Module 1: Crash Prevention
Lesson 3: Weather Information systems
Programming Activity Using Arduino – Teacher Resource
Grade 6 - 8

Time Required

- Weather Information Systems is a 120 minute lesson plan (90 minutes without the FARS activity)
- Arduino lesson plan is two – 60 minute sessions or 2 hours, by itself

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.),
- Arduino's 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

- 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 device employs 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. See the diagram at the end of this document.

*This lesson works well setting up the activities as stations.

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

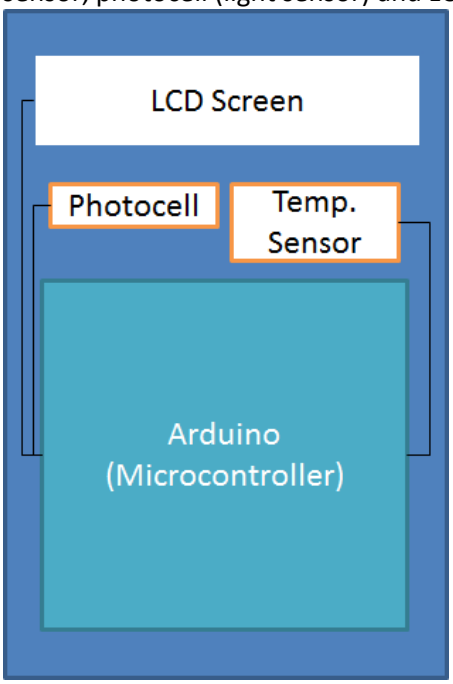
Now, let's say you're a transportation engineer and you want to program a Weather Information Device on your own, we'll start by developing your own algorithm for the device.

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:
        i. Photocell (light sensor) (Input.)
        ii. LCD screen (Output.)
        iii. Temperature sensor (Input.)
        iv. Arduino (Neither.)
    c. Are there any existing systems the device will integrate? (No, but in the real world there

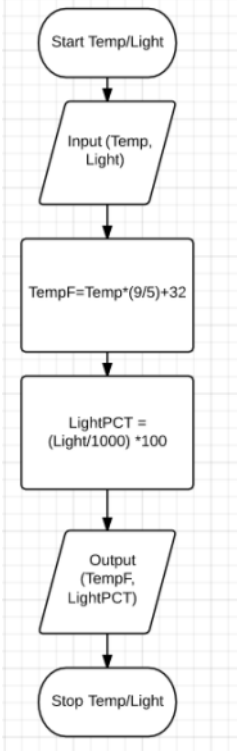
likely would be.)

d. 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.)

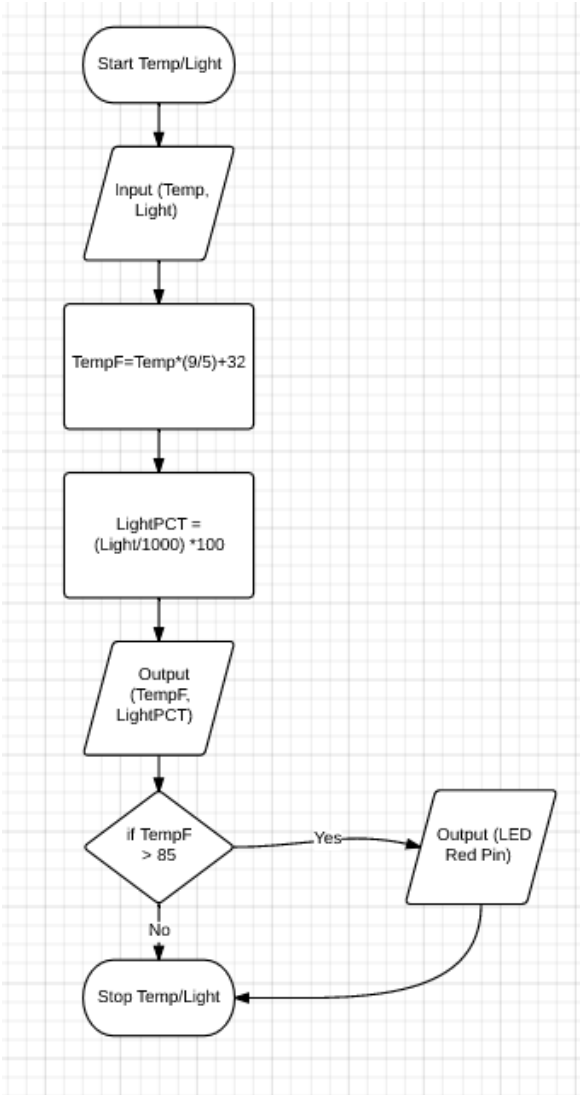
2. Draw a block diagram of the device including power, the microcontroller (Arduino), temperature sensor, photocell (light sensor) and LCD screen.



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. (The maximum reading for the photocell is 1000.) ***Remember:*** Declare, then, Input, Process, Output (IPO).
*It is recommended that the first flow chart activity be a group activity (reference the slides)



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 but, 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. (Variables can have any name the student chooses, they should be declared at the top. For this exercise the students can just list the variable names at the top, typically the variables are given a data type at initialization, but the data types aren't covered in the MS lesson. It's important that they know to declare or initialize (give a value to) variables and this was covered on slide 16.)

```
TempRead
TempReadV
TempC
Light
TempF
LightPCT

void setup()
{
}
```

b.
```
void loop()
{
TempRead =  Read port 1;
Light = Read port 0;
TempReadV = TempRead * (5.0 /1024.0);
TempC = (TempReadV – 0.5) * 100;
TempF = Temp C *(9.0/5.0) +32.0;
LightPCT = (Light/1000)*100;

print("Temp     F");  //There are a lot of spaces in there so that the temperature can be printed in between the "Temp" and the "F" – if we were programming they would need to be separated due to their different data types, but for purposes of pseudocode, the students can simply say print(Temp: TempF F) or something like that.

go to 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)

print(TempF);

go to 0,1;  // this is a location – the 0th character on the second row,
print("Light     %");
go to 6,1;  // this is a location – the 6th character on the second row,
print(LightPCT);
}
```

Helpful pointer:

- Psuedocode does not need to have correct syntax, it is just for you to follow as you program.

Part 3: Extra Credit

If you finished Part 2 and want a bigger challenge, see if you can add to your pseudocode 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 develop pseudocode to program the LCD to change color? How would you program the LCD to turn purple? (LED on pin 6, turn on/0; LED on pin 3, turn on/0)

Conclusion

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

1. List three ways your pseudocode 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. There may be other creative ways the students come up with.)
   b. Are there ways you could make your pseudocode more readable/user friendly? (Add more comments break up the sections differently, the idea here is that students start to realize while the syntax (the grammar and capitalization used to program) is strict there are many ways of programming and getting the same result.)
   c. What are some other ways your algorithm or pseudocode 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, foggy, etc., turn 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. There may be other clever responses here as well.)
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, rain (flooding), etc.)

If students have extra time after completing this activity – they can visit the Arduino website and look at the syntax reference for programming or other projects they could make with an Arduino:

Syntax reference: https://www.arduino.cc/en/Reference/HomePage.

Arduino homepage: https://www.arduino.cc/

Completed program with RGB code

This code is included in the event that one of the Weather Information Devices require reprogramming – the program below can be copied and pasted into the Arduino software/IDE, verified and uploaded to the device to reset it.

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

#include <LiquidCrystal.h>

int tempPin = 0;
int lightPin = 1;

LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

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(6, 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