

# Arduino For Amateur Radio

By Julie VK3FOWL and Joe VK3YSP

- Introduction to Arduino
- Arduino Models, Books and Add-Ons
- Amateur Radio Applications
- Arduino Pro Micro
- Introduction to Arduino Programming
- More on Arduino Programming
- Workshop

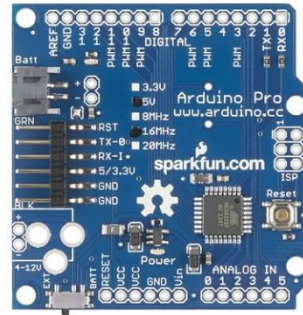
# Introduction to Arduino

- Atmel 8, 16 or 32-bit microcontrollers
- Inexpensive: \$3 Pro Mini - \$6 Pro Micro
- Cross platform: Windows/Linux/MacOSX
- Open Source Hardware and Software
- Free Integrated Development Environment (IDE)
- USB programmer built in
- C/C++ Compatible (almost)
- I/O: Digital, Analog, I2C, SPI, USB, TTL Serial

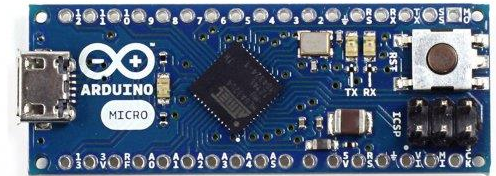
# Arduino Models



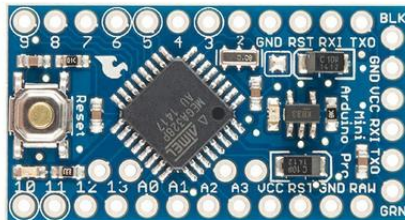
UNO



PRO



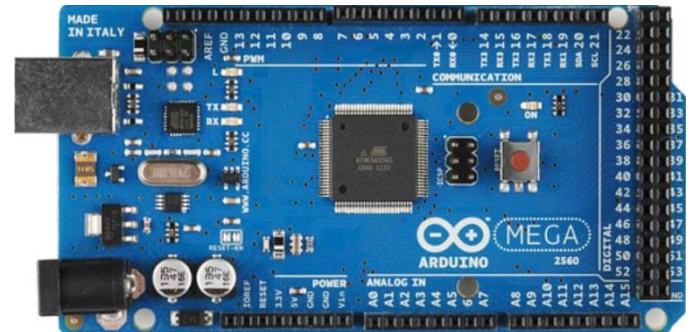
MICRO



PRO MINI



NANO



MEGA



# Arduino Books



...PLUS  
MANY  
MORE

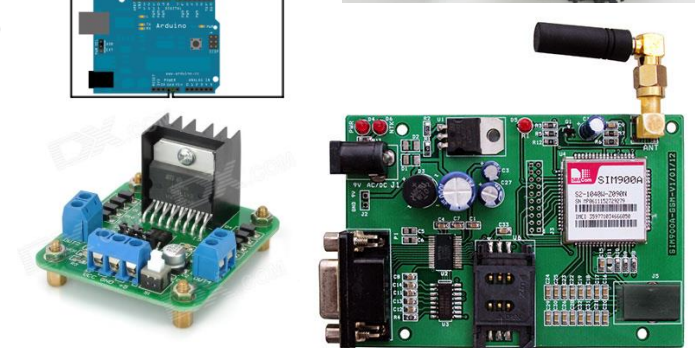
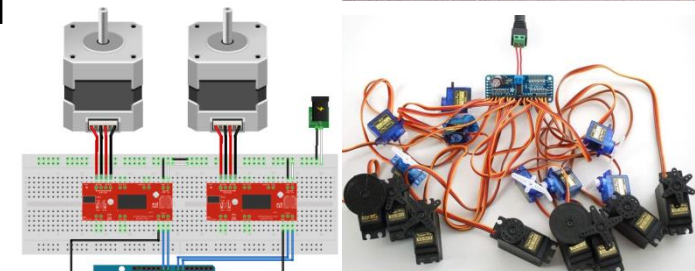
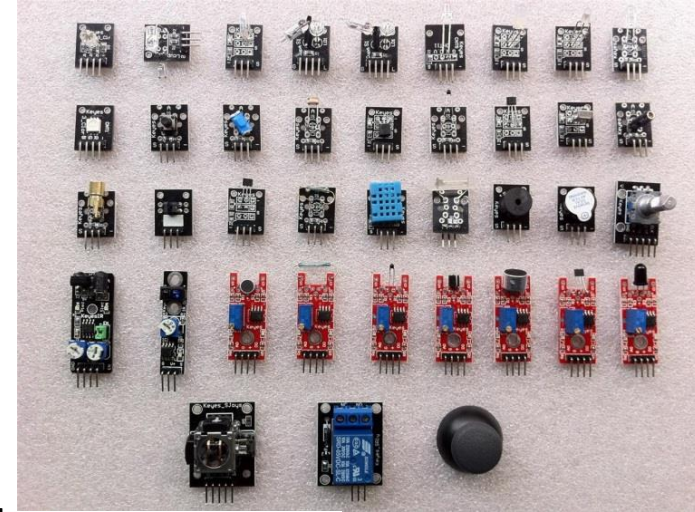


# Arduino Add-Ons

## Sensors

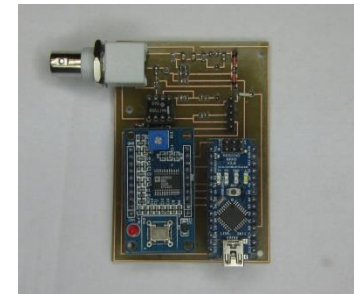
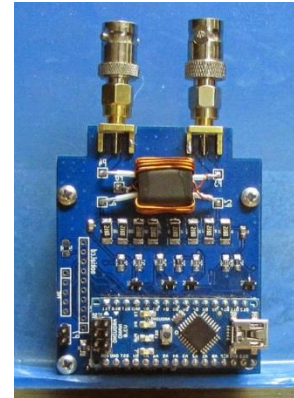
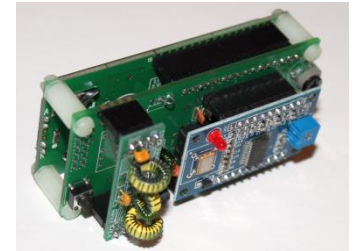
- GPS Receiver
- Accelerometer
- Magnetometer
- Barometer/Altimeter
- Anemometer/Airspeed
- Rate Gyroscope
- Thermocouple/IR/PIR
- Vibration/Tilt
- Hygrometer/Humidity
- Acoustic/Ultrasonic
- Force/Flex/Weight/Strain
- Gas/Water/Light/Colour/UV
- Fluid Level/Fluid Flow/pH
- Proximity/Line/Touch
- Pulse/Heartbeat/BP
- ECG/Brainwave
- Ionizing Radiation
- Switch/Pot/Encoder
- Keyboard/Joystick/Touchpad
- Reed/Hall Effect Switch
- CAM/LIDAR/FLIR

- Gesture/Fingerprint
  - Hydrophone/Geophone
  - Barcode/RFID reader
  - Mag/Memory card reader
  - Coin Acceptor
  - Voltage/Current/Power
  - Analog to Digital Converter
- Drivers
- Linear/rotary actuators
  - Stepper motor/Servomotor
  - AC/DC Motor/Relay/Solenoid
  - Solenoid Valve
  - Buzzer/Bell/Siren
  - Digital to Analog Converter
- Displays
- Monochrome/Colour
  - Text/graphics/LED/LCD/OLED
- RF/Data Link/Remote Control
- DDS VFO
  - WiFi/MANET
  - GSM/3G/4G
  - IR/RF Remote



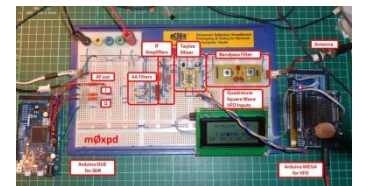
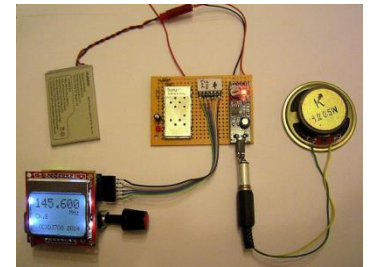
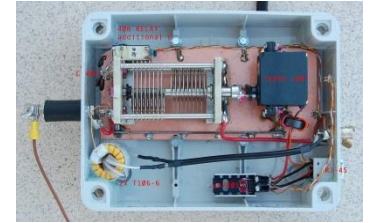
# Amateur Radio Applications

- DDS VFO Controller
- WSPR/QRSS/APRS beacon
- Digital SWR/Power meter
- Digital Antenna/Network Analyser



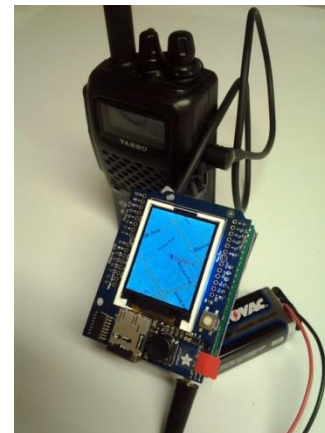
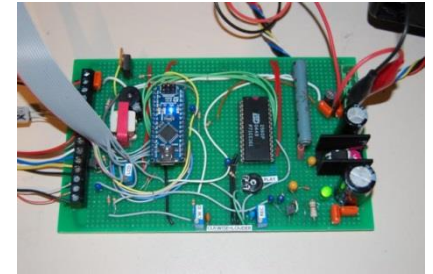
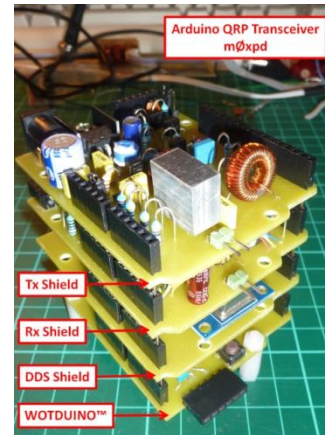
# Amateur Radio Applications

- Automatic Antenna Tuner
- Transceiver module controller
- Antenna rotator controller
- SDR Controller



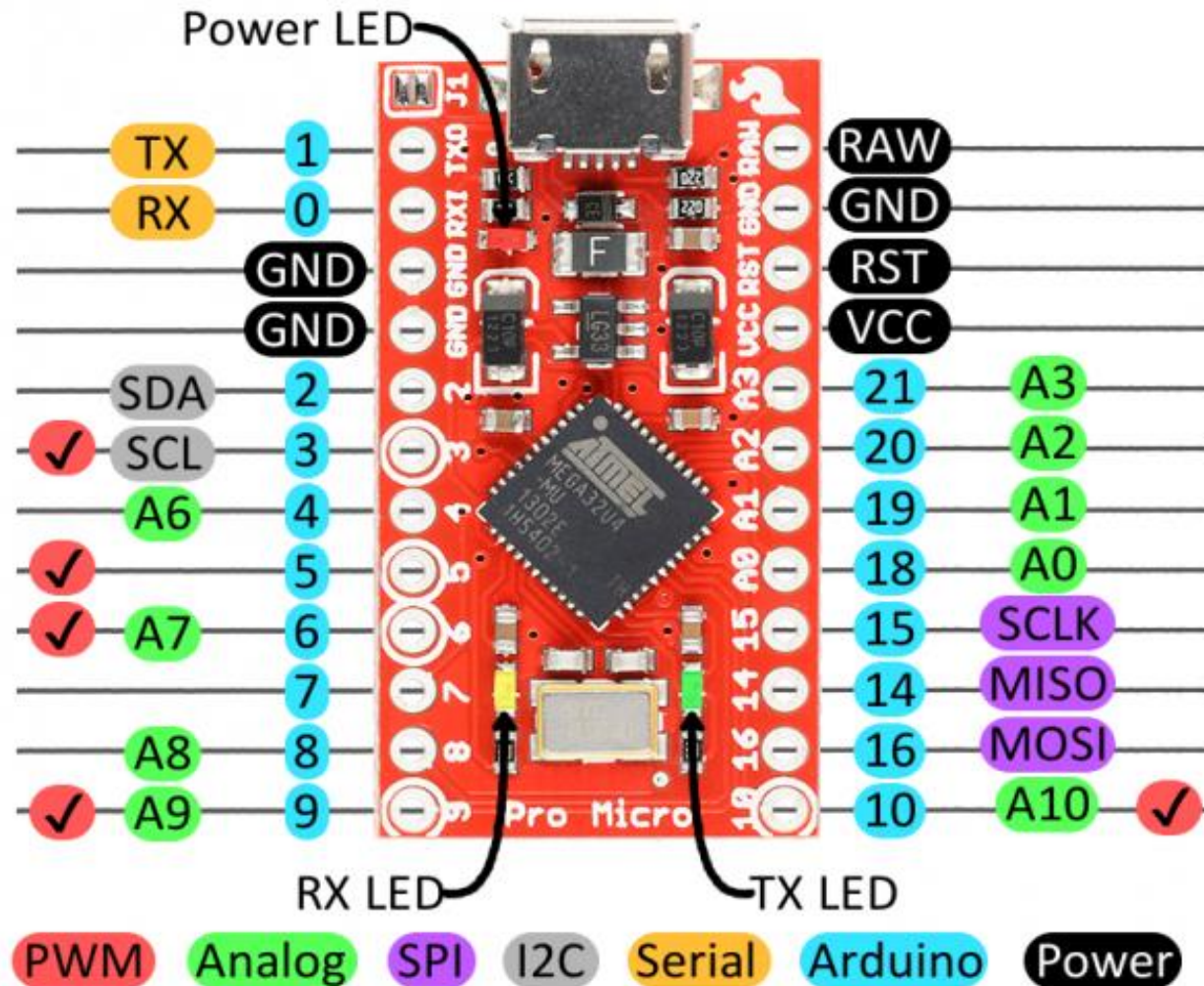
# Amateur Radio Applications

- Repeater controller
- QRP Transceiver
- Morse keyer/encoder/decoder
- APRS Display





# Arduino Pro Micro



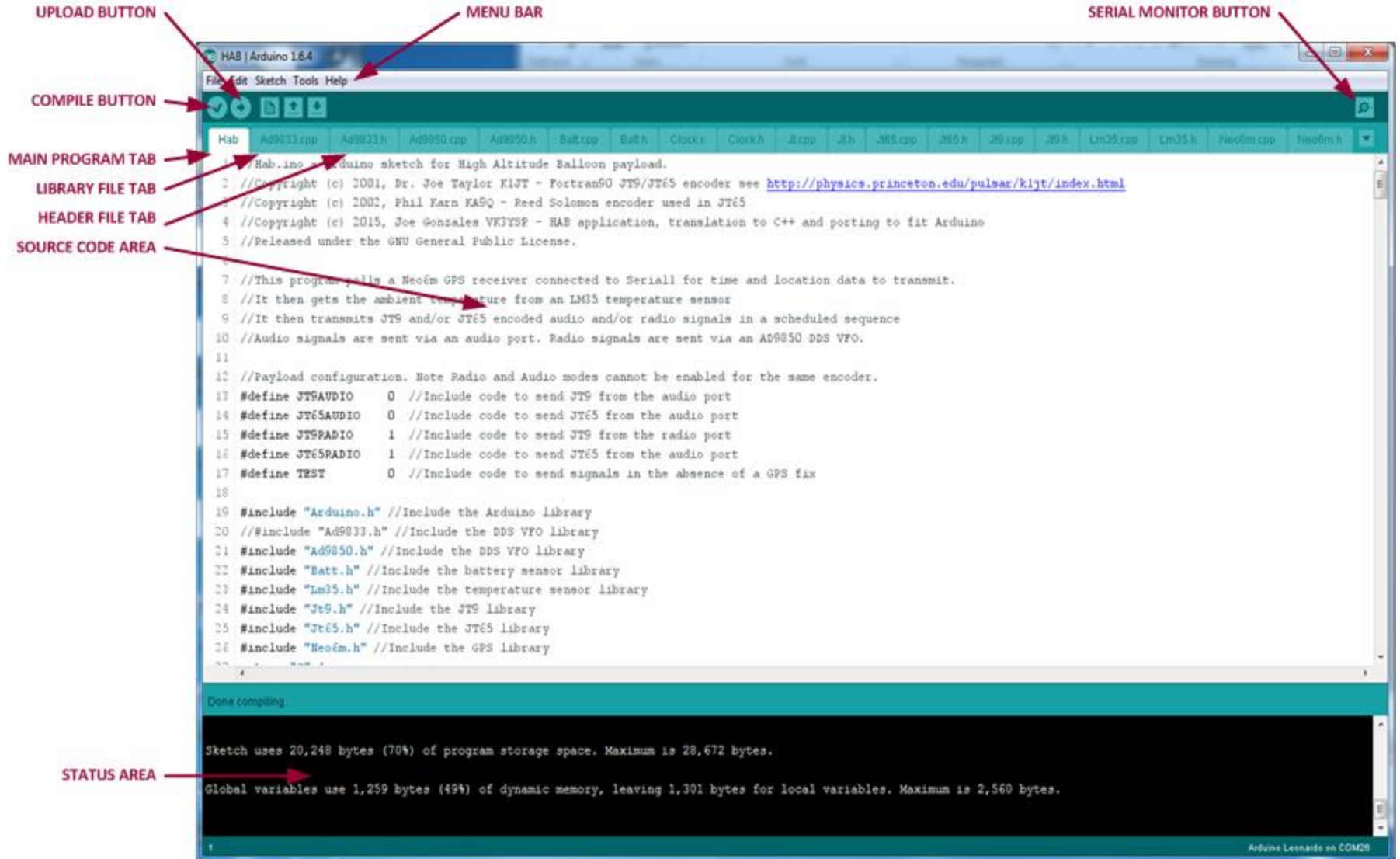
# Arduino Pro Micro

- High Performance, Atmel ATmega32U4, 16MHz, 8-Bit Microcontroller
- Advanced RISC Architecture, Low Power AVR<sup>®</sup>
- 5V–12V External Power or USB Power.
- 32KB Program memory, 2.5KB SRAM, 1KB EEPROM
- One 8-bit, two 16-bit and one 10-bit Counter/Timer/Prescaler/Capture/Compare
- USB 2.0 port for programming/serial monitor (No switch required for programming)
- Eighteen 5V Input/Output Ports for Digital TTL I/O
- Five 8-bit PWM Ports for PWM or Analog Output
- Nine 10-bit ADC Port for Analog Input with 2.5V or 5V reference
- Programmable USART for Serial TTL I/O
- 3-wire, SPI Serial Peripheral Interface. 2-wire, I2C Inter-Integrated Circuit Interface.
- Interrupts with wake-up on pin change.

# Introduction to Arduino programming

- Arduino Integrated Development Environment (IDE)
- Arduino Program Development
- Comments
- Statements
- Data Types
- Constants
- Variables
- Functions
- Programs
- Library Functions
- Program Design
- Program Code

# Arduino IDE





# Arduino Program Development

- Plug the Arduino into the computer USB port
- Start up the IDE application
- Load an existing source code (.ino/.c/.cpp/.h) file; or
- Type in a new source code file and save it
- Compile the source code to check it
- Fix any errors and re-compile it if necessary
- Upload the executable code to the Arduino
- Watch the Arduino reset and start running immediately
- Start the Serial Monitor for any serial I/O
- Unplug the Arduino and it will run stand-alone on power up

# Comments

- Comments are used to provide explanatory notes inline with the source code. They are ignored by the compiler and are not executed by the processor.

```
//This is a single line comment. It is not executed.
```

```
/*
```

```
These are comments over several lines.
```

```
They are not executed either.
```

```
*/
```

# Statements

- Statements are executable instructions.
- Statements are executed in order.
- Each statement finishes with a semicolon.
- Multiple statements executed as a group are enclosed in parentheses {}

```
{  
    a = 1;      //Statement 1  
    a = 2;      //Statement 2  
    b = a + 2    //Statement 3  
}
```

- Note: No semicolon required after the last statement

# Data Types

- Data types identify the format of information used in statements.
- Common data types:
  - `char` (character): A,a,B,b,C,c... !,@,#... 1,2,3...
  - `string` (strings of characters): "Hello"
  - `int` (integers): ...-1,-2,0,1,2...
  - `byte` (8-bit bytes): 0..255
  - `float` (floating point numbers): 1.234
  - `bool` (boolean): true/false
  - `void` (nothing):
- Information with different data types is typically not compatible, but can sometimes be converted.
- Data types are coloured `blue` in the IDE.



# Constants

- Constants are pre-defined words used instead of numbers in statements.
- They represent permanent values using more meaningful names.
- Constants are used to refer to Arduino pins
  - A0 refers to pin 18. SCLK refers to pin 15.
- Constants are used in Arduino functions
  - HIGH refers to 1 and LOW refers to 0
  - INPUT and OUTPUT refers to different pin modes

# Variables

- Variables are used to store transient values.
- They are declared by name with their data type:

```
int cup1; float cup2;
```

- Variables can then be assigned values:

```
cup1 = 2; cup2 = 1.5;
```

- Variables can be initialised (once):

```
int cup1 = 2; float cup2 = 1.5;
```

- Variables can be operated upon:

```
cup1 = cup1 + 2;  
cup1++ 2;
```

# Functions

- Functions define a set of frequently executed statements.
- Values can be passed to functions. Functions can return a value.
- Functions can be declared with different parameter and return data types.

```
float julie(int cup1, float cup2) {  
    float cup3 = cup1 + cup2; //Note: Here cup1, cup2 and cup3 are “local” variables  
    return cup3;  
}
```

- Functions can be called by passing parameters and assigning return values.

```
float cup3 = julie(2, 1.5); //Note: This cup3 is different. How about: float cup3 = julie(1.5, 2);?
```

- Functions can also be declared with no parameters and no return value.

```
void julie() {  
    //Statements  
}
```

- These functions can be called as follows:

```
julie();
```

# Programs

- All Arduino “sketches” are essentially C/C++ programs, but instead of the normal ANSI C main function, `int main()`, they must have two declared functions `void setup()` and `void loop()`. Each function takes no parameters and returns no result.

```
//This setup function is called once automatically on Arduino startup
void setup() {
  // put your setup code here, to run once:

}

//This loop function is called repeatedly after setup has completed
void loop() {
  // put your main code here, to run repeatedly:

}
```



# Library Functions

- The Arduino IDE provides many useful pre-defined library functions including:  
Digital I/O, Analog I/O, Serial I/O, Advanced I/O, Time, Maths, Trig, Random numbers, Interrupts and USB emulation.
- The IDE provides a web-based reference:  
Select Help | Reference to browse it.
- Library functions are **orange** in the IDE.

# More on Arduino programming

- Declaring more complex data types
- Compiler directives
  - #define, #include, #ifdef
- Control structures:
  - if/then/else/while/switch
- Operators:
  - comparison/arithmetic/boolean
  - bitwise/pointer/compound
- Interrupts
- More library functions
- Object oriented programming (C++)

# Program 1 Design

This program will flash a LED on and off.

The LED is connected between Pin 2 (+) and GND (-).

The setup function initializes pin 2 as a digital output pin and sets it LOW (so the LED will be off).

The loop function will repeatedly:

- Set Pin 2 HIGH (the LED will go on) then wait 100ms.

- Set Pin 2 LOW (the LED will go off) then wait 100ms.

Note: A series current-limiting resistor is required for the LED.  $R = (5 - V_f) / I_f$

# Library Function: pinMode

`void pinMode(int pin, int mode);`

Configures the specified pin to behave either as an input or an output.

- pin: The Arduino pin number
- mode: INPUT, OUTPUT or INPUT\_PULLUP

Note: The pull-up mode enables internal pull-up resistors (20kΩ - 50kΩ) to +5Vdc on the specified pin.

Example:

```
pinMode(2, OUTPUT); //Initialize pin 2 as an output
```

# Library Function: digitalWrite

`void digitalWrite(int pin, int value);`

Set the voltage on an output pin to the specified value. HIGH = 5V, LOW = 0V.

- pin: The Arduino pin number
- value: HIGH, LOW

Examples:

```
digitalWrite(2, HIGH); //Turn the LED on.
```

```
digitalWrite(2, LOW); //Turn the LED off.
```



# Library Function: delay

```
void delay(int ms);
```

Suspends program execution for the specified number of milliseconds.

- ms: The delay time in milliseconds

Example:

```
delay(100);    //Wait for 100ms
```

# Program 1 Code

```
//This program flashes a LED on and off.  
//Hardware: Arduino Pro Micro. LED with resistor connected between pin2 and GND  
//This program is free. It is released under the GNU General Public License.  
  
//Setup the Arduino.  
void setup() {  
    pinMode(2, OUTPUT);           //Initialize digital pin 2 as an output pin.  
    digitalWrite(2, LOW);         //Turn the LED off.  
}  
  
//Flash the LED.  
void loop() {  
    digitalWrite(2, HIGH);        //Turn the LED on.  
    delay(100);                   //Wait for 100ms.  
    digitalWrite(2, LOW);         //Turn the LED off.  
    delay(100);                   //Wait for 100ms.  
}
```

# Arduino Workshop

- Coffee break
- IDE installation
- Discussion, Questions and Answers
- Program 1 Development and Test

If time permits:

- Program 2 Design and Coding
- Program 2 Development and Test

# IDE Installation

- Install the IDE application on your laptop.
- Connect the Arduino to your laptop USB port.
- Start the IDE application and test the compiler.
- Test program upload and execution.
- Program 1: Blink a LED.
- Program 2: Measure the ambient light level.

# Program 2 Design

Reusing the code from the previous example, this program will measure the voltage on an analog pin and display it using the IDE serial monitor. It will then flash the LED at a rate proportional to the ambient light level.

A light dependent resistor ( $2\text{k}\Omega \sim 50\text{k}\Omega$ ) is connected between pin A0 and GND to provide a dynamic voltage input.

The setup function will initialize pin A0 as an analog input pin and enable an internal pull-up resistor ( $20\text{k}\Omega - 50\text{k}\Omega$ ).

The loop function will repeatedly:

Turn the LED on for 100ms.

Read the integer value of the 10-bit analog to digital converter (0 – 1023)

Compute the equivalent voltage =  $5 * (\text{value} / 1024.0)$

Send the voltage information to the serial port so it can be displayed by the serial monitor.

Use the analog value as the off delay time for the LED.

# Library Function: analogRead

```
int analogRead(int pin);
```

Reads the value of the specified analog pin.

- pin: The Arduino pin number
- returns: 0 – 1023, representing 0 – 5 volts

Example:

```
int value = analogRead(A0);
```



# Library Function: Serial.begin

`void Serial.begin(speed);`

Sets the speed of the Arduino serial port.

— speed: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200 bps.

Example:

```
Serial1.begin(9600);
```

# Library Function: Serial.print

`void Serial.print(val); void Serial.println(val);`

Sends data to the Arduino serial port.

– val: Any integer, float, character or string value

Examples:

<code>Serial.print(78);</code>	<code>//Prints 78</code>
<code>Serial.print(1.23456);</code>	<code>//Prints 1.23</code>
<code>Serial.print("N");</code>	<code>//Prints N</code>
<code>Serial.println("Hello");</code>	<code>//Prints Hello\r\n</code>

# Program 2 Code

```
//This program will measure the voltage on an analog pin and display it using the IDE serial monitor.  
//It will also flash the LED at a rate determined by the ambient light level  
//Hardware: Arduino Pro Micro. LED with resistor connected between pin2 and GND. Photoresistor connected between pin A0 and GND.  
//This program is free. It is released under the GNU General Public License.  
  
//Setup the Arduino serial port speed and the analog input pin  
void setup () {  
  Serial.begin(9600);           //Start up the serial port at a speed of 9600bps  
  pinMode(A0, INPUT_PULLUP);    //Initialize analog pin A0 as an input pin with internal pullup resistor  
  
  pinMode(2, OUTPUT);           //Initialize digital pin 2 as an output pin.  
  digitalWrite(2, LOW);         //Turn the LED off.  
}  
  
//Repeatedly read and display the input voltage. Note: It varies with the ambient light intensity  
void loop () {  
  int value = analogRead(A0);   //Read the raw analog value from the A0 pin  
  float voltage = 5 * value / 1024.0; //Compute the actual voltage  
  Serial.print("The voltage is: "); //Print the heading over the serial port  
  Serial.println(voltage);       //Print the voltage and a new line  
  
  digitalWrite(2, HIGH);        //Turn the LED on.  
  delay(100);                   //Wait for one hundred milliseconds  
  digitalWrite(2, LOW);         //Turn the LED off.  
  delay(value);                 //Wait for a variable amount of time determined by the ambient light level.  
}
```