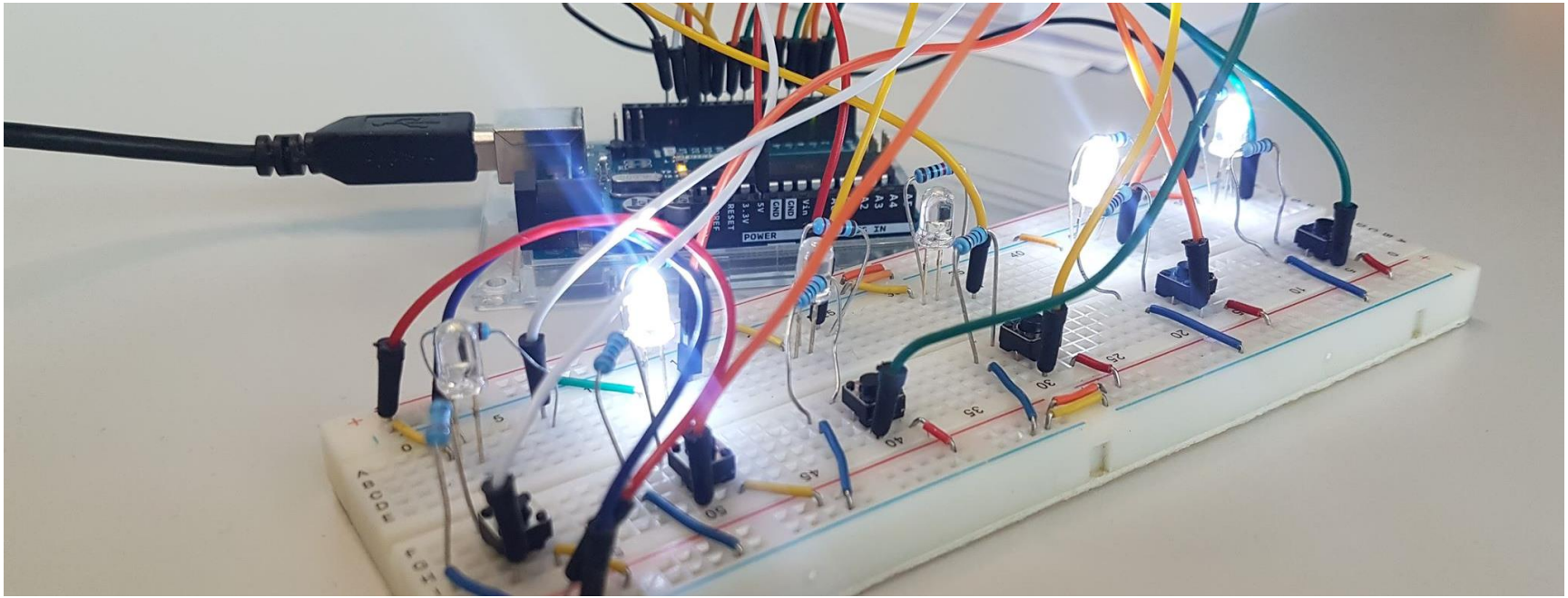
IAT 884 – Week 3 – Workshop 3

Alissa Antle and Annemiek Veldhuis (*ahv1@sfu.ca*)



Microcontrollers

Microcontrollers

Basics

An **I/O board** is a device that acts as a conduit between various electronic devices. I/O boards generally utilize a microprocessor to analyze and transmit data packets between the attached devices.

A **microprocessor** is a silicon chip that contains a CPU. Microprocessors control the logic of almost all digital devices, from clock radios to fuel-injection systems for automobiles.

A **microcontroller** is a low power consumption, self sufficient microprocessor. They typically integrate read/write memory, ROM memory, and EEPROM for permanent data storage.

Microcontrollers

Arduino

Arduino is a cheap, robust I/O board based on the ATmega328P chip.

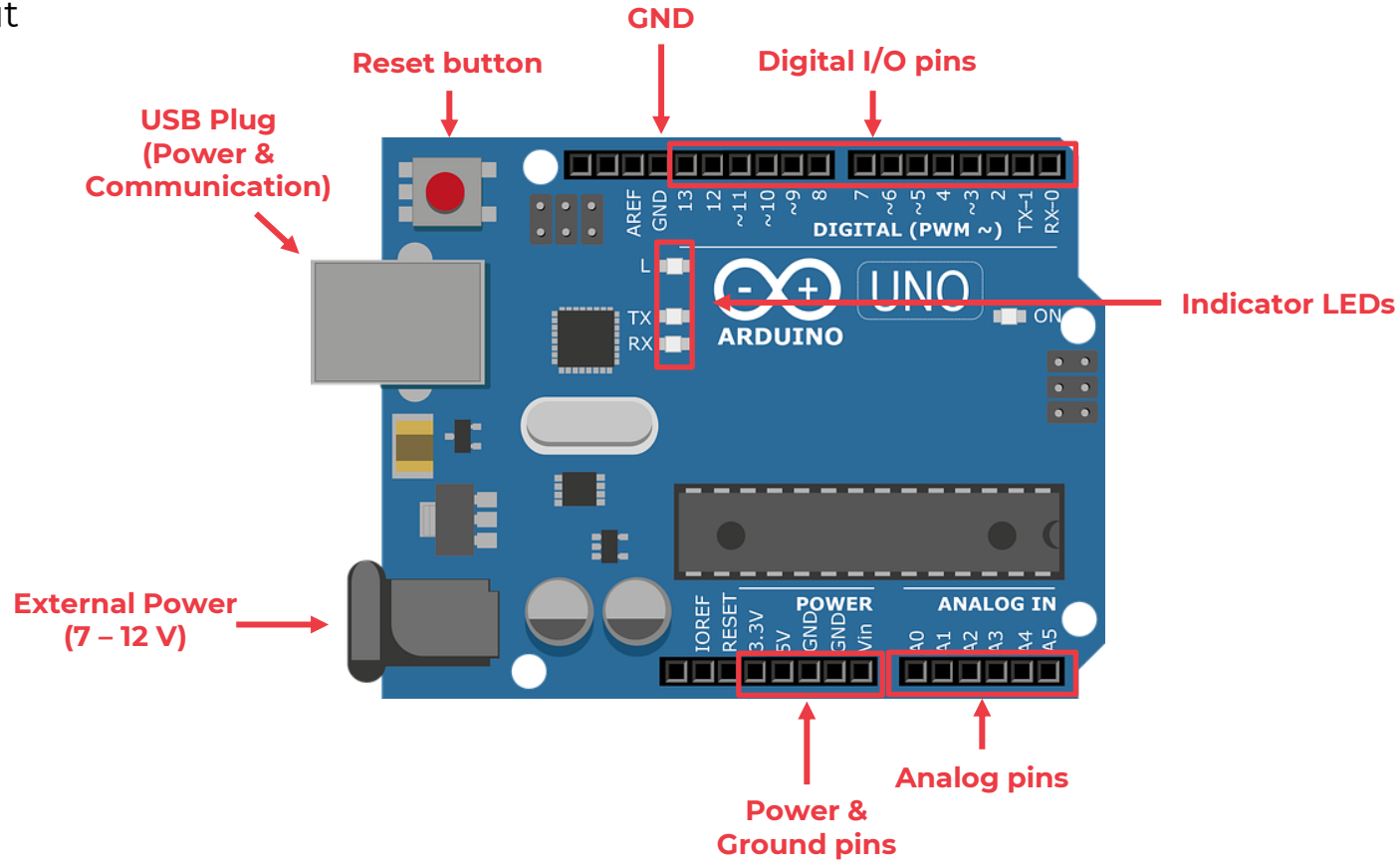
The Arduino board can function connected to a computer (in Serial communication mode) or as a stand-alone CPU that can drive a hardware application.

Arduino Uno Rev3 – Tech specs:

Microcontroller: Atmega 328P
Operating Voltage: 5V
Input Voltage (recommended): 7-12 V
Digital I/O Pins: 14 (of which 6 provide PWM output)
Analog Input Pins: 6
DC Current per I/O Pin: 20 mA
Flash Memory: 32KB (of which 0.5 KB used by bootloader)
Clock Speed: 16 MHz

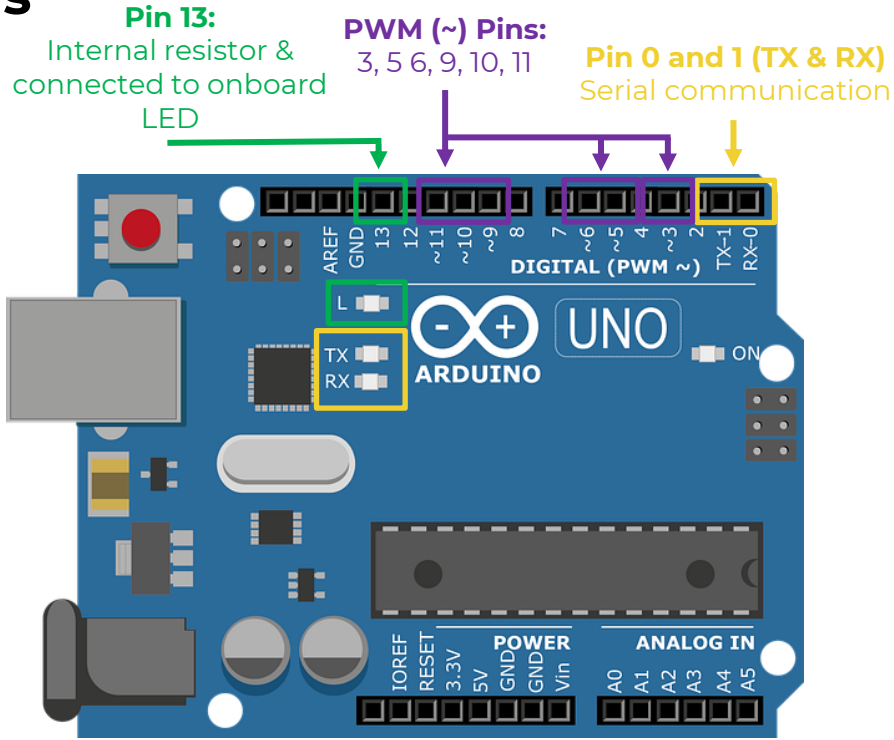
Microcontrollers

Pin-out



Microcontrollers

Pin-out



Microcontrollers

Output

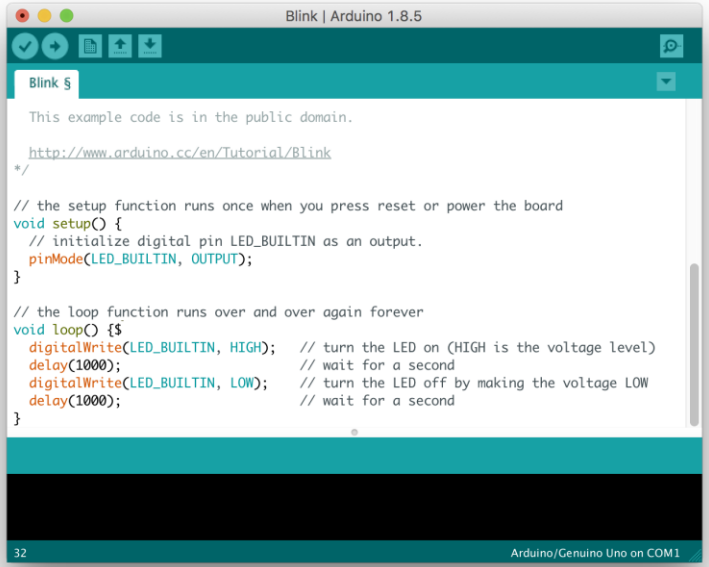
Output from the Arduino can replace the use of a battery for powering electronic circuits.

Unlike a battery which has a preset output, the Arduino can be **programmed** to send out a specific amount of voltage.

The Arduino can output from 0 – 5 V

Microcontrollers

Arduino IDE



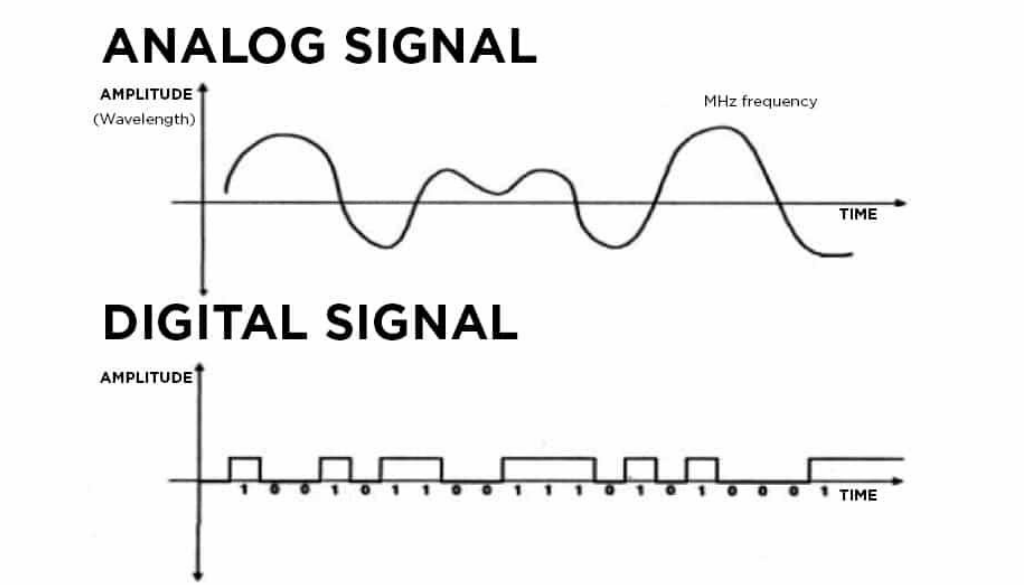
Digital & Analog Output

Digital & Analog

Differences

Analog allows for setting the amount of voltage as a specific value from 0 – 5 volts (continuous)

Digital works like a switch. It is either:
ON (sending voltage) or
OFF (not sending voltage)



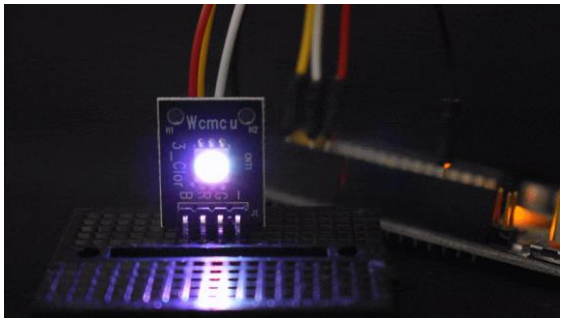
Digital & Analog

Information

This is an important distinction since it affects the way that information will be displayed to your user.

Digital output is good for alerts: Is something on? Is there danger?

Analog Output is good for conveying continuous and subtle information: How much? Volume, Speed of rotation, levels.



Digital & Analog

Digital Out

Binary is 0 or 1
For digital signals we use **high** or **low**.

High = 5v
Low = 0v

When you set a digital pin to high it will begin sending out 5 volts until it has its state changed to low.

When you set a digital pin to low it will stop sending voltage until it has its state changed to high.

There are 14 dedicated digital out pins on the Arduino Uno. You can additionally use the 6 analog in pins as digital output pins if necessary. These are referenced as digital pins 14-19.

Digital & Analog

Structure of an Arduino program

Global variables;

void setup(){

Setup environment variables;
Setup Pins for digital/analog input/output;

}

void loop(){

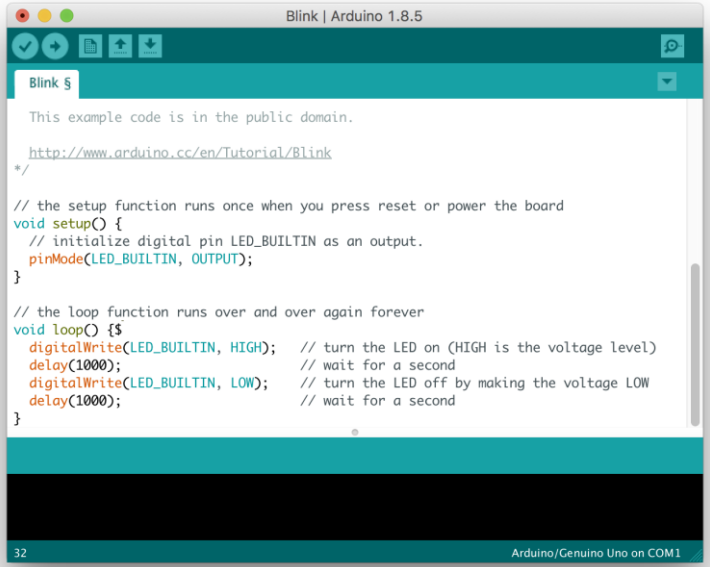
This is the stuff that creates the interaction;
Generally, you will either be checking the state of a sensor
Or changing the state of an attached device (LED, Motor, etc)
This is also where you call other functions;

}

void function(var 1, var 2){

Do something;

}



Digital & Analog

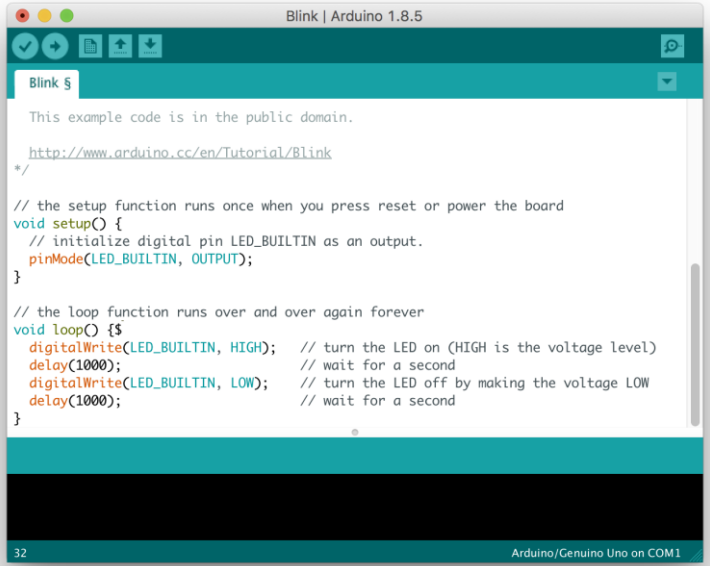
Digital output

```
int ledPin = 2;

void setup(){
                pinMode(ledPin, OUTPUT);
}

void loop(){
                digitalWrite(ledPin, HIGH);
                delay(1000);
                digitalWrite(ledPin. LOW);
                delay(1000);
}
```

Give a name to the pins you will be using



Digital & Analog

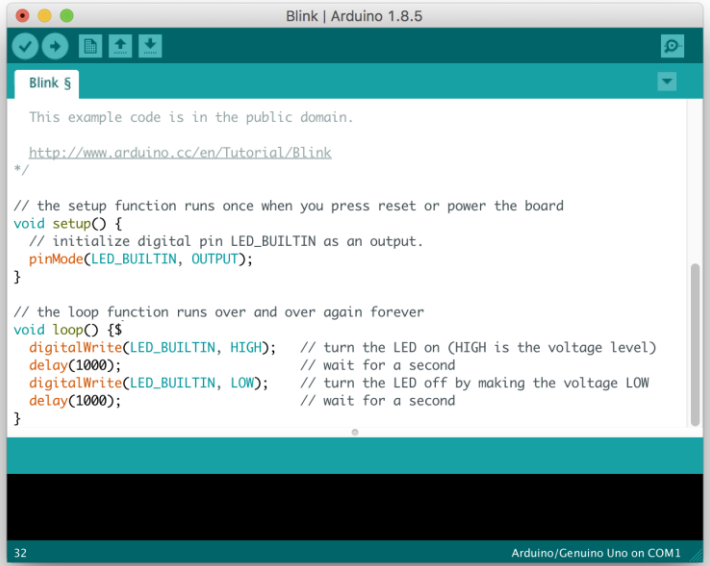
Digital output

```
int ledPin = 2;

void setup(){
                pinMode(ledPin, OUTPUT);
}

void loop(){
                digitalWrite(ledPin, HIGH);
                delay(1000);
                digitalWrite(ledPin. LOW);
                delay(1000);
}
```

Tell the program whether you will use this pin as an output or input pin



Digital & Analog

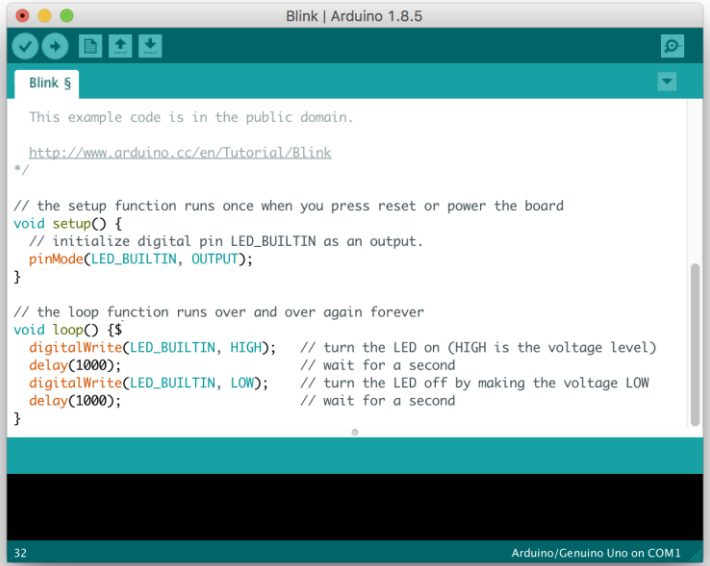
Digital output

```
int ledPin = 2;

void setup(){
                pinMode(ledPin, OUTPUT);
}

void loop(){
                digitalWrite(ledPin, HIGH);
                delay(1000);
                digitalWrite(ledPin. LOW);
                delay(1000);
}
```

The command digitalWrite allows you to change the amount of voltage being sent to the pin



Digital & Analog

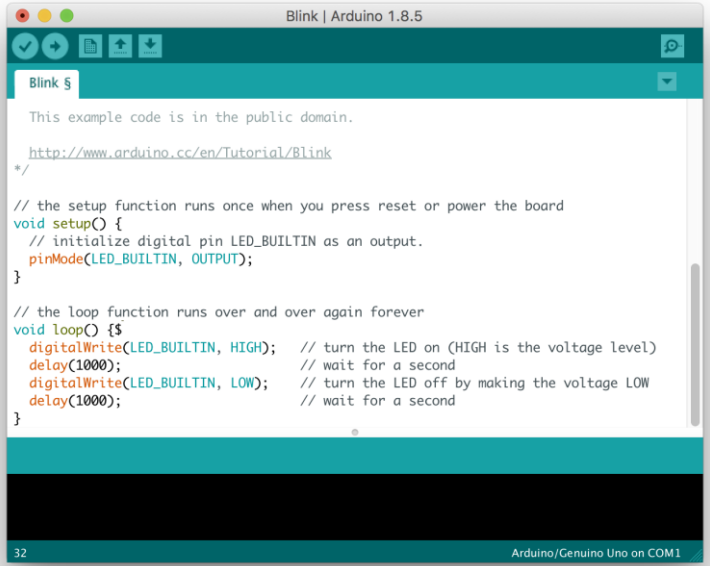
Digital output

```
int ledPin = 2;

void setup(){
                pinMode(ledPin, OUTPUT);
}

void loop(){
                digitalWrite(ledPin, HIGH);
                delay(1000);
                digitalWrite(ledPin. LOW);
                delay(1000);
}
```

HIGH = 5V
LOW = 0V



Digital & Analog

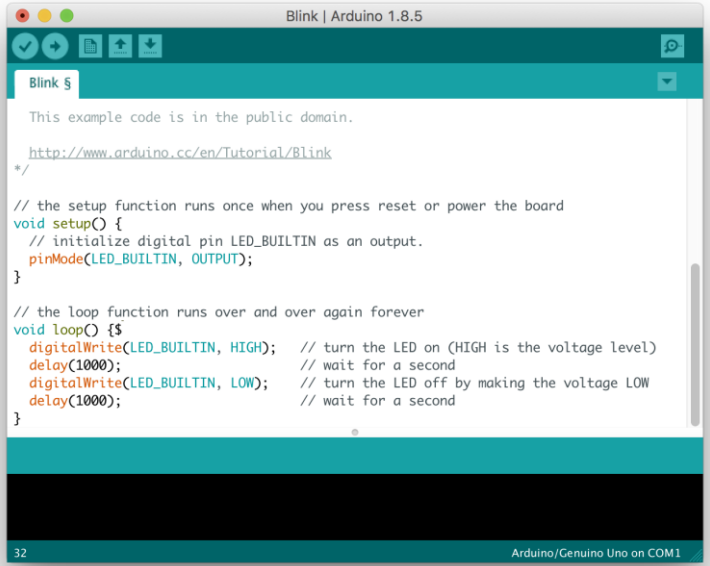
Digital output

```
int ledPin = 2;

void setup(){
                pinMode(ledPin, OUTPUT);
}

void loop(){
                digitalWrite(ledPin, HIGH);
                delay(1000);
                digitalWrite(ledPin. LOW);
                delay(1000);
}
```

Between turning the ledPin (pin 2) high and low, we wait 1000ms (1s)



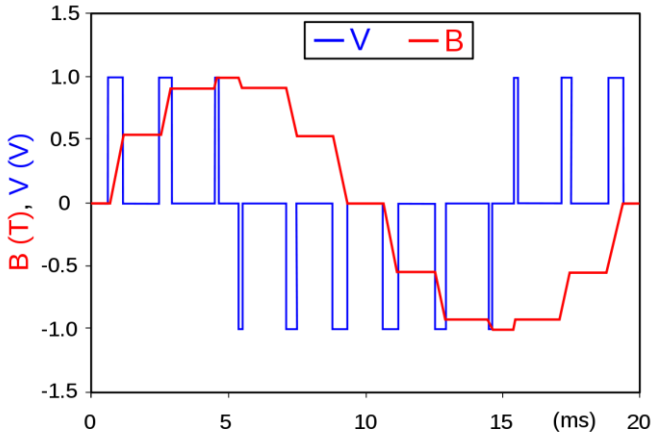
Digital & Analog

Analog output: PWM

The Arduino Analog Output pins **do not generate a true analog output** in the sense that the microcontroller does not have a resistive diver to create the voltage.

Instead, it uses **a digital PWM signal** that can be smoothed to create an average voltage, which does result in an “analog output”.

Pulse Width Modulation (PWM) modulates the duty cycle of a square wave to control the amount of power sent out to a load.



Digital & Analog

Analog output

You do not need to set PINMODE for analog output pins

analogWrite(pinNumber, value);

PinNumber must be one of the PWM pins (3, 5, 6, 9, 10, 11)
Value is between 0 - 255 (0v – 5v)
(The maximum value is 255 since the PWM register is 8 bits wide (2 in the power of 8 is 255)).

```
int ledPin = 9;

void setup(){
        //nothing needs to be here
}

void loop(){
        //send out ~2.5v on Pin 9
        analogWrite (ledPin, 150);
}
```

Digital & Analog

Analog output

```
int ledPin = 3;

void setup(){
        //nothing needs to be here
}

void loop(){
        for(int i = 0; i < 255; i++){
                analogWrite(ledPin, i);
                delay(10);
        }
}
```

Digital & Analog

Analog output

```
int ledPin = 3;
int val = 0;
boolean up = true;

void setup(){
}

void loop(){
                if(up){
                                val++;
                                if(val == 255){
                                                up = false;
                                }
                } else {
                                val--;
                                if(val == 0){
                                                up = true;
                                }
                }

                analogWrite(ledPin, val);
                delay(10);
}
```

Debugging

Debugging

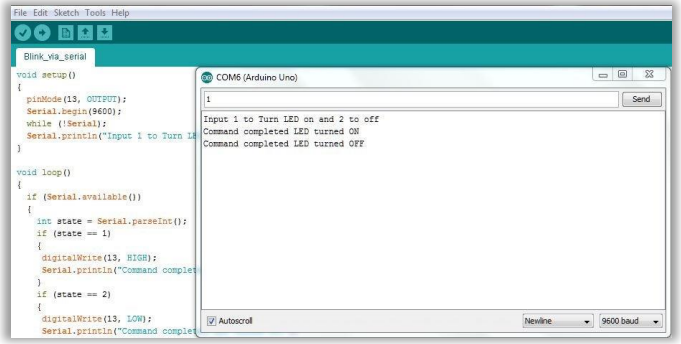
Serial monitor

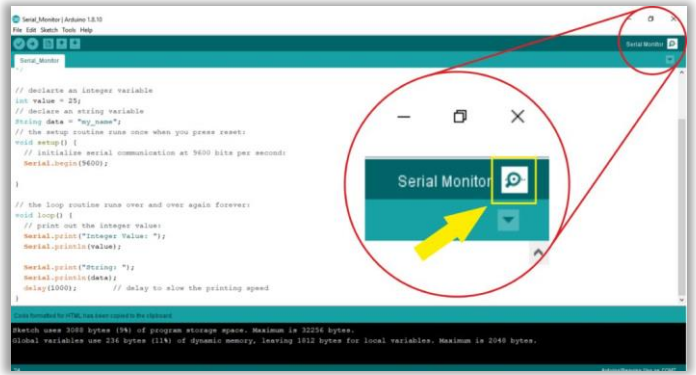
Serial.print() and **Serial.println()**

Send data out over the serial communication channel. This is the primary way to send characters and numbers out from the Arduino board.

Because serial communication is used, the data sent is available to any application capable of retrieving serial data communications. So, this also works as a way of sharing data between the Arduino board and Max/MSP, Processing, and other programming environments.

Serial.println() is the same as Serial.print() except it appends both a carriage return character(ASCII 13) and a newline character (ASCII 10).





Debugging

Serial monitor

Step 1: Initialize in **Setup**

```
// initialize serial communication at 9600 bits per second:
Serial.begin(9600);
```

Debugging

Serial monitor

Step 2: Print text and values in **Loop**

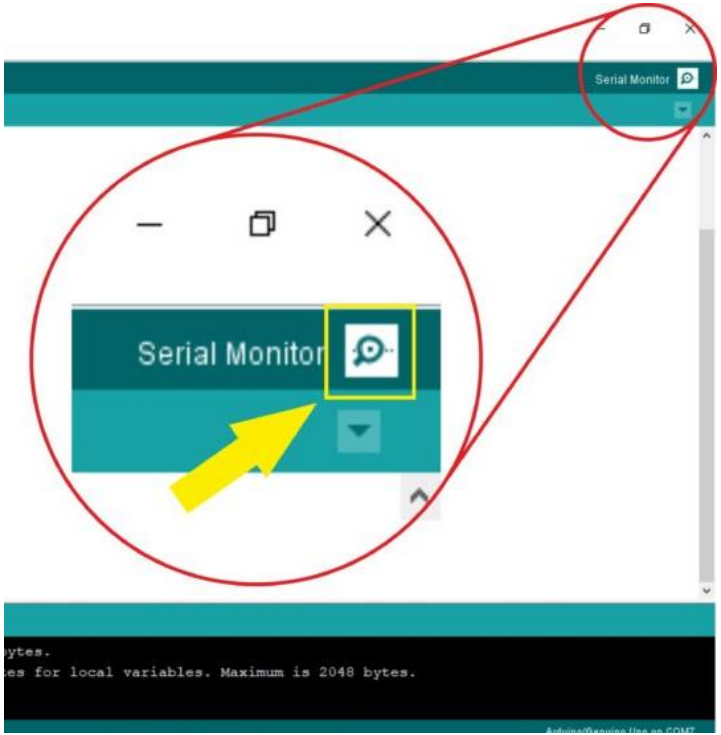
```
Serial.print("Integer Value: ");
Serial.println(value);
```

Debugging

Serial monitor

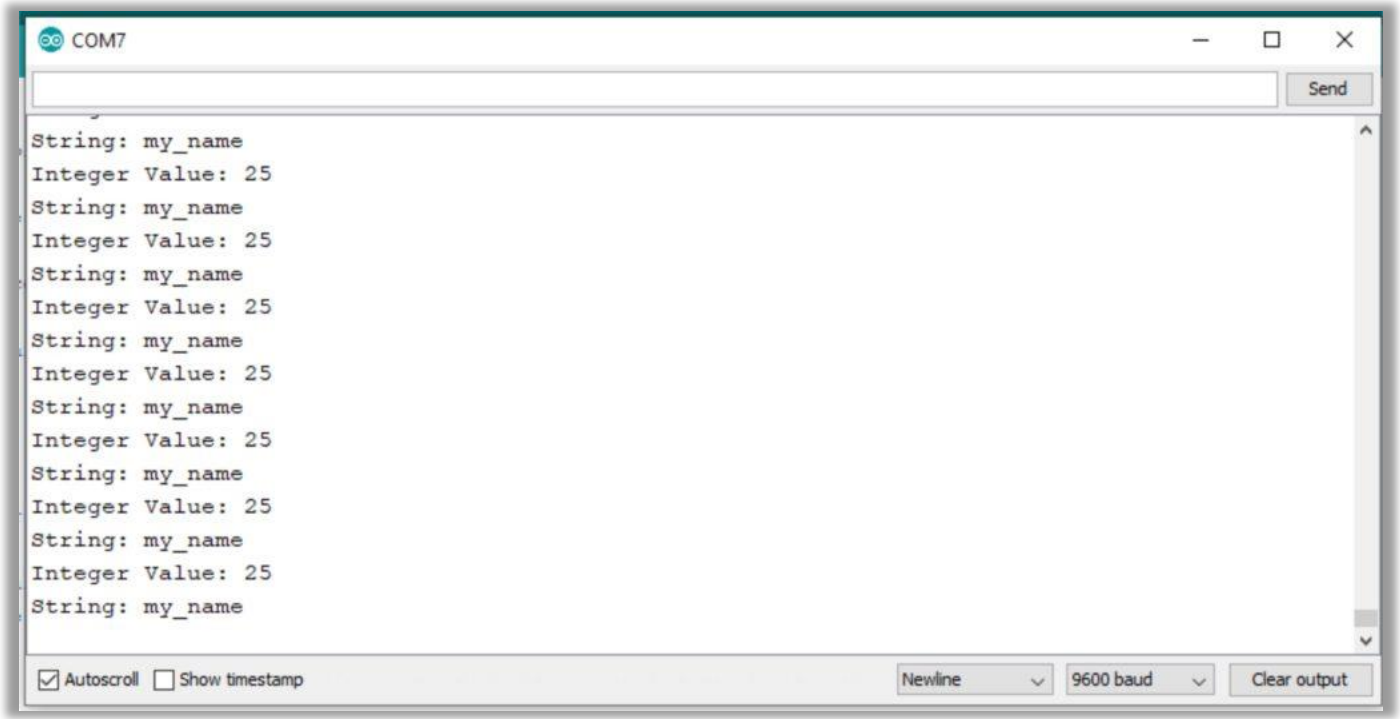
Step 3: Run code

Step 4: open the Serial Monitor



Debugging

Serial monitor



Debugging

```
int ledPin = 3;

void setup(){
        Serial.begin(9600);
}

void loop(){
        for(int i = 0; i < 255; i++){
                analogWrite(ledPin, i);
                Serial.println(i);
                delay(10);
        }
}
```

Output?

Exercises

Exercises

Download the Week 3 Handout on the wiki.

Get as far as possible with the exercises during the lecture time. Complete them at home if you can't finish.

During remote learning: create the circuits through Tinkercad circuits.

Send a document with your name, screenshots of your circuits, and answers to the questions to ***ahv1@sfu.ca***.
Deadline: Sunday 11.59pm