Intro

1) Create an Arduino image that writes “Hello world” in the Serial Monitor. Make sure it only happens once.

2) Create an Arduino image that writes “Hello world” in the Serial Monitor. Make sure it once per frame.

3) Create an Arduino image that writes “Hello world” once in the beginning in the Serial Monitor. After that write your name in the serial monitor every frame.

4) Create an Arduino image that writes “Hello world” in the Serial Monitor. Write “Hello” and “world” into different parts. Write your name after that on the next line.

Variables

1) Create an Arduino image that writes the sum of two numbers.

2) Create an Arduino image that writes the result of dividing two numbers.

3) Write an Arduino image that writes the result of the specified operations.
   EXAMPLE:
   -1 + 4 * 6
   (35+ 5) % 7
   14 + -4 * 6 / 11
   2 + 15 / 6 * 1 - 7 % 2

   RESULT:
   23
   5
   12
   3

4) Create an Arduino image to swap two numbers.
   EXAMPLE
   Input the First Number : 5
   Input the Second Number : 6
   Expected Output:
   After Swapping :
   First Number : 6
   Second Number : 5

5) Create an Arduino image to print the output of multiplication of three numbers which..
   EXAMPLE:
   Input the first number to multiply: 2

Input the second number to multiply: 3
Input the third number to multiply: 6
Expected Output:
2 x 3 x 6 = 36

6) Create an Arduino image that converts a given number from euros into dollars. Use this exchange rate: 1000 EUR = 1190$. Use Integer numbers. Output the result

7) Create an Arduino image that converts degrees celsius into fahrenheit using this formula: F = (9/5) x C + 32. Output the result (check your findings by googling some numbers and convert them to fahrenheit.)

8) Create an Arduino image that calculates how old you are based on the year you were born (don't incorporate the exact date, just the year) Output the result.

9) Create an Arduino image that converts seconds into hours, minutes and seconds.
   EXAMPLES:
   70 sec equals 1 min and 10 sec
   3601sec equals 1 hour, 0 mins and 1 sec
   4000 sec equals 1hour, 6 mins and 40sec

10) Create an Arduino image that writes "Hello world, NAME" in the Serial Monitor. Change NAME with your name. Save your name in a variable and use this variable.

11) Create an Arduino image that writes the length of a word which you save in a predefined variable.

12) Create an Arduino image that writes the first character of a word which you save in a predefined variable.

13) Create an Arduino image that writes the last character of a word which you save in a predefined variable.

14) Create an Arduino image that writes the middle character of a word which you save in a predefined variable.

Selections

1) Make an Arduino image with two integers and check whether they are equal or not. Print out the result.

2) Make an Arduino image to check whether a given number is even or odd.Print out the result.

3) Make an Arduino image to check whether a given number is positive or negative.Print out the result.

4) Make an Arduino image to find whether a given year is a leap year or not.Print out the result.

5) Make an Arduino image to read the value of an integer and display the value 1 when it is larger than 0, 0 when it is 0 and -1 when it is less than 0.Print out the result.

6) Make an Arduino image to read temperature in centigrade and display a suitable message according to temperature state below :

    Temp < 0 then Freezing weather
    Temp 0-10 then Very Cold weather
    Temp 10-20 then Cold weather
    Temp 20-30 then Normal in Temp
    Temp 30-40 then Its Hot
    Temp >=40 then Its Very Hot

7) Make an Arduino image to check a grade and declare the equivalent description :

Grade	Description
E	Excellent
V	Very good
G	Good
A	Average
F	Fail

8) Make an Arduino image to check any day number saved in a variable and display the day name in the word (use if statements).

    EXAMPLE :
    4
    Expected Output :
    Thursday

9) Make an Arduino image to check any Month number saved in a variable and display Month name in the word.

    EXAMPLE :
    4
    Expected Output:
    April

Loops

1) Write an Arduino image to display the first 10 natural numbers (using a while, then using a for loop).

Expected Output :
1 2 3 4 5 6 7 8 9 10

2) Write an Arduino image to find the sum of the first 10 natural numbers.
   Expected Output :
   The first 10 natural number is :
   1 2 3 4 5 6 7 8 9 10
   The Sum is : 55

3) Write an Arduino image to display the cube of the number up to 10.
   Expected Output :
   Number is : 1 and cube of the 1 is :1
   Number is : 2 and cube of the 2 is :8
   Number is : 3 and cube of the 3 is :27
   Number is : 4 and cube of the 4 is :64
   Number is : 5 and cube of the 5 is :125
   …

4) Write an Arduino image to display the multiplication table of a number saved in a variable.
   Test Data :
   Number : 15
   Expected Output :
   15 X 1 = 15
   …
   …
   15 X 10 = 150

5) Write an Arduino image to display the multiplication table vertically from 1 to 10.
   Expected Output :
   Multiplication table from 1 to 8
   1x1 = 1, 2x1 = 2, 3x1 = 3, 4x1 = 4, 5x1 = 5, 6x1 = 6, 7x1 = 7, 8x1 = 8, 9x1 = 9, 10x1 = 10
   ...
   1x10 = 10, 2x10 = 20, 3x10 = 30, 4x10 = 40, 5x10 = 50, 6x10 = 60, 7x10 = 70, 8x10 = 80, 9x10 = 90, 10x10 = 100

6) Write an Arduino image to display the first 10 odd numbers and their sum.
   Expected Output :
   The odd numbers are :1 3 5 7 9 11 13 15 17 19
   The Sum of odd Natural Number up to 10 terms : 100

7) Write an Arduino image to determine whether a saved number is prime or not (remainder of a division of any number between that number and 1 equals to 0).
   Test Data :
   Number: 13
   Expected Output :
   13 is a prime number.

8) Write an Arduino image (nested loop) that prints out this:
   *
   **
   ***
   ****
   *****
   ******

9) Write an Arduino image (nested loop) that prints out this:

   0****
   01***
   012**
   0123*
   01234

10) Write an Arduino image (nested loop) that prints out this:

    ****1
    ***22
    **333
    *4444
    55555

Basic methods

1) Write an Arduino image to create a method that shows the following output:
   Expected Output :
   Welcome Friends!
   Have a nice day!

2) Write an Arduino image to create a method with a string parameter (a name).
   Test Data :
   Name : John
   Expected Output :
   Welcome friend John !
   Have a nice day!

3) Write an Arduino image to create a method that writes out the sum of two numbers (pass the numbers from main to the function using 2 parameters).
   Test Data :
   Number 1: 15
   Number 2: 16
   Expected Output :
   The sum of two numbers is : 31

4) Write an Arduino image to create a method that writes out the quotient of two numbers (pass the numbers from main to the function using 2 parameters).
Test Data :
Number 1: 10
Number 2: 4
Expected Output :
The sum of two numbers is : 2.5

5) Write an Arduino image using a method to count the number of spaces in the string.
Test Data :
Phrase: This is a test string.
Expected Output :
"This is a test string." contains 4 spaces

6) Write an Arduino image to create a function to calculate the result of raising an integer number to the power of another.
Test Data :
Base number: 3
Exponent : 2
Expected Output :
So, the number 3 ^ (to the power) 2 = 9

7) Write an Arduino image to create a function to check whether a number is prime or not.
Test Data :
Number : 31
Expected Output :
31 is a prime number

8) Write an Arduino image to create a method to find the factorial of a given number.
Test Data :
Number: 5
Expected Output :
The factorial of 5! is 120

Methods with return type

1) Write an Arduino image to create a method with a string parameter (a name).
Test Data :
Name : John
Expected Output :
Welcome friend John !
Have a nice day!

2) Write an Arduino image to create a method that returns the sum of two numbers to the main(pass the numbers from main to the function using 2 parameters). The result is outputted by the main method (not the function).
   Test Data :
   Number 1: 15
   Number 2: 16
   Expected Output :
   The sum of two numbers is : 31

3) Write an Arduino image using a method that returns the number of spaces in the string.
   Test Data :
   Phrase: This is a test string.
   Expected Output :
   "This is a test string." contains 4 spaces

4) Write an Arduino image to create a function to calculate the result of raising an integer number to the power of another. Return the result to write it in the serial monitor.
   Test Data :
   Base number: 3
   Exponent : 2
   Expected Output :
   So, the number 3 ^ (to the power) 2 = 9

5) Write an Arduino image to create a function to check whether a number is prime or not. Return the result to write it in the serial monitor.
   Test Data :
   Number : 31
   Expected Output :
   31 is a prime number

6) Write an Arduino image to create a method to find the factorial of a given number. Return the result to write it in the serial monitor.
   Test Data :
   Number: 5
   Expected Output :
   The factorial of 5! is 120

Digital

1) Stopping lights

    a) Make an Arduino image where you check if the button is pressed or not. The button is connected **with the blue wire** to **digital pin 4**.
    When the pin button is pushed you write “PUSHED” in the serial monitor.
    When it is not you write “NOT PUSHED”.

b) Make the button a trigger so every time the button gets pushed we only register it once instead of every frame. Write “CLICK” once in your serial monitor every time you press the button.

c) When pressing the button toggle the 3 LEDs off and on. (Click and they go on, click again and they go off,...).
The red LED is connected **with the pink wire** to **digital pin 5**.
The orange LED is connected **with the orange wire** to **digital pin 6**.
The green LED is connected **with the green wire** to **digital pin 7**.

d) Change your code so every time you click on the button the next light turns on (and the others turn off).

2) To send a nuke to switches needs to be flipped. we want a notification when a switch is on and we want an extra notification when both are on.

a) Make an Arduino image where you check if a switch is on or not.
The first switch is connected **with the blue wire** to **digital pin 3**.
The second switch is connected **with the green wire** to **digital pin 4**.
Log which switch is on in the serial monitor so you can check yourself.

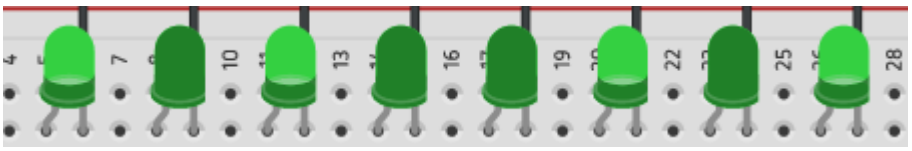
b) When the first first switch is on turn on the left orange LED.
The left orange LED is connected **with the yellow wire** to **digital pin 5**.

c) Repeat the same for the second switch and the right orange LED.
The right orange LED is connected **with the orange wire** to **digital pin 6**.

d) When both switches are on turn on the red LED.
The red LED is connected **with the pink wire** to **digital pin 7**.

e) When the red LED is turned on the LEDs must remain on even when we put one or both switches off again.

3) Binary numbers are made out of 0’s and 1’s. We can represent them by switches and LEDs. When you open the exercise it may look a bit overwhelming. There are 8 switches and 8 LEDs. These represent 1 byte. A byte can contain 256 different values (from 0 to 255). When we turn a switch on the corresponding LED goes on as well and it represents a 1. If the switch is off the LED is off and a 0 is represented.
Example:



LED	LED 1	LED 2	LED 3	LED 4	LED 5	LED 6	LED 7	LED 8
ON/OFF								

Bin value	1	0	1	0	0	1	0	1
Represented dec value	128	64	32	16	8	4	2	1

So this means that the decimal value is equal to:
1 X 128 + 0 + 64 + 1 X 32 + 0 X16 + 0 X 8 + 1 X 4 + 0 X 2 + 1 X 1 =
128 + 32 + 4 + 1 = 165

a) Check if everything works as you expect. Turn every switch on and off and check if the corresponding light goes on and off as well.

b) Read the value of the first switch and write it out in the Serial monitor.
LED 1 is connected **with a green wire** to **digital pin 9**.

c) Do the same for the other switches.
LED 2 is connected **with a yellow wire** to **digital pin 8**.
LED 3 is connected **with a green wire** to **digital pin 7**.
LED 4 is connected **with a yellow wire** to **digital pin 6**.
LED 5 is connected **with a green wire** to **digital pin 5**.
LED 6 is connected **with a yellow wire** to **digital pin 4**.
LED 7 is connected **with a green wire** to **digital pin 3**.
LED 8 is connected **with a yellow wire** to **digital pin 2**.

d) Make a variable to keep track of the decimal value. Add the represented values of the switches who are on just like in the example above and write out the decimal value.

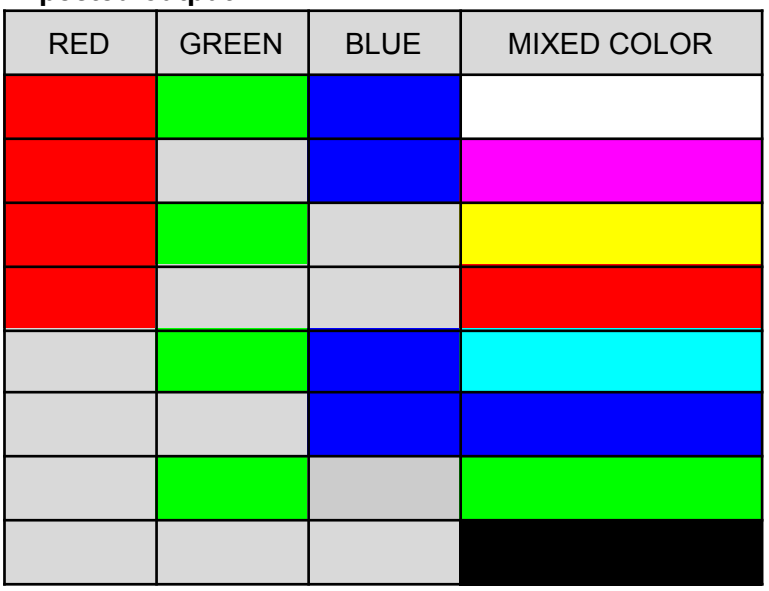
4) In this exercise you see 2 breadboards. On the first you see 3 switches and 3 LEDs. Every switch is connected with a LED of a different color. On the second breadboard we see a special LED and a push button. This is an rgb LED we can make it shine in every color on the rgb spectrum. We want to use this setup to change the color of the rgb LED to the mixed color of the other LEDs when we push the button.
There is already some code in this exercise. You do not need to know how this part works yet and may just leave it be.

a) Read the value of the switch of the red LED and write it out in the serial monitor.
Red switch is connected **with a pink wire** to **digital pin 3**.

b) Do the same for the other switches.
Green switch is connected **with a green wire** to **digital pin 4**.
Blue switch is connected **with a blue wire** to **digital pin 5**.

c) When we push the button change the color of the rgb LED. You can change the color by using the ShowColor() method. This method has 3 parameters, one for each color channel value. When red is on, assign value 1 to the red parameter and if it is off assign 0. Do the same for the other two channels.

Expected output:



d) Make the button a trigger so every time the button gets pushed we only register it once instead of every frame.

Analog

1) The darker it is, the lighter we want to make it.

   a) Read how light it is through the photoresistor. Write it out in the serial monitor. The photoresistor is connected **with the blue wire** to **analog pin 0**.

   b) Write an analog signal to the LED depending on the signal we receive from the photoresistor. In this step we are going to make the LED brighter when there is more light. Keep in mind that an incoming signal is between 0 and 1023 and an outgoing signal should be between 0 and 255.
   The LED is connected **with the yellow wire** to **digital pin 3**.

   c) Reverse the outgoing signal so that the LED becomes brighter when there is less light.

2) The potentiometer in this exercise should divide a voltage of 5V over the two resistors. The multimeters can show how much voltage there is over the resistors. The more we drag the potentiometer counter clockwise the more voltage we will see on the left multimeter. The more we drag it clockwise the more voltage we'll see on the right multimeter.

   a) Read the value of the potentiometer and write it out in the serial monitor. The potentiometer is connected **with a blue wire** to **analog pin 0**.

   b) We know that we read analog values between 0 and 1023 and we send values between 0 and 255. Translate the read value to a value to send

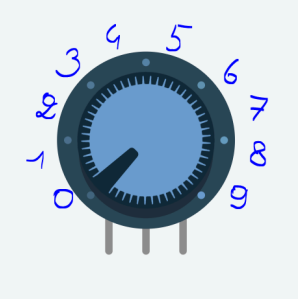
through the left resistor. Send a value through the right resistor so the sum of the 2 values is 255.
The sum of the values on the multimeters should always be 5V. (Mind that when a value becomes really small it will be in mV instead of V.)

3) To crack this vault 3 dials must be set to the right stand before pushing the button.

   a) Start by making a trigger of the button. Write out something in the serial monitor to check if it works.
   The button is connected **with the yellow wire** to **digital pin 5**.

   b) When the button is pressed we check what the dials say. Do this in a method. For now just write out the value of the analog value.
   The left potentiometer is connected **with a blue wire** to **analog pin 2**.
   The middle potentiometer is connected **with a blue wire** to **analog pin 1**.
   The right potentiometer is connected **with a blue wire** to **analog pin 0**.

   c) Translate the values of the dials which go from 0 to 1023 to a value between 0 and 9.
   This is a sketch for how the dials should be read.



   d) When the button is clicked we want to check if the mapped values are equal to the code (which is saved in digOne, digTwo and digThree). When a dial is set correctly, light the corresponding LED.
   The left button is connected **with the green wire** to **digital pin 2**.
   The left button is connected **with the green wire** to **digital pin 3**.
   The left button is connected **with the green wire** to **digital pin 4**.

   e) When we later on move the dial again we want the LEDs to turn off again.

   f) Try again after changing the code.

4) Just as on a mixer we want the more we open the potentiometer the more LEDs go on. They always start at the bottom (green) and go to the top (red). When the potentiometer is completely closed no LEDs are on.

   a) Read the value of the potentiometer and write it out in the serial monitor.
   The potentiometer is connected **with a blue wire** to **analog pin 0**.

b) Check if the value exceeds the threshold for each LED to go on. There are 9 LEDS so we divide the max value by 8. Each threshold is ⅛ of the max value bigger than the previous and the first one starts at 1.
The first LED is connected **with a green wire** to **digital pin 2**.
The second LED is connected **with a green wire** to **digital pin 3**.
The third LED is connected **with a green wire** to **digital pin 4**.
The fourth LED is connected **with a yellow wire** to **digital pin 5**.
The fifth LED is connected **with a yellow wire** to **digital pin 6**.
The sixth LED is connected **with a yellow wire** to **digital pin 7**.
The seventh LED is connected **with an orange wire** to **digital pin 8**.
The eight LED is connected **with an orange wire** to **digital pin 9**.
The last LED is connected **with the pink wire** to **digital pin 10**.

0 - 1	1 - 1/8	1/8 - 2/8	2/8 - 3/8	3/8 - 4/8	4/8 - 5/8	5/8 - 6/8	6/8 - 7/8	7/8 - 8/8	8/8 - …
NO LEDS	LED 1	LED2	LED 3	LED 4	LED 5	LED 6	LED 7	LED 8	LED 9

c) If not already try doing this with an array for the LEDs and a loop to check which ones need to be on. This will be a lot shorter.

Non blocking code

1) With a click on the button we put the sirens of the police car on.

   a) Make sure that when you click on the button you show it in the serial monitor. This should be a trigger.
   The button is connected **with the green wire** to **digital pin 3**.

   b) Switch every 0.5 seconds between the red LED and the blue LED. This means one is on and the other is off. Make sure this is non blocking code because we want to use the button to make it stop.
   The blue LED is connected **with the blue wire** to **digital pin 2**.
   The red LED is connected **with the pink wire** to **digital pin 4**.

   c) If not already make a click on the button start and stop the sirens.

   d) Make sure the processor is not working on 100% of its capacity by delaying 20 milliseconds at the end of the loop.

2) Let's make a game that counts down and we should press the button as close to the 0 as possible.
   a) We will only be using 1 button to both start and stop the game. Start by making a trigger for this button.
   The button is connected **with the blue wire** to **digital pin 2**.

b) Make a boolean that checks if we are counting down. This will decide what we are doing when pressing the button. (When we press it when the boolean is true it stops the timer. When it is false it restarts the timer.)

c) Make a method that handles everything when the button is triggered.

d) When the button is pressed the first time we want an analog signal of 5V (255) to be sent out of pin 3 into the resistor (with the multimeter over it).
The resistor is connected **with the yellow wire** to **digital pin 3**.

e) Once the previous signal is sent lower the signal by 1 after 20 milliseconds. Repeat it until it is 0.

f) When the button is pressed during the countdown the countdown should be stopped. So make sure it is non blockable code. (If it is hard to check if it works, change the duration between two different analog signals to 1000 milliseconds.)

g) When the countdown reaches 0 before the user clicks the button the red LED should light up.
The red LED is connected **with the pink wire** to **digital pin 4**.

h) When the user clicks the button before the countdown reaches 0 the green LED should light up.
The red LED is connected **with the green wire** to **digital pin 5**.

i) When the timer is reset and the countdown restarts both LEDs should be off.

j) Make sure the processor is not working on 100% of its capacity by delaying 10 milliseconds at the end of the loop.

3) We want to use the potentiometers to decide how much of every color we want to mix. The rgb LED will show that color for 10 seconds when we press the button. After 10 seconds the LED goes off again. When the LED is on and we change the color and press the button again the color should change and stay there for 10 seconds as well.

   a) Make a trigger for the button.
   The button is connected **with the yellow wire** to **digital pin 2**.

   b) When the button is triggered we want to check the incoming signal of the potentiometers for each color channel.
   The left potentiometer is connected **with a pink wire** to **analog pin 2**.
   The middle potentiometer is connected **with a green wire** to **analog pin 1**.
   The right potentiometer is connected **with a blue wire** to **analog pin 0**.

c) Use the values from the potentiometers to send analog signals to the rgb LED. Keep in mind that the incoming and outgoing analog signals are mapped differently.
The red pin of the LED is connected **with a pink wire** to **analog pin 9**.
The green pin of the LED is connected **with a green wire** to **analog pin 11**.
The blue pin of the LED is connected **with a blue wire** to **analog pin 10**.

d) Make sure that after 10 seconds the LED goes off again. You can do this by sending an analog signal of 0 into every color channel of the rgb LED. Do this with non blocking code because we still want to be able to change the color during these 10 seconds.

e) Make sure that you can change the color during the 10 seconds by pressing the button again. When the color is changed it should once more take 10 seconds before the LED goes off again.

f) Make sure the processor is not working on 100% of its capacity by delaying 20 milliseconds at the end of the loop.

4) This exercise is a bit more advanced because there is already some code inside. A LED “moves” around the ring. we want when we press the button the moving stops and when we press the button again it continues where it left off.
(The LED doesn’t actually move but the LEDs take turns being on.)

a) Try to understand what the code does.

b) Make a trigger for the button. Do this outside of the for loop.
The button is connected **with the blue wire** to **digital pin 2**.

c) Change the delay to non blocking code so we can use the trigger to stop and restart the moving of the LED.

d) Make sure the processor is not working on 100% of its capacity by delaying 20 milliseconds at the end of the loop.