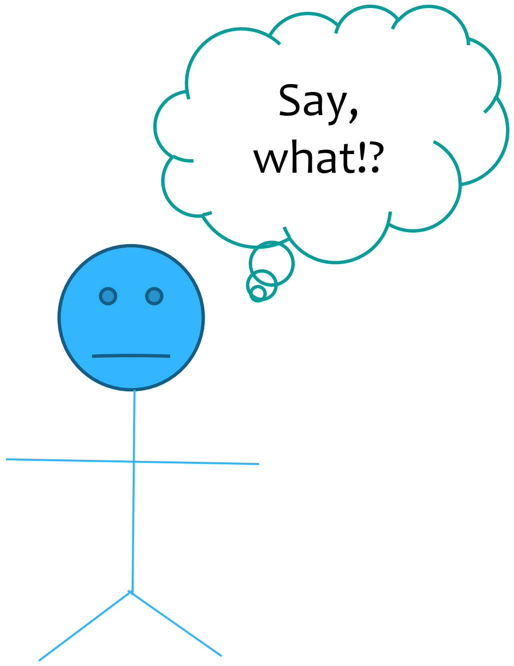
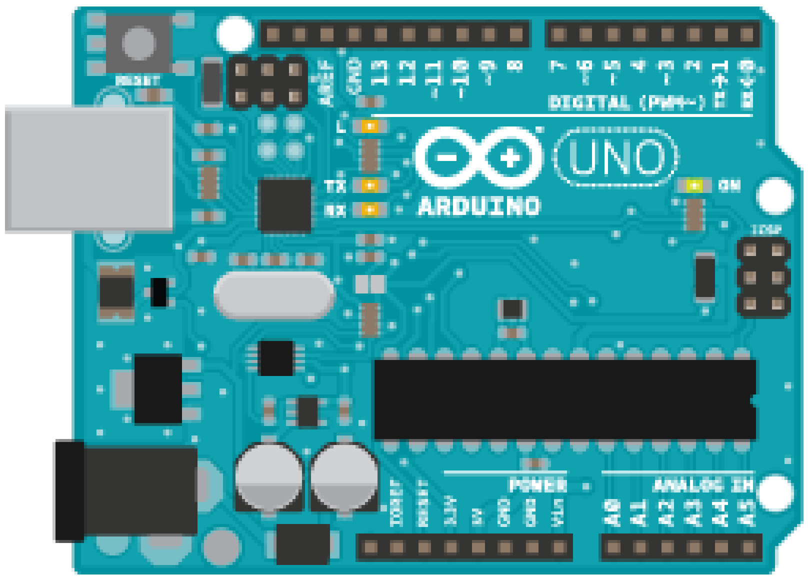
Introduction to Programming

Writing an Arduino Program

What is an Arduino?

* It’s an open-source electronics prototyping platform.





Let's Define It Word By Word…

- **Open-source:** "Resources that can be used, redistributed, or rewritten free of charge. (Often software or hardware.)"
- **Electronics:** "Technology which makes the use of controlled motion of electrons through different media."
- **Prototype:** "An original form that can serve as a basis or standard for other things"
- **Platform:** "Hardware architecture with software framework on which other software can run."

Verbiage from Jody Culkin, Arduino comic

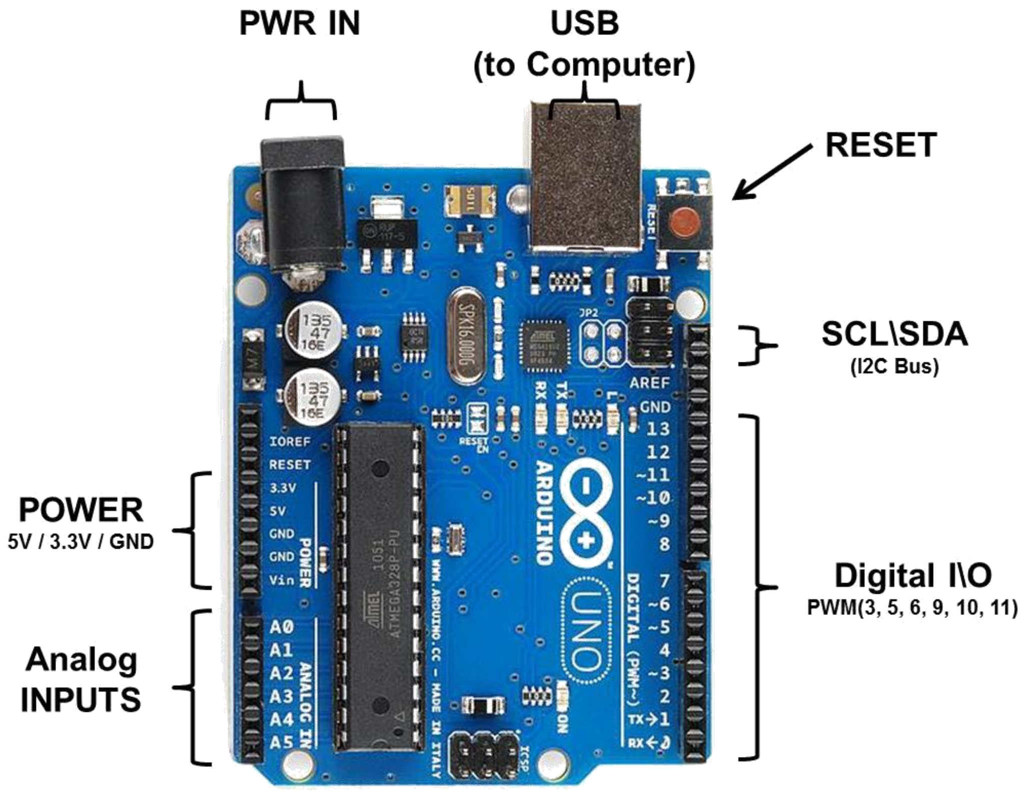
How does it work?

The Arduino contains a ***microchip***.

- A ***microchip*** is a very small computer that you can program.
  - You can attach sensors to it, to measure conditions
    - Such as the amount of light in a room
  - It can control how other objects react to those conditions
    - Room gets dark, a light turns on

Hardware Parts of the Arduino

- Power In – where the Arduino is powered (battery or wall charger)
- USB - allows you to connect Arduino to computer to program it
- Reset – restarts the program running on the board, same as unplugging and plugging it back in.
- Analog & Digital – covered later



How does this relate to the weather information system device?

The weather information device is powered by an Arduino.

If we try to “reverse engineer” the weather device, we need to figure out what method is used by the device to report the temperature and light conditions.

Basic Programming Concepts

What is a program? Code? Algorithm?

A program is a set of instructions given to a microcontroller or computer to carry out a process.

Code is lingo used to describe a program.

An algorithm is an ordered set of steps required to complete task(s) or solve problem(s). Algorithms set the basis for a program. Though you likely use algorithms daily – brushing teeth, walking, putting deodorant on, making a sandwich, etc.

What are Variables?

In programming, variables are used to refer to and hold a specific value. The value can vary – hence variable.

Just like algebra variables with added rules:

- Can be named whatever you like,
- Can't have spaces,
- Are case and spelling sensitive,
- When performing calculations the variable name that will hold the answer goes on the left of the =
  - Temp_F = Temp_C *(9.0/5.0) + 32.0;
  - ***Note:*** Temp_C is a variable that already has a value, Temp_F will get its value from the equation above.

Variables can be thought of as storage boxes

Let’s watch a video to better conceptualize what role variables play in programming and how they work

https://www.youtube.com/watch?v=T6OMJIIsEFE

Logic Flow Chart – Basic Shapes

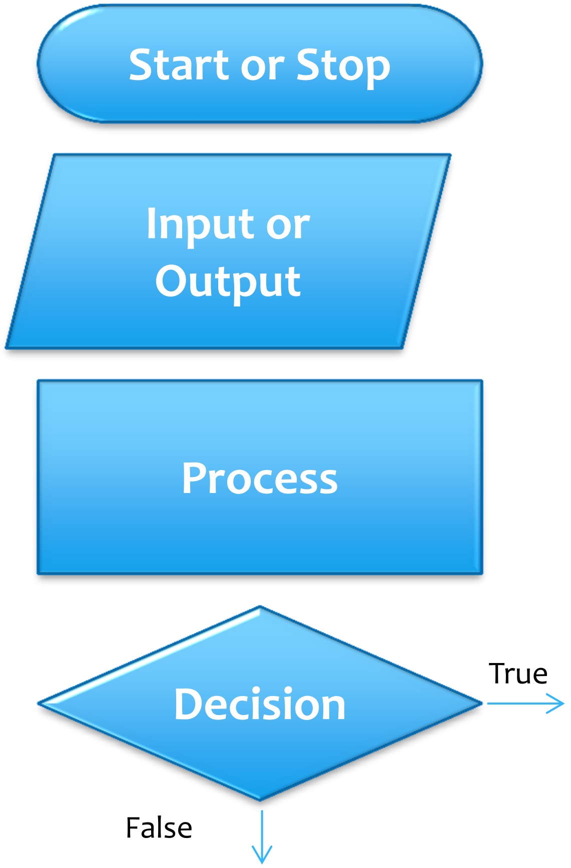
Before engineers code, they develop flow charts to outline the algorithm required to solve a problem & they use variables to hold values.

- **Start or stop:** begin or end an algorithm
- **Input or Output:** inputs to or outputs from the microcontroller (i.e. temp. sensor, LCD screen)
- **Process:** Calculations on inputs (i.e. Fifteen = 10 + 5)
- **Decision:** One example is an "if statement" – decides if a statement is true, usually followed by a process or output if the condition is true.

Example:

```
if (armpits = stinky)
{
 take shower now;
}
```

armpits is a variable!



Input / Output

Referenced from the perspective of the microcontroller (Arduino).

Input is a signal / information going into the Arduino.

Output is any signal exiting the Arduino.

What are some examples of Inputs & Outputs?

Examples: Buttons Switches, Light Sensors, Flex Sensors, Humidity Sensors, Temperature Sensors…	Examples: LCD screen, LEDs, motors, a buzzer, relay

Program Flow

Just as a flow chart flows in the direction of the arrows, programs have a "flow."

The computer or microcontroller complete the steps programmed or coded, line by line.

* For instance to make the LCD screen say: "Temp   F" we would write:

```
lcd.print("Temp      F  ");  \\The quotes indicate a string of text
```

* Before going to the next line, the microcontroller prints "Temp F" on the screen.
* On the next lines we can tell it to go to a specific location on the LCD screen (6,0) and output the converted temperature value.

```
lcd.setCursor(6, 0);
lcd.print(tempF);
```

Program Flow Continued

Since the computer will do exactly what you tell it to, as the programmer, it's important not to skip steps.

Imagine you were telling someone on the phone how to make a BLT, but you didn't tell them to cut the tomato or lettuce first and they took your directions literally. What would happen?

Order of Code

The computer or microcontroller will process each command one at a time and in the order that you tell it to.

But, each step happens so fast that we couldn’t detect this.

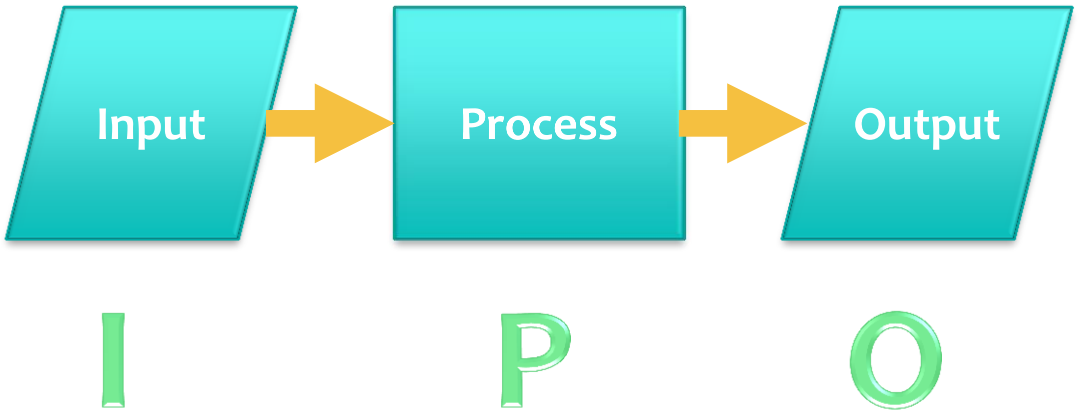
For instance, if we wanted to print, “Temp   F” on the LCD screen and have it stay there when we insert the temperature output between the Temp and F, we could do that.

First, we tell the computer to print “Temp   F” with spaces for the temperature – then we tell the computer to go to the position on the LCD with space and print the temperature reading.

Basic Programming Flow Model

The following model is the most basic programming method; it helps ensure that steps are not missed.

* Declare or Initialize Variables, then
* I-P-O:

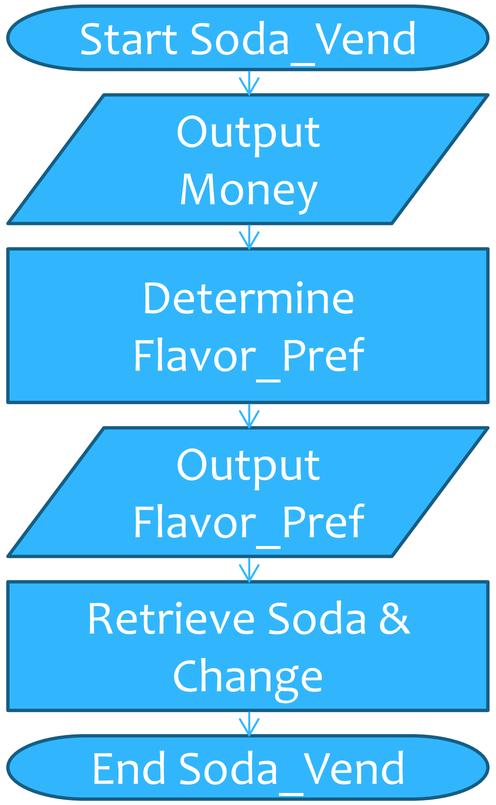


* Sometimes, the flow repeats in a "feedback loop" where the Output continues to feed the Input.

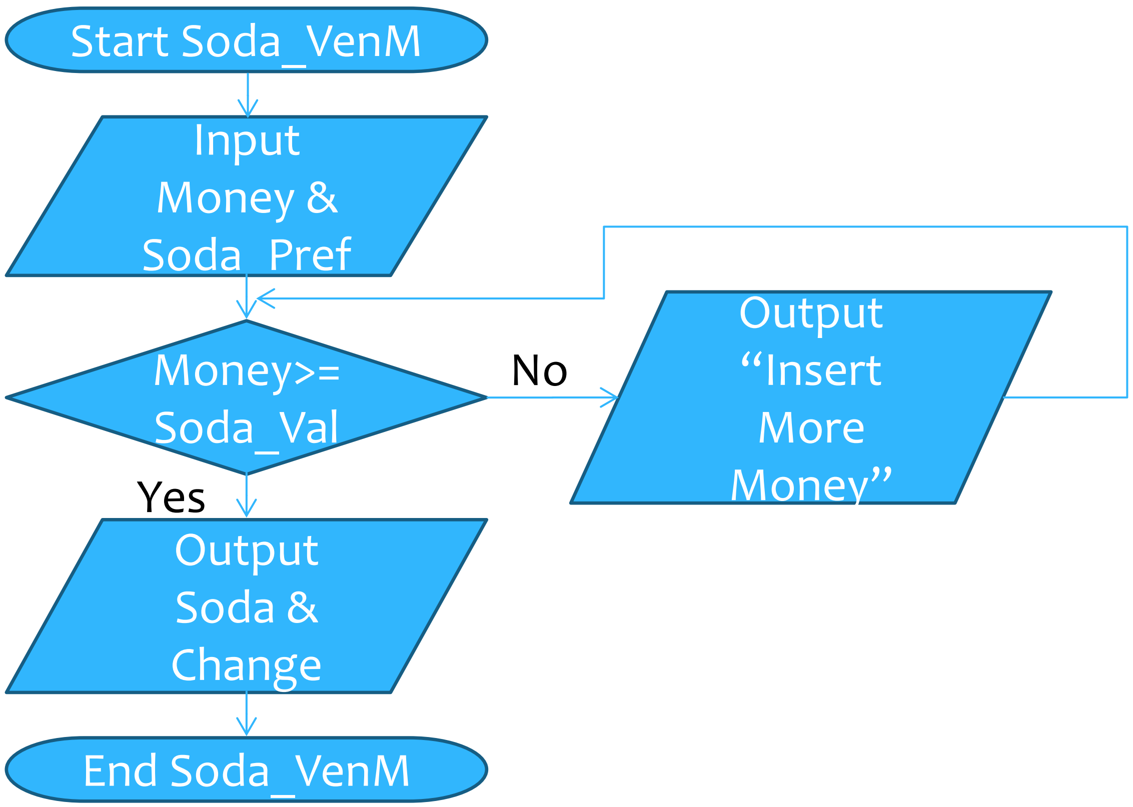
Flow Charting - Tying it Together

- Class participation exercise - create a flowchart for purchasing a soda from a vending machine.
  - From your perspective
  - From the vending machine's perspective

Flowchart for Soda Vending (Your Perspective)



Flowchart for Soda Vending (Machine's Perspective)



“Psuedocode”

After a flowchart, many beginner programmers find it helpful to write pseudocode.

Psuedo: meaning informal, not proper.

In psudocode, you write <u>generally</u> what you will write in your code to bridge between your flowchart / algorithm & your code.

Additional Programming Concepts

Variables are Categorized by Data Type so the Microcontroller “Knows” How to Handle Them

Whenever you declare or define a variable you must tell the microcontroller how to handle the data you assign to the variable. Here are common data types:

* **Integer (int)** – whole numbers, just like integers in you learned about in your math class
* **Float (float)** – numbers with high accuracy including decimal places
* Character (char) – letters
* **String (str)** – a string of letters, usually defined using quotation marks

When performing operations (for the Arduino) on values that have the float data type for an Arduino, you must use decimals to keep it as a “float” data type. For example:

Temp_C = (TempRead – 0.5) / 1024   -> Temp_C will be an integer

Temp_C = (TempRead – 0.5) / 1024**.0** -> Temp_C will remain float

Analog / Digital

When dealing with input (sensors) and output (LCD) we need to know whether they are analog or digital.

- **Digital** signals are either ON or OFF or HIGH or LOW. This type of input can also be called – discrete.

- **Analog** signals are anything that can be a full range of values.

Can you think of some examples for each?

What type would the light sensor & temperature sensor be considered?

Sensor Output & Conversion

When you look into reverse engineering something it’s important to know information about the “hardware” – in this case sensors.

The temperature sensor requires two steps to convert the output voltage into a temperature.

The photocell / light sensor requires knowledge of the maximum output as well.

Coding / Programming the Microcontroller (Arduino)

What is an Integrated Development Environment (IDE)?

It's the program (*environment*) on the computer that allows you to *develop*, compile (check) and upload (*integrate*) your program or sketch to the hardware (Arduino).

Syntax

Syntax is the way that words are put together in a language. To program, we need to be sure to follow the "grammar" rules for the Arduino "language."

Here are some general syntax rules for the Arduino.

- Statements must end in a semicolon;
  - The semicolon let's the microprocessor know where the end of the statement is.
  - Things that are grouped together are grouped with "curly brackets": { } (examples to follow)
  - Multiplication is performed with the asterisk: *
  - Division is performed with the forward slash: /

Comments

Comments are an organizational tool that programmers use to make "code" legible & understandable.

It's also a way to document what your program does and modifications you've made to it.

These lines are ignored by the microcontroller and can say anything. To let the microcontroller "know" to ignore these phrases, lines or sentences, specific syntax is used to make a comment.

For the Arduino, the syntax is \\ anything following the \\ **will turn grey in the IDE** indicating that the comment will not be read by the microcontroller.

Comments

What information do you think you might include in the comments of your first program?

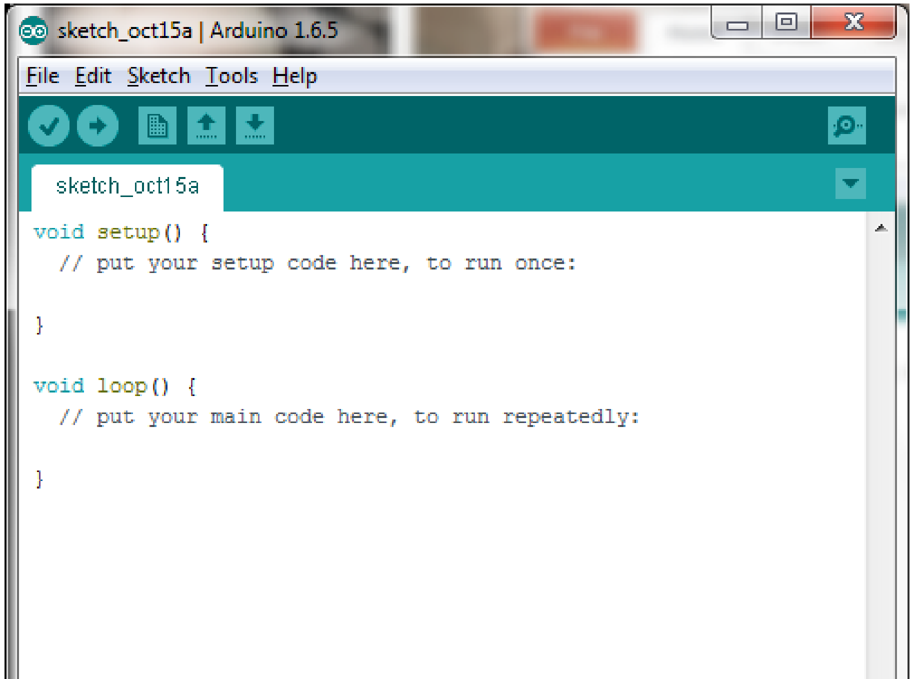
How could you use comments to organize your first program?

Setup & Loop

Arduino programs require two things at minimum to run.

1

2



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

}

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

}
```

Notice the comments are grey.

Syntax Cues

The IDE will provide visual cues when the syntax is typed correctly.

Structure statements turn GREEN:

- setup()
- loop()

Constants & data types performed on Variables turn TEAL – and built-in functions turn ORANGE:

- int tempReading = analogRead(tempPin);
- float lightReading = analogRead (lightPin);

Example Code

```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and
ground.

 This example code is in the public domain.
 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

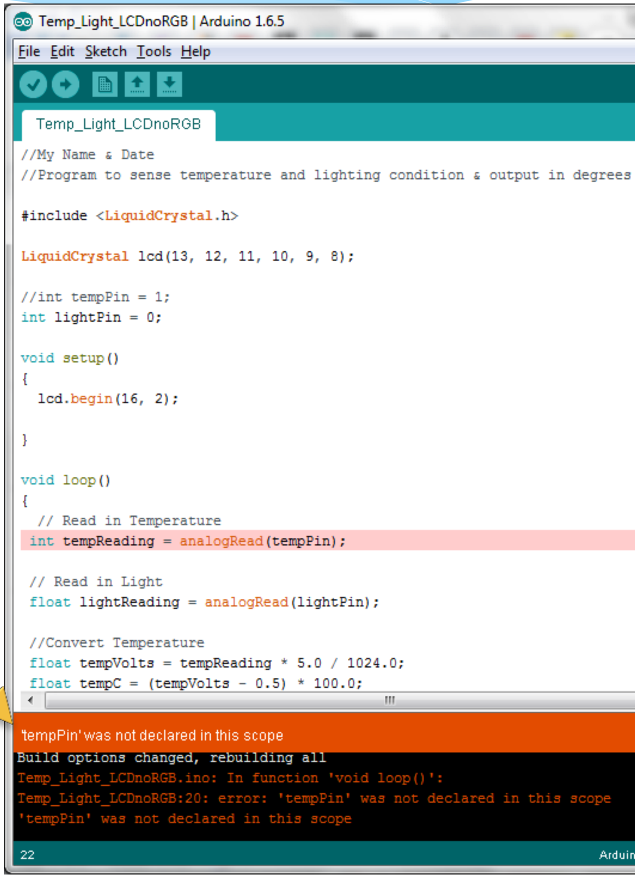
Debugging

Just like any work you do, it's important to debug or check your work.

The Arduino IDE can "verify" or check the program you wrote prior to uploading it to the board. **(Be sure to save your "sketch" first.)**

If you have errors (or bugs) in your program, they will be listed at the bottom of the screen.

The process of finding the "bugs" and removing them is called – debugging.


Temp_Light_LCDnoRGB | Arduino 1.6.5
File Edit Sketch Tools Help
Temp_Light_LCDnoRGB

```
//My Name & Date
//Program to sense temperature and lighting condition & output in degrees

#include <LiquidCrystal.h>

LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

//int tempPin = 1;
int lightPin = 0;

void setup()
{
  lcd.begin(16, 2);

}

void loop()
{
  // Read in Temperature
 int tempReading = analogRead(tempPin);

 // Read in Light
 float lightReading = analogRead(lightPin);

 //Convert Temperature
 float tempVolts = tempReading * 5.0 / 1024.0;
 float tempC = (tempVolts - 0.5) * 100.0;
```

```
'tempPin' was not declared in this scope
Build options changed, rebuilding all
Temp_Light_LCDnoRGB.ino: In function 'void loop()':
Temp_Light_LCDnoRGB:20: error: 'tempPin' was not declared in this scope
'tempPin' was not declared in this scope
```

Debugging

From the information listed on the bottom of your screen in the Arduino IDE, you should be able to get an idea of WHERE in your code you have a bug.

```
YourProgramName.ino: In function 'void loop()':
```

In this example, the error is in the loop section. Under that it will tell you what the error is.

```
YourProgramName.ino:26: error: 'tempPin' was not declared in this scope
```

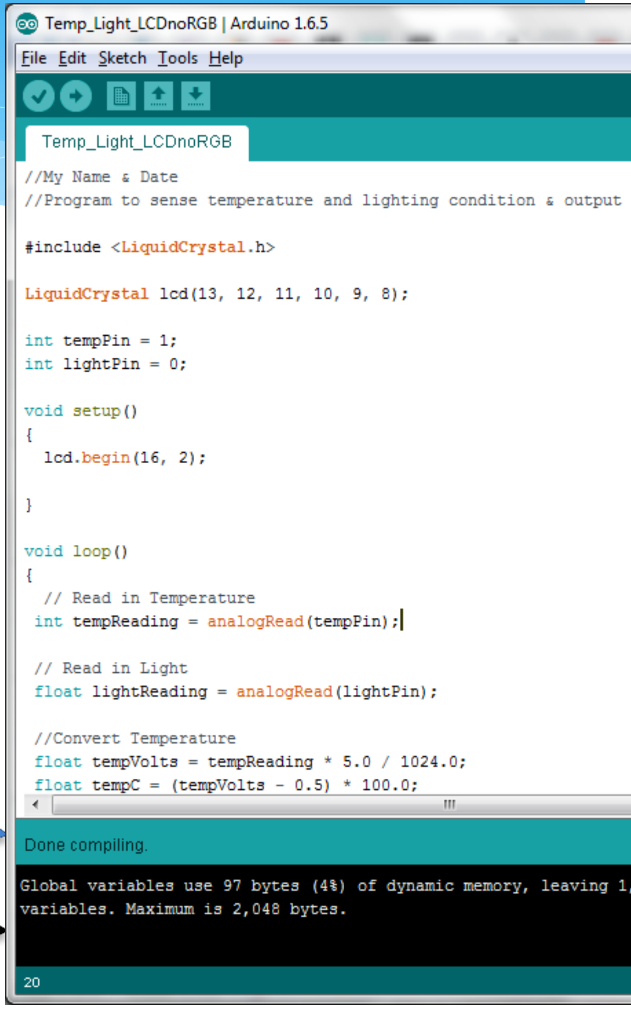
In this example, the variable tempPin wasn’t initialized properly.

Debugging

Once you have fixed the code for the error found, you can "verify" your "sketch" again.

When all errors are gone, the bar under the code will say done compiling and will stay teal.

Also the text at the bottom will be white, not orange.


Temp_Light_LCDnoRGB | Arduino 1.6.5
File Edit Sketch Tools Help
Temp_Light_LCDnoRGB

```
//My Name & Date
//Program to sense temperature and lighting condition & output

#include <LiquidCrystal.h>

LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

int tempPin = 1;
int lightPin = 0;

void setup()
{
  lcd.begin(16, 2);

}

void loop()
{
  // Read in Temperature
 int tempReading = analogRead(tempPin);

 // Read in Light
 float lightReading = analogRead(lightPin);

 //Convert Temperature
 float tempVolts = tempReading * 5.0 / 1024.0;
 float tempC = (tempVolts - 0.5) * 100.0;
```

```
Done compiling.

Global variables use 97 bytes (4%) of dynamic memory, leaving 1,
variables. Maximum is 2,048 bytes.

20
```

Programming an LCD Screen

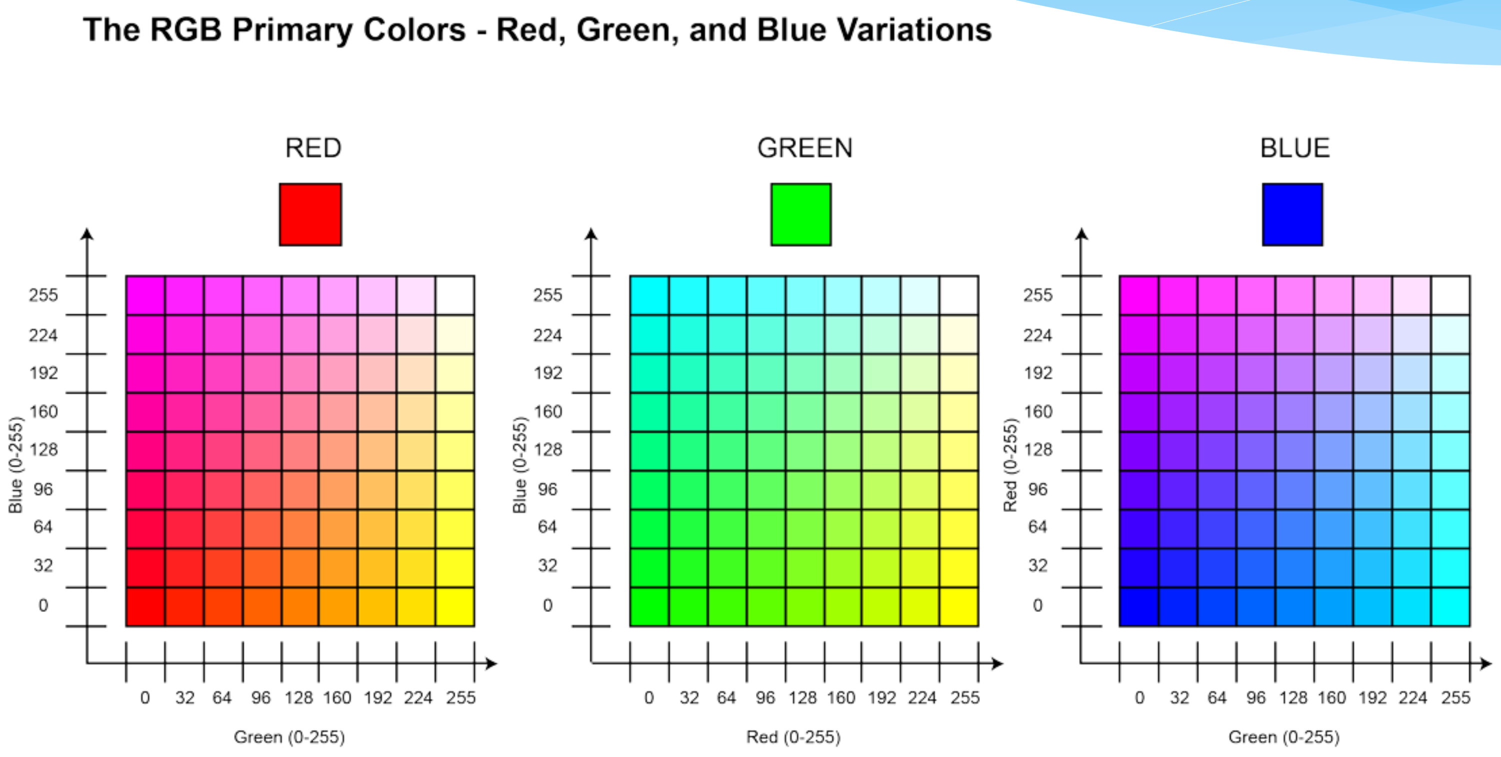
Let's watch a YouTube video tutorial on programming an LCD screen

http://youtu.be/TirVG6tmTnQ

Now, we are ready to develop a program!

* Let’s start developing algorithms and programming a simulated Weather Information Device.

“Extra Credit” Help on RGB Coordinates



HSB(0,100,100)
RGB (255,0,0)
CMYK(0,99.4,100,0)
HEX#FF0000

HSB(120,100,100)
RGB (0,255,0)
CMYK(62.76,0,100,0)
HEX#00FF00

HSB(240,100,100)
RGB (0,0,255)
CMYK(88.37,76.92,0,0)
HEX#0000FF