Pre-lab 12: Introduction to Arduino

Digital circuits transfer information not as voltage levels or frequencies, but as binary levels. Today we will use a microcontroller, which can be thought of as a tiny computer. We will focus on using the inputs and outputs of a microcontroller to interface with circuits, and learn the basics of coding instructions to an Arduino microcontroller.

Glossary of terms

Binary Base-2 counting system used to transfer information digitally. One "bit" of information is a single digit (0 or 1). One "byte" of information is 8 bits, which can represent 2^8 integer values between 0 and $2^8 - 1 = 255$.

Logic levels The voltages that encode binary 0 and 1. Different systems use different logic levels; your Arduino runs on 3.3 V logic. This means that logic LOW is 0 V and logic HIGH is 3.3 V, though there is a "cushion" of values on either side (so voltages as low as 2.2 V will still be read as HIGH, and voltages as high as 1.1 V will be read as LOW).

Parallel and Serial Electrically, binary information can be transferred in parallel (a set of wires each with a place value in a binary number) or serially (zeros and ones are sent along fewer wires, at a known rate). There are many different protocols for serial communication, defining how the sender and receiver agree on several things: which party is sending and which is receiving, what rate the bits are sent at, etc.

UART (Universal Asynchronous Receiver/Transmitter) We will see several different serial protocols in the coming weeks. The first is the way the Arduino communicates with the computer, which is called UART. This protocol is "asynchronous," meaning there is not a wire sending a dedicated clock signal between the communicating devices.

Arduino MKR Zero A microcontroller board built around the Cortex-M0 32-bit SAMD21 chip. It can be programmed via a USB connection. It can send and receive information as digital or analog electrical signals through various input/output (I/O) pins. You can find the full pinout in the Datasheets folder on Canvas; an image is shown in Figure 1.



Figure 1: Arduino MKR Zero.

Programming terms

Today we will be writing programs (called "sketches," but you may be more familiar with the term "scripts") in the Arduino language, which is similar to C++. If you have programmed before in any language, refresh your memory on the concepts below. The important thing to remember is that a computer will do exactly what you tell it to do, but it interprets your commands in a very specific way. Our job is to be sure we know what we are telling it to do.

Variable A named container for a piece of data. For example, we might want the program to refer to a particular pin on the Arduino (say, pin 9) that has an external LED attached to it. We would create a variable named `extLEDPin` and assign it a value of 9. Thereafter, when the code is run, the name `extLEDPin` will be understood by the Arduino to mean 9.

Variable assignment To assign a value to a variable, use the symbol "=". To minimize confusion, pronounce this symbol as "gets". "`x=1`" is pronounced "x gets 1" and means that the variable `x` now has the value 1. This is to distinguish from "`x==1`," pronounced "x equals 1," which is a logical expression that evaluates to `True` if `x` is 1 and `False` if `x` is anything else.

Type The kind of value (integer, float [i.e. decimal number], character, etc) a variable refers to. In C++ (and consequently in Arduino), all variables must be declared to have a type, and can only accept values of that type. Thus, if we want to store an integer value given by a sensor, we must declare that a variable called `sensorValue` will take an integer value:

```
int sensorValue;
```

Later, we can assign an integer to that variable:

```
sensorValue = 1;
```

There is also a short-hand notation for doing both steps (declaration and assignment) at once:

```
int sensorValue = 1;
```

Array A data structure that holds a collection of values. To access a value within an array, you "index into" the array by using square brackets containing the index. If I declare and assign an array:

```
int myArray[] = {12,24,31,4};
```

I can later access the first value, 12, by writing `myArray[0]` (pronounced "myArray sub zero"), or the second value, 24, with `myArray[1]`. Note that the **index starts from 0** for the first element of the array.

Function Much like a mathematical function, a function is a programming construct that takes inputs and does something with them. It may or may not return outputs. The syntax, from the Arduino documentation, is shown in Figure 2.

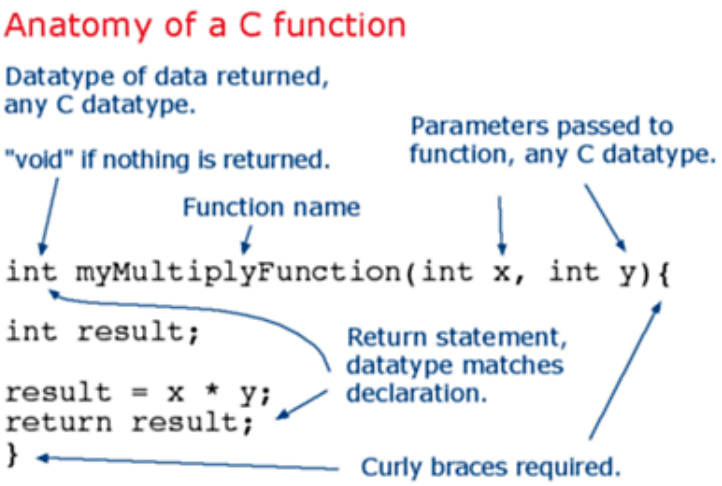


Figure 2: Anatomy of a function, from Arduino documentation.

Variable scope The portion of your code in which a variable is visible. Variables can be global or local.

Global variable A variable that is visible throughout the sketch. We declare these at the top of the sketch, before any functions, and use them for things like "which pin is the LED connected to," which is a value that may be needed anywhere in the sketch. Any function will be able to refer to these variables and change their values.

Local variable A variable that is only visible within a function. These are declared within a function. You cannot refer to them or change them from within other functions. The variables `x`, `y`, and `result` in the `myMultiplyFunction` example above are local variables.

This page is intentionally left blank

Pre-lab 12 Hand-in

Name: _____

Pre-lab 12.1 Download the Arduino Desktop IDE (integrated development environment) and install it on your laptop. Open the program and go to Tools > Board > Boards Manager. Search for and install the Arduino SAMD Boards (32-bits ARM Cortex-M0+) package, which we will need to run our MKR Zero. Then, open the Blink example from the Examples menu (File > Examples > 01.Basics). From the code, deduce what unit of time the `delay()` command takes as an argument (seconds? milliseconds? nanoseconds?).

Side note: Bring your laptop to class for the next few weeks.

Pre-lab 12.2 In the following sketch, which variables are global and which are local?

```
1   int extLEDPin = 3;
2
3   // the setup function runs once when you press reset or power the board
4   void setup() {
5     pinMode(extLEDPin, OUTPUT);
6   }
7
8   // the loop function runs over and over again forever
9   void loop() {
10    blinkLED(1000);
11  }
12
13  void blinkLED(int waitTime) {
14    digitalWrite(extLEDPin, HIGH);  // turn the LED on (HIGH is the voltage level)
15    delay(waitTime);                // wait for the specified time
16    digitalWrite(extLEDPin, LOW);   // turn the LED off by making the voltage LOW
17    delay(waitTime);
18  }
```

What does this sketch do?

From the last lab

Pre-lab 12.3 What is the input impedance of the integrator of Figure 2 in Lab 11?

Pre-lab 12.4 That might have been too easy. Now, remember the time when the output of the integrator saturated at $\pm 14\,\mathrm{V}$. What is the input impedance when that was happening?