Concept Session 1 - Arduino
Course Notes

Welcome to Concept, a program of sessions based around the Arduino. Seven sessions cover the introduction to Arduino, Amateur Radio applications, building a digital VFO and an SDR receiver.

Objectives

This first session aims to start off from wherever you are today. Maybe you have some experience in the Arduino, maybe you are a beginner. You will learn step-by-step, with experienced users catching up on the details and newcomers learning the basics for the first time.

You will use your own Laptop or Desktop computer, which can be a Microsoft Windows PC or an Apple Mac. You will download and install the Arduino Integrated Development Environment (IDE), which is free software from Arduino.cc.

The whole course aims to bring back an interest in hands-on Amateur Radio, based on building things and understanding how they work. You will be able, through software development, to contribute to the practical construction and operation of the Arduino shields and sketches that we will build.

Kit 1

For this first session you will need a kit of parts, the basic Arduino UNO, an experiment board, wires and a power supply - which can be the USB port of your computer or an external 9V battery.

USB Stick

On the USB stick provided with this course are the slides for all sessions .Together with a

host of program sketches, help and documentation files.

Kit 2

In addition, to do the experiments and build prototypes you will need some more parts, which includes a VFO synthesiser module, a Real Time Clock module, a Rotary Encoder and an LCD display.

Kits 3 & 4

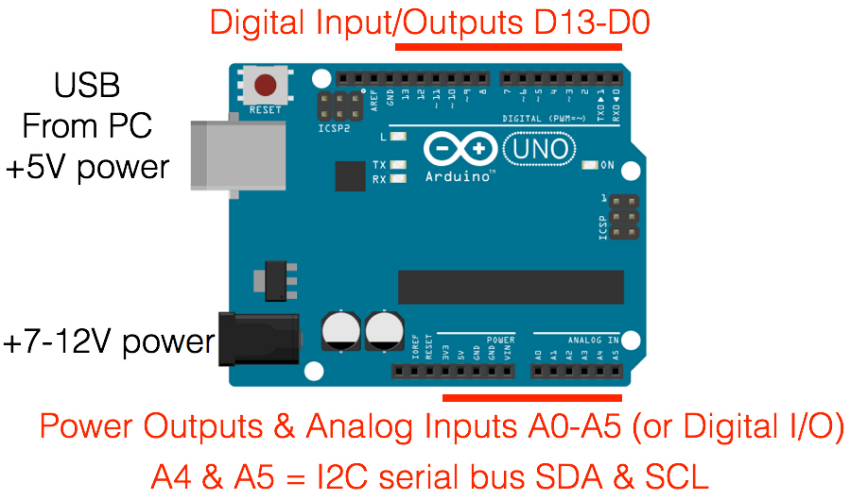
These final kits provide all the components and PCBs to build a digital VFO and a Software Defined Radio.

The parts included in the kits are shown in the table on the next page. x = new to this kit, o = from previous kit.

	Starter	Learner	VFO & RTC	VFO+ROT	VFO+ROT+LCD	VFO_RTC_IQ	SDR_40M
Arduino UNO	x						
400 point BB	x						
Jumper wires	x						
LED		x					
220R		x					
Piezo buzzer		x					
Si5351 module			x	o	o		
Rotary Encoder				x	o		
I2C LCD			x		o		
RTC module			x				
CR1220 battery			x				
F - M wires			x		o		
PCB VFO_RTC_IQ3						x	
Si5351 module						o	
SN74AC74N						x	
2 x 100nF						x	
1 x 10uH						x	
2x6 & 3x8 pin headers						x	
4 & 5 pin headers right angle						x	
Rotary Encoder						o	
RTC module						o	
I2C LCD						o	
F-F wires						x	
PCB SDRX_40M1							x
2 x 100pF							x
1 x 120pF							x
1 x 1500pF							x
3 x 10nF							x
3 x 100nF							x
2 x 220nF							x
1 x 10uF							x
2 x 100pF							x
1 x 10uH							x
2 x T30-6 Toroids							x
Wire 28swg 50 & 60 cm							x
2 x 39R							x
3 x 1k							x
2 x 3k9							x
2 x 22k							x
Right Angle header 3 pin							x
2x6 & 3x8 pin header kit							x
FST3253							x
TLV2462							x

The Arduino UNO

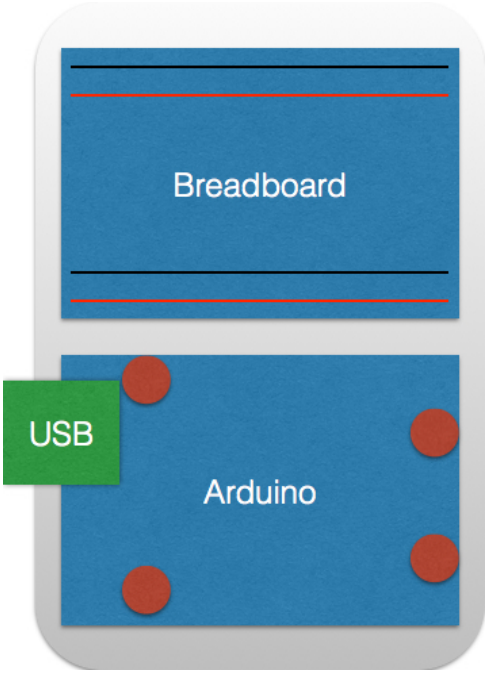
The Arduino UNO is a single board microcomputer with both Digital and Analog Inputs/Outputs. It is powered by either your computer USB connection or an external supply of 7-12V.



The Analog inputs A0-A5 can also be used for Digital input/output signals (0/5V), A4 and A5 are often used for communicating using the serial I2C bus system with Data SDA and Clock SCL, which we will use to talk to our LCD display, Real Time Clock and digital synthesiser VFO.

Build your experiment board

Mount the Arduino UNO on the support board using the screws, spaces and nuts. Then stick on the experimenters breadboard, making sure to get the black -ve and red +ve rows correct (see below)



Set up your PC or Mac

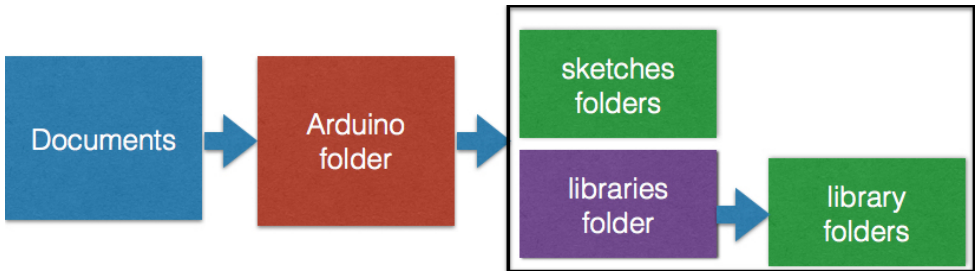
In order to set up your computer it must be connected to the Internet. Go to the web site

arduino.cc

Chose “Download” and download and install the version of the Arduino IDE suitable for your computer.

Next, make a new folder under Documents, called “Arduino”, this is where your program sketches and software Libraries will reside, plus help and other documents.

Copy the contents of the USB stick to this folder.



Run the Arduino IDE program, and chose “Preferences”, you must let the IDE software know where it can find your sketches and libraries. Under “Sketchbook Location” browse and find the folder you made Documents/Arduino. Select it.

Then QUIT the Arduino program, and RE-OPEN it.

Lastly plug-in your Arduino UNO to a USB port on your computer.

With the Arduino program open chose `Tools > Board > Arduino Uno`.

Then chose `Tools > Serial Port > ...`

... and chose the name of the port of your USB connection (this will be a “COMx” port on the PC or a “/dev/tty/usbmodemxxx” port on the Mac).

Your computer and Arduino are now ready for action.

What is a sketch?

A sketch is what Arduino people call programs. This is a simple sketch, note that most things are in lower case, programs are case sensitive:

```
// My_Do_Nothing is a sketch outline, the sketch does nothing

// example of an included library
#include <Arduino.h>

// example of a constant define
#define LED 13

// setup function, executed once on upload (->)
void setup() {

}

// loop function, executed over and over
void loop() {

  myFunction(); // call a user defined function

}

// your own function, called by loop()
int myFunction() {

}
```

Open the sketch now in your IDE program, you will find it at:

```
File > Sketchbook > My_Do_Nothing
```

All sketches have a basic structure of five parts

`"include"` - external libraries to be included in your code

`"define"` - names of values to be substituted in your code

`"setup"` - a series of instructions which is run only once

`"loop"` - a series of instructions which are repeated, over-and-over

`"myfunctions"` - one or more of your own functions you write to solve the problem

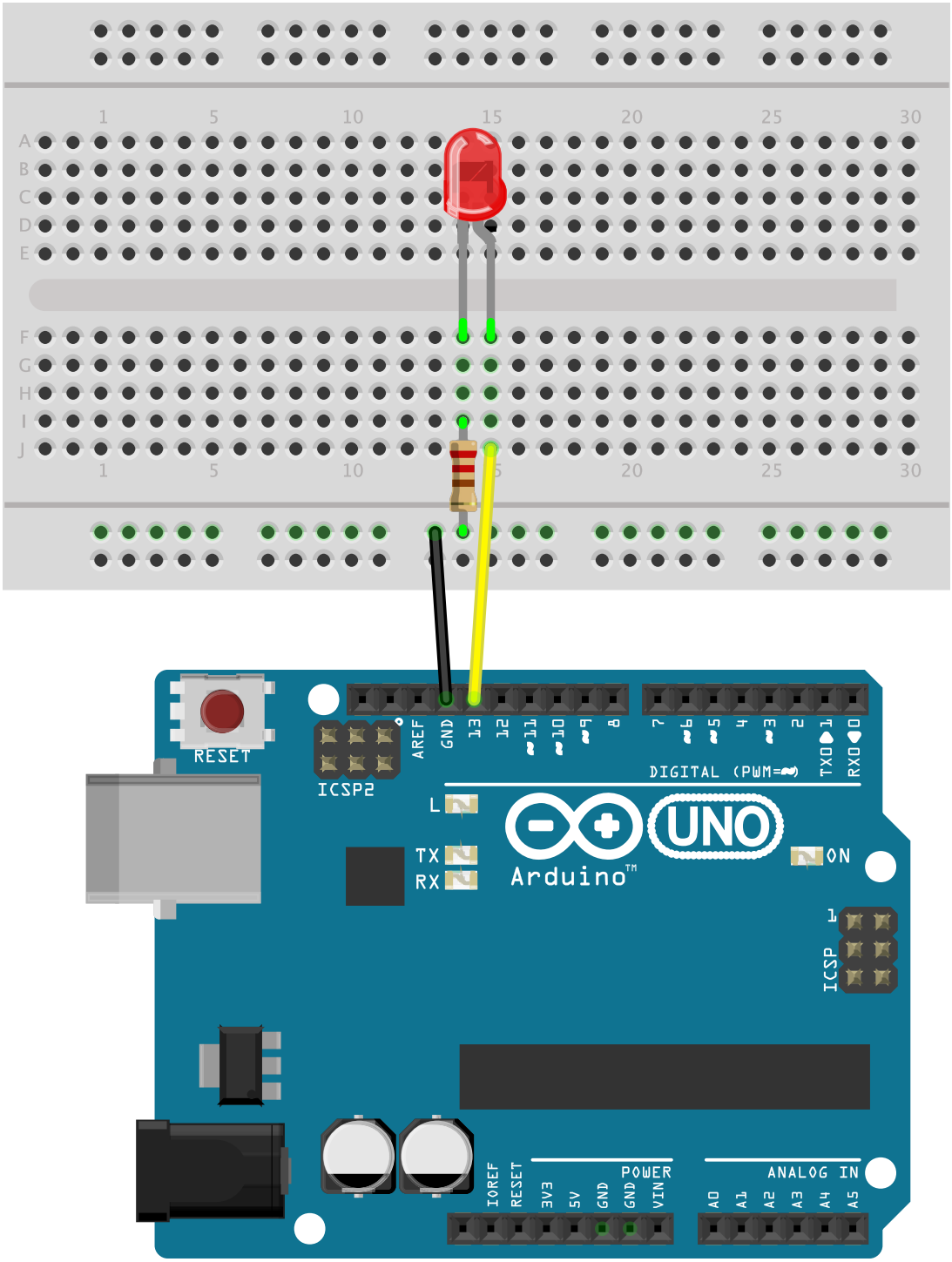
All the sketches you will see later have similar outlines, if you wish open another sketch and take a look at the code.

Practical sketches

We will build a couple of very simple experiments to illustrate the Arduino and IDE, these will allow newcomers to learn the trade, and get more experienced people back into the swing of things.

Blink an LED

In the world of Arduino this is always the very first sketch that anyone ever runs. An LED is connected to one of the Digital Outputs (D13) of the board and made to flash on/off under control of a sketch. Wire this up:



The +ve terminal of the LED is the longer wire and goes to pin D13, and the -ve goes to a 220R resistor to ground GND.

Open the sketch

```
File > Sketchbook > My_Blink
```

and click the top menu bar right arrow button. This will compile your sketch into the machine code which the Arduino microprocessor uses, and upload it to your board. The LED will start to blink!

The sketch code is below:

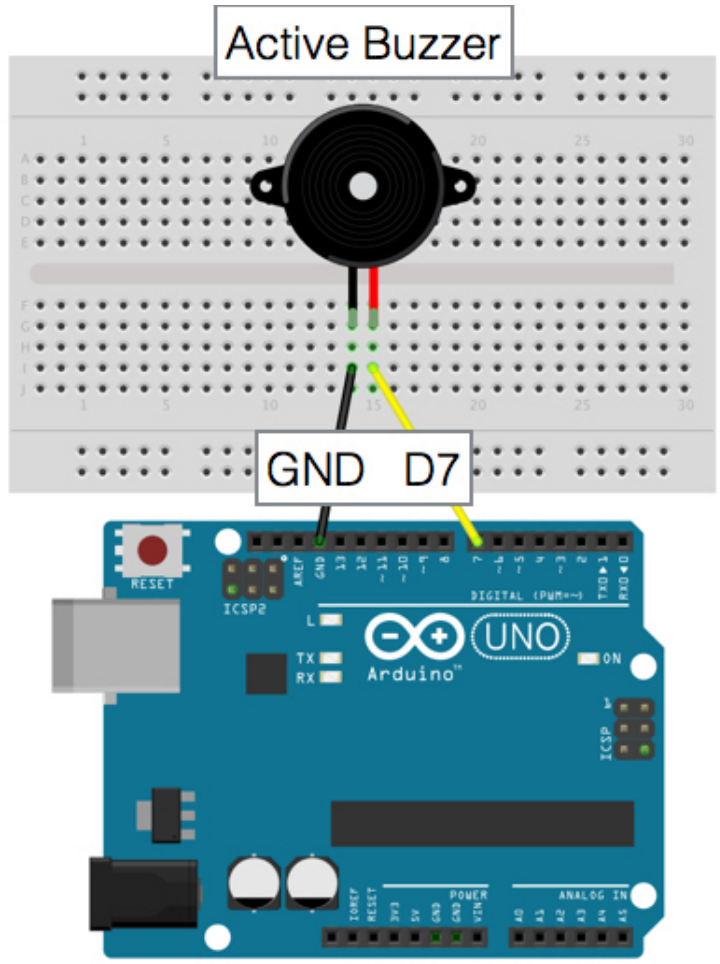
```
// My_Blink
// flashes a LED on pin 13

// pin number
#define LED 13

// the setup routine runs once when you upload (->) the sketch
void setup() {
  // initialise the digital pin 13 as an output
  pinMode(LED, OUTPUT);
}

// the loop runs over and over again, forever
void loop() {
  digitalWrite(LED, HIGH);  // turn the LED on (HIGH voltage level)
  delay(1000);              // wait for 1 second (1000ms)
  digitalWrite(LED, LOW);   // turn the LED off (LOW voltage leve)
  delay(1000);              // wait for 1 second
}
```

There is no "included” library for this sketch. The LED pin 13 connection is "defined" and called “LED”. There is a "setup" function, which defines the mode of the LED pin as an output, and finally there is a "loop" which alternately writes a HIGH or LOW to the LED pin with a delay of 1sec between steps. This loop repeats for ever.

Send some morse code

The next sketch is a real Amateur radio application. It is able to generate good morse code from any text you type in on your computer.

Wire up the active buzzer to pin D7.



Open and upload the sketch

```
File > Sketchbook > My_Morse_KB
```

This will compile your sketch into the machine code which the Arduino microprocessor uses, and upload it to your board.

Take some time to read the sketch code and understand it. Notice that this time there is an “include” library, which provides the code to generate the morse signals. This library was freely downloaded from the web, where you will find innumerable other useful libraries.

Note also that the code creates a “morseOut” object. This uses the library to create an object which inherits the class behaviours specified in the library, so we can now use it, for example like this

```
morseOut.setspeed(WPM); // set the WPM
morseOut.encode(); // set encode mode
```

```
morseOut.write(text); // send out as morse
```

Now open the “Monitor” which you can do by clicking on the “magnifying lens” button at top right of the IDE menu bar. Make sure the boxes at the bottom of the window are “Newline” and “9600” to set the correct protocol for communication to your sketch.

Now enter some text, for example “CQ DE <your call sign> K” at the top and hit return. Your buzzer will send the morse of this message. Try changing the WPM and reload the program.

You have completed the first session.

Next session - VFO prototype

In the next session we will prototype a digital VFO