Module 1: Crash Prevention
Lesson 3: Weather Information systems
Programming Activity Using Arduino – Teacher Resource
Grade 9 - 12

Time Required: 3 – 60 minute sessions or 3 hours

Required Materials

- Computers (enough for groups of no more than three),
- Access to the internet for flowcharting using mxGraph: https://www.draw.io or stand-alone software for flowcharting if you do not have internet access (i.e. Microsoft Visio, Microsoft PowerPoint, etc.),
- Weather information devices (powered by Arduinos) set up with temperature and photocell sensors that output to LCD screens
    - Classroom kits available for purchase from NanoSonic, Inc.
    - For DIY instructions, see: https://learn.adafruit.com/adafruit-arduino-lesson-12-lcd-displays-part-2/overview & https://learn.adafruit.com/character-lcds/rgb-backlit-lcds
- 9V batteries or wall outlets and the corresponding wire to connect to the Arduino,
- Introductory lesson to programming or slides provided by NanoSonic, Inc. with the Arduino kits

Optional Materials

- Flashlight,
- Heat source (e.g. hot plate or hair dryer – use caution, as this gets very hot)
- Cold source (e.g. 2 ice packs to sandwich the sensor and get the temperature down to freezing)

Preparation

- Label the Weather Information Devices so that students can identify their Arduino device during later sections.
- Install the latest version of the Integrated Development Environment (IDE) or compiler software from: http://arduino.cc/en/Main/Software on each computer.

  Hints:

    - **For PC Users**
        - Let the installer copy and move the files to the appropriate locations, or
        - Create a folder under C:\Program Files (x86) called Arduino. Move the entire Arduino program folder here.
    - **For Mac Users**
        - Move the Arduino executable to the dock for ease of access.
        - Resist the temptation to run these from your desktop.

Description of the Weather Information Arduino Device:

The pre-programmed and easily reprogrammable devices employ a temperature sensor and a photocell (light sensor) and displays the temperature in degrees Fahrenheit and light as a percentage on the liquid crystal display (LCD) screen. The photocell is nominally facing the "back of the device." The LCD is programmed to be backlit in green at room temperature, red at temperatures above 90°F and cold at temperatures below 32°F.

Prior to starting the activity: It may be useful to highlight the example syntax/code on the front page of the student activity sheet.

Part 1: Analyze the Weather Information Device –

First let's explore what the Weather Information Device does:

1. Break into groups of 3 or less.
2. Obtain the following:

Materials:	
Weather Information Device	**Heat Source**
Power source (battery and wire)	**Cold Source**
Computer	
Flash light (*optional*)	

3. Power the Weather Information Device by plugging the power source into the bottom of the device. (Batteries and wires to connect to the Arduino are provided with the teaching kits. Wall plugs are available for additional purchase.)
4. What happens to the light output when you turn out the lights? (The number decreases.)
5. What happens when you shine light on the device? (The number increases.)
6. Where do you think the photocell (light sensor) is located? (Facing the back of the device.)
7. What happens when you expose the device to a hot temperature? (The temperature reading increases, the screen turns red.)
8. What about cold? (The temperature reading decreases, the screen turns blue.)

Part 2: Algorithm Development with Pseudo Code

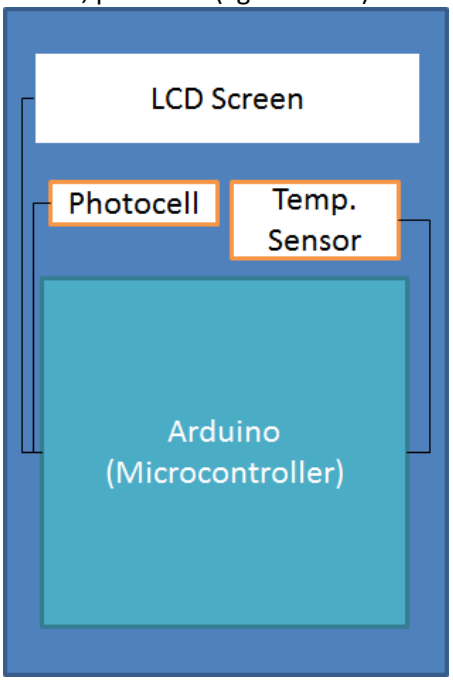
Now, let's say you're a transportation engineer and you want to program a Weather Information Device on your own, we'll start by developing your own algorithm for the device.

1. Short description:
    a. Write a brief description of the device (i.e. what will the device do?). (Measure the temperature and lighting conditions in the room.)
    b. From the following list, identify the inputs and outputs for the device (see diagram at the end of this document):
        i. Photocell (light sensor) (Input.)
        ii. LCD screen (Output.)
        iii. Temperature sensor (Input.)

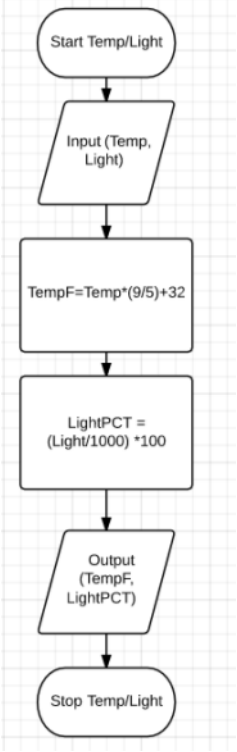
iv. Arduino (Neither.)

c. Are the inputs analog or digital? (Analog since they are not discrete (i.e. HIGH/LOW).)
d. Are there any existing systems the device will integrate? (No, but in the real world there likely would be.)
e. Include some brief descriptive notes about the device. (The device is rectangular and about the size of two decks of cards, it has an LCD screen and power and USB ports on the bottom.)

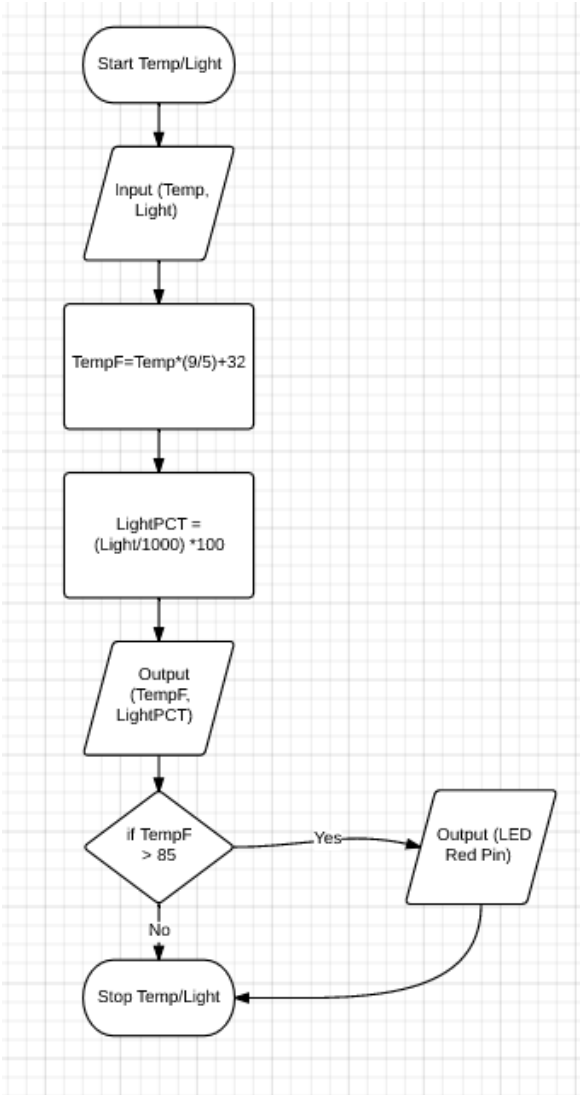
2. Draw a block diagram of the device including power, the microcontroller (Arduino), temperature sensor, photocell (light sensor) and LCD screen.



3. Create a logic flow chart for a program to read the temperature and lighting condition where the temperature and lighting condition is displayed on an LCD screen in degrees F and a percentage of the possible lighting conditions. Assume that the temperature will be read into the microcontroller in degrees C and need to be converted to degrees F. ***Remember:*** Declare, then, Input, Process, Output (IPO).



4. Edit the logic chart to include an “if statement” to select a colored LED or the color of the LCD screen to match with the temperature conditions (e.g. if the temperature is HOT, the LCD screen turns red).



a. Where should the "if statement" go? (After the calculation before the stop.)
b. Decide what temperature condition you are going to look for.
    i. For instance if the temperature is above a certain point that you define, the temperature is HOT, or below a certain point that you define the temperature is COLD.
c. There may be a number of places it could be included, can you identify where the "if statement" should NOT go? (Cannot go before input or calculation of the temperature.)

5. Using the logic flow chart identify which sections will go in the loop section of the Arduino code. (Which parts will we need the microcontroller to process more than once?) ***Hint:*** Does the weather information system device just read and output the temperature and light conditions once? (All sections in the flowchart will be repeated so that the output will update constantly.)

6. From the logic flow chart and the Arduino code examples above, write the pseudo code for the Weather Information Device. Since we assumed that the temperature was read in in °C, but it is actually read in in volts, we need to add lines and variables. See the front page for the formulas.

Be sure to include the two necessary pieces of Arduino code.

a.
```
int tempReading;
float tempVolts;
float TempC;
float TempF;
float Light;
float LightPCT;  (Variables can have any name the student chooses.)

void setup()
{

}
```

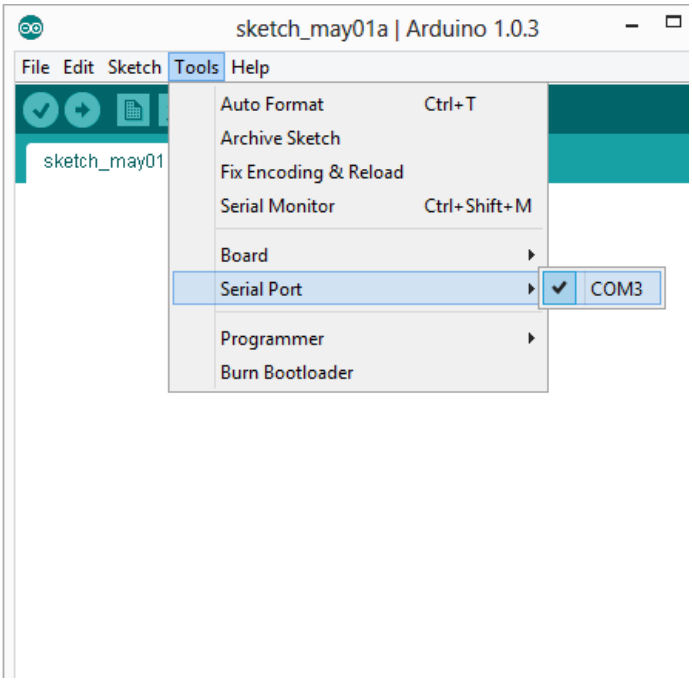
b.
```
void loop()
{
tempReading = analogRead port 1;
tempVolts = tempReading * 5.0 / 1024.0;
tempC = (tempVolts - 0.5) * 100.0;
tempF = tempC * 9.0 / 5.0 + 32.0;

Light = analogRead port 0;
LightPCT = (Light/1000)*100;

lcd.print("Temp   F");
lcd cursor 6,0; // this is a location – the 6th character on the first row, Arduino counts the
first row as the zeroth row (this procedure is outlined on slide 13.  The students will get
help with this later on in the activity, so for now, it's just important that their
pseudocode says output the temperature.
lcd.print(TempF);

lcd cursor 0,1;
lcd.print("Light    %");
lcd cursor 6,1;
lcd.print(LightPCT);
}
```

Helpful pointers:

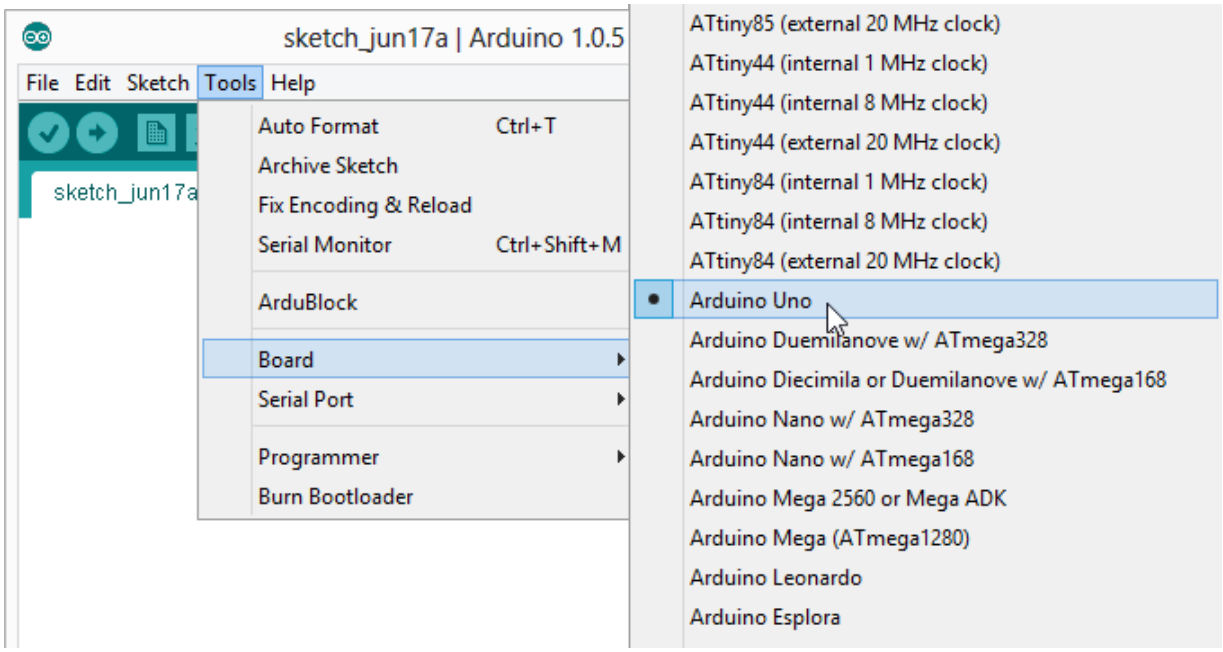
- How many variables are needed? Make sure you remember to declare them / give them a data type. The data type for the temperature sensor should be integer; it is a characteristic of the sensor. The light sensor is different – its output contains a decimal.
- Psuedocode does not need to have correct syntax, it is just for you to follow as you program.

Part 3: Programming

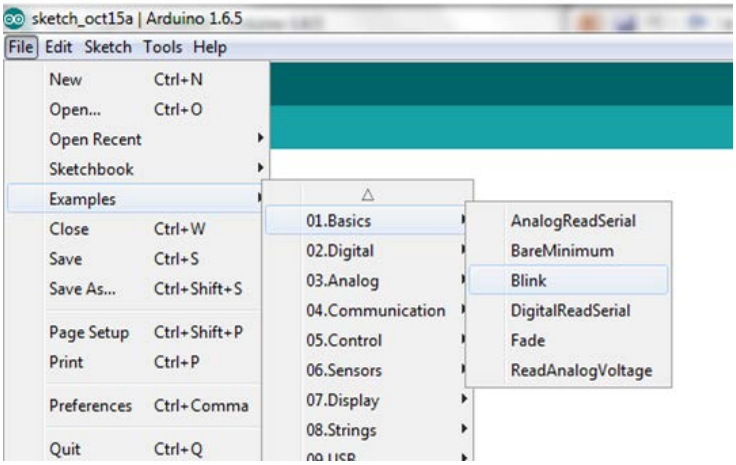
1. Connect Computer to USB port on the Weather Information device.
2. Open the Arduino program:
3. Find the Arduino connection on the computer: Tools → Serial Port
   Your computer communicates to the Arduino microcontroller via a serial port → through a USB-Serial adapter. Check to make sure that the drivers are properly installed.



4. Double-check that the proper board is selected under the Tools→Board menu. (The Weather Information Device has an Arduino Uno inside of it.)



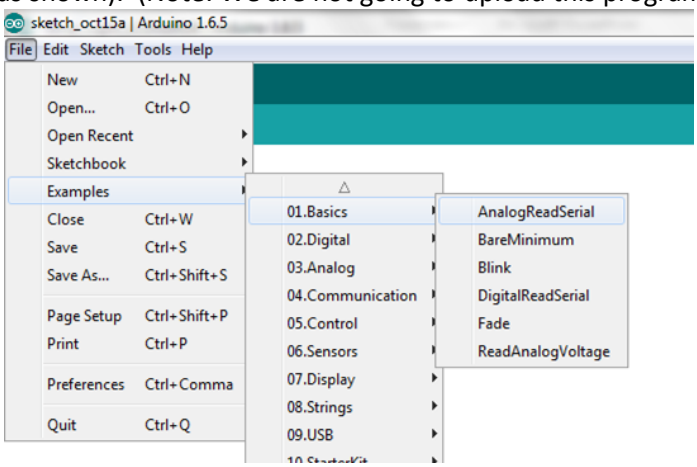
5. First, we'll start by practicing verifying and uploading code (an example included with the IDE) to the Arduino.
    a. Open the example "Blink" by clicking on the following: File -> Examples -> Basic -> Blink (as shown).



    b. Examine the code, read the comments – some might seem complicated but that's ok, look at the statements/code that are listed in the loop section. This is an example of a well-written code. The comments at the top tell you what the program does.
    c. Then, click on the verify button. (Check mark at the top.)
        i. You shouldn't get any errors unless you typed something else in the window.
    d. Click upload, it looks like a right pointing arrow (next to the verify button).
    e. This program should make an LED on the Arduino board blink – look in the Arduino

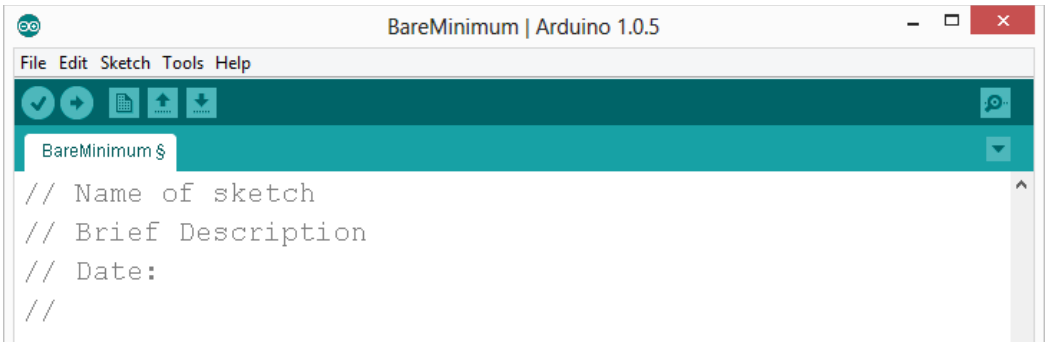
powered device and see if you can see the blinking light. (It will be on the teal board inside the Weather Information Device.)

6. Next, we'll open an example code that may help you with your syntax and code. Open the "AnalogReadSerial" example by clicking on the following: File -> Examples -> Basic -> AnalogReadSerial (as shown). (Note: We are not going to upload this program to the Arduino.)



   a. In the line: **int** sensorValue = **analogRead**(A0);
      i. Identify the variable name: sensorValue
      ii. Identify what data type the variable is assigned: integer
      iii. Identify the Arduino function used to read in the sensor value: analogRead
         1. Remember: analog indicates that this is data that has a value, it is not just on or off.
      iv. Identify the pin on the Arduino board that the sensor is connected to in this example. A0 or 0 both are accepted.
      v. Note that the data type of the variable is declared in the same line of code as a value is assigned to the variable, this reduces the lines of code needed.
      vi. Also, notice that the variable data type is declared within the loop() section, this is because the program will read in the sensor value more than once and if we do not define the data type each time we may get errors.
   b. Now that you dissected that line of code, you are ready to write your own code for the weather information device.

7. Open a new "sketch" in the IDE. A sketch is just what the Arduino IDE calls a program or code. Just like any work you do – put your name and date at the top in a comment along with a brief description:



8. Enter the two required functions / methods / routines as shown above and provided below (they may automatically be included in new sketches depending on the Arduino IDE version):

```
void setup()
{
        // runs once <- this is a comment
}
void loop()
{
        // repeats
}
```

There are syntax examples on the front page and more can be found on the following web page: https://www.arduino.cc/en/Reference/HomePage.

9. **Initialize (first step in programming flow):** Because we will be outputting to the LCD screen on the Arduino, we will need to include the LCD library, tell the Arduino which pins the LCD is connected on and initialize the LCD screen. To do that enter the following above the setup step:

```
#include <LiquidCrystal.h>

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

Also, we will need to include the initialize statement for the LCD screen in the **setup**() routine since it only needs to run once. Enter the following statement in your **setup**() routine:

```
lcd.begin(16, 2);
```

This code lets the Arduino know that the LCD has 16 characters in length and 2 rows.

10. We've initialized the LCD, the next step is to **Input (second step in programming flow):** Determine whether your sensor read statements belong in the setup() routine or the loop() routine. Place them there. (loop() See example code in the completed basic program section.)
    a. What data type should we assign to the sensor values; integer or float? Look back at the helpful pointers in section 2) (Temperature sensor: int initially then float during conversions, photo sensor: float)
        i. Just like the analog example:
            1. You can declare a data type for your sensor values at the same time as

reading them in,
2. We will use the built in **analogRead** function.
3. See the front page & the analog example for help

ii. Remember: the photocell (light sensor) is on the Arduino pin 0, the temperature sensor is on the Arduino pin 1. These are analog pins on the Arduino.

11. What calculations do you need to perform on your inputs? (See example code in the completed basic program section.)
    a. Before converting the temperature from °C to °F we will have to convert it from a reading to a voltage using the following: **tempVolts =** [your temperature variable name] *** 5.0 / 1024.0;** (*Note: if we do not use the decimals, the data type will change to integer.*)
    b. To convert from the voltage to °C we will also need to perform a conversion using formula in the formulas section. (*Note: if we do not use the decimals, the data type will change to integer.*)
    c. Now, you can convert the temperature from °C to °F.
    d. Remember to calculate the percentage of light. (***The maximum light value is 1000.***)

12. Determine whether your LCD output statements belong in the **setup**() routine or the **loop**() routine. Place them there. (loop() See example code in the completed basic program section and slide 13.)
    a. The first page can provide some help with syntax – here you will use a built in function to print your variable output.
    b. So that users can understand what the output means, it is important to give the output some context, but printing string or just plain text (not a variable value) has to be done separately because text has a data type called "string":
        i. Remember to include quotation marks around any text that is a string. This is like declaring the data type; you're letting the computer know not to look for variables, but just to print what you actually typed.
        ii. To do this you can first print the string/text indicating what the output means and then you can specify a specific location on the LCD screen to output your variable's value.
        iii. An easy way to change the location of the LCD output is by using the command: lcd.setCursor(CHAR, ROW); where CHAR is the number of the character and ROW is the number of the row. Since the Arduino starts counting at 0, when 1 is used for the ROW in the command, you are actually referencing the second row.
        iv. Remember to limit your characters to 16 across the line or you'll "overflow" the LCD output.
        v. An example:
        ```
        lcd.print("Humidity        %");
        lcd.setCursor(6, 0);
        lcd.print(HumidPCT);
        ```

13. So that we can read the output on the LCD a delay is typically used, the delay is given to the Arduino in milliseconds. Setting a delay will delay how often the sensors are read, calculated and the results are output. A delay of at least 1 second will make the output readable, but you may want to experiment with this. (Convert 1 second to milliseconds – see the front page for syntax and help.) (delay(1000);)

14. When complete, verify your code using the Arduino IDE (click the check mark).
    a. If your code didn't produce any errors on the bottom, upload the code to the Arduino (click the right pointing arrow) and see if it works.
    b. If it produced errors look for missing parenthesis and semicolons. Remember: the orange text at the bottom will help you find where the error(s) are located.
    c. If it does not work or produces an error, hypothesize why it is not working, double check to see if you missed any important steps.

Part 4: Extra Credit

If you were successful at having the device display the temperature and percent lighting, see if you can add code to change the color of the screen for a given condition (i.e. turn red when hot). To do this, you will use the modified flow chart you made in Part 2.

The RED LED on the LCD is wired to port or pin 6, the GREEN LED is on port 5 and the BLUE LED is wired to port 3. The LED's are what "backlight" the LCD screen. Outputting a 0 value to the LED will turn the LED on, conversely outputting 255 to an LED will keep the LED off. These outputs are analog outputs that require the analogWrite (PIN, VALUE); command. The colors are defined in the red, green, blue color space so you could turn the LCD a fancier color by giving the LEDs values between 0 and 255. (You can Google coordinates in RGB color space if you have time.)

In programming if statements allow you to tell the microcontroller or computer to do something if a certain condition is true. Hence, they are really if, then do statements and the syntax varies from language to language. But, in Arduino programming, if statements do not use semicolons and look like this:

```
if (CONDITION)
{
\\ insert what you want to happen here if the condition is true (i.e. turn the RED LED on)
}
```

Can you program the LCD to change color? How would you program the LCD to turn purple? Which LEDs would you turn on?
(analogWrite(6,0); analogWrite(3,0);)

Syntax help for if statements can be found here: https://www.arduino.cc/en/Reference/If

Conclusion

Based on the results, form a conclusion as to whether your hypothesis was supported or rejected and explain.

Analysis questions

1. List three ways your code could be improved.
    a. Are there ways to reduce the number of lines (make it more efficient)? (Variables can be declared at the same time that they are first being used, the first two calculations for the voltage and the temperature conversion could be combined, but that would reduce the ability to debug potential problems.)
    b. Are there ways you could make your code more readable/user friendly? (Add more comments break up the sections differently, the idea here is that students start to realize

while the syntax is strict there are many ways of programming and getting the same result.)

c. What are some other ways your code could be improved? (An open ended thought provoking question, to again help students start to realize while the syntax is strict there are many ways of programming and getting the same result.)

2. Name one thing that you learned about programming that you did not previously know.
3. What are some weather-related warning messages that infrastructure-to-vehicle systems could send to a driver?
   a. Name at least one temperature specific warning, (Ice ahead, road buckling possible ahead.)
   b. Name at least one light specific warning. (It's dark, overcast, etc. turn your headlights on.)
4. How can infrastructure-to-vehicle communication help to prevent weather-related crashes? (Preparing drivers so that they have enough time to react or get off of the road.)
5. What else can sensors detect that might provide useful information to drivers?
   - Name one thing that future iterations of connected vehicles might sense to:
     1. Increase safety (Warn drivers of impending storms or weather that can adversely affect travel.)
     2. Increase awareness (Warnings that warn of snow drifts, winds, rainfall (flooding), etc.)

Completed program with RGB code

This is example code provided so that you, the instructor can help students out if they are stuck along the way.

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

#include <LiquidCrystal.h>
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

int lightPin = 0;
int tempPin = 1;

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;
```

```
 float tempF = tempC * 9.0 / 5.0 + 32.0;

 // Convert Light
 float lightPCT = (lightReading/1000.0)*100;

 // Output Temperature
  lcd.print("Temp       F  ");
  lcd.setCursor(6, 0);
  lcd.print(tempF);

  // Output Light on second row
  lcd.setCursor(0, 1);
  lcd.print("Light       %");
  lcd.setCursor(7, 1);
  lcd.print(lightPCT);

  // Set frequency that the LCD updates
  delay(1500);
}
```

Completed program with RGB code – to upload on Arduinos following the lab:

This is code is provided so that you, the instructor, can reset the Arduino powered Weather Information Systems following a programming activity with your students so they'll be ready for the first part of the activity for the next group.

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

#include <LiquidCrystal.h>
#define REDLITE 6
#define GREENLITE 5
#define BLUELITE 3

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

int tempPin = 1;
int lightPin = 0;
int R;
int G;
int B;

void setup()
{
  lcd.begin(16, 2);
  pinMode(REDLITE, OUTPUT);
  pinMode(GREENLITE, OUTPUT);
```

```
  pinMode(BLUELITE, OUTPUT);
}

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;
 float tempF = tempC * 9.0 / 5.0 + 32.0;

 // Convert Light
 float lightPCT = (lightReading/1000.0)*100;

 // Output Temperature
  lcd.print("Temp        F  ");
  lcd.setCursor(6, 0);
  lcd.print(tempF);

  // Output Light on second row
  lcd.setCursor(0, 1);
  lcd.print("Light        %");
  lcd.setCursor(7, 1);
  lcd.print(lightPCT);

  // Select Backlight Color Based on Temperature
  if (tempF >= 90)
  {
  R = 0; //Backlight is red if TempF is HOT
  G = 255;
  B = 255;
  }
  else if (tempF <= 32)
  {
  R = 255;
  G = 255;
  B = 0;  //Backlight is blue if TempF is COLD
  }
  else
  {
  R = 255;
  G = 0;  // Backlight is green if the TempF isn't defined as HOT or COLD
  B = 255;
```

```
  }

  analogWrite(REDLITE, R);
  analogWrite(GREENLITE, G);
  analogWrite(BLUELITE, B);

  // Set frequency that the LCD updates
  delay(1500);
}
```

Arduino Diagram