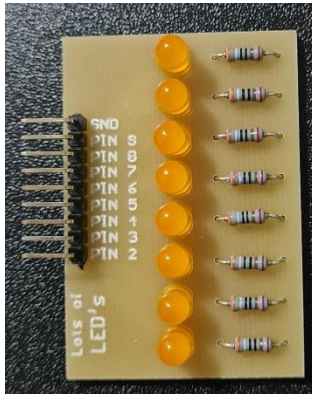


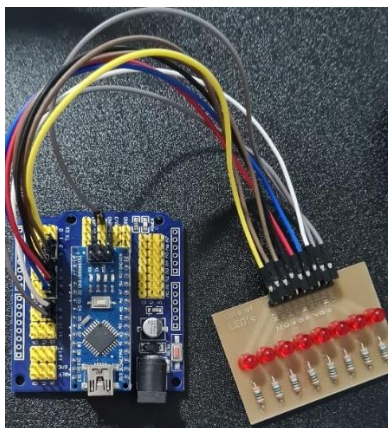
## Getting started...

### Connecting LED's

In this workshop, we are going **connect LEDs** to the Arduino and write a small program to light up the LEDs. We are also going to create different **patterns of light** (for example flashing lights). Next, we are going to also connect the **distance sensor/ultrasonic ranger** to the Arduino and combine it with the lights, so when it measures a distance, the corresponding LEDs will light up.



You'll notice the different pins, numbered from 9 to 2 next to the LEDs- we are going to use cable connectors to connect the pins on the LEDs to the corresponding pins on the Arduino. Make sure you connect the Ground pin first and then connect the other pins to the pins in the signal column ("S") on the Arduino. If you are unsure about this, you can refer to the instructions from Session 1.



Once connected, this is what your set up should look like.

Now we need to write some code so that we can turn the lights on/off. First, open the Arduino IDE and **create a new sketch**. Save the sketch and make sure you give it a proper name so that you can recognize the file later on.

It should automatically set up two different functions- **void setup()** and **void loop()**. Any code that we place in void setup() will only run once, whereas code in the void loop ()

function runs repeatedly. Every function has a **return type**, in this case void, which means that the function does not return anything.



```

sketch_jun25b.ino
1  void setup() {
2    // put your setup code here, to run once:
3  }
4
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8  }
9
  
```

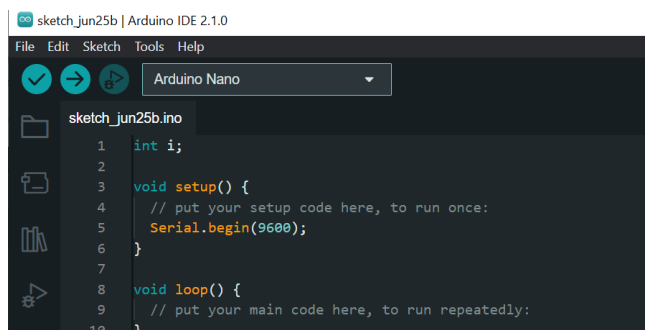
Next, we need to create a variable which is going to refer back to the pins and the input/output. Place **int i;** before void setup(). It needs to be a **global variable**, so it can be accessed by any other function later in the program. If we place it inside the void setup() function, it becomes a **local variable** and it would only be possible to access it from within that function.



```

sketch_jun25b.ino
1  int i;
2
3  void setup() {
4    // put your setup code here, to run once:
5  }
6
7  void loop() {
8    // put your main code here, to run repeatedly:
9  }
  
```

As always, we need to begin our program with **Serial.begin(9600);** in the void setup function. This allows the serial monitor and the Arduino to communicate with each other, at a data transfer rate of 9600 bit/second.



```

sketch_jun25b.ino
1  int i;
2
3  void setup() {
4    // put your setup code here, to run once:
5    Serial.begin(9600);
6  }
7
8  void loop() {
9    // put your main code here, to run repeatedly:
10 }
  
```

We want our program to **light up the LEDs from pin 2 to pin 9**, so instead of writing the same code for all pins, we are going to use a **for loop**. Inside the void setup() function we set the variable i to 2, so that the program starts with LED number 2, and it continues as long as i is greater than or equal to 9, since we have no pin 10. We are then going to **increment** i by writing i++, so that after lighting up LED 2, the program automatically lights up LED 3 next, then 4 and so on. We also need to set the **pin mode** so that the program knows whether we are referring to input or output here. In this case, we use **OUTPUT**.

```
for (i = 2; i <= 9; i++) {
  pinMode(i, OUTPUT);
}
```

Make sure you remember to place all brackets and semicolons 😊.

Your code should now look like this:

```
sketch_jun25b.ino
1  int i;
2
3  void setup() {
4    Serial.begin(9600);
5    for (i = 2; i <= 9; i++) {
6      pinMode(i, OUTPUT);
7    }
8  }
9
```

Now we go into the void loop() function and switch on the LEDs.

As we want to switch all of them on, from LED 2 through to LED 9, we need to create a for loop again, as we did in the last step. You can just copy and paste that line of code.

```
for (i = 2; i <= 9; i++) {
}
```

Inside that loop, we tell the program to turn on the LED referring to i (remember, since this increments after each LED, we only have to write one line of code for ALL LEDs instead of code for each of the LED's).

```
digitalWrite(i, HIGH);
```

**digitalWrite** sends information to the digital pin that our LEDs are connected to. We set the variable for this to i earlier one. We also set the setting to HIGH, which means that the LED will light up brightly. If you set it to LOW, it will remain dark.

The code should now look like this:

```

sketch_jun25b | Arduino IDE 2.1.0
File Edit Sketch Tools Help
┌───┴───┐
┌───┴───┐ Arduino Nano
┌───┴───┐
┌───┴───┐ sketch_jun25b.ino
1  int i;
2
3  void setup() {
4    Serial.begin(9600);
5    for (i = 2; i <= 9; i++) {
6      pinMode(i, OUTPUT);
7    }
8  }
9
10 void loop() {
11   // put your main code here, to run repeatedly:
12   for (i = 2; i <= 9; i++) {
13     digitalWrite(i, HIGH);
14   }
15 }
  
```

Well done! 😊

You can now upload it to your Arduino.

What happens if you upload it?

## Create different light patterns

In the last step, the LEDs would all light up at the same time. That's cool, but also pretty boring because it doesn't do much else yet. Lets change that!

Have you got any ideas for different light patterns that we could create?

Here are some ideas for you to try:

- We could light them up **one at a time**
  - o To do this, we just need to add a **delay** after lighting up each of the LEDs.  

`delay(500);`

 500 means that there is a delay of 500 milliseconds between each LED. You can change this to make the LEDs light up quicker or slower, for example 200 or 1000. What happens if you do this?
- If you try the above, you might notice that once a LED is switched on, it stays that way. At some point, all of them are going to be switched on and nothing else happens. How about we **switch them off again after switching them on?**

You can do this by changing the setting from **HIGH** to **LOW**  
 Just add another line of code after adding a **delay**.

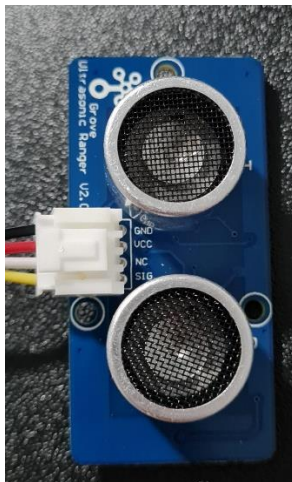
```
digitalWrite(i, LOW);
```

This will switch them on and off and create a cool pattern. You can also change the delay to make them switch on and off much faster.

Can you think of more patterns?

## Add ultrasonic ranger/distance sensor

In this part of the workshop, we are going to **combine the LEDs with a distance sensor**, so once a certain distance is measured, the corresponding LED with light up. For example, if the distance is 2 cm, LED1 lights up. If the distance is 5cm, LED 2 lights up, and so on.



If you take a close look, you might notice that the different cables going out from the ultrasonic ranger are labelled. **When you connect it to the Arduino, connect it as follows:**

GND – GND on the Arduino

VCC – 5V on the Arduino

NC – does not get connected

SIG (Signal) – connect to digital pin 10

First, **create a new Sketch** and save it under a file name so that you can recognize it later on. The initial setup is going to be the same as before – void setup () and void loop().

Since we are now going to use a sensor that returns input back to our serial monitor, we need to include

```
#include "Ultrasonic.h"
```

in our file, before void set up().

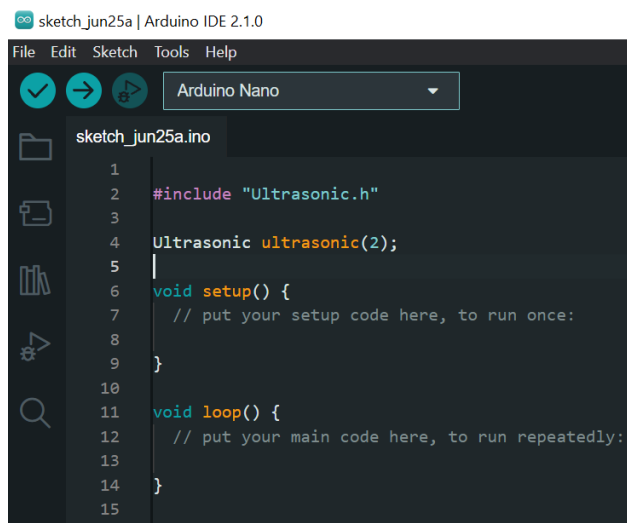
We also need to set up a variable that contains the data from the sensor. We are going to name it **Ultrasonic** and since it is produced by the ultrasonic sensor, we need to include that too, as well as the pin that the sensor is connected to.

```
Ultrasonic ultrasonic(10);
```

+++ Please make sure you set the Ultrasonic ultrasonic () to PIN 10 instead of PIN 2

```
Ultrasonic ultrasonic(10)
```

This also happens before the set up() function.



If your code looks like this, you are ready to move on to the next step.

First, we need to include **Serial.begin(9600);** in the set up() function again, so that the Arduino can communicate with the serial monitor.

Once that is done, we can move to the loop() function.

Our program needs to store the distance measured by the ultrasonic ranger/distance sensor so we can work with it later. For this, we are going to create a variable named **RangeInCentimeters**, which a data type of **long**, inside void loop().

```
long RangeInCentimeters;
```

At the moment, the value would be 0, because the ultrasonic ranger cannot measure anything yet. To assign the value measured by the sensor to the RangeInCentimeters variable, we are going to set the variable to an in-built function for the sensor.

```
RangeInCentimeters = ultrasonic.MeasureInCentimeters();
```

At the moment our Arduino receives measurements from the ultrasonic ranger, but we can't see it yet. To be able to see the results on the serial monitor, we need to print out the results, by adding

```
Serial.print(RangeInCentimeters);
```

to our code. This will print the value of RangeInCentimeters to our serial monitor.

If you like, you can add "Serial.println(" cm") so that it also prints "cm" after the value.

```
Serial.println(" cm");
```

Your code should now look like this:

```

6 void setup()
7 {
8
9   Serial.begin(9600);
10 }
11
12
13 void loop()
14 {
15   long RangeInCentimeters;
16   RangeInCentimeters = ultrasonic.MeasureInCentimeters();
17   Serial.println("Distance in centimeters:");
18   Serial.print(RangeInCentimeters);
19   Serial.println("cm");

```

What happens if you **upload it** to your Arduino and **check the output** on the serial monitor?

Exactly- it prints different outputs in a very quick sequence. To make the reading a little easier, you can add **delay(250);** to your code, which means that the ultrasonic ranger will only take a new measurement every 250 milliseconds.

Well done! 😊

## Combine lights and ultrasonic ranger

In the last part of this workshop session, we are going to **combine the ultrasonic ranger with lights**. For example, if the distance is 2 cm, LED1 lights up. If the distance is 5cm, LED 2 lights up, and so on.

First, save your old sketch and create a new one.

You then need to **connect the LEDs and the ultrasonic ranger** to your Arduino (as you'd have done in the first part of this workshop). The connections are going to be similar - for the ultrasonic ranger it would be



GND – GND on the Arduino

VCC – 5V on the Arduino

NC – does not get connected

SIG (Signal) – connect to digital pin 11.

For the signal, the ultrasonic ranger pin gets connected to pin 11 on the Arduino as pin 2 would be connecting to the LEDs already.

For the lights, connect the different LEDs to the corresponding pins on the Arduino. Make sure you also connect the GND ("Ground") pin.

To get started, we first need to include `#include <Ultrasonic.h>` for the ultrasonic ranger at the beginning of our file. As we are including the LEDs too, it is important that we create a **global variable i** again – also at the beginning of the file before setup void (); .

The pin for the ultrasonic ranger we are going to set up for pin 11.

**Serial.begin(9600);** in the void setup function is also going to remain unchanged.

```

rangerWithLights.ino
1  #include <Ultrasonic.h>
2  int i;
3
4  Ultrasonic ultrasonic(11);
5
6  void setup()
7  {
8
9      Serial.begin(9600);
10 }
11
  
```

Moving to the void loop() function, to set up the ultrasonic ranger you can refer to the first part of the workshop and copy and paste the code, however, make sure you change the delay to delay(2000); - this is required so there is enough time to switch the LEDs on and off after measuring the distance.

This is what your code should now look like:

```

rangerWithLights.ino
1  #include <Ultrasonic.h>
2  int i;
3
4  Ultrasonic ultrasonic(11);
5
6  void setup()
7  {
8      Serial.begin(9600);
9  }
10
11 void loop()
12 {
13     long RangeInCentimeters;
14     RangeInCentimeters = ultrasonic.MeasureInCentimeters();
15     Serial.println("Distance in centimeters:");
16     Serial.print(RangeInCentimeters);
17     Serial.println("cm");
18     delay(2000);
  
```



Next, we want the corresponding LEDs to get switched on if the ultrasonic ranger measures a certain distance. We have 8 LEDs and the ultrasonic ranger is most accurate with distances up to 15cm.

## If the distance is:

- Less than 3cm: switch LED 2 on and off
- More than 3cm and less than 5cm: switch LED 3 on and off
- More than 5cm and less than 7cm: switch LED 4 on and off
- More than 7cm and less than 10cm: switch LED 5 on and off
- More than 10cm and less than 12cm: switch LED 6 on and off
- More than 12cm and less than 15cm: switch LED 7 on and off
- More than 15cm: switch LED 8 on and off

For this, we need to create an if statement and a number of if-else statements.

For example, for a distance less than 3cm:

```
if (RangeInCentimeters <=3) {  
    digitalWrite (2, HIGH);  
    delay (500);  
    digitalWrite (2, LOW);  
}
```

This means that if the distance is less than 3cm, the LED connected to pin 2 is set to HIGH. We also include a delay before switching the LED off again, so that we can actually see it lighting up.

For the remaining distances we use else-if statements, for example:

```
else if (RangeInCentimeters >= 3 && RangeInCentimeters <= 5) {  
    digitalWrite (3, HIGH);  
    delay (500);  
    digitalWrite (3, LOW);  
}
```

If you feel confident, you can finish up the remaining else-if statements and upload the code to your Arduino. Good job! 😊

If you'd like to refer to the rest of code, you can find it on the next page

```
if (RangeInCentimeters <=3) {  
    digitalWrite (2, HIGH);  
    delay (500);  
    digitalWrite (2, LOW);  
  
} else if (RangeInCentimeters >= 3 && RangeInCentimeters <= 5) {  
    digitalWrite (3, HIGH);  
    delay (500);  
    digitalWrite (3, LOW);  
  
} else if (RangeInCentimeters >= 5 && RangeInCentimeters <= 7) {  
    digitalWrite (4, HIGH);  
    delay (500);  
    digitalWrite (4, LOW);  
  
} else if (RangeInCentimeters >= 7 && RangeInCentimeters <= 10) {  
    digitalWrite (5, HIGH);  
    delay (500);  
    digitalWrite (5, LOW);  
  
} else if (RangeInCentimeters >= 10 && RangeInCentimeters <= 12 ) {  
    digitalWrite (6, HIGH);  
    delay (500);  
    digitalWrite (6, LOW);  
  
} else if (RangeInCentimeters >= 12 && RangeInCentimeters <= 15) {  
    digitalWrite (7, HIGH);  
    delay (500);  
    digitalWrite (7, LOW);  
  
} else if (RangeInCentimeters <= 15) {  
    digitalWrite (8, HIGH);  
    delay (500);  
    digitalWrite (8, LOW);  
}  
}
```