

—


Recap from yesterday

What did we learn?

Programming I

Samuel Bechara, PhD

Our Journal Process for Coding:

1. Describe program in words
 2. Write Pseudocode
 3. Sketch Diagrams and Flow of Code
 4. Program
 5. Test
 6. Repeat
- 
- A detailed image of a microchip with a grid of pins, resting on a blue circuit board background with intricate white circuit traces.

Arduino Syntax Recap

- Comments `//` or `/* */`
- Variables need a special data type and must be declared and defined
`float var1 = 10.23;`
- Semicolons end lines of code `;`
- Code in functions must be contained within `{ }`
- Functions determine if variables are “Global” or “Local”

Think about what this program does. Please don't shout anything out. Write it down in your journal.

A screenshot of the Arduino IDE interface. The title bar at the top reads 'alternate_bulitin_singleLED | Arduino 1.8.3'. Below the title bar is a toolbar with icons for checking, running, serial monitor, and file operations. The main text area contains the following C++ code:

```
int blueLED=10;

void setup()
{
  pinMode(blueLED, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(blueLED, HIGH);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
  digitalWrite(LED_BUILTIN, HIGH);
  digitalWrite(blueLED, LOW);
  delay(1000);
}
```

Your third arduino program

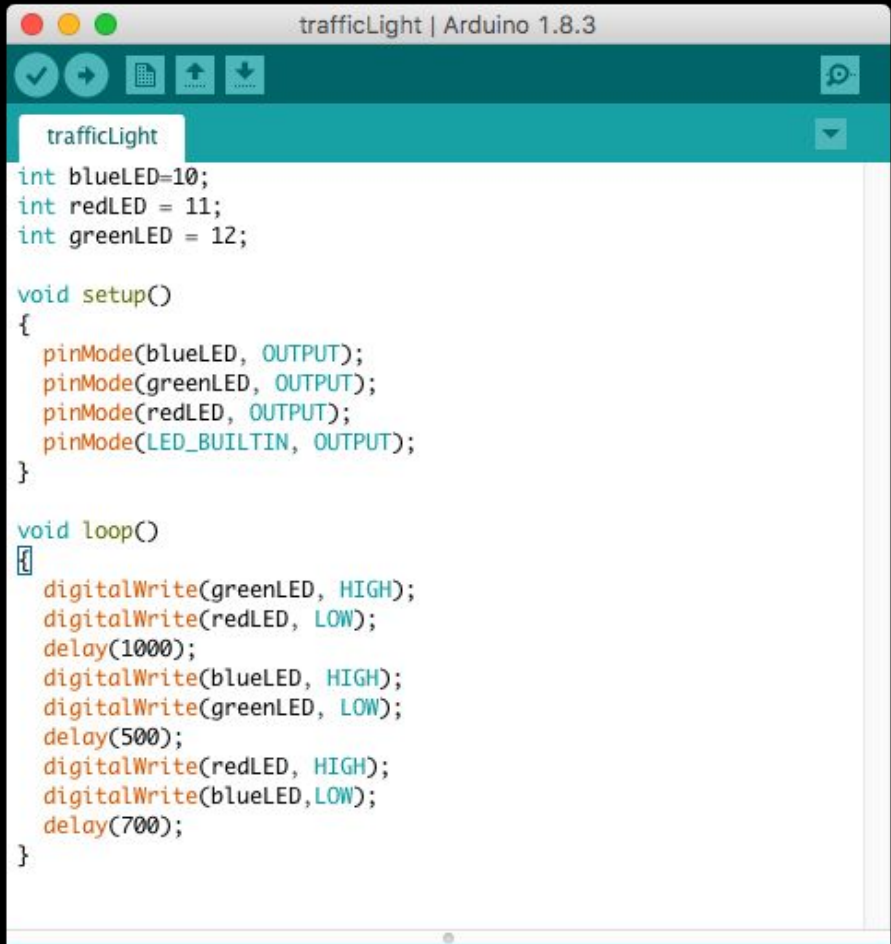
This time we are only going to give you the description. Make sure you go through the whole process. (Describe program in words, Write Pseudocode, Sketch Diagrams, Program, Test, Repeat). You can use what you have done so far to help! Be sure to save it and name it something that makes sense.

Description:

Hook up the other 2 LED lights. Alternate them like a traffic light. Some of you will not have Red, Green, Yellow LEDs. Make due with what you have.

Make whatever program that you want that uses all three LEDs!

What your program
should look like
(you can change the
delay to whatever
you want)

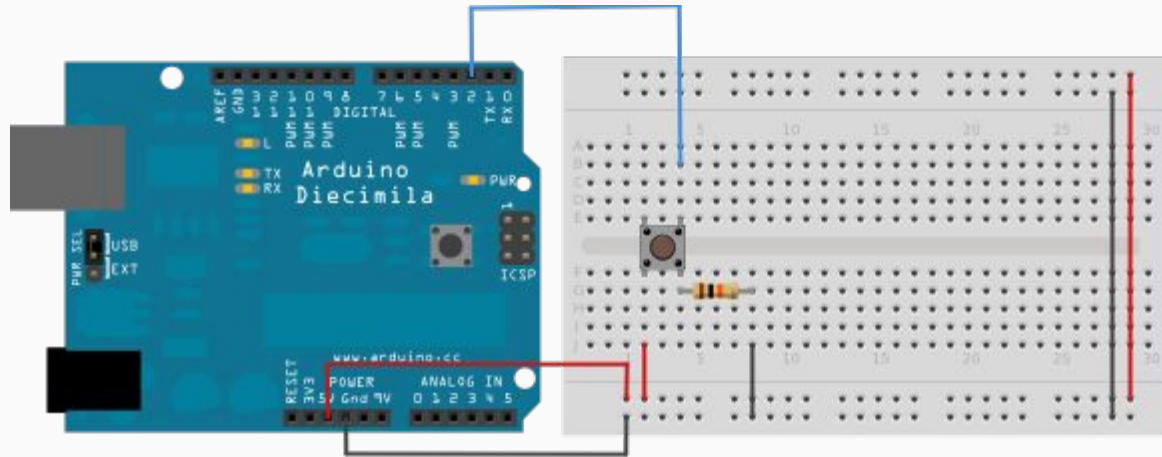
A screenshot of the Arduino IDE interface. The title bar at the top reads "trafficLight | Arduino 1.8.3". Below the title bar is a toolbar with icons for checking, running, serial monitor, and file operations. The main text area contains the following C++ code:

```
int blueLED=10;
int redLED = 11;
int greenLED = 12;

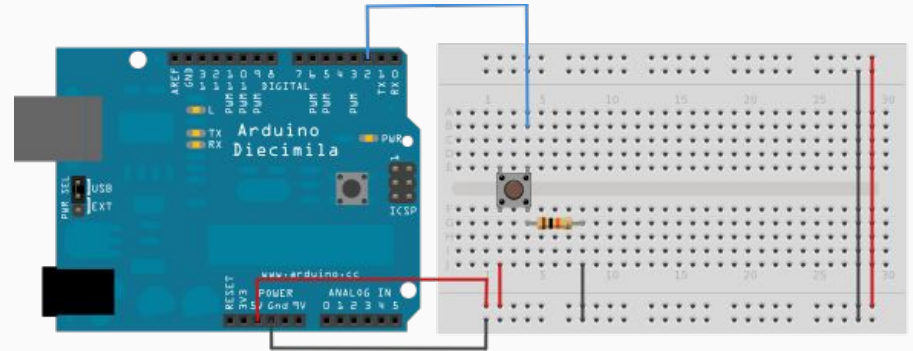
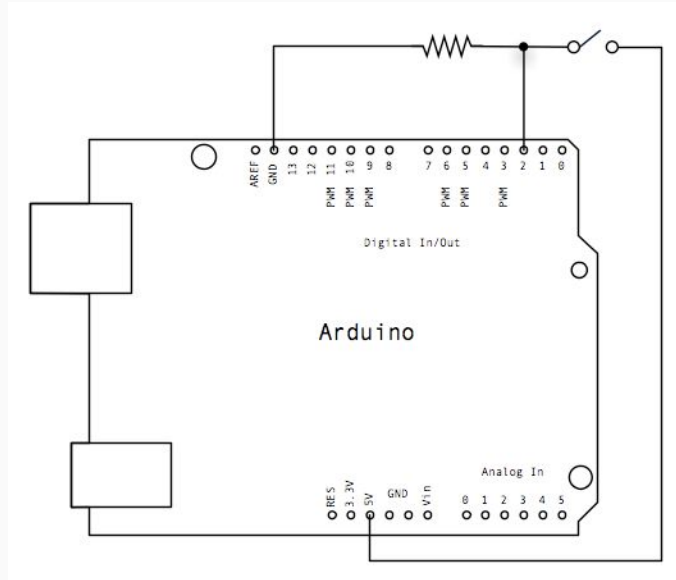
void setup()
{
  pinMode(blueLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(greenLED, HIGH);
  digitalWrite(redLED, LOW);
  delay(1000);
  digitalWrite(blueLED, HIGH);
  digitalWrite(greenLED, LOW);
  delay(500);
  digitalWrite(redLED, HIGH);
  digitalWrite(blueLED, LOW);
  delay(700);
}
```

You MIGHT have to move things around to make room... That's ok!
Let's add a button.



Diagram



Your fourth arduino program

Description:

What do you want your button to do? You should have 3 LEDs connected. Your button just acts as an input. Once the arduino registers the input, it can do something.

Everyone's program should be a little different.

Pseudocode our fourth arduino program!

Everyone's program will be different. Mine flashes all the LED's when the button is pushed.

Pseudocode:

1. Turn all LEDs off
2. If the button is pressed
 - a. Turn on all LEDs
 - b. Wait 500 milliseconds
 - c. Turn off all LEDs
3. Return to step 1

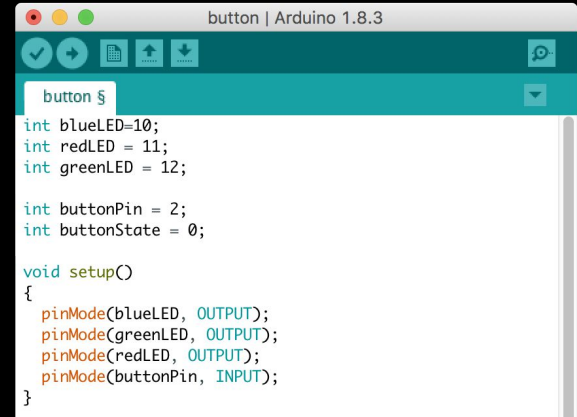
Start writing the fourth **arduino** program

Step 1: Like always, include a comment that describes your program

Step 2: As we have done before, let's set up int variables for all the pins. (the numbers to the right are just for illustration, yours are probably different!)

Step 3: Now we have a button. So let's add 2 new variables. One for the button pin, and one for its "state"

Step 4: In the setup, we make the LEDs OUTPUT as usual. But this time we need to make the button...

A screenshot of the Arduino IDE interface. The title bar reads "button | Arduino 1.8.3". The toolbar at the top includes icons for running, saving, and other standard IDE functions. The code editor shows a program with the following code:

```
button §  
int blueLED=10;  
int redLED = 11;  
int greenLED = 12;  
  
int buttonPin = 2;  
int buttonState = 0;  
  
void setup()  
{  
  pinMode(blueLED, OUTPUT);  
  pinMode(greenLED, OUTPUT);  
  pinMode(redLED, OUTPUT);  
  pinMode(buttonPin, INPUT);  
}
```

What are we missing?

We need to have the Arduino know when the button is pressed. How can we do that?

Conditionals and comparison operators!

Conditionals evaluate to either TRUE or FALSE

```
if (someVariable > 50)
{
    // do something here
}
```

Comparison Operators

`x == y` (x is equal to y)

`x != y` (x is not equal to y)

`x < y` (x is less than y)

`x > y` (x is greater than y)

`x <= y` (x is less than or equal to y)

`x >= y` (x is greater than or equal to y)

Consider the following. Let: `x = 4`, `y = 5`

`x == y` False

`x != y` True

`x < y` True

`x > y` False


`x <= y` True

`x >= y` False

Lets look at the rest of the fourth arduino program

Now we have a statement that checks the state of the button
(HIGH corresponds to being pressed)

IF it is pressed, something happens.

A screenshot of the Arduino IDE interface. The title bar reads "button | Arduino 1.8.3". The code editor shows the following code:

```
int blueLED=10;
int redLED = 11;
int greenLED = 12;

int buttonPin = 2;
int buttonState = 0;

void setup()
{
  pinMode(blueLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop()
{
  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH)
  {
    digitalWrite(greenLED, HIGH);
    digitalWrite(redLED, HIGH);
    digitalWrite(blueLED, HIGH);
    delay(500);
    digitalWrite(greenLED, LOW);
    digitalWrite(redLED, LOW);
    digitalWrite(blueLED, LOW);
  }
```

What if you want something to happen if the button isn't pressed?

You can use an `if else` or an `else!`

```
if (someVariable > 50)
{
}

else if (variable2 < 20)
{
}

else
{
}
```


Finish up the fourth arduino program



```
button | Arduino 1.8.3  
button $  
int blueLED=10;  
int redLED = 11;  
int greenLED = 12;  
  
int buttonPin = 2;  
int buttonState = 0;  
  
void setup()  
{  
  pinMode(blueLED, OUTPUT);  
  pinMode(greenLED, OUTPUT);  
  pinMode(redLED, OUTPUT);  
  pinMode(buttonPin, INPUT);  
}  
  
void loop()  
{  
  buttonState = digitalRead(buttonPin);  
  
  if (buttonState == HIGH)  
  {  
    digitalWrite(greenLED, HIGH);  
    digitalWrite(redLED, HIGH);  
    digitalWrite(blueLED, HIGH);  
    delay(500);  
    digitalWrite(greenLED, LOW);  
    digitalWrite(redLED, LOW);  
    digitalWrite(blueLED, LOW);  
  }  
}
```

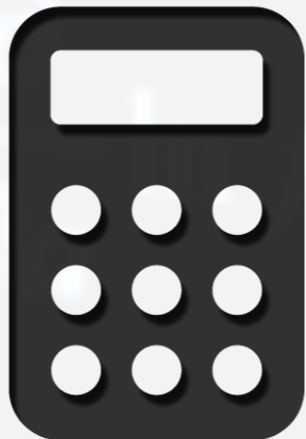
Your fifth arduino program

Description:

For this program lets tie everything together. Lets make the button change the lit LED to a random LED. Then, lets “count” the number of button presses. If the press is a multiple of 5 i.e. (5, 10, 15, 20, etc) let's turn on all three LEDs.

Let's also output the number of presses to the Serial port.

Remember this slide?



All sorts of things! You can perform arithmetic just like you would with a calculator.

- + Addition
- − Subtraction
- / Division
- * Multiplication
- % **Remainder**

Pseudocode our fifth arduino program!

Pseudocode:

Setup: Choose random LED and light. Start Serial.

1. Light up random LED
2. If the button is pressed, increment buttonCounter, print to serial
 - a. If (buttonPresses % 5 == 0)
 - i. Blink all LEDs*
 - b. Else
 - i. Turn off LED(s) that is on
 - ii. Turn on random LED ** for now, don't worry if it is the same LED as before*
 - iii. Remember what LED is on
3. Return to step 1

Arduino Random

The random() function generates pseudo-random numbers

Syntax

`random(max)` OR `random(min, max)`

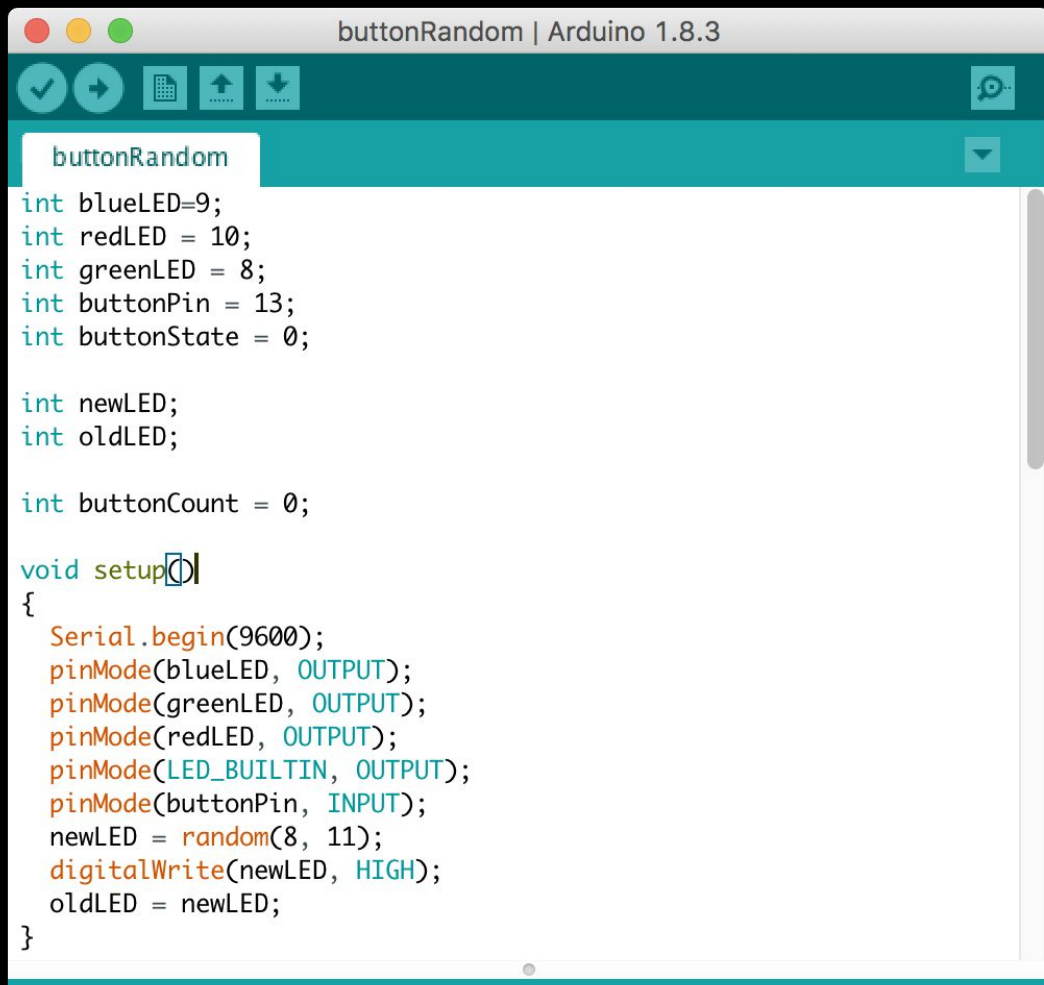
Returns

The function RETURNS a random number between `min` and `max - 1`

Start writing the fifth arduino program

This one is pretty tough, you can see what I called variables and how I setup the program on the right.

Don't forget to comment and sketch!

A screenshot of the Arduino IDE interface. The title bar at the top reads 'buttonRandom | Arduino 1.8.3'. Below the title bar is a toolbar with icons for checking, running, serial monitor, and file operations. The main text area contains the following C++ code:

```
buttonRandom

int blueLED=9;
int redLED = 10;
int greenLED = 8;
int buttonPin = 13;
int buttonState = 0;

int newLED;
int oldLED;

int buttonCount = 0;

void setup()
{
  Serial.begin(9600);
  pinMode(blueLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(buttonPin, INPUT);
  newLED = random(8, 11);
  digitalWrite(newLED, HIGH);
  oldLED = newLED;
}
```

Speed of Light Game



—

Your goal:

Make a game that works with the button.

You will get more LED's to hook up. Arrange them so that they are in a row. You should have 5 in total.

The lights should “bounce” back and forth. The goal of the player is to stop the light on the middle LED.

They get 5 tries.

If they stop it on the outside LEDs -> 100pts

If they stop it on the inside LEDs -> 250 pts

If they stop it on the middle LED -> 500 pts



Take a minute to reflect.



Tip

Write down whatever you want.

What did you learn today?

What do you hope to learn tomorrow?