Introduction to Arduino

This workshop aims to introduce you to using the Arduino IDE as well as how to use basic components such as servos and ultrasonic sensors. These are important skills as they are components that will aid you throughout the competition.

Resources

- 1x Arduino Uno + Cable
- 1x Ultrasonic Sensor
- 1x Servo Motor
- 1x Breadboard
- Jumper Wires
- 1x LED
- 1x Resistor

Setting Up the Arduino IDE

1. Download the Arduino IDE from the official website.
2. Install the CH340 Driver (if needed) for serial communication:
    - [Windows Driver]
    - [Mac Driver]

The board should come up automatically at the top of the screen, to connect the Arduino Board to your computer.

The Arduino Board consists of many different Pins that can be used as Input (read signals from sensors) or Output (controls servos):

Type	Quantity	Use
Analog Pins	6	Can read or write an analog signal from 0-1023. Where 1023 is equivalent to HIGH and 0 to LOW
Digital Pins	14	Can read or write a binary HIGH or LOW signal

PWM Pins (~)	6	Digital PIN that can send an analog signal by modulating the voltage that is sent

These Pins are used to interact with components, used to either read data that is sent from a component or sent data to a component.

The Arduino has a reset button, next to the cable port. This will cause the code to restart. It also has a power socket that can have a 9V battery plugged into it. This can be used to power the Arduino rather than the USB-B port.

Basic Arduino Programming

The Arduino IDE uses C++ code. If you have never coded in C++ before, make sure to ask the demonstrators for help and use the internet whenever you run into issues.

Key C++ Syntax

- End lines with ;
- Declare variables with types:

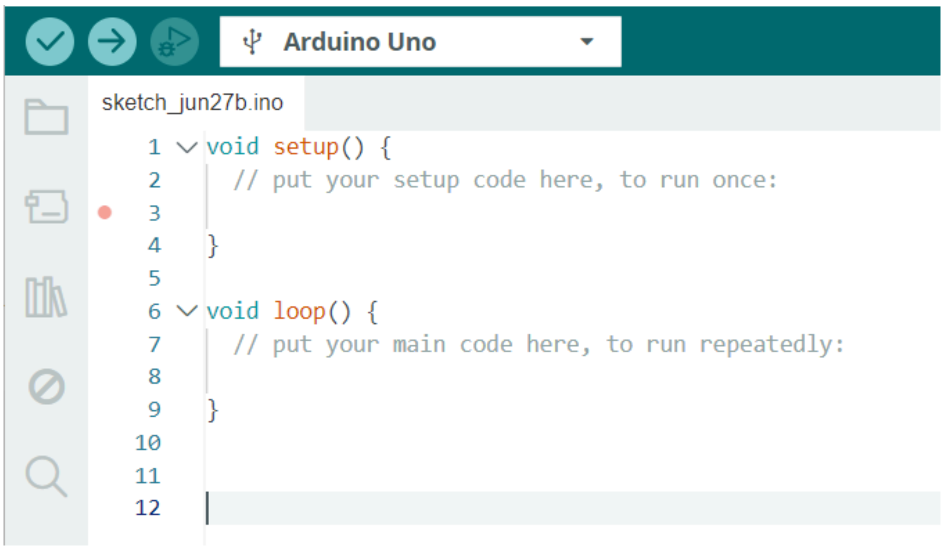
```
C/C++
int number = 4;           // Integer
char character = 'a';     // Character
bool sun_is_hot = true;   // Boolean
```

- Loops: When you need to repeat the same action multiple times you can use a for loop or while loop

```
C/C++
// For loop
for (int i = 0; i < max_value; i++) { /* Code */ }

// While loop
while (condition) { /* Code */ }
```

Arduino Program Structure

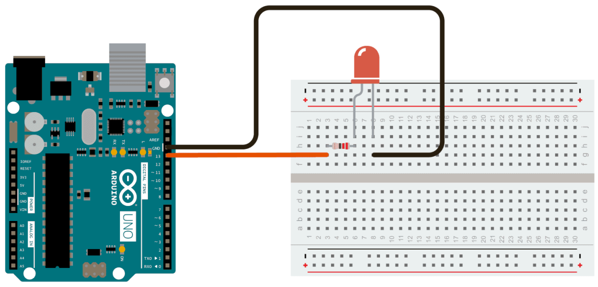
This is the screen you should see when you open a new file. Note: it may be slightly different for Mac but the main parts are the same.



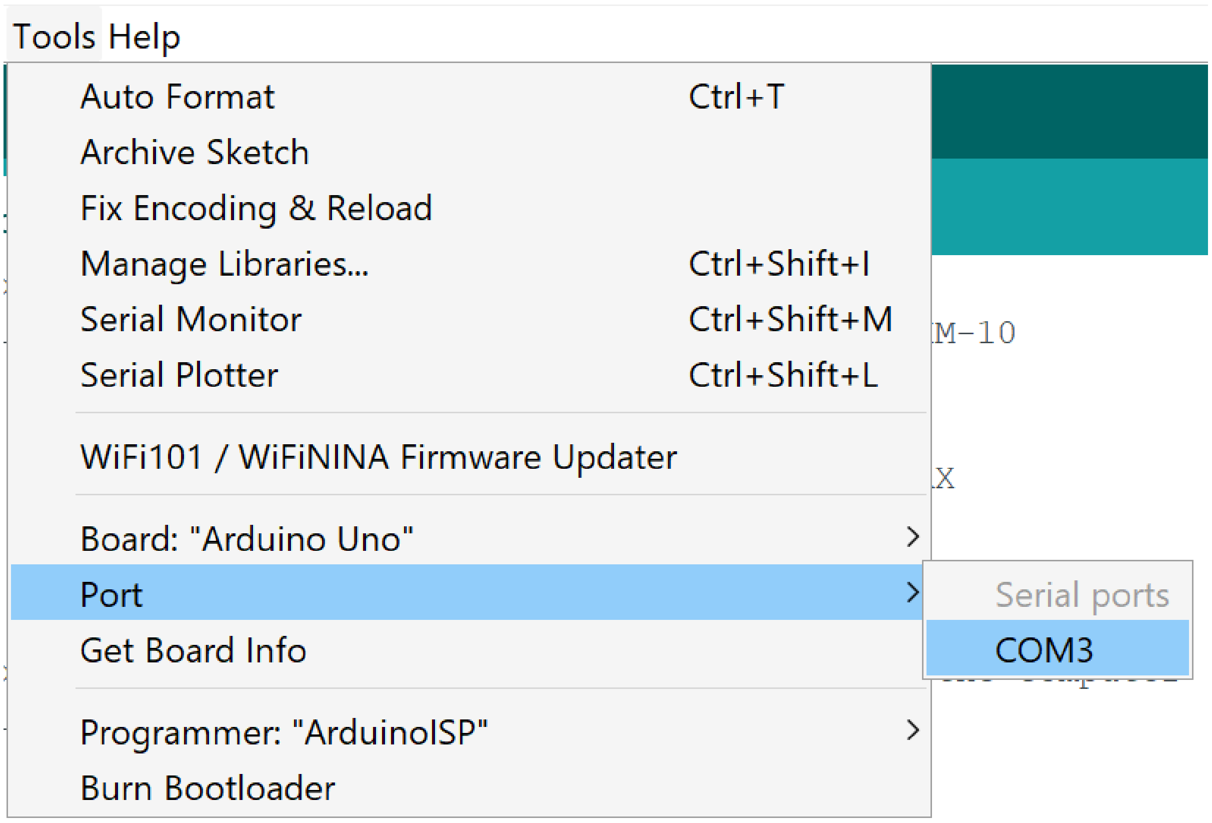
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Tick Button	Compiles the code. You can use this to test for any errors.
Arrow Button	Compiles and then uploads the code onto the Arduino.
Void setup():	This function runs just once initially. You can use it to set the Mode of the Arduino pins, run startup functions, or declare variables.
Void loop():	This function runs continuously after setup() runs. This polling loop is where the bulk of your code will be, such as checking Input/Output Pins.

To see how this works, start by building the circuit below. It consists of any Digital Pin connected to a resistor, which is then in series with the long leg of an LED. The short leg is then connected to the GND Pin. Be careful when using the LED that you connect it the right way.



Be sure to set your board and port correctly (port can vary between devices)



Once you have completed this, copy the code below and run it. Make sure you understand how it works.

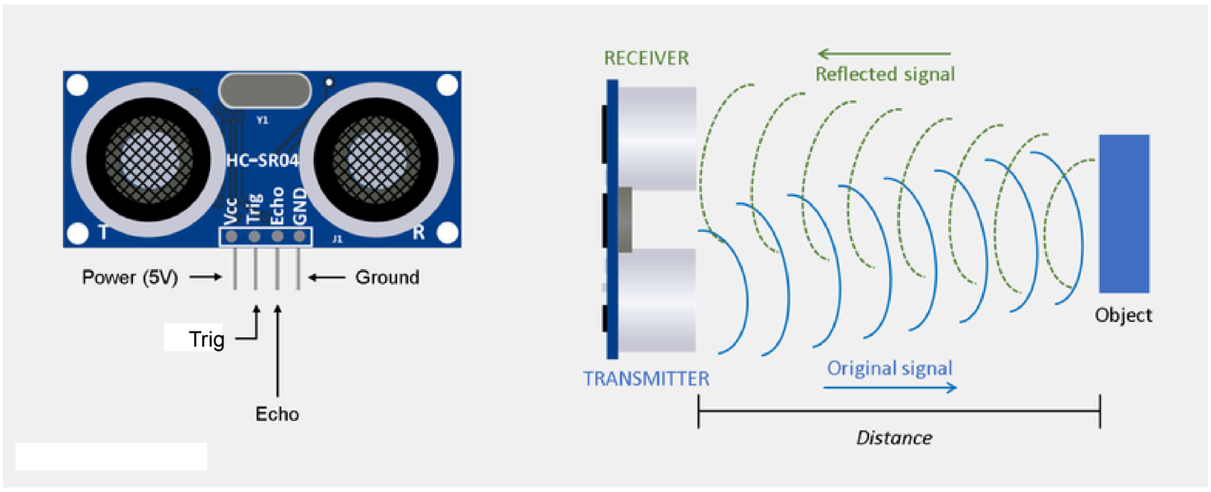
C/C++
```
#define LED_PIN 13

// the setup function is the first function which runs on the Arduino
void setup() {
  // tell the arduino to set LED_PIN to output mode.
  pinMode(LED_PIN, OUTPUT);
}

// the loop function runs over and over again forever
```

```
void loop() {
  digitalWrite(LED_PIN , HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for 1000 milliseconds
  digitalWrite(LED_PIN , LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for 1000 milliseconds
}
```

Ultrasonic Sensors

Ultrasonic Sensors are used to measure distance. They do this by sending out sound pulses and determining how long they take to bounce back. Ultrasonics use 4 Pins:

VCC	Power: Connect to Arduino 5V
Trig	Output: Sends pulses when HIGH
Echo	Input: Stays HIGH for the length of time it took for the waves to travel
GND	Ground: Connect to Arduino GND



However, we care more about the distance an object is away and so we must convert time to distance. The speed the sound waves travel at is 0.0343 cm/microsecond. We must also divide by 2 as the sound waves travel to the object and back.

$distance = speed * time = 0.0343 * duration/2$

To print the results to your computer you can use the Serial Monitor. To use this in the setup function run the following:

Serial.begin(int baud_rate): Sets the data rate in bits per second (baud) for serial data transmission. For communicating with Serial Monitor, make sure to use one of the baud rates listed in the menu at the bottom right corner of its screen. You only need to run this function once, in the setup() function of your Arduino program. Use a Baud rate of 9600.

Serial.println(val): Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n').

Your Turn!
Have a try at implementing this on your own! If you get stuck make sure to ask the demonstrators.
Hint #1: the function pulseIn() can be used to see how long a pin (perhaps the ECHO Pin) has been high. (Function Documentation)
Skeleton Code:

```
C/C++
#define trigPin <Insert Pin Number>
#define echoPin <Insert Pin Number>
int duration, distance;

void setup() {
//Ultrasonic Setup
pinMode(trigPin, <Insert type>);
pinMode(echoPin, <Insert type>);

//Serial Monitor Setup
Serial.begin(9600);

}

void loop() {

  //Send a LOW signal through the trig Pin then delay for some microseconds

  //Send a HIGH signal through the trig Pin then delay for some microseconds
  delayMicroseconds(10);
  //Send a LOW signal through the trig Pin

  //Calculating distance
//Use pulseIn on the Echo Pin
 //Convert duration to distance

  //Print to Serial Monitor
  Serial.print("Distance: ");
  Serial.println(distance);
}
```

To open the Serial Monitor you can find it in Tools menu or click on this link:

Set baud rate as 9600 Then in the loop function you can run this function to print a string or character followed by a new line.

Servo Motor (SG90)

Servo Motors are a very useful piece of equipment that rotate when directed. This can be used to create large amounts of force to push objects. They consist of 3 Wires:

Red	Power: Connect to Arduino 5V
Brown	Ground: Connect to Arduino GND
Yellow	Signal: Controls the angle of the servo motor

Controlling them in Arduino is also quite easy by using the Servo.h library. To import this add the line "#include <Servo.h>" at the start of your code.

To use a servo:

- Define the servo as Servo myServo at the start of the file
- Use myServo.attach(SignalPin) to connect signal wire to Arduino in the setup function
- Set the angle of the Servo using myServo.write(angle) in the loop function

Note: Angle should be between 0 and 180. This specific servo can not rotate 360 degrees.

Your Turn!
Try to implement this yourself by creating two for loops that increase the angle from 0 to 180 and then back again. Make sure to add a small delay, eg delay(15), between each iteration of the loop.

C/C++
```cpp
#include <Servo.h>
Servo myServo;

void setup() {
  myServo.attach(9);  // Signal pin
}

void loop() {
  for (int angle = 0; angle <= 180; angle++) {
    myServo.write(angle);
    delay(15);
  }
  for (int angle = 180; angle >= 0; angle--) {
    myServo.write(angle);
    delay(15);
  }
}
```

Putting it all together

Use what you have learnt so far to have a try at building a circuit that triggers a servo motor to move whenever the ultrasonic sensor detects motion. An example of how the circuit could look is shown below.