# 

# **Laboratory 2: Hello Serial & LED**

## Objectives

1. Understand the concept of serial communication
2. Understand the ASCII code
3. Understand the concept of data format
4. Understand the concept of Light Emitting Diode.
5. Understand how to properly connect LED to the M5Stack development kit.
6. Practice more complex sketches

## Digital Logic

***Digital Logic*** is an alternative term for ***Digital Electronic (or Digital Circuits)***. Digital is a way for representing the electronic signals as High (True) and Low (False) rather than the real analog values (voltages). Thus, small changes in the analog levels (voltages), such as noise from circuits or communication channels, does not effect the digital values. This allows electronic devices to switch to known states than producing specific values (voltages).

Usually, a signal level (voltage) near a reference ground level is called Low or False or 0. A signal level (voltage) near the supply voltage is called High or True or 1. With this notion in mind, we can simply refer to our digital value as high or low.

## Light Emitting Diode (LED)

LED is short for ***Light Emitting Diode***. It is a special kind of diodes (unipolar electronic devices) that will emit photon (light) where there is an electronic current flow in a correct direction (from anode to cathode). Illustration 3 and Illustration 2 show the symbol and the picture of an LED.

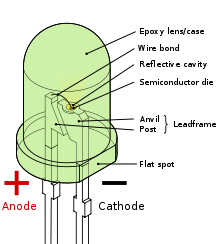


Illustration 2: A picture of LED (a picture from wikipedia.org)



Illustration 3: Electronic Symbol of LED

Due to its characteristics, LED is an ideal choice for using as an output of a digital pin. Using LED as a digital output, we can easily observe the logic from the light. There are several ways to connect the LED to an output PIN. Here are two simple methods presented (see Illustration 4). Depending on the electronic properties of the device, the quality may vary. For most cases, LED1 should provide better quality (brighter).

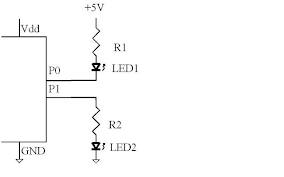


Illustration 4: Two methods for interfacing LED

In particular, the LED1 (connected to P0) will emit the light when the output of P0 is LOW. The LED 2 will emit the light when the output of P1 is HIGH. To determine the correct logic, a programmer must understand the schematic of the circuits.

## Arduino and Digital Input/Output (I/O) Pins

Arduino supports 13 digital input/output pins. They are called D0 to D13 respectively. See Illustration 5 for the pin mapping. However, each pin can be either input or output at a time (specifying by the ***pinMode*** function). There are three values that can be specified as a parameter for the pinMode function: ***INPUT, OUTPUT, INPUT\_PULLUP*** (case sensitive). In this lab, you will only use the OUTPUT mode with the LED. You will get more opportunities to explore the other two input modes as well in later labs,.

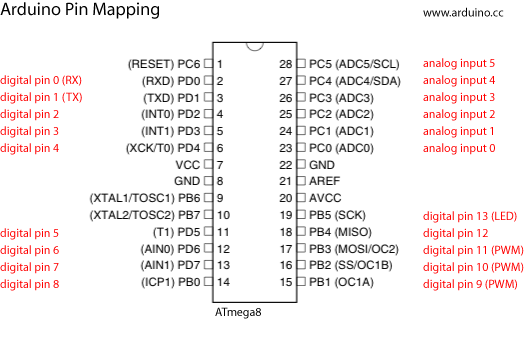


Illustration 5 : Arduino Pin Mapping (picture taken from http://www.arduino.cc)

Arduino provides two constant values for the two digital logic (True/False). They are ***HIGH*** and ***LOW*** (case sensitive). Like others constant values provided by the system, they are all case sensitive.

Here are three functions that is required for managing the digital Output of the Arduino: pinMode and digitalWrite. Here are the references taken from the Arduino site. Please note that these functions do not return any values.

## pinMode()

#### Description

Configures the specified pin to behave either as an input or an output. See the description of [digital pins](http://arduino.cc/en/Tutorial/DigitalPins) for details on the functionality of the pins.

As of Arduino 1.0.1, it is possible to enable the internal pullup resistors with the mode INPUT\_PULLUP. Additionally, the INPUT mode explicitly disables the internal pullups.

#### Syntax

pinMode(pin, mode)

#### Parameters

pin: the number of the pin whose mode you wish to set

mode: [INPUT](http://arduino.cc/en/Reference/Constants), [OUTPUT](http://arduino.cc/en/Reference/Constants), or [INPUT\_PULLUP](http://arduino.cc/en/Reference/Constants). (see the [digital pins](http://arduino.cc/en/Tutorial/DigitalPins) page for a more complete description of the functionality.)

#### Returns

None

## digitalWrite()

#### Description

Write a [HIGH](http://arduino.cc/en/Reference/Constants) or a [LOW](http://arduino.cc/en/Reference/Constants) value to a digital pin.

If the pin has been configured as an OUTPUT with [pinMode](http://arduino.cc/en/Reference/PinMode)(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

If the pin is configured as an INPUT, writing a HIGH value with digitalWrite() will enable an internal 20K pullup resistor (see the [tutorial on digital pins](http://arduino.cc/en/Tutorial/DigitalPins)). Writing LOW will disable the pullup. The pullup resistor is enough to light an LED dimly, so if LEDs appear to work, but very dimly, this is a likely cause. The remedy is to set the pin to an output with the pinMode() function.

NOTE: Digital pin 13 is harder to use as a digital input than the other digital pins because it has an LED and resistor attached to it that's soldered to the board on most boards. If you enable its internal 20k pull-up resistor, it will hang at around 1.7 V instead of the expected 5V because the onboard LED and series resistor pull the voltage level down, meaning it always returns LOW. If you must use pin 13 as a digital input, use an external pull down resistor.

#### Syntax

digitalWrite(pin, value)

#### Parameters

pin: the pin number

value: [HIGH](http://arduino.cc/en/Reference/Constants) or [LOW](http://arduino.cc/en/Reference/Constants)

#### Returns

none

## Connecting LED to the M5Stack Development Kit

Illustration 6 show the side/top-view locations of the digital I/O pins in the M5Stack Development kit.

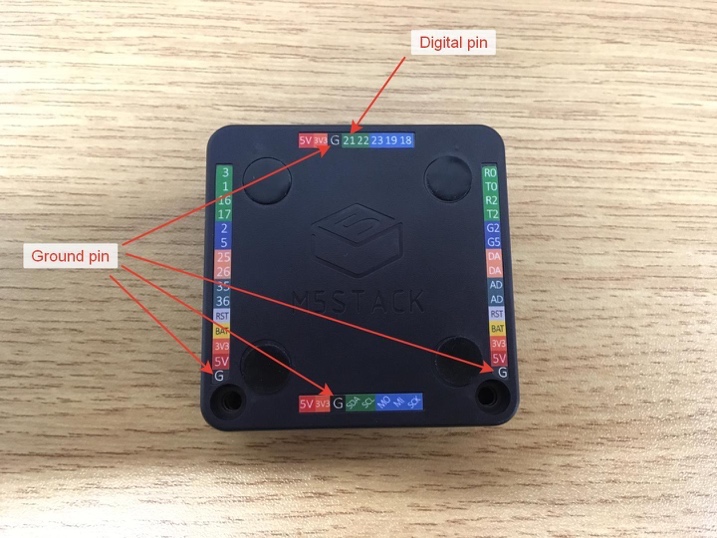


Illustration 6: Side/top-view locations of the digital I/O pins in the M5Stack development kit

You will be using the digital PIN 21 and the Ground pin PIN GD next to it to connect to the LED. In this lab, we will provide you with a small kit with 8 LEDs that are prewired with resistors.

A close up of a device

Description automatically generated

Illustration 7: Kit with 8 LEDs prewired with resistors

## M5Stack Buttons

As previously mentioned, we only use LED to connect to an output digital port, because we will use a built-in input in this lab: the three buttons on the M5Stack development kit to control the LED. When you look at the kit with the LCD screen on top of the three buttons, the three buttons: A, B, C, from left to right, can be accessed in a sketch using the variable M5.BtnA, M5.BtnB, M5.BtnC, respectively. In addition, we can detect the status of each button in the sketch using the following functions:

* M5.BtnA/B/C.read(): returns the state of the button, True==pressed and

False==released.

* M5.BtnA/B/C.isPressed(), M5.BtnA/B/C.isReleased(): check the button

state when it was last read, and return False or True

accordingly. They do not cause the button to be read

* M5.BtnA/B/C.wasPressed(), M5.BtnA/B/C.wasReleased(): check the button

state to see if it changed between the last two reads

and return False and True accordingly. They do not

cause the button to be read

Reference: <https://github.com/m5stack/M5Stack/blob/master/src/utility/Button.cpp>

Please note that with the use of the buttons, you want to add the function call M5.update() to the end of your loop() function as well.

## Serial communication

Serial communication is a method for using a sequence of bits to transfer a byte of data. In the digital system, we use 8 bits to represent a byte of data. A byte is generally equal to a character. Thus, there are (theoretically) 256 possible characters in a byte. One standard that provides association between a value and a physical character is ASCII (see next section for more details).

With a serial communication, the data transfer between two devices can be done with just two wires (one for sending and the other one for receiving). This kind of communication is cheaper comparing to parallel communication where 8 wires are needed to transmit a byte of data. Thus, to send a byte of data from one device to another using serial communication, we have to send (at least) 8 times (one for each bit). With no clock nor control signal involved in the communication, the key to the success of serial communication is timing. Both devices have to be configured for the same clock speed (specified as bit per second or bps) and the same protocol in order to observe the right data at the right time. This is the basic of network communication that we used today.

## 

## ASCII

ASCII is short for the American Standard Code for Information Interchange. It is a character-encoding scheme based on English alphabets, numbers and punctuation marks. It is nowadays used by several digital devices as a way to represent codes for characters. Thus, when type a character 'A', the digital computer system simply records an ASCII code to the memory.

## Serial communication in Arduino

Arduino has already supported serial communication with the serial library. To use it, a programmer has to first specify a baud rate (bit per second). Note that both devices have to be configured for the same baud rate. For example, to specify the baud rate of 9600 bps, use function Serial.begin(9600). To send data from Arduino to the computer, you could use three functions: Serial.write(byte): for sending a byte of data; Serial.print(data, [format such as BIN, OCT, DEC, HEX])**:** to print a data; Serial.println(data, [format such as BIN, OCT, DEC, HEX]): to print a data with new line at the end. The print and println functions are similar to those of the Java programming language. To read data from Arduino, a programmer may use if (Serial.available() > 0) { } to see if there is a data sending from the host. Once some data exists, use the Serial.read() to read a byte from the serial. The following example is a “Hello, World” sketch for serial communication in Arduino.

#include <M5Stack.h>

void setup() {

M5.begin();

Serial.begin(9600);

Serial.println(" ");

Serial.println("Hello World");

M5.update();

}

void loop() {

if (Serial.available()>0) {

int readByte=Serial.read();

Serial.print('x');

}

M5.update();

}

## Arduino Control Structures (Decisions)

## Arduino is based on the C/C++ programming language. Thus, it shares the same control structures used by C/C++ and those used by Java (and several programming languages). We provide a few examples here to ease your understanding.

### 'if'

Thinking of 'if' as an action to perform when the test expression is true.

Example

if (someVariable > 10)

{

// do something here

}

### 'if/else'

The 'if/else' allows greater control over 'if' by grouping multiple tests together.

Example

if (someVariable > 10)

{

// do something here

}

else if (someVariable >5)

{

// do something here

}

else

{

// do something here

}

### 'for'

The 'for' is useful for repeating a block of statements. In this example, the system will repeat the println function for 100 times (0 to 99).

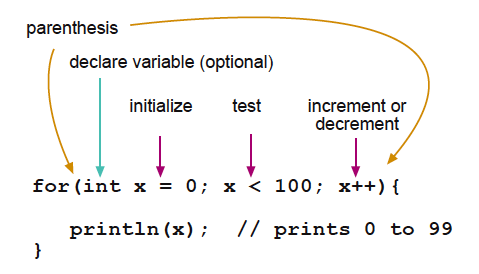


Illustration 11: Description of for loop (picture taken from http://www.arduino.cc)

### 'while'

The 'while' loop will run indefinitely as long as the test expression is true.

Example

int a=0;

while (a<100)

{

a++;

}

while (true)

{

// this is an infinite loop

}

### 'do .. while'

The 'do .. while' loop is similar to the while loop. However, the test expression is tested at the end of the loop. This means the loop will always run at least once.

Example

int a=0;

do

{

a++;

} while (a<100);

## Lab Exercises

In this laboratory, you have to:

* Connect the LED and its resistor to PIN 21 like in the Laboratory 2

and create sketches that use serial communication to send/receive data from the main Arduino tool.

**Task 1**

1. Connect your M5Stack to the computer using the USB cable, open the Arduino IDE, select M5Stack as your board by going to the menu ***Tools > Boards > M5Stack-Core-ESP32*** and choose the appropriate COM port that the computer connects to M5Stack by going to the menu ***Tools > PORT*** and select the right COM port. You can see the right COM port by inspecting Ports in the device manager.
2. Connect the LED kit in Illustration 7 to the M5Stack development kit according to the LED2 configuration in Illustration 4.
3. Create a new Arduino sketch as follows.

#include <M5Stack.h>

int ledPin = 21;

int ledState = LOW;

void setup(){

M5.begin();

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, ledState);

M5.update();

}

void loop() {  
 if (M5.BtnA.wasPressed()){

ledState = !ledState;

digitalWrite(ledPin, ledState);

}

M5.update();

}

Verify with the command button . Once there is no error, upload the sketch

to Arduino with the command button .

1. Modify the code so that the LED state is toggled when the Button B and C is pressed instead, respectively.

**Task 2**

1. Create a sketch that use serial communication to send from the M5Stack development kit.   
   In your sketch, read data from Button A (leftmost one when standing the M5Stack kit up with buttons below the LCD screen) and send the result to the serial output to display.

#include <M5Stack.h>

void setup() {

M5.begin();

Serial.begin(9600);

Serial.println(" ");

M5.update();

}

void loop() {

Serial.print("The button is ");

if (M5.BtnA.isPressed()) {

Serial.println("Pressed");

} else {

Serial.println("Released");

}

M5.update();

}

Change the isPressed() function to wasPressed(). Observer the difference

and answer the question below.

1. Modify the sketch in Exercise 1 to print the word “click” when Button A is clicked. Also print the number of clicks so far. To ease understanding, the pseudocode is given.

LET count=0;

SETUP:

Serial.begin(9600);

LOOP:

if (clicked)

count=count+1;

Serial.print("click ");

Serial.println(count);

end

1. Create a new sketch (by modifying the following program) that will read data from serial communication port with the read command. For each byte read, print the binary value, the hexadecimal value, and the decimal value to the screen. You may use this sketch to answer some following question as well.

void setup() {

Serial.begin(9600);

Serial.println("Exercise 3 : Ascii Converter");

}

int readData=0;

void loop() {

if (Serial.available()>0) {

readData = Serial.read();

Serial.print("READ '");

Serial.write(readData);

Serial.print("' - ");

Serial.print(" HEX: 0x");

Serial.print(readData,HEX);

Serial.print(", BIN: ");

Serial.print(readData,BIN);

Serial.println(" .. done");

}

}

If you got it right, the output should look like this.

Exercise 3: Ascii Converter

READ '1' - DEC: 49, HEX: 0x31, BIN: 110001 .. done

READ '2' - DEC: 50, HEX: 0x32, BIN: 110010 .. done

1. Create a new sketch that will read a value from the serial communication port. If a value is between '0' and '9', light up the LED, otherwise turn it off. For example, the following example should light up the LED.

Exercise 4: Enter a number to light up the LED

Please input a number.

> 4

1. Use the knowledge from previous laboratories, exercises and the switch control structure to create a sketch that will read a value from the serial communication port and do the associated works as stated by the following pseudocode. When display the name(s) and student id(s) to the serial port, please also light up the LED.

SETUP:

Serial.begin(9600);

LOOP:

Read key from the serial port

switch (key)

'1' : display name and student id of the first member of your

group to the serial port and light up the LED

'2' : display name and student id of the second member of your

group to serial port and light up the LED

'3' : display both names and student ids of your group to the

serial port and light up the LED

default: display ‘-’ to the serial port and turn the LED off

Hint: you may modify the following program to your benefits.

#include <M5Stack.h>

void setup() {

M5.begin();

Serial.begin(9600);

menu();

}

int readData=0;

void loop() {

if (Serial.available()>0) {

readData = Serial.read();

Serial.write(readData);

Serial.println("");

switch (readData) {

case '1' : student1(); break;

case '2' : student2(); break;

default : turnLEDOff();

}

menu();

}

}

void menu() {

Serial.println("Exercise 5 : Multi-functions program");

Serial.println("\nPlease select your function.");

Serial.println(" 1 - First member ");

Serial.println(" 2 - Second member ");

Serial.print("\n> ");

}

void turnLEDOff() {

Serial.println(" - ");

/\* turn LED off \*/

}

void student1() {

Serial.println("1. ID: xxx1 Mr. First Last");

/\* turn LED on \*/

}

void student2() {

Serial.println("2. ID: xxx2 Mr. SecondFirst SecondLast");

/\* turn LED on \*/

}

Section: \_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_

**Members**

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Student ID: \_\_\_\_\_\_\_\_\_\_\_\_

**2190151 Computer Programming Laboratory.**

**Laboratory 2:**

**Task 1**

1. LED toggling

Graded by ….................................................

**Task 2**

1. BASIC serial communication output (print and println)

Graded by ….................................................

1. Clicks with serial write

Graded by ….................................................

1. Serial read to ASCII values

Graded by ….................................................

1. Serial Read to LED

Graded by ….................................................

1. Multi-function program

Graded by ….................................................

1. Explain the difference between the two M5Stack functions related to buttons: isPressed() , wasPressed().
2. Complete the following function to wait for a key from the serial port. For each pressed on a key '!', print your name and student id to the serial console. (Assuming that port is all set up.)

void myInfo() {  
 if (Serial.available()>0) {  
 int key=Serial.read();

}

1. Use the sketch in Exercise 3 to complete the following table.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Character** | **Ascii (Decimal)** | **Ascii (Hex)** | **Character** | **Ascii (Decimal)** | **Ascii (Hex)** | **Character** | **Ascii (Decimal)** | **Ascii (Hex)** | | **0** |  |  | **A** |  |  | **(** |  |  | | **1** |  |  | **B** |  |  | **)** |  |  | | **2** |  |  | **C** |  |  | **@** |  |  | | **3** |  |  | **a** |  |  | **!** |  |  | | **4** |  |  | **b** |  |  | **(space)** |  |  | | **5** |  |  | **c** |  |  | **+** |  |  | | **6** |  |  | **Y** |  |  | **-** |  |  | | **7** |  |  | **Z** |  |  | **\*** |  |  | | **8** |  |  | **x** |  |  | **/** |  |  | | **9** |  |  | **z** |  |  | **=** |  |  | |

1. What is ASCII? What is it good for? Please explain.