A short introduction to Arduino Programming

Fauzan

July 6, 2016

1 What's the program?

Armed with an LED, you want to banish darkness at the push of a button. Because just using a battery and wires and sticky black tape is so fifth grade stuff, you are going to use the mighty Arduino as your saviour. Congratulations, you have defined your **Problem**.

```
I want to banish darkness at the push of a button with the Arduino
    as my saviour.
```

You know what you want to do, but how are you going to do it? How is a *dumb someone else* (lets call hime Dumbo) going to do it? Dumbo may have no moral obligations of banshing darkness and elightening your path. Actually, Dumbo does not even know why darkness is evil or enlightenment is virtuos. You have to give line by line instructions to Dumbo, explaining what to do. Tricky, isn't it? This instruction set is the **Algorithm**. The algorithm to the above problem is given below.

```
1  Hey Dumbo!
2
3  Check the button.
4
5  If button is pressed and LED is OFF, turn LED ON.
6  If button is released and LED is ON, turn LED OFF.
7
8  Repeat lines 3-8.
```

But our Dumbo, the Arduino does not know about LEDs and buttons. It has I/O pins and knows how to read and set voltages. Let us try to write our algorithm in terms of pins and voltages.

```
#Button terminal A is at 5V pin
#Button terminal B is at pin 2
#LED cathode is at pin 13

Set pin 2 to input mode
```

```
6      Set pin 13 to output mode
7
8      Create a variable named "state"
9
10     state = value at pin 2
11
12     If state = HI, Set pin 13 to HI
13     If state = LO, Set pin 13 to LO
14
15     Goto line 10
```

We have another obstacle. Dumbos come in all shapes and sizes. Worst, hardly any of them are fluent in English. Dumbos understand languages which are, by today's standards, extraordinarily tedious for us to directly write in. Arduino understands hex code which looks like this.

```
:1001A000611108C0F8948C91209582238C938881EA
:1001B00082230AC0623051F4F8948C91322F30952A
:1001C00083238C938881822B888304C0F8948C913C
:1001D000822B8C939FBFDF91CF9108950F931F9334
:1001E000CF93DF931F92CDB7DEB7282F30E0F90110
:1001F000E859FF4F8491F901E458FF4F1491F90138
:10020000E057FF4F04910023C9F0882321F0698350
:100210000E948C006981E02FF0E0EE0FFF1FE25595
:10022000FF4FA591B4919FB7F8948C91611103C0D1
:100230001095812301C0812B8C939FBF0F90DF917C
:10024000CF911F910F910895CF93DF93282F30E026
:10025000F901E859FF4F8491F901E458FF4FD49117
:10026000F901E057FF4FC491CC2391F081110E9416
:100270008C00EC2FF0E0EE0FFF1FEC55FF4FA59127
:10028000B4912C912D2381E090E021F480E002C014
:100290000080E090E0DF91CF91089508950E94A7013A
```

However, we can **Program** in easier languages which can be translated to Dumbo language. Or, high level languages can be compiled to low level languages. Programs written in a high level language, C++ can be compiled to a hex file, which is then uploaded to the arduino. You can also write code in Assembly, a low level language, and compile it to hex.

The Arduino sketch for our algorithm looks like this. It will be explained in further sections.

```
const int buttonPin = 2;
const int ledPin = 13;

int buttonState = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

You can not upload the above code directly to the Arduino. It has to be compiled to a lower level language (C++ is a high level language). For that we need a **Compiler**. Some people have combined an editor, compiler, uploader and other nice things into a single package, the Arduino IDE.

Note that the IDE is not necessary to work with, but recommended. You can use your own language, editor, compiler and uploader.

2 Setup the Arduino IDE

2.1 Download and install the IDE

The relevant packages can be obtained from `www.arduino.cc/en/Main/Software`. In the time the package downloads, you can go through the next section.

2.1.1 Windows

Download and run the installer.

2.1.2 Linux

Download the `arduino-1.X.X-linuxXX.tar.xz` archive. Open the terminal and run the below commands one by one. Navigate to the `Downloads` directory, extract the archive, move it to `opt` directory and execute script to create desktop shortcut, menu item and file associations.

```
          cd Downloads
          tar -xvf arduino-1.X.X-linuxXX.tar.xz
          sudo mv arduino-1.X.X /opt
          cd /opt/arduino-1.X.X/
          ./install.sh
```

2.1.3 Browser

Go to `create.arduino.cc/editor/`

2.2 Test

Open the IDE. Connect the Arduino to the computer using a USB A to USB B cable. Under File, Examples, 01.Basics, select Blink. Under Tools, Board, select the proper board. Uner Tools, Port, select a port. Below the menu bar, click on the second circle with an arrow. Check the Arduino board. An LED should be blinking.

If you are getting errors, reset the Arduino board and reconnect it to the computer.

3 Arduino sketches

3.1 Some basics

Code that you write in the IDE is a sketch. It is not the same as C++ code, but very similar. Any sketch is made up of statements. All statements must end with a `;`.

3.1.1 Variables

You can store data in **variables**. There are different types of variables: integers, decimal numbers, strings, chararcters, booleans, etc. Before you can use a variable, you have to create it by writing a declaration statement.

Snippet 1: Declaration

```
        int apples;
```

The above statment creates a variable called `apples` of type `integer`. All `integer`s are granted `2 bytes` of memory in Arduino. Presently, the memory contains a `garbage value`[1]. You can change the value stored in `apples` with the help of an assignment statement.

Snippet 2: Assignment

```
        apples = 7;
```

You can declare and assign value to a variable in a single statement.

Snippet 3: Assignment

```
        int apples = 7;
```

You can declare different types of variables.

```
        int apples = 5;
        float applejuice = 4.5;
```

[1]You will see later that global variables are by default initialised to zero.

```
        char initial = 'A';
        bool isfruit = true;
```

3.1.2 Expressions

You can do mathematical operations in the sketch.

Snippet 4: Expressions

```
        1+2;           //Evaluates to 3
        5-9;           //Evaluates to -4
        apples*5;      //Evaluates to 28, since apples = 7
        24/4;          //Evaluates to 6
        sizeof(int);   //Evaluates to 4
```

`+`, `-`, `*`, `/` and `sizeof` are examples of operators. All operstors need one or more ardguments.

Another operator is `%`, the `modulo` operator.

```
        1%2;           //Evaluates to 0
        2%2;           //Evaluates to 1
        3%2;           //Evaluates to 0
        13%apples;     //Evaluates to 6
```

There are also boolean operators, AND `&&`, OR `||` and NOT `!`.

```
        1&&0;          //Evaluates to 0
        true||0;       //Evaluates to 1
        !3;            //Evaluates to 0
        !false;        //Evaluates to 1
        apples||false; //Evaluates to 1
```

In boolean logic, zero is equivalent to `false`, and any non zero value is equivalent to `true`.

What will this expression evaluate to?

```
        1+5*3%4;
```

While all operators have a precedence level, it is not easy to remember the order. We can instead use brackets to control the order of evaluation. The below expression is much easier to understand.

```
        % 1+((5*3)%4);
```

As you may have guessed, variables can be assigned as well used as arguments in an expression.

```
            float applejuice = apples * 0.9; //Creates applejuice and sets it to 4.5
```

3.2 Some code!

On opening the IDE, you will see the below snippet in the code window:

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

These are two **functions**. The `setup` function is run once when the Arduino is powered up or reset. The `loop` runs repeatedly untill the Arduino is powered down.

3.2.1 Hello World

Let us now write a program that will print `Hello World` on our computer screen. Our computer and the Arduino are two separate computers. We will need to communicate between the two. Before we can do so, we need to set the speed of communication. Add the following line to the setup function.

```
            Serial.begin(9600);
```

`Serial.begin()` is a function which configures the Arduino to send data through the serial port at a baud rate wihich is given as an argument. This function takes one argument of integer type. We have provided `9600` as an argument. Thus the Arduino will send data at a rate of 9600 baud.

Now we can send data which will printed on our screen. Add the following line to the loop function.

```
            Serial.println("Hello World");
```

`Serial.println()` is a function that takes a value and sends it through the serial port. In this case, we want to send a string. A string is enclosed within two `"`. `Hello World` and not `"Hello World"` will be sent through the serial port.

Our code looks like this:

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Hello World");
}
```

Now we are ready to upload! Click the upload button below the menubar. This will compile the code and upload it to the Arduino.

Before we can see anything on the screen, we need to set up a serial monitor. Thankfully, it is included in the IDE. Click on the magnifying glass on the right end below the menubar. Select a baud rate equal to the one at which Arduino is sending us data, ei., 9600. You will see this on the screen:

```
          Hello World
          Hello World
          Hello World
          Hello World
```

What is happening here? The Arduino is printing the string once, then again, and again as fast as it can. The speed of printing is limited by the processor. What if we want a controlled output?

3.2.2 Hello World . . . one second please

Say we want to print `Hello World` once every second. There is a function `delay()` that can help us. Add this line below `Serial.println()`.

```
          delay(1000);
```

`delay()` takes one argument, the number of milliseconds to wait. When the Arduino encounters this statement, it waits fo rthe specified interval.

Upload the code and check the serial monitor.