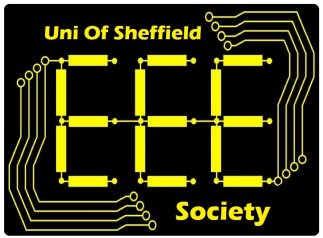
UOS EEE Society - Arduino Labs

Introduction to Arduino



The University Of Sheffield.

Session Overview

1. What is a Microcontroller?
2. What is Arduino?
3. What can Arduino be used for?
4. The Arduino Uno
5. The Arduino Programming Language
6. Program Structure
7. Data Types
8. The Line Termination Character
9. Variables
10. Arithmetic Operators
11. Comparison Operators

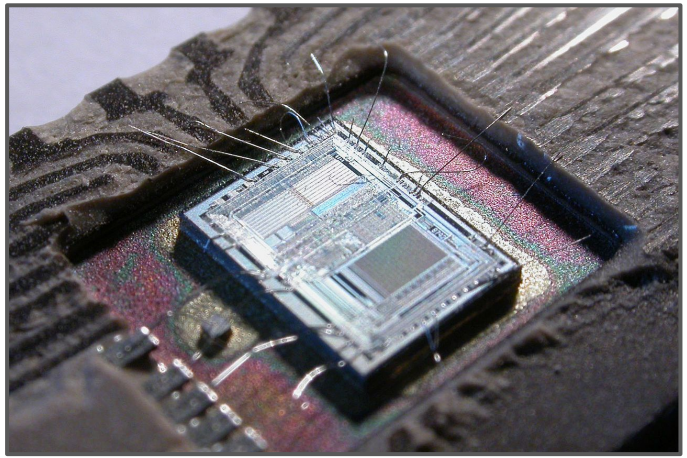
Post and Share about the Sessions as much as possible on Social Media!

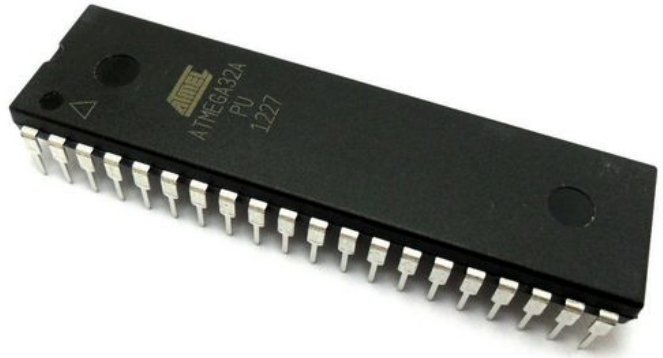
Who has used a Microcontroller before?

What is a Microcontroller?

What is a Microcontroller?

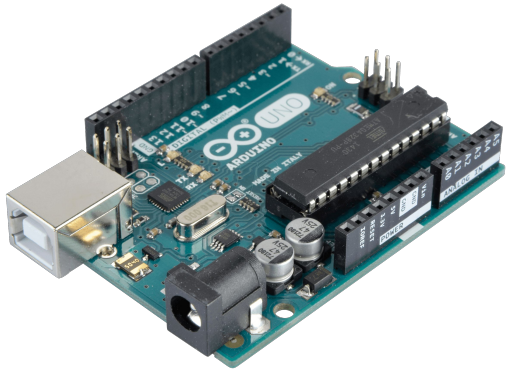
- A small computer on a single Integrated Circuit
- Contains a Processor, Memory and Input/Output Peripherals
- Programmable using programming languages such as C and Assembly
- Used to control things in the real world (Motors, LEDs, Relays)



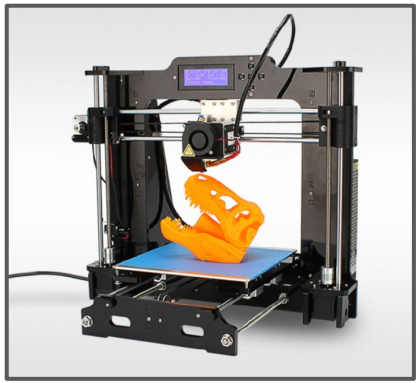


What is Arduino?

- Microcontroller Platform
- Open Source Hardware Designs (Creative Commons Licence)
- Free to use Programming Language
- Easy to use Addon Boards (Shields)
- **Used for Rapid Prototyping**

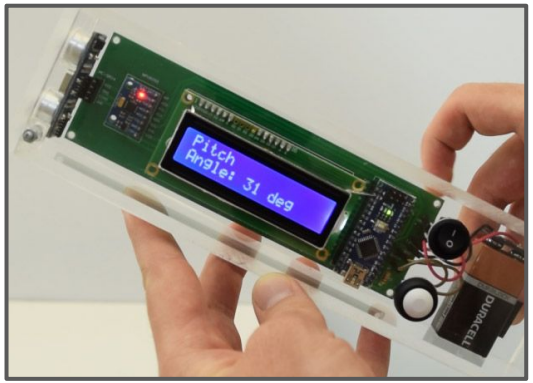


What can Arduino be used for?



3D Printer Controllers

Vending Machine



Spirit Level and Range Finder

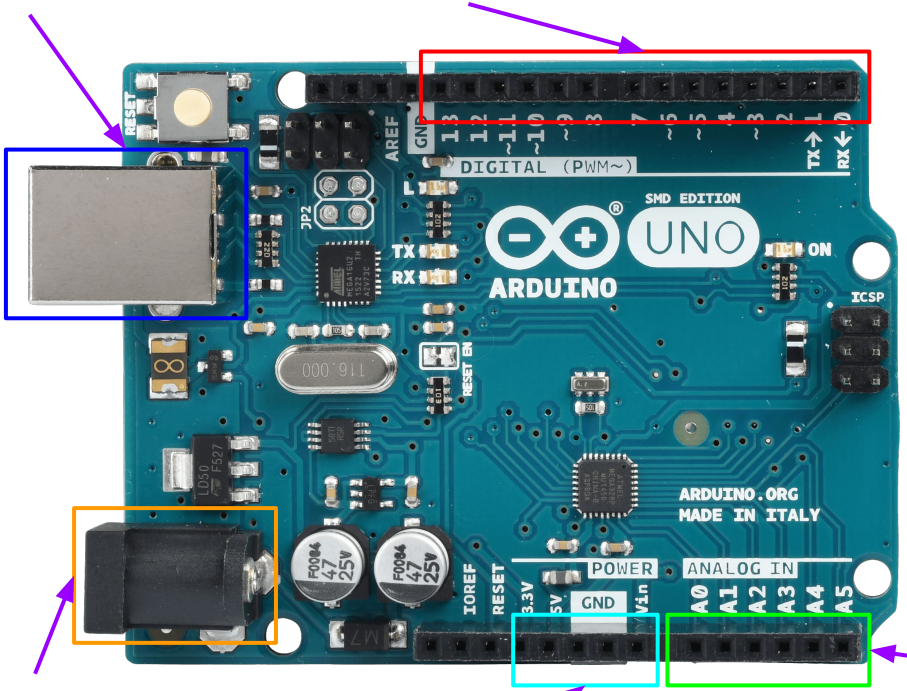
Plus loads of other amazing things!

Look at the Arduino Blog for more projects!

The Arduino Uno

USB-B Port

Digital I/O Pins



Barrel Jack

Power Pins

Analog Input Pins

Digital I/O Pins - Used to control external **Digital** circuitry. Pins 11, 10, 9, 6, 5 and 3 have PWM (Pulse Width Modulation) capability.

Analog Input Pins - Converts an Analog Input to a Binary Number that can be used in the Microcontrollers program.

Power Pins - Provides power to external circuitry.

Barrel Jack - Provides power to the Arduino Uno using an external Power Supply.

USB-B Port - Used to provide power to the Arduino Uno and for programming.

[Arduino Specification and Production Page]

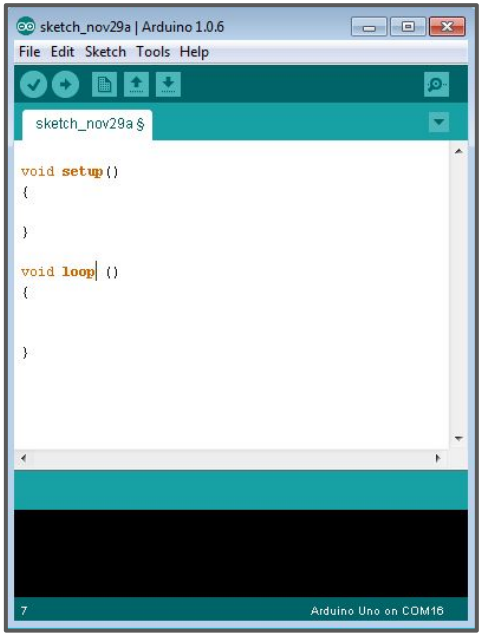
The Arduino Programming Language

- A simplified version of C++.
- Uses functional programming constructs.
- Allows for libraries.
- Easy to learn and fast to write!

I will briefly cover all aspects of the Programming Language in this and the following Lecture. For a more in depth tutorial see this Tutorials Point Page.

Program Structure

- Arduino Programmes consist of the following basic sections:
  - Setup Function
  - Loop Function
- The Setup Function:
  - Runs once.
  - Is mostly used for initial Hardware Setup.
- The Loop Function:
  - Loops continuously until the Arduino is powered off.
  - Executed after the setup function.
  - Used to **implement** the main section of your code.



Data Types

- Boolean (bool) - Takes a value of True (1) or False (0)
- Character (char) - Takes a value of -128 to 127 that represents a Character
- Byte (byte) - Takes a eight bit binary value
- Integer (int) - Takes a numerical value from -32,768 to 32,767
- Float (float) - Takes a floating point value from -3.4028235E+38 to 3.4028235E+38
- String (String) - A special type used to hold text (not the same as a Character Array)

More information can be found HERE

Line Termination Character

- An important thing to remember when writing code using the Arduino Programming Language:

Programming lines always end with a Semicolon! (;)

Variables

- Variables store information inside the program.
- They can be of many different types, as shown on the previous slide.
- Variables are created using the assignment operator.
- They can either take constant values or be generated using mathematics within the program:

```
string HelloWorld = “Hello World”;

int MyNumber = a_number + another_number;
```

Arithmetic Operators

Operator Name	Operator Symbol	Description
Assignment Operator	=	Stores the value to the right of the equal sign in the variable to the left of the equal sign.
Addition	+	Adds two operands.
Subtraction	-	Subtracts two operands.
Multiplication	*	Multiplies two operands together.
Division	/	Divides the left operand (numerator) by the right operand (denominator).
Modulo	%	Gives the remainder of an integer division.

Comparison Operators

Operator Name	Operator Symbol	Description
Equal to	==	Evaluates whether the operand on the right hand side is equal to the operand on the left hand side.
Not Equal to	!=	Evaluates whether the operand on the right hand side is not equal to the operand on the left hand side.
Less than	<	Evaluates if the left operand is less than the right operand.
Greater than	>	Evaluates if the left operand is greater than the right operand.
Less than or Equal to	<=	Evaluates if the left operand is less than or equal to the right operand.
Greater than or Equal to	>=	Evaluates if the left operand is greater than or equal to the right operand.

Thanks for listening!

Next week we will be briefly covering:

1. Control Statements and Loops
2. Functions
3. Arrays and Strings
4. Libraries
5. Debugging using the Serial Monitor
6. The Arduino Editor
7. Online Resources
8. The LED Shield we are using in Labs!

Follow us on Social Media:

- https://www.facebook.com/uoseeesoc
- https://twitter.com/uoseeesoc
- Snapchat - uoseeesoc
- Gmail - eeesoc@sheffield.ac.uk