Summary:

Students will familiarize themselves with the Arduino IDE by writing simple programs that introduce the serial monitor, as well as basic coding architecture for Arduino.

Materials:

Each student should have the following materials, equipment, and supplies:

- Laptop with Arduino IDE installed
- USB-AB programming cable
- Arduino Mega microcontroller
- Lecture 6 PowerPoint on Introduction to programming

Procedure:

Example 1 – Write a simple program to display "Hello World!" in the serial monitor

1. Open the Arduino IDE and plug in the USB programming cable from the laptop to the microcontroller. Under tools, select the board (Arduino/Genuino Mega or Mega2560), processor (ATMega2560), and COM port.
2. Write a simple program to establish communication with the serial monitor and display the message "Hello World!"

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.print("Hello World!");
}
```

Figure 1: Code for "Hello World!"

3. Open the serial monitor. Make sure the baud rate in the pop-up window matches what you set in your code. You should see "Hello World!" scroll horizontally across the monitor.
4. Edit you the text in Serial.print command to include a "\r\n" at the end and reupload the code. This should print a new line every time Hello World is displayed.

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.print("Hello World! \r\n");
}
```

Figure 2: Code for "Hello World!" with control characters

5. Uncheck the Autoscroll box and observe that the print becomes stationary.
6. Change the baud rate to 19200 and observe what happens. The monitor will display incorrectly when this occurs.

Example 2 – Write a simple program using arithmetic with conditional statements

1. Write a simple program to declare a global volatile long integer variable (a variable that is expected to change within the code) set equal to 0, and have it increment by two infinitely. Use the serial monitor to display the results. The result should be n + 2 indefinitely.

```
volatile int myVar = 0;           // Global variable that is expected to change

void setup() {
  Serial.begin(9600);
}

void loop() {
  myVar = myVar + 2;              // Increment the value by plus 2
  Serial.print(myVar);            // Displays result in serial monitor
  Serial.print(" \r\n");          // Adds line feed and carriage return
  delay(500);                     // Sets a time to repeat every 5 s
}
```

Figure 3: Code to increment a global variable by 2.

2. Copy and paste your variable directly under the void loop and observe how this changes the output. Instead of incrementing by n + 2, it repeatedly gives a value of 2. As a local variable, the value is being reset back to 0 after each loop; therefore, it will always be the value of 0 + 2.

```
void setup() {
  Serial.begin(9600);
}

void loop() {

  volatile long int myVar = 0;      // Global variable that is expected to change

  myVar = myVar + 2;                // Increment the value by 2
  Serial.print(myVar);              // Displays result in serial monitor
  Serial.print(" \r\n");            // Adds line feed and carriage return
  delay(500);                       // Sets a time to repeat every 5
}
```

Figure 4: Code to increment a local variable by 2.

3. Define the variable as global again (move it back to the top). Change the operation to multiply by 3. Change the value of the global variable to 2. Use the serial monitor to observe the changes. Be sure to let the program run until you see negative numbers. The reason you see negative numbers while using an integer data type is that you have exceeded the memory limit of the microcontroller.

```
volatile long int myVar = 2;            // Global variable that is expected

void setup() {
  Serial.begin(9600);
}

void loop() {

  myVar = myVar * 3;                // Increment the value by plus 2
  Serial.print(myVar);              // Displays result in serial monitor
  Serial.print(" \r\n");            // Adds line feed and carriage return
  delay(500);                       // Sets a time to repeat every 5
}
```

Figure 5: Code to multiply in increments of 3

4. Set the global variable back to 0. Remove the arithmetic function under the main loop. Add a for loop that will count from 0 to 9 and display the results in the serial monitor. Open the serial monitor to observe the results. You should see that it counted from 0 to 9, and then stopped. A for loop reads the condition once, and then stops when it is finished.

```
volatile long int myVar = 0;      // Global variable that is expected to change

void setup() {
  Serial.begin(9600);
}

void loop() {

  for(myVar; myVar < 10; myVar++) {

  Serial.print(myVar);            // Displays result in serial monitor
  Serial.print(" \r\n");          // Adds line feed and carriage return
  delay(500); }                   // Sets a time to repeat every 5
}
```

Figure 6: Code to use a for loop as a counter

5. Change the global variable to 5 and observe the changes. It should start incrementing at 5 instead of 0.
6. Change the for condition to set the local variable equal to 0. Observe how this affects the value. Even though the global variable is set to 5, the counter should start at 0. It should also repeat instead of stopping because the variable is reset at the end of each loop which causes it to meet the condition of being less than 10 again. Having this defined as a local variable also changes the way it reads the condition. You should notice that the counter now scrolls vertically instead of horizontally because it is able to observe the control characters.

```
volatile long int myVar = 5;      // Global variable that is expected to change

void setup() {
  Serial.begin(9600);
}

void loop() {

  for(myVar = 0; myVar < 10; myVar++){

    Serial.print(myVar);              // Displays result in serial monitor
    Serial.print(" \r\n");            // Adds line feed and carriage return
    delay(500);                       // Sets a time to repeat every 5
  }
}
```

Figure 7: Code to use a for loop as a counter

7. Add a while statement to control when the for loop is executed. Define the condition to run when the value of the variable is equal to 5. The code should run the same as it did without the while statement. Change the value of the variable to anything other than 5. Observe that nothing happens because the while condition is not met.

```
volatile long int myVar = 5;      // Global variable that is expected to change

void setup() {
  Serial.begin(9600);
}

void loop() {

  while(myVar == 5){

    for(myVar = 0; myVar < 10; myVar++){

    Serial.print(myVar);             // Displays result in serial monitor
    Serial.print(" \r\n");           // Adds line feed and carriage return
    delay(500);                      // Sets a time to repeat every 5
    }
  }
}
```

Figure 8: Code to add a while statement

8. Add an if/else statement that will run the for loop when the value of the variable is less than or equal to 5 or a counter that increments by 2 if the value is greater than 5. Change the while condition to be greater than or equal to 0. First, run the code with the variable equal to 5. You should observe the same result as before from the for counter. Then, change the variable to 3. You should observe the number incrementing by 2.

```
volatile long int myVar = 3;       // Global variable that is expected to change

void setup() {
  Serial.begin(9600);
}

void loop() {

  while(myVar >= 0){

    if(myVar <= 5){

      for(myVar = 0; myVar < 10; myVar++){

      Serial.print(myVar);              // Displays result in serial monitor
      Serial.print(" \r\n");            // Adds line feed and carriage return
      delay(50);                        // Sets a time to repeat every 5
      }
    }
    else if(myVar > 5){
      myVar = myVar + 2;                // Increment the value by 2
      Serial.print(myVar);              // Displays result in serial monitor
      Serial.print(" \r\n");            // Adds line feed and carriage return
      delay(50);
    }
  }
}
```

Figure 9: Code to add an if/else statement

9. The full code these examples were pulled from is available for download. You are encouraged to review this in your free time. Try saving a backup copy and experiment by changing the variables and parameters of the condition. Add additional if/else or while statements to gain a better understanding of how these functions work.

Expected Outcomes:

Each team should complete the activity with the knowledge and skills to do simple digital output and input with the Arduino Mega microcontroller.