Julia Frothingham

Lab 17 - Arduino Project 1

What does pinMode do?

`pinMode` designates a pin as an output or an input pin. Basically, it tells the arduino whether the pin will be used to supply a voltage to a component (output) or if it will need to have a value from a component measured (input).

What does digitalWrite do?

`digitalWrite` sets a given output pin to either HIGH or LOW. If it is set to HIGH, the pin will output 5V. If it is set to LOW, the pin will output 0V.

What's the difference between digitalWrite and analogWrite?

`digitalWrite` is used for digital pins, and deals with digital signals (i.e., signals that are only one of two possible values.) `analogWrite` is used for analog signals, which can have more than two possible values.

What does analogRead do?

`analogRead` takes in an analog signal on a specified pin and converts it into a digital signal that can be read by a computer. It reads a signal and assigns it a value between 0 and 1023, where 0 corresponds to an input of 0V and 1023 corresponds to an input of 5V.

What good is the serial monitor? What can you use it for?

The serial monitor is useful for debugging. You can use it to print the values being measured and evaluate whether the values being measured are expected values. You can also use it to check your output values and determine whether unexpected circuit behavior is a code problem or a wiring problem.

What's the difference between serial.print("hi") and serial.println("hi")?

`serial.print("hi")`, if repeated, will print "hihihihihihihihihihi…..". This can be difficult to read.
`serial.println("hi")`, if repeated, will print
"hi"
"hi"
"hi"
"hi"
…
This is much easier to read, and to tell which output goes with which repetition.

Does an "if" statement have to have an "else" following it?

No, it does not.

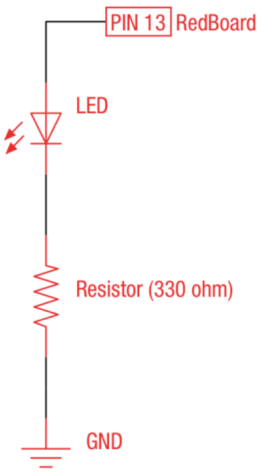
Circuit 1D had user-defined functions. What are those? Why would you want to define a function like that?

User-defined functions are blocks of code that a user writes that can be executed as functions. They're useful for code that's repeated, or that will be repeated with different inputs. With user-defined functions, it's more efficient because you can write just the function in the body of your code rather than every single line contained in the function definition.

Name 4 commands you used that require that a circuit element is connected to a particular pin on the Arduino

pinMode(), digitalWrite(), analogWrite(), analogRead()

Figure 1. Project 1 Circuit 1A diagram and code.

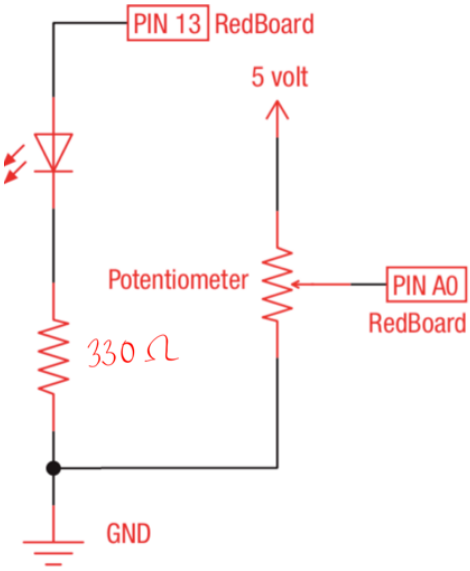


Circuit_1A_Blink

```
/*
 * Circuit 1A-Blink
 * Turns on LED connected to pin 13 on and off. Repeats forever
 */

void setup() {
  pinMode(13, OUTPUT);  // set pin 13 as an output
}

void loop() {
  digitalWrite(13, HIGH);  // force pin 13 to output HIGH, turning on LED
  delay(2000);             // pause 2000ms
  digitalWrite(13, LOW);  // force pin 13 to output LOW, turning off LED
  delay(2000);            // pause 2000ms

}
```

This code caused the LED to blink repeatedly with 2s between blinks. The frequency could be adjusted by changing the length of time in each `delay` call. A video of the circuit in action can be viewed here: https://youtu.be/kUb0enSqdW8

Figure 2. Project 1 Circuit 1B diagram and code.

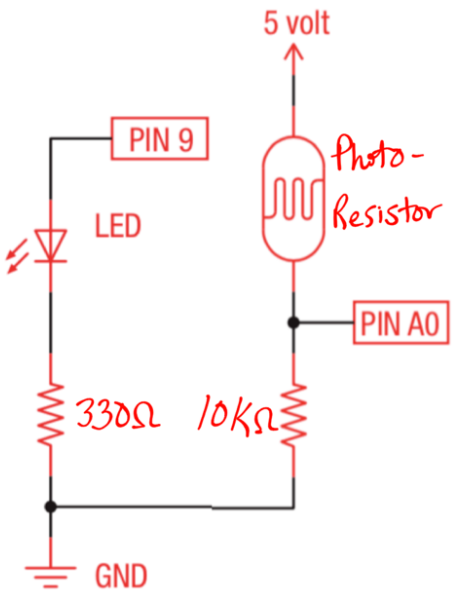


Circuit_1B_Potentiometer

```
/*
 * Circuit 1B-Potentiometer
 * Changes how fast an LED blinks, based on a potentiometer input
 */

int potPosition;        //create global variable - can be used in loop

void setup() {
  Serial.begin(9600);   //begin serial communication at 9600 bits/sec
  pinMode(13, OUTPUT);  //set pin 13 as an output
}

void loop() {
  potPosition = 2*analogRead(A0);   //read the signal on pin A0 (range from 0-2046)
  Serial.println(potPosition);    //print the read voltage in the serial monitor

  digitalWrite(13, HIGH);         //output HIGH on pin 13, turning LED on
  delay(potPosition);             //delay by an amount determined by potentiometer position

  digitalWrite(13, LOW);          //output LOW on pin 13, turning LED off
  delay(potPosition);             //delay by an amount determined by potentiometer position
}
```

This circuit causes the frequency of the LED to change based on input from a potentiometer. In the code above, the range of the delay is multiplied by 2, increasing it from 0-1023 seconds to 0-2046 seconds. A video of the circuit in action can be viewed here: https://youtu.be/y9H39u5FPnc

Figure 3. Project 1 Circuit 1C diagram and code.



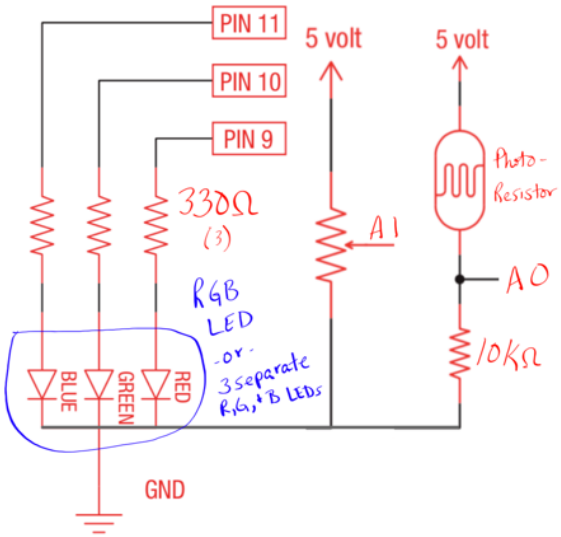
Circuit_1C_Photoresistor

```
/*
 * Circuit 1C-Photoresistor
 * Use a photoresistor to monitor how bright a room is, and turn on an LED when it gets dark
 */

int photoresistor = 0;    //create variable for photoresistor value

//change this to change how dark room needs to be for LED to turn on
int threshold = 750;      //create variable for room brightness threshold

void setup() {
  Serial.begin(9600);    //serial monitor starts communicating at 9600 baud
  pinMode(13, OUTPUT);   //set pin 13 as an output pin
}

void loop() {
  photoresistor = analogRead(A0);   //read value from photoresistor
  Serial.println(photoresistor);    //print read photoresistor value

// if the room is dark, turn on the LED. otherwise, keep it off
// wait 100ms before checking again
  if (photoresistor < threshold){
    digitalWrite(13, HIGH);
  } else {
    digitalWrite(13, LOW);
  }
  delay(100);
}
```

The circuit turns on an LED if the darkness of a room, measured with a photoresistor, drops to a certain level. Covering up the photoresistor causes the LED to turn on. A video of the circuit in action can be viewed here: https://youtu.be/SYVB9AdPEgg

Replacing the 10KΩ resistor with an LED shows the effects of Ohm's Law. When the photoresistor is uncovered, the LED "resistor" is on at what appears to be maximum brightness. This shows that the resistance across the photoresistor is low, so the current through that branch of the circuit is high, which powers the LED "resistor." When the photoresistor is covered, the LED "resistor" dims. This shows that the resistance across the photoresistor is increased, so the current through that branch of the circuit is decreased, which means less current flows through the LED "resistor," so it receives less current. A video of this modified circuit in action can be viewed here: https://youtu.be/JJ0yd8i-mR4

Figure 4. Project 1 Circuit 1D diagram and code.



Circuit_1D_RGBNightlight

```
/*
 * Circuit 1D-RGB Nightlight
 * Turns an RGB LED on or off based on the light level read by a photoresistor.
 * Change colors by turning the potentiometer.
 */

//variables to store values of photoresistor and potentiometer
int photoresistor = 0;
int potentiometer = 0;

int threshold = 800;  //controls how dark it needs to be for light to turn on

//pins driving or reading each component
//assigned here as variables for minimal code changes if wiring is changed
int RedPin = 9;
int GreenPin = 10;
int BluePin = 11;
int photoresistorPin = A0;
int potentiometerPin = A2;
void setup() {
  Serial.begin(9600); //begin serial communication

  //set pins to drive LEDs - variables used instead of hardcoded values in case wiring changes
  pinMode(RedPin, OUTPUT);
  pinMode(GreenPin, OUTPUT);
  pinMode(BluePin, OUTPUT);
}

void loop() {
  //read values from sensors - variables used instead of hardcoded values in case wiring chang
  photoresistor = analogRead(photoresistorPin);
  potentiometer = analogRead(photoresistor);

  //print measured values to serial monitor - useful for debugging and setting threshold
  Serial.print("Photoresistor value:");
  Serial.print(photoresistor);
  Serial.print("  Potentiometer value:");
  Serial.print(potentiometer);
  if(photoresistor < threshold){
    // what to do if photoresistor is measuring enough darkness to turn on LED
    //run different methods depending on potentiometer position
    //low end of dial sets LEDs to red, then goes through rainbow order as dial is turned
    if(potentiometer >= 0 && potentiometer <= 150)
      red();
    if (potentiometer > 150 && potentiometer <= 300)
      orange();
    if(potentiometer > 300 && potentiometer <= 450)
      yellow();
    if(potentiometer > 450 && potentiometer <= 600)
      green();
    if(potentiometer > 600 && potentiometer <= 750)
      cyan();
    if(potentiometer > 750 && potentiometer <= 900)
      blue();
    if(potentiometer > 900)
      magenta();
  }
  // what to do if the photoresistor isn't measuring enough darkness to turn on LED
  else {
    turnOff(); // keep the LED off
  }
  delay(500); // wait 500ms to read values and check conditions again
}
```

Figure 4 continued.

```
// define methods for each LED color state

void red(){
  // set LED to red
  // turns on red leg of LED at max value
  analogWrite(RedPin, 200);
  analogWrite(GreenPin, 0);
  analogWrite(BluePin, 0);
  Serial.println("  red");  //print color to serial monitor - useful for debugging
}

void orange(){
  // set LED to orange
  // turns on red leg of LED at max value and green leg at half value
  analogWrite(RedPin, 200);
  analogWrite(GreenPin, 100);
  analogWrite(BluePin, 0);
  Serial.println("  orange");  //print color to serial monitor - useful for debugging
}

void yellow(){
  // set LED to yellow
  // turns on red leg and green leg of LED at max value
  analogWrite(RedPin, 200);
  analogWrite(GreenPin, 200);
  analogWrite(BluePin, 0);
  Serial.println("  yellow");  //print color to serial monitor - useful for debugging
}

void green(){
  // set LED to green
  // turns on green leg of LED at max value
  analogWrite(RedPin, 0);
  analogWrite(GreenPin, 200);
  analogWrite(BluePin, 0);
  Serial.println("  green");  //print color to serial monitor - useful for debugging
}

void cyan(){
  // set LED to cyan
  // turns on green leg and blue leg of LED at max value
  analogWrite(RedPin, 0);
  analogWrite(GreenPin, 200);
  analogWrite(BluePin, 200);
  Serial.println("  cyan");  //print color to serial monitor - useful for debugging
}

void blue(){
  // set LED to blue
  // turns on blue leg of LED at max value
  analogWrite(RedPin, 0);
  analogWrite(GreenPin, 0);
  analogWrite(BluePin, 200);
  Serial.println("  blue");  //print color to serial monitor - useful for debugging
}

void magenta(){
  // set LED to magenta
  // turns on red leg and blue leg of LED at max value
  analogWrite(RedPin, 200);
  analogWrite(GreenPin, 0);
  analogWrite(BluePin, 200);
  Serial.println("  magenta");  //print color to serial monitor - useful for debugging
}

void turnOff(){
  // turns off LEDs
  // sets all legs to low
  analogWrite(RedPin, 0);
  analogWrite(GreenPin, 0);
  analogWrite(BluePin, 0);
}
```

The circuit turns **on** an RGB LED based on a photoresistor reading. If the room is dark or the photoresistor is covered up, the LED turns on. The color of the LED is controlled by a potentiometer, transitioning through the rainbow from red to magenta as the dial is turned. A video of this circuit in action can be viewed here: https://youtu.be/53CoBhtEObs

Figure 5. PWM output from Arduino measured in Scopy.