The Serial Monitor

Now that you've been adding inputs to your Arduino, it would be useful to be able to have a method for monitoring the information that is being sent from the inputs. In the previous project, the photoresistor constantly sends information based on the light level in the room. You then mapped the values being sent from the photoresistor to the LED brightness. However, these values have been "hidden" from you in the sense that they are essentially passing from component to component.

This is fine as long as your project is working as intended, but sometimes it's helpful to see the values that the Arduino is reading. In the previous example, it would've been useful to actually monitor the range of values that the Arduino reads from the photoresistor.

Arduino has a feature called the *serial console* for this purpose. This console is built into the Uno that you're using in this course. There are other Arduino boards with this feature, but it's important to note that not every board supports serial output.

Basically, the serial monitor is used to listen to the communication between the Arduino board and a computer.

For this project, you'll be using the circuit setup from the previous photoresistor project. Adding the monitor is done in the Arduino IDE, so for this project there is no need to change your connections.

Code Additions

Note: This example will be using the code from the potentiometer sketch.

There is some code that you need to add in order to use the serial monitor. Let's build it out step by step:

1. In **void setup()**, add the following: Serial.begin(9600); Your setup should look like this:

```
void setup() {
// put your setup code here, to run once:
pinMode(ledPin, OUTPUT); //Setup LED pin for output
Serial.begin(9600);
}
```

2. In **void loop()**, add the following *after* **potentValue**:

```
Serial.print("Analog Read: "); //Output to serial console
Serial.println(potentValue); //Output potentValue to seri
```

The **void loop()** should now look like this:

```
void loop() {
  // put your main code here, to run repeatedly:
  potentValue=analogRead(potentPin); //Read the value of the
  Serial.print("Analog Read: "); //Output to serial console
  Serial.println(potentValue); //Output potentValue to seria
  brightnessValue=map(potentValue,0,1023,0,255); //Map the po
  analogWrite(ledPin,brightnessValue); //Set the brightness o
}
```

The complete sketch should look like this:

```
//Variable to store pin of LED
int potentPin=0; //Variable to store pin of potentiometer
int potentValue=0; //Variable to store last known value of po
int brightnessValue=0; //Variable to store LED brightness

void setup() {
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT); //Setup LED pin for output
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  potentValue=analogRead(potentPin); //Read the value of the
  Serial.print("Analog Read: "); //Output to serial console
  Serial.println(potentValue); //Output potentValue to seria
  brightnessValue=map(potentValue,0,1023,0,255); //Map the po
  analogWrite(ledPin,brightnessValue); //Set the brightness o
}
```

New Commands

This project introduced 3 new commands: Serial.begin(), Serial.print(), and Serial.println(). Let's review each command before moving to the next project.

- **Serial.begin()** : This sets up the data rate for the serial connection. The typical value that you'll use is 9600. This number needs to match the value in the Serial Monitor window or else it won't work.
  - **Serial.print()** : This is the command that actually prints to the serial port.
  - **Serial.println()** : This command prints to the serial port and is followed by a new line. In the example code this is the command that prints and then creates a space after the entry.

Quick Test: Go ahead and change the Serial.println() in the code to be

Serial.print() and see what happens!