Here are some interesting facts about the Programming Language the Arduino’s use known as “C”. C is probably the coolest programming language. It’s a bit like rock n’ roll: It has been around since the 1960s but we keep making new cool things with it every year! Almost all professional programming languages are versions of C. The Arduino understands C: so if you speak to it in C, it will do what you tell it to do. Getting started is really easy as long as you remember to type things a special way. Here are some things to get you started. If you need help, just ask Google!

Input and output functions

The most important words when speaking to Arduino tell it what is connected to the wires we are poking it with!

**pinMode(pin, mode)** - Set a digital pin to behave either as an input or an output. Example:

pinMode(7,INPUT); // turns pin 7 into an input

**digitalWrite(pin, value)** - Turns a digital pin either on or off. Pins must be made into an output using pinMode before digitalWrite will have any effect. Example:

digitalWrite(8,HIGH); // turns on digital pin 8

**int digitalRead(pin)** - Reads the state of an input pin, returns HIGH if the pin senses some voltage or LOW if there is no voltage applied. Example:

val = digitalRead(7); // reads pin 7 into val

**int analogRead(pin)** - Reads the voltage applied to an analog input pin and returns a number between 0 and 1023 that represents the voltages between 0 and 5 V. Example:

val = analogRead(0); // reads analog input 0 into val

**analogWrite(pin, value)** - Changes the PWM (Pulse Width Modulation) rate on one of the pins marked PWM: pin may be 11, 10, 9, 6, 5, 3: value may be a number between 0 and 255 that represents the scale between 0 and 5 V output voltage. Example:

analogWrite(9,128); // Dim an LED on pin 9 to 50%

Variables

Variables are where you “store” (save for later) data that you can use in your “sketch” (your computer program). They can be changed as many times as you like in you sketch. When you “declare” (type the first time) a variable you have to specify its type. This means telling the Arduino the size of the value you want to store. T 2 most useful types are :

**Int -** Any number between –32,768 and 32,767. This is the most common data type used in Arduino. It uses 2 “bytes” of memory. (A byte is the word used to describe the space used by a computer to remember something.)

**char** A single character, such as the big letter A or the euro symbol €. Like any computer, Arduino stores it as a number, even though you see text. When chars are used to store numbers, they can hold values from –128 to 127.

Special symbols

Arduino includes a number of symbols to delineate lines of code, comments, and blocks of code.

**; (semicolon)** - Every instruction (line of code) is terminated by a semicolon. This syntax lets you format the code freely. You could even put two instructions on the same line, as long as you separate them with a semicolon. (However, this would make the code harder to read.)

Example:

delay(100);

**{}** (curly braces) - This is used to mark blocks of code. For example, when you write code for the loop() function, you have to use curly braces before and after the code.

Example:

void loop()

{

Serial.println("ciao");

}

// (comments) - These are portions of text ignored by the Arduino processor, but are extremely useful to remind yourself (or others) of what a piece of code does.

Example:

//this text is ignored until the end of the line

Control Structures

Arduino includes keywords for controlling the flow of your sketch.

**if . . . else** - This structure makes decisions in your program. if must be followed by a question specified as an expression contained in parentheses. If the expression is true, whatever follows will be executed. If it’s false, the block of code following else will be executed. It’s possible to use just if without providing an else clause. Example:

if (val == 1)

{

digitalWrite(LED,HIGH);

}

**while** - Similar to if, this executes a block of code while a certain condition is true.

Example:

// blink LED while sensor is below 512

sensorValue = analogRead(1);

while (sensorValue < 512)

{

digitalWrite(13,HIGH);

delay(100);

digitalWrite(13,HIGH);

delay(100);

sensorValue = analogRead(1);

}

**for** - Lets you repeat a block of code a specified number of times. Example:

for (int i = 0; i < 10; i++)

{

Serial.print("ciao");

}

**return** - Stops running a function and returns from it. You can also use this to return a value from inside a function. For example, if you have a function called computeTemperature() and you want to return the result to the part of your code that invoked the function you would write something like:

int computeTemperature()

{

int temperature = 0;

temperature = (analogRead(0) + 45) / 100;

return temperature;

}

Arithmetic and formulas

You can use Arduino to make complex calculations using a special syntax. **+** and **–** work like you’ve learned in school, and multiplication is done using with an **\*** and division with a **/**. Examples:

a = 2 + 2;

light = ((12 \* sensorValue) - 5 ) / 2;

remainder = 3 % 2; // returns 2 because 3 / 2 has remainder 1

Comparison Operators

When you specify conditions or tests for if, while, and for statements, these are the operators you can use:

**==** (equal to)

**!=** (not equal to)

**<** (less than)

**>** (greater than)

**<=** (less than or equal to)

**>=** (greater than or equal to)

Time functions

Arduino includes functions for measuring elapsed time and also for pausing the sketch.

**delay(ms) -** Pauses the program for the amount of milliseconds specified. Example:

delay(500); // stops the program for half a second

**delayMicroseconds(us)**

Pauses the program for the given amount of microseconds. Example:

delayMicroseconds(1000); // waits for 1 millisecond

Many thanks to team at CoderDojo Dalkey for this source material.