Analyze the code sample provided in the appendix below to answer the following questions.

1. Summarize, at a high level, what the program does.

The program uses the input from the user in the serial monitor to lower the brightness of 3 LEDs connected to pins 3, 5, and 6 simultaneously, then writes out the hexadecimal equivalent of the decimal value that is the brightness of the LEDs at that time.

1. Explain, in more detail, what each of the specific code sections does. There are six code sections identified by comment lines and highlighted in blue.

Section 1: Sets integer values to the variable names that represent each LED, with the pin numbers of the corresponding LEDs. Begins the setup function.

Section 2: Sets the baud value (speed) of the serial monitor, defines each variable mentioned earlier as outputs, essentially setting the LED pins as outputs. Starts the big loop and stops to check for user input in the serial monitor.

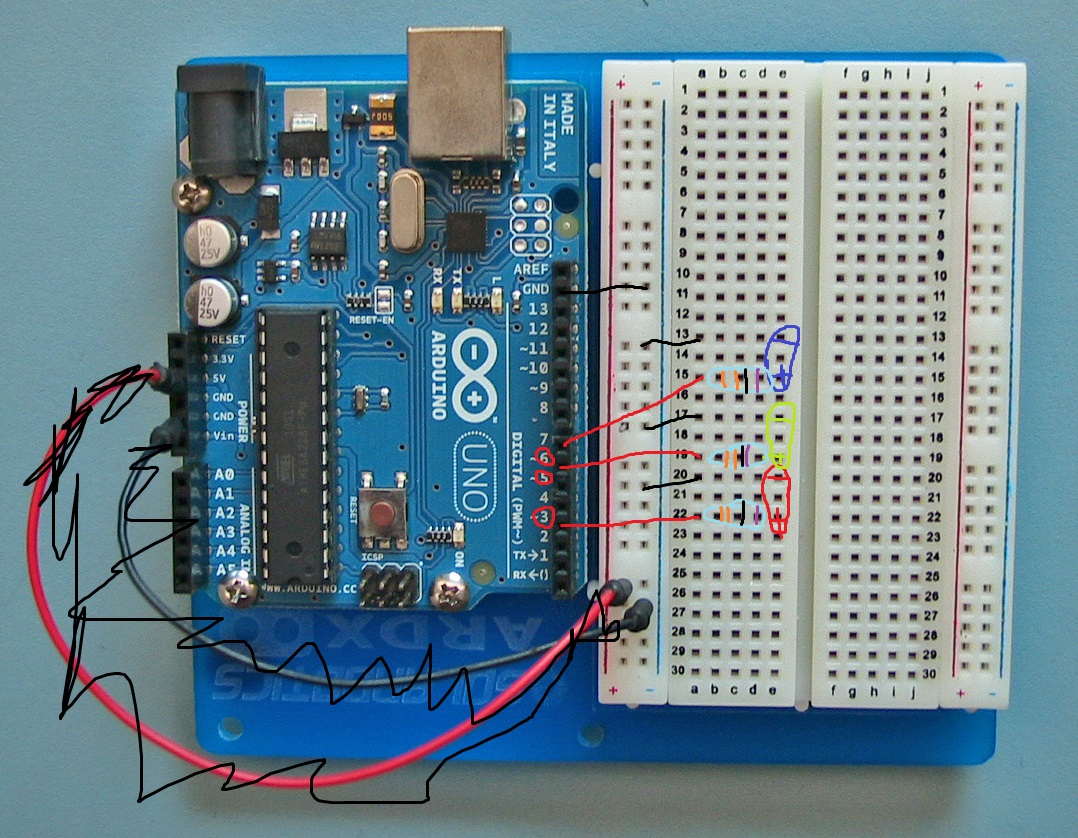
Section 3: Assigns a number to the variables called red, green, and blue as the number entered on the serial monitor. Checks if there is a newline to continue the program.

Section 4: Assigns a new value to the afformentioned variables in section 3, as 255 (Max brightness value available with this command) minus the number that was previously put into the monitor, as well as changing the number if it was outside the parameters of 0-255 by decreasing numbers over the cap to 255 and raising the numbers below 0 to 0.

Section 5: Sets the brightness of the LEDs to the value decided in section 4.

Section 6: Writes out the colours and their decimal values from section 4 and 5 as their hexadecimal equivalents.

1. Draw a diagram of the wiring diagram for the connection of LEDs to the Arduino board. Make sure to label and identify all pin numbers and assignments.



1. List all of the outputs of the program. Use a table similar to what you did in Module B.1. Make a table listing all of the outputs and their associated meaning.

|  |  |
| --- | --- |
| **Output** | **Meaning** |
| Pin 3 | Red LED turns on/off/changes brightness |
| Pin 5 | Green LED turns on/off/changes brightness |
| Pin 6 | Blue LED turns on/off/changes brightness |
| Red, Blue, Green and hex values. | The brightness of each LED converted from decimal to hexadecimal, and the names of each colour. |

1. List all of the inputs to the program. Use a table similar to what you did in Module B.1. Make a table listing all of the inputs and their associated action.

|  |  |
| --- | --- |
| **Input** | **Action** |
| Inputting a number into the serial monitor. | Dims the LEDs from 255 by inputted value capped at 255 and minimum at 0, so any value entered outside these gets set at 0 if it’s a negative number, and 255 if it is above. |

1. Provide an example of console input that would cause the program not to work properly. (i.e. Input that would cause an error.)

Any string value that is not an integer, for example, “red”, “blue”, “Toyota”, “Banana”, etc.

Start of Code Appendix

**// Code Section 1:**

const int redPin = 3;

const int greenPin = 5;

const int bluePin = 6;

void setup() {

**// Code Section 2:**

Serial.begin(9600);

pinMode(redPin, OUTPUT);

pinMode(greenPin, OUTPUT);

pinMode(bluePin, OUTPUT);

}

void loop() {

while (Serial.available() > 0) {

**// Code Section 3:**

int red = Serial.parseInt();

int green = Serial.parseInt();

int blue = Serial.parseInt();

// The character '\n' is a newline character appended to the typed in message   
 // from the serial console.

if (Serial.read() == '\n') {

**// Code Section 4:**

red = 255 - constrain(red, 0, 255);

green = 255 - constrain(green, 0, 255);

blue = 255 - constrain(blue, 0, 255);

**// Code Section 5:**

analogWrite(redPin, red);

analogWrite(greenPin, green);

analogWrite(bluePin, blue);

**// Code Section 6:**

Serial.print(red, HEX);

Serial.print(green, HEX);

Serial.println(blue, HEX);

}

}

}

\*\*\* End of Code Appendix