**Arduino Integrated Development Environment (IDE)**

Now Let’s take a closer look at the Arduino software.

![Arduino IDE Interface Image]

**Diagram Notes:**
- Sketch Name
- Code Area
- Serial Monitor
- Verify
- Upload
- New
- Save
- Open
RedBoard Setup

Arduino Integrates Development Environment (IDE) Board

There are many, many kinds of Arduino boards in the marketplace and each one operates differently. It is important that we select the correct board for our application.

You will not find our SparkFun Redbaord listed as one of the options, but since it is based upon Arduino Uno board we select this option.
Arduino Integrates Development Environment (IDE)

Communication Port (Com Port)

In order to get the Redboard and your computer talking to each other we need to ensure the correct port is selected. This usually happens automatically, but it is good to double check at the start of every class. If the wrong port is highlighted simply click on the port that has the Arduino connected to it.

Testing the Connection

Prior to starting your coding and circuit building, it is good practice to make sure the computer and Redboard are functioning properly.

To do simply upload the basic sketch that is provided upon opening IDE and upload it to the Redboard.

If all goes well nothing should happen, but if you get any error messages then we know there might be a problem.
Getting Started

Arduino Integrates Development Environment (IDE)

Examples

Since we are not all coding experts Arduino has kindly provided us with the basic framework for many of the projects we will be building and also serve as a great resource to copy and paste from in the future as you design your own projects.

We will be using example regularly and then modifying or customizing the code to suit our needs.
Arduino Integrates Development Environment (IDE)

Libraries
Libraries are where all the complex behind the scenes stuff happens, like complex algorithms, mathematical computations, etc... Just like examples, the IDE has a number of build in libraries, but we need to add more to the system as we add new components.

Arduino IDE

Every sketch in IDE also for one (1) setup and one (1) loop... nothing more!

In the Setup you tell the program what to do, in the loop you tell the program how to do it!
**Arduino IDE**

**Setup**

In the setup this is where you tell setup the parameters for the program, variable, operations, etc...

**Loop**

In the loop, the program is repeated over and over again until the program is stopped. This generally where you will either send or collect data.

---

**Syntax**

Syntax refers to the spelling and grammar of a programming language. Computers are inflexible machines that understand what you type only if you type it in the exact form that the computer expects. The expected form is called the syntax.

Each program defines its own syntactical rules that control which words the computer understands, which combinations of words are meaningful, and what punctuation is necessary to be a correctly structured document.
Syntax - Semi-Colon

A semicolon needs to follow every statement written in the Arduino programming language.

Syntax - Double Backslash

When you want to annotate your code with comments on a single line a double backslash is used.

Good code is commented well. Comments are meant to inform you and anyone else who might stumble across your code, what the heck you were thinking when you wrote it.
Syntax – Star Backslash

When you want to annotate your code with comments longer than one lone long use the backslash-star surrounding the text

/*Text*/

Syntax – Curly Brackets

Curly braces are used to enclose further instructions carried out by a function (we discuss functions next). There is always an opening curly bracket and a closing curly bracket. If you forget to close a curly bracket, the compiler will not like it and give you an error code.
Syntax - int

Allows you define a variable to an integer data type.

Example:

```c
int RedLED = 13
```

Syntax - Functions

Functions are pieces of code that are used so often that they are encapsulated in certain keywords so that you can use them more easily.

In Arduino, there are certain functions that are used so often they have been built into the IDE. When you type them, the name of the function will appear orange.
Syntax - Functions

Many functions require arguments or information to work so it knows what to process and how. These arguments are placed in parentheses at the end of each function.

Syntax - Digital Functions

The function `pinMode()` is a common function used to designate the mode of an Arduino pin.

Example:
`pinMode(13, OUTPUT)`

Notice that the word OUTPUT is blue. There are certain keywords in Arduino that are used frequently and the color blue helps identify them. The IDE turns them blue automatically.
Syntax - Digital Functions

The function `digitalWrite()`, is a common function used to designate a pin as On (HIGH) or off (LOW)

Example:
`digitalWrite(5, HIGH)`

Syntax - Digital Functions

The function `digitalRead()`, is a common function used to read if a digital pin as On (HIGH) or off (LOW)

Example:
`digitalRead(7)`
Syntax – Analog Functions

`analogReference()` configures the reference voltage used for analog input (i.e. the value used as the top of the input range). The default analog reference of 5 volts (on 5V Arduino boards) or 3.3 volts (on 3.3V Arduino boards).

Example:
```
analogReference(9)
```

Syntax – Analog Functions

`analogRead()` reads the value from the specified analog pin. The Arduino board contains a 6 channel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit. The input range and resolution can be changed using `analogReference()`.

Example:
```
analogRead(9)
```
Syntax - Analog Functions

`analogWrite()` writes an analog value to a PWM pin (3, 5, 6, 9, 10 & 11). Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to `analogWrite()`, the pin will generate a steady square wave of the specified duty cycle until the next call to `analogWrite()` (or a call to `digitalRead()` or `digitalWrite()` on the same pin).

Example:

`analogWrite(11, 855)`

855 could be any value between 0 and 1023

Syntax - Other Functions

`delay()` pauses the program for the amount of time (in milliseconds) specified as parameter. Note: there are 1000 milliseconds in a second.

Example:

`delay(300)`
Syntax - Other Functions

Serial.begin() is a command you give to the Arduino to begin serial communication. Inside the IDE, there’s a serial monitor that outputs data that you configure it to output. It won’t work unless you begin the serial communication using. Typically a baud rate of 9600 is highly suggest.

Example:

Serial.begin(9600)

Syntax - Other Functions

serial.println() is a command that the program gives to the Arduino to print the label to go with the to the serial monitor data.

Example:

delay("Voltage")
An *if()*/*else()* statement is used when you want to compare two things, similar to a true/false comparison...if true do this...if false do that. Note: when programming you use two (2) equal signs (==) when comparing to items...a single equal signs sets the the value.

Example:

```cpp
if(1 == LOW) {
    digitalWrite(3, HIGH)
    digitalWrite(4, LOW)
} 
else {
    digitalWrite(3, LOW)
    digitalWrite(4, HIGH)
}
```

### Types of Comparisons:

- `x == y` (x is equal to y)
- `x != y` (x is not equal to y)
- `x < y` (x is less than y)
- `x > y` (x is greater than y)
- `x <= y` (x is less than or equal to y)
- `x >= y` (x is greater than or equal to y)
Syntax - For Loop

The `for()` statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The `for()` statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins. There are three parts to the `for()` loop:

for(initialization; condition; increment) {
}

Example:

```c
for(int x = 0; x < 0; x++) {
}
```