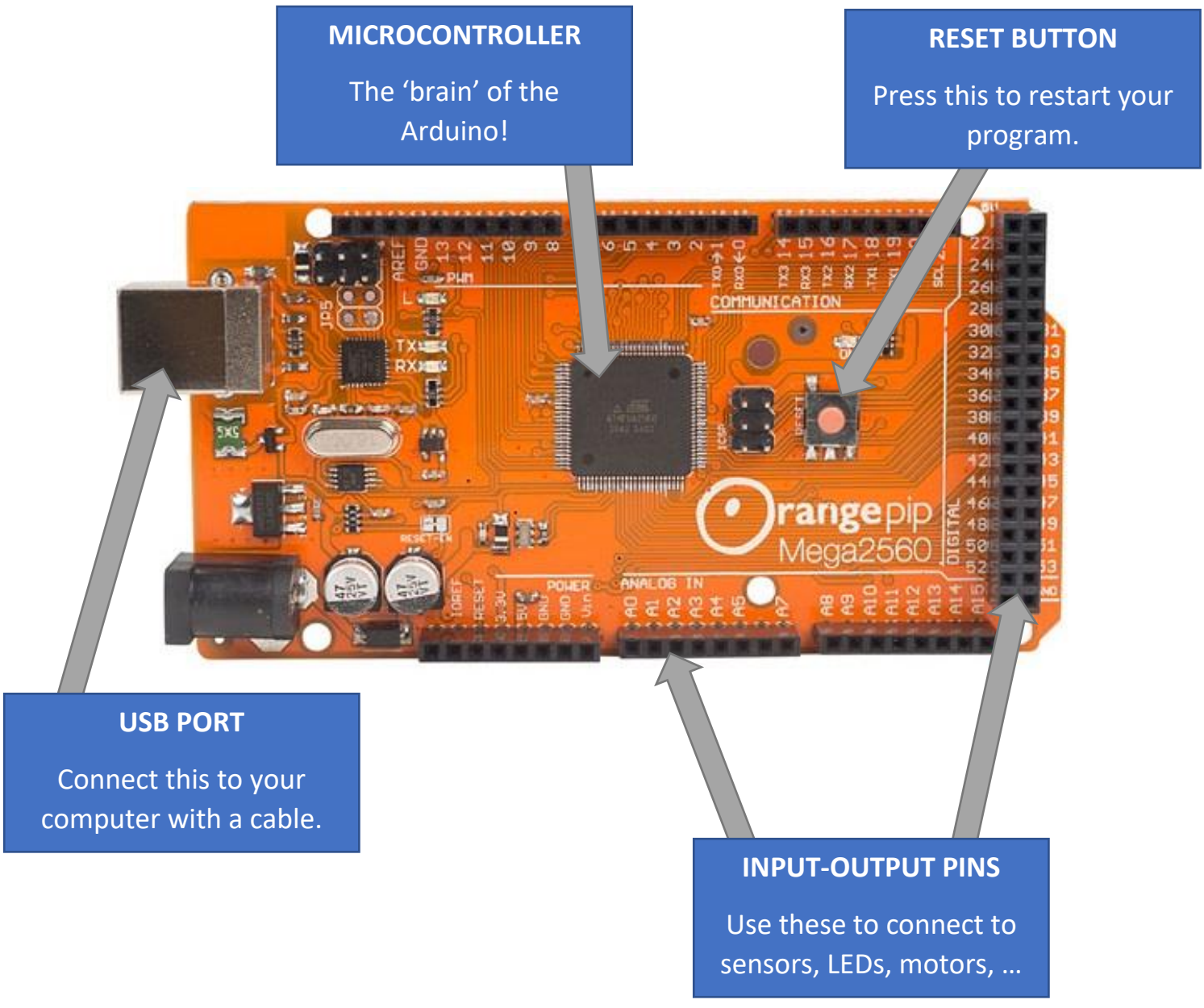
- A set of jumper wires and a USB cable

What is an Arduino?

An Arduino is a programmable device which allows people to easily develop electronics projects.

It can be programmed to do many different things - for example, read data from sensors, control motors, and display messages on a screen.

Arduinos are used all over the world in some amazing projects such as: autonomous robots, DIY 3D printers, and drones!



How do I program an Arduino?

The Arduino is programmed using a computer and a piece of software called an IDE (Integrated Development Environment).

This is where you will enter your code, check it for errors, and then upload it to the Arduino via a USB cable.

VERIFY AND UPLOAD BUTTONS

Click to check your code for errors and upload your code to the Arduino.

CODE WINDOW

This is where you enter your Arduino code.

DEBUG WINDOW

Status and error messages are shown here.

What does setup() and loop() mean?

By default, the *code window* will already contain two *functions (blocks of code with a given name)*:

setup()

and

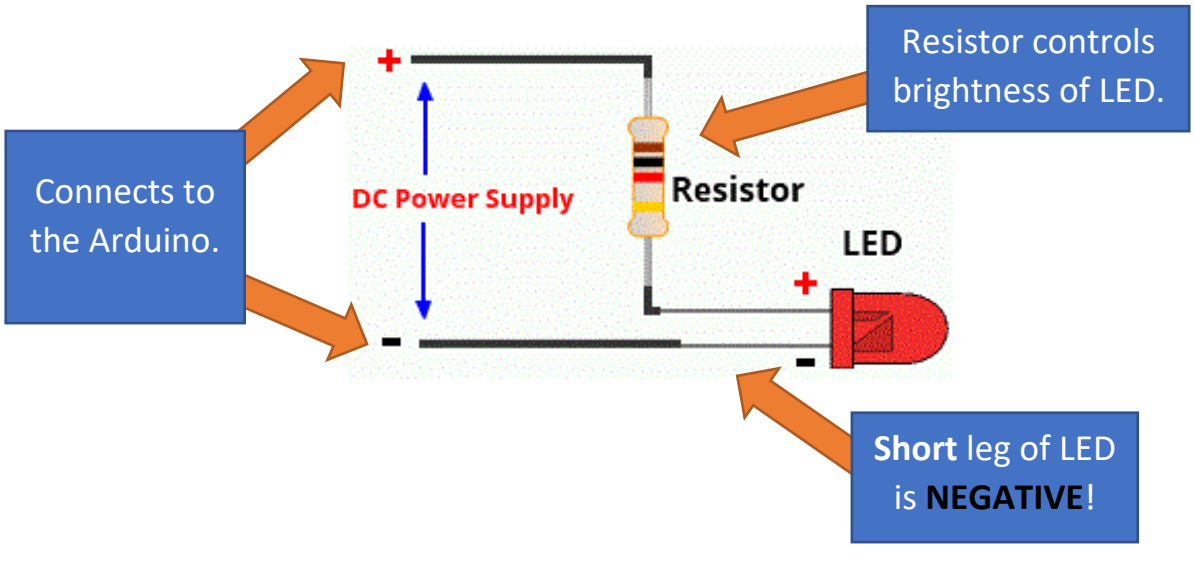
loop()

The code written in the ***setup* function is run only once** – when the Arduino is powered on. We will use the *setup* function to tell the Arduino which sensors and LEDs we are connecting it to.

After that, the code contained in the ***loop* function is run repeatedly** – over and over again until the power is disconnected from the Arduino. The *loop* function will be used for our main code, doing things such as reading from the sensors, calculating distances, and turning LEDs on and off.

What is an LED?

A LED (light-emitting diode) is an electrical device that can light up. LEDs come in many colours and are a great way of displaying simplified messages to a user.



What is ultrasound?

Ultrasound uses reflections of waves to detect distances between objects. It is called *ultrasound* because the waves have a frequency higher than 20 kHz – which is outside the range of human hearing.

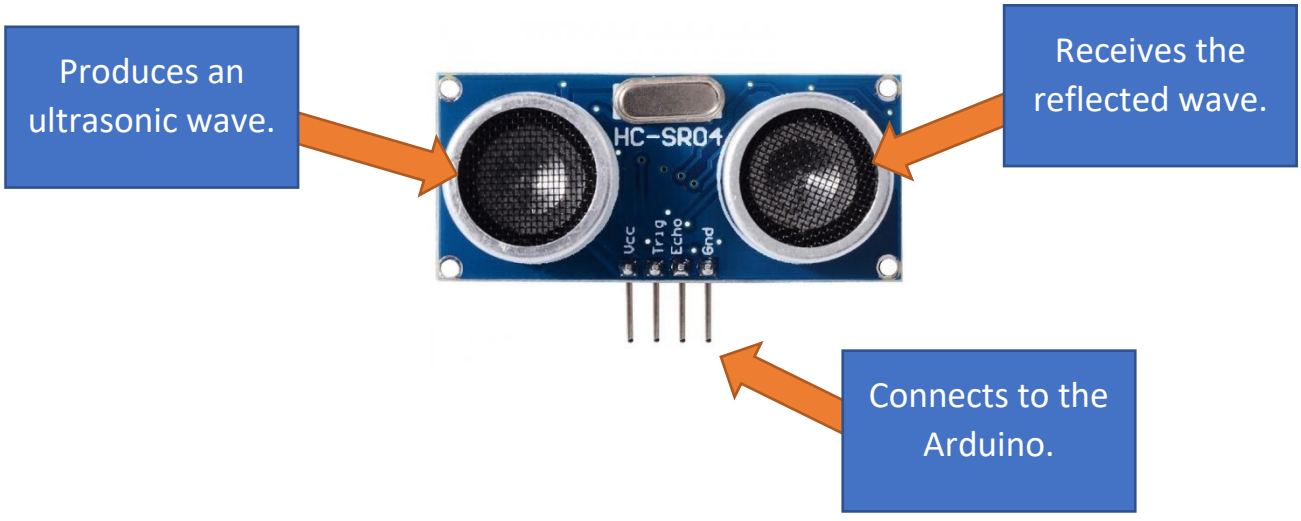
You may have seen ultrasound devices used in hospitals to examine babies in a mother's womb, or ultrasound used by bats to navigate the environment.

An ultrasonic sensor works as follows:

1. An ultrasonic wave is produced by the sensor.
2. It hits a surface and is reflected back.
3. The ultrasonic sensor picks up this reflected wave.

Distance = Speed x Time

We can then use the time taken between production and reception of the wave, as well as the speed of the wave, to calculate the distance.



ACTION – (Time to Get Started!)

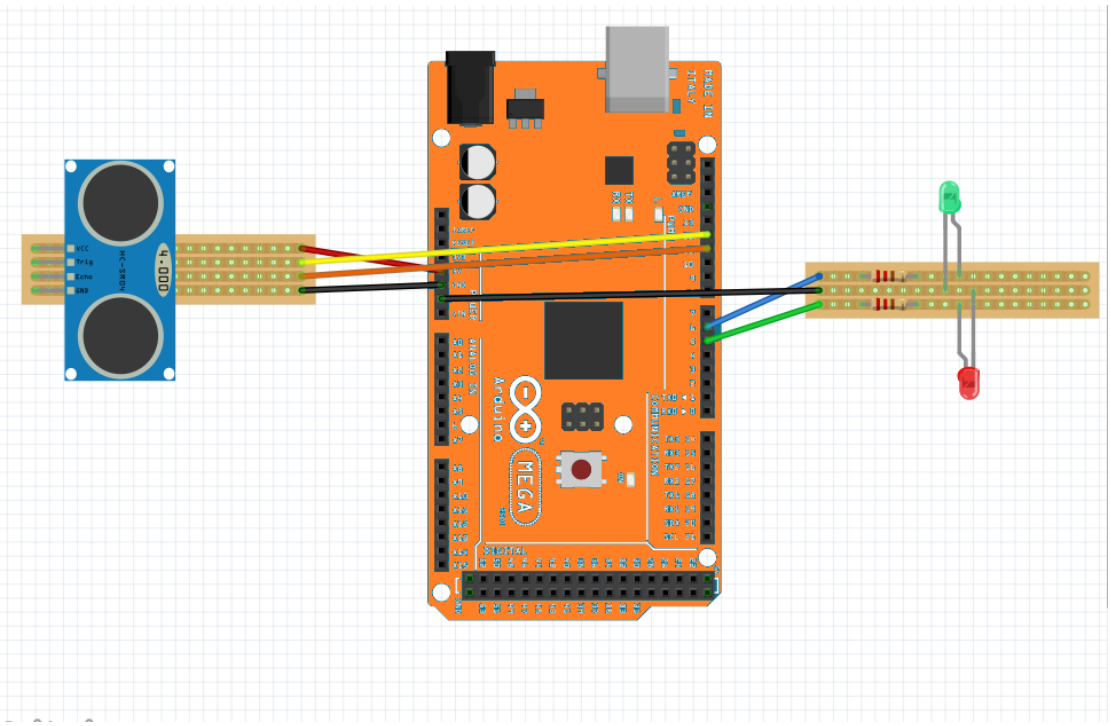
Now that you know the basics of the devices we will be using today, it's time to start building and coding!

There are four tasks, each increasing in difficulty and building on the previous one. Before you get started however, let's connect the hardware and setup the software environment.

Remember, if you need help with anything, simply ask one of us!

Hardware

First, let's connect the ultrasonic sensor, as well as the LEDs and resistors to the Arduino. You can see which pins we need to connect wires to using the picture below.



Once you have finished, ask one of us to check your connections before moving on to the next part!

Software

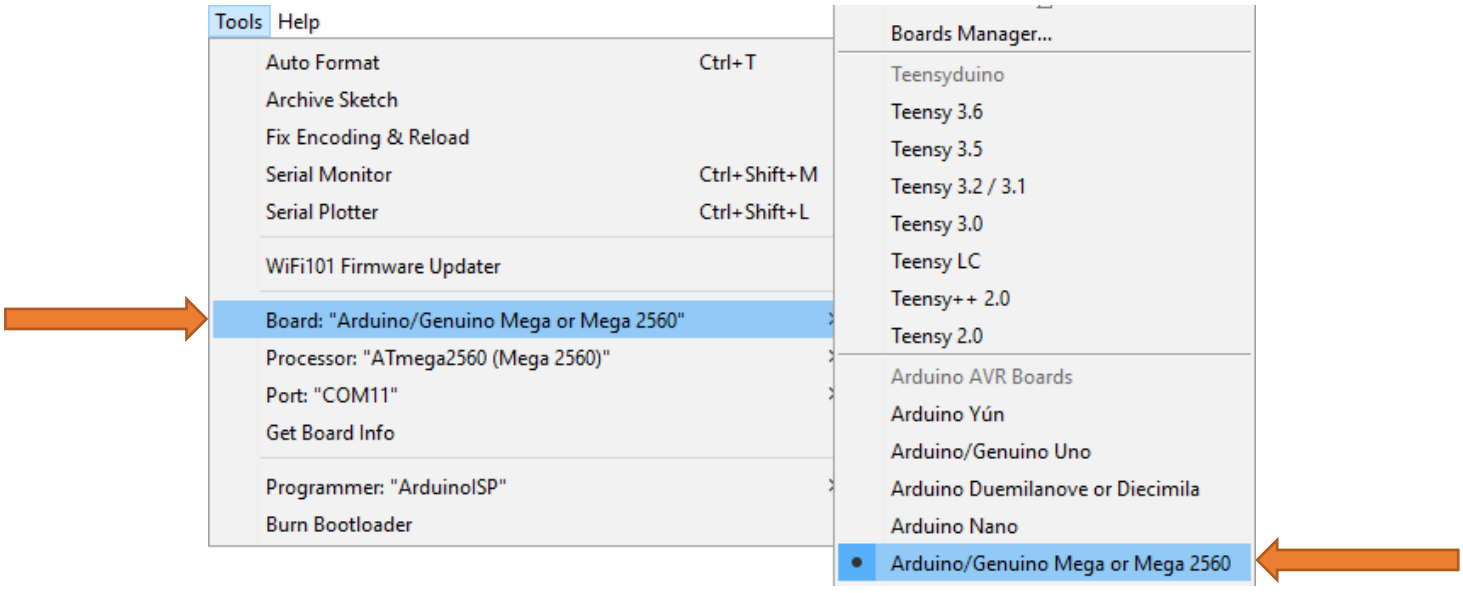
Once you have correctly connected all pieces of hardware together, connect the Arduino to your computer via the USB cable. The Arduino should now be powered on.

Then, start the *Arduino IDE* on the computer.

We now need to tell the IDE how to communicate with the Arduino. This can be achieved in two steps:

1. In the IDE, click on the *Tools* menu bar and then go to *Board:…* . Select the following:

   Arduino/Genuino Mega or Mega 2560



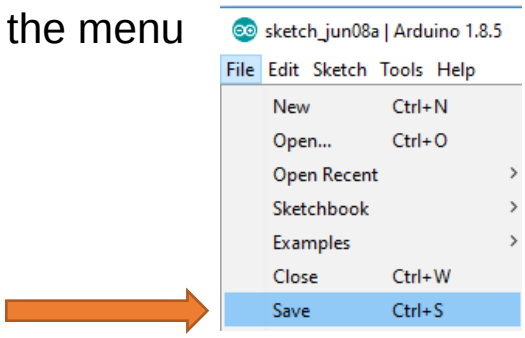
2. Next, still in the *Tools* menu bar, click on *Port:…* . A new drop-down menu should open. Select the menu item which has Arduino in its name.

Now the computer and the Arduino should be able to communicate.

We're almost ready to start now. Before we do you should know that for each task you should perform the following sequence:

1. Write your code into the *code window*.

2. Save your code to a file by selecting the menu bar *File* and then selecting *Save*.



3. Once you are happy with your code, click the following icon to check your code for errors: 

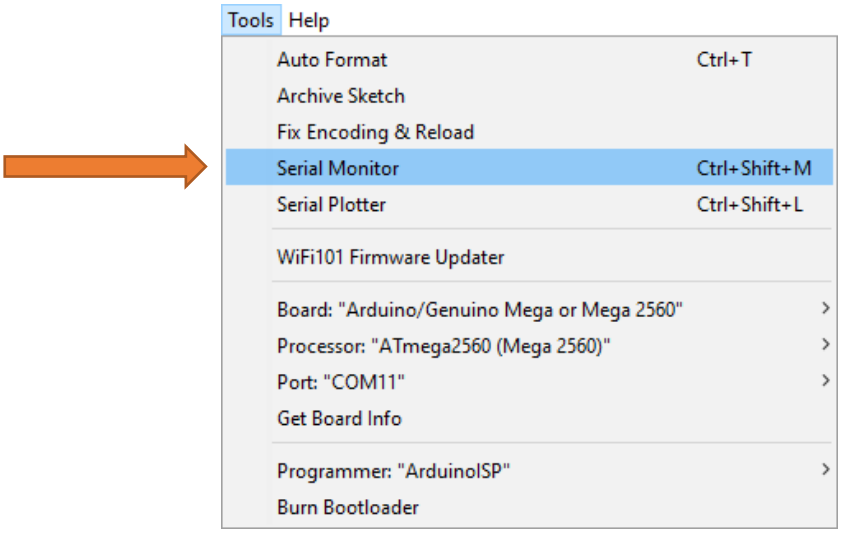
4. If no errors have been found, you can now upload your program to the Arduino by pressing this button: 

Now you should have everything set up and it's time to try out a couple of programs!

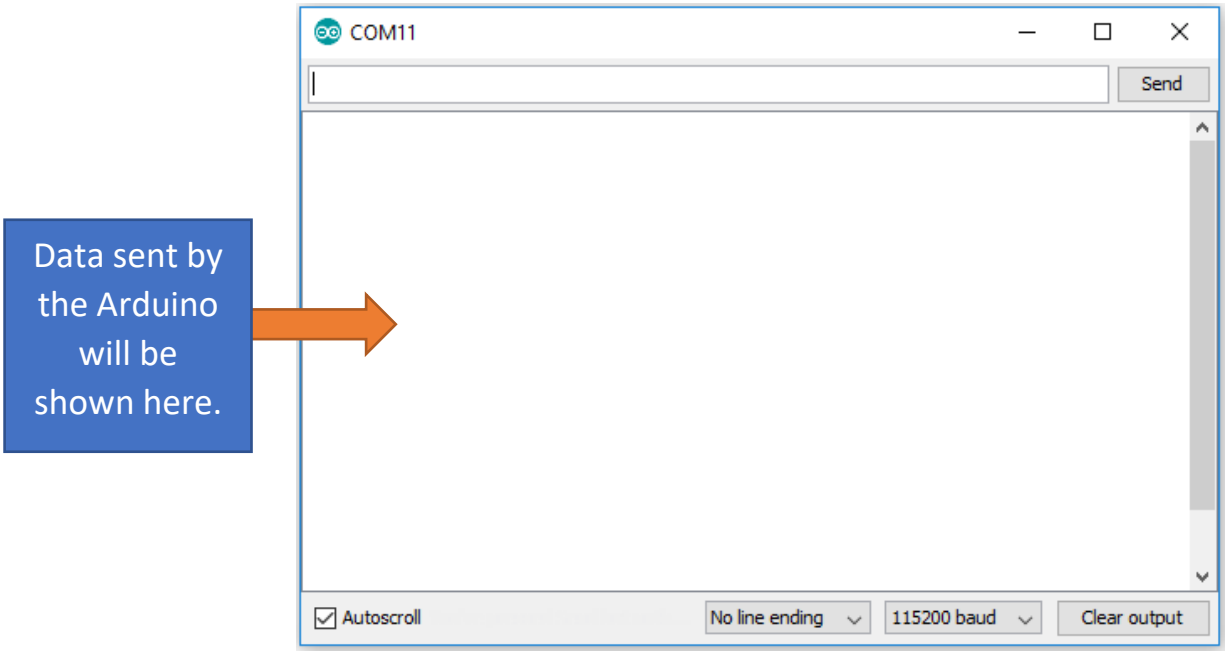
Task 1 – Reading from the ultrasonic sensor

First off, we will try to get distance measurements from the ultrasonic sensor and display them in what is known as a *Serial Monitor*.

The Serial Monitor displays information sent from the Arduino to the computer via the USB cable. You can view it by selecting the menu bar *Tools* and then selecting *Serial Monitor*.



You should now see a new window appear that looks like this:



TIP: Press CTRL+SHIFT+M to bring this window up quickly!

Open the "task1.ino" file. The code lines in `standard courier font` should have been typed in for you.

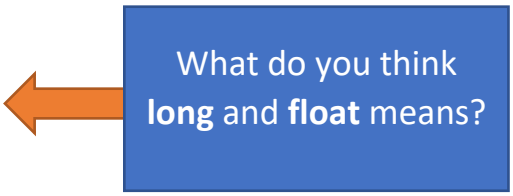
Don't worry if you don't understand some (or most) of it – a lot of it is simply telling the Arduino how to communicate with the sensor. You will be writing the more interesting parts of the program yourself!

1. First, let's tell the Arduino which input-output pins we wish to use for the ultrasonic distance sensor:

```
int trigPin = 12;
int echoPin = 11;
```

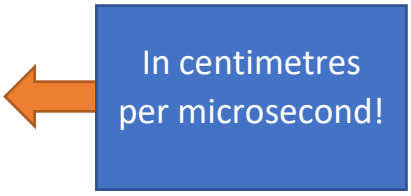
2. Then, we need to create some *variables*. Variables are containers in memory that can store things such as numbers, or text. We will need two: one to store the *duration* between transmission and reflection, and one to store the *distance* we calculate.

```
long duration;
float distance;
```



3. To calculate the distance, we also need to know the speed of the wave! This is a constant however, because it does not change.

```
const float wave_speed = 0.0343;
```



4. We are now ready to add code to the setup function. Remember, this is a function that is run only once when the program starts. Here we are going to tell the Arduino which pins will be used as inputs, which pins will be used as outputs, and that we wish to start the Serial connection.

```
void setup() {
      Serial.begin (9600);

      pinMode(trigPin, OUTPUT);
      pinMode(echoPin, INPUT);
}
```

9600 tells the Arduino to send data back to the computer at a speed of 9600 bits per second.

5. Finally, we can fill in the code for the loop function. This is the code that will run over and over again, reading from the ultrasonic sensor and calculating the distance.

```
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);

  /* TODO: Convert Delay (microseconds) to Distance
                              (cm) */

  Serial.print("Distance ");
  Serial.print(distance);
  Serial.println("cm");

  delay(100);
}
```

This transmits a 10 millisecond pulse.

We now listen for the echo and measure the delay in microseconds.

Code enclosed with /* and */ is ignored by the Arduino.

The Arduino transmits the data back to the computer.

We let the Arduino wait 100 milliseconds before running the code again!

Now, it's your turn! You may have noticed that the code above is missing something (hint: *TODO*).

We need to convert the measured delay of the reflected wave (given in microseconds) to a distance in centimetres.

Can you think of a way of calculating the distance to the object given that you know the speed of the wave and the overall time it took to the object and back?

Once you have an answer, enter your code before the Serial commands. If you have any trouble or need a hint – ask one of us!

If your code is working and has been checked by one of us, move on to *Task 2*.

Task 2 – Lighting up an LED

Instead of using the Serial Monitor to show the exact distance measurement, we now would like to make a **red** LED light up when the distance from the sensor is below 20cm.

Again, we'll give you the code for the setting up the LED (open "task2.ino"). Then, you'll have to do the more interesting bits.

1. Before the setup function and before the variable definitions, there is a new line:

```
int redLEDPin = 5;
```

2. Additionally, in the setup function the following line of code is added:

```
pinMode(redLEDPin, OUTPUT);
```

That's all we need for now. Your task is as follows:

Can you think of a way of turning on the red LED when the distance measured is below 20cm and turning off the red LED when distance is above 20cm?

Here are some ***tips and hints*** you may find useful for this task:

a) You can turn on an LED by applying a voltage across it, which is done by setting the output of the pin connected to the LED to 5v.

   This is done in code using:

```
digitalWrite(redLEDPin, HIGH);
```

b) Conversely to turn the LED off you set the output pin to 0v, which is done using:

```
digitalWrite(redLEDPin, LOW);
```

c) To execute a line of code only if a certain condition is met (such as the distance being less than 20cm) an **if statement** can be used. An example would be:

```
if (a == 6) {

     // Code here runs if a is equal to 6

} else {

     // If a is not 6 the code here runs
}
```

Once you have an answer, enter your code after your distance calculation. If you have any trouble or need help – ask one of us!

If your code is working and has been checked by one of us, move on to *Task 3*.

Task 3 – Changing the brightness of the LED

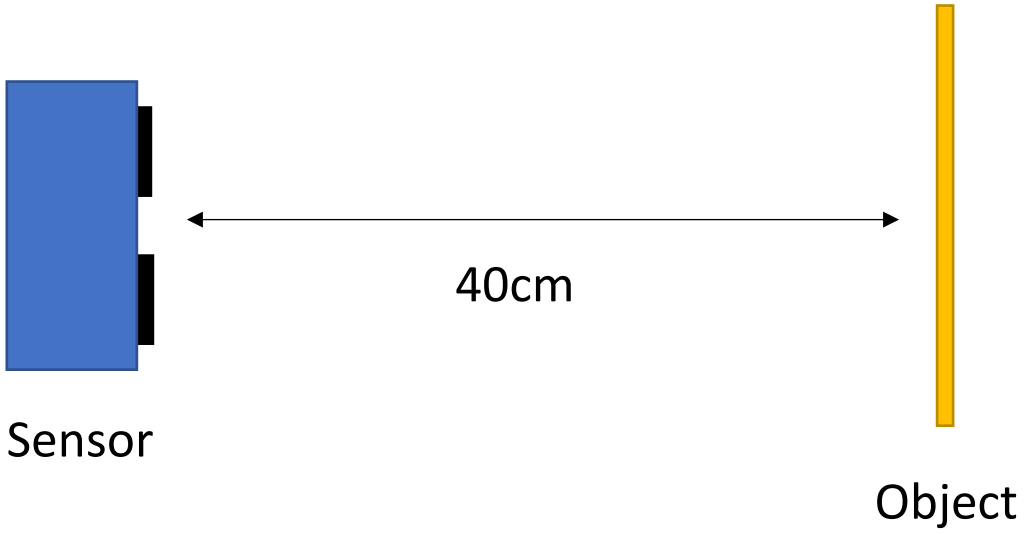
Now, open “task3.ino”. Instead of simply turning the red LED on and off, we wish to change the brightness of it depending on the distance measurement acquired by the ultrasonic sensor.

In particular, …

Can you think of a way of increasing the brightness of the red LED as the distance decreases?

And conversely, decreasing the brightness as the distance increases?

The LED should be at minimum brightness when the object is 40cm away and maximum brightness when the object is 0cm away.



Here are some ***tips and hints*** you may find useful for this task:

a) You can turn on change the brightness of an LED by applying *analogue* voltage. The greater the voltage, the brighter the LED.

   This is done in code using:

```
analogWrite(redLEDPin, brightness);
```

b) The brightness must be a whole number and can vary from **0 to 255** with 255 corresponding to maximum brightness of the LED.

c) The distance to the object can be mapped to an appropriate brightness level using the following function:

$$\text{brightness} = \frac{(\text{a} - \text{distance})}{\text{a}} \times \text{b}$$

Can you think of a value for **a** and **b** so that the brightness is 255 when the distance is 0, and 0 when the distance is 40?

Once you have an answer, replace your old LED on/off code with your new LED code. If you have any trouble or need a hint – ask one of us!

If your code is working and has been checked by one of us, move on to *Task 4*.

Task 4 – Controlling a Second LED

You should now have programmed the Arduino to increase the brightness of the red LED when the distance measured by the ultrasonic distance sensor decreases.

As a concluding task, we wish to add a second, **green** LED. This LED should decrease in brightness as the distance measured decreases – opposite to the red LED!

Now it's your turn again. Open "task4.ino". You should be able to re-use code from the previous tasks to do the following:

1. **Alter the brightness of the green LED depending on the distance measured - as stated above.**

2. **The green LED should be at full brightness at 80cm and turn off at 40cm when the red LED begins to turn on.**

As before - once you have an answer, enter your code after your previous red LED code. If you have any trouble or need a hint – ask one of us!

Once you're done ask one of us to check your final task.

Well done for completing all of the tasks!