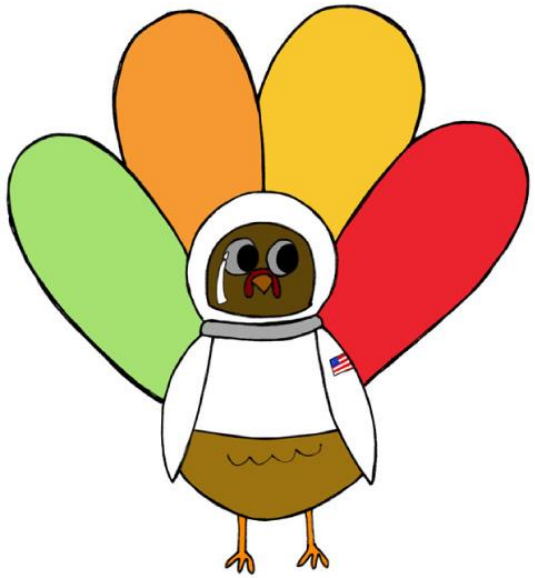
Measurements, Sensors and Data Logging Course

Week 3

Upcoming Weeks

- Office Hours
  - Monday Nov. 23rd @ 7:00 PM
  - Monday Nov. 30th @ 7:00 PM

- Weekly Session
  - ~~Thursday Nov. 26~~ Thanksgiving, no session
  - Thursday Dec. 3rd @ 7:00 PM



Review of Homework Activities

Button Activity 1

Turn OFF LED when Button is pressed

```
const byte buttonPin = 6; // pushbutton is on D6
const byte ledPin = 4; // LED is on D4

byte buttonValue = 0; // global variable for storing the value of the button

void setup()
{
  pinMode(buttonPin, INPUT); // initialize the button as an input
  pinMode(ledPin, OUTPUT); // initialize the LED as an output
}

void loop()
{
  // read state of the button and store it in variable buttonValue
  buttonValue = digitalRead(buttonPin);

  if(buttonValue == LOW)  // Check if button is not pressed
  {
    digitalWrite(ledPin, HIGH); // Turn ON LED
  }
  else  // if it is not pressed
  {
    digitalWrite(ledPin, LOW);  // Turn OFF LED
  }
}
```

Button Activity 2

Turn ON LED when Button is pressed without using a conditional

```
const byte buttonPin = 6; // pushbutton is on D6
const byte ledPin = 4; // LED is on D4

byte buttonValue = 0; // global variable for storing the value of the button

void setup()
{
  pinMode(buttonPin, INPUT); // initialize the button as an input
  pinMode(ledPin, OUTPUT); // initialize the LED as an output
}

void loop()
{
  // read state of the button and output it to the LED
  digitalWrite(ledPin, digitalRead(buttonPin));
}
```

Button Activity 3

Toggle the LED when Button is pressed

```
const int buttonPin = 6; // pushbutton is on D6
const int ledPin = 4; // LED is on D4
const int debounce = 25; // ms for debounce delay

bool buttonState = LOW; // initialize button state

void setup()
{
  pinMode(buttonPin, INPUT); // initialize the button as an input
  pinMode(ledPin, OUTPUT); // initialize the led as an output
}

void loop()
{
  if((buttonState == LOW) && (digitalRead(buttonPin) == HIGH)); // detect button rising edge
  {
    digitalWrite(ledPin, not(digitalRead(ledPin))); // toggle ledPin
    delay(debounce); // allow for some button contact bouncing without triggering
    buttonState = HIGH; // Set buttonstate high to avoid constant triggering
  }
  if((buttonState == HIGH) && (digitalRead(buttonPin) == LOW)) // detect button falling edge
  {
    delay(debounce); // Allow for some contact bounce on release
    buttonState = LOW; // reset button press trigger
  }
}
```

Pot Activity 1

LED PWM gets brighter with clockwise rotation of pot

```
const byte ledPin = 4;  // LED is on pin D4
const byte potPin = A0; // Potentiometer is on pin A0
const int analogHigh = 1023;  // maximum analogRead value
unsigned int potValue = 0;

void setup()
{
  pinMode(ledPin, OUTPUT); // set the ledPin to be an output
}

void loop()
{
  potValue = analogRead(potPin);  //read the potentiometer
  // Create the PWM signal
  digitalWrite(ledPin, HIGH);  // Write the LED pin high
  delayMicroseconds((analogHigh - potValue) * 16);  // wait for the remainder of
the period
  digitalWrite(ledPin, LOW); // Write the LED pin low
  delayMicroseconds(potValue * 16);  // delay by the pot value microseconds *16
}
```

Pot Activity 2

LED PWM uses the Light Sensor instead of the Pot

```
const byte ledPin = 4;  // LED is on pin D4
const byte lightPin = A6; // Light Sensor is on pin A6
const int analogHigh = 1023;  // maximum analogRead value
unsigned int lightValue = 0;

void setup()
{
  pinMode(ledPin, OUTPUT); // set the ledPin to be an output
}

void loop()
{
  lightValue = analogRead(lightPin);  //read the light sensor
  // Create the PWM signal
  digitalWrite(ledPin, HIGH);  // Write the LED pin high
  delayMicroseconds(lightValue * 16);  // delay by the lightValue microseconds *16
  digitalWrite(ledPin, LOW); // Write the LED pin low
  delayMicroseconds((analogHigh - lightValue) * 16);  // wait for the remainder of
the period
}
```

Lesson 6: Temp and Humidity Sensor

Using a Library to read a sensor

Temperature Sensor Introduction

Lesson 6: Temp and Humidity

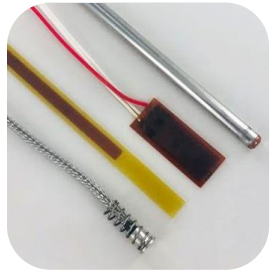
- Temperature Sensor: Measure temperature of object, substance, medium
- Where are they used: Everywhere! fluids, gases, thermostats, electronics, appliances
- How are they used: Depends.
  - For fluid, air and object temp measurement: sensor needs to be in the medium (fluid/air) or against the object being measured
  - For IR sensors: sensor remotely mounted, a signal reflects against surface of the object being measured

Temperature Sensor Introduction

Lesson 6: Temp and Humidity

- Sensor Types: 4 most common
  - Thermocouples: 2 dissimilar metals joined together, produce a voltage (Seebeck effect)
    - Very small voltage produced, need to use an amplifier to measure.
    - Many different “types”, ex: K-Type, J-Type

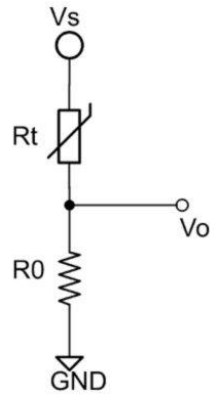
  - RTD (Resistance Temperature Detector): Resistance changes with temperature
    - Current drivers required to measure



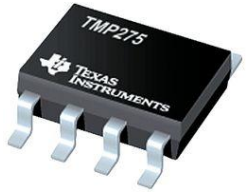
- More Information: https://www.digikey.com/en/blog/types-of-temperature-sensors#:~:text=There%20are%20four%20types%20of,based%20integrated%20circuits%20(IC

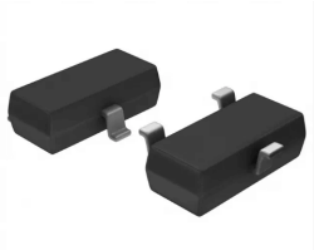
Temperature Sensor Introduction

Lesson 6: Temp and Humidity



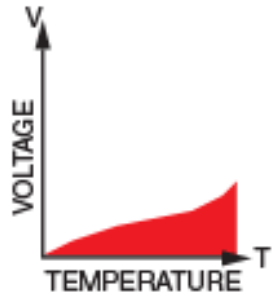
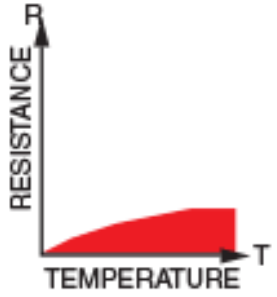
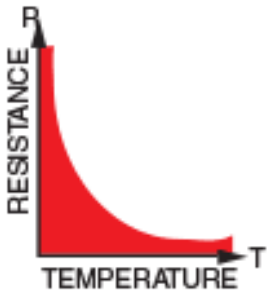
- Sensor Types: 4 most common
  - Thermistor: Resistance changes with temperature
    - Use voltage divider to measure
  - Semiconductor based ICs: measure the physical properties of a transistor





Temperature Sensor Introduction

Lesson 6: Temp and Humidity

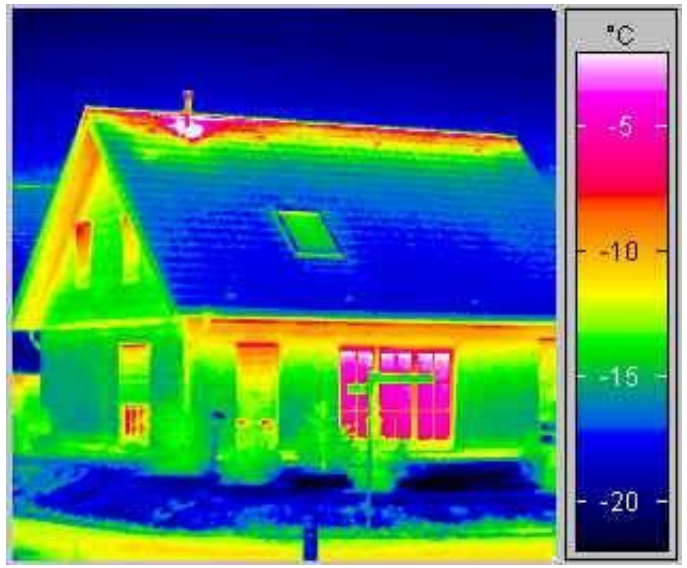
	Thermocouple	RTD	Thermistor	I. C. Sensor
				
Advantages	▪ Self-powered ▪ Simple ▪ Rugged ▪ Inexpensive ▪ Wide variety ▪ Wide temperature range	▪ Most stable ▪ Most accurate ▪ More linear than thermocouple	▪ High output ▪ Fast ▪ Two-wire ohms measurement	▪ Most linear ▪ Highest output ▪ Inexpensive
Disadvantages	▪ Non-linear ▪ Low voltage ▪ Reference required ▪ Least stable ▪ Least sensitive	▪ Expensive ▪ Current source required ▪ Small Δ R ▪ Low absolute resistance ▪ Self-heating	▪ Non-linear ▪ Limited temperature range ▪ Fragile ▪ Current source required ▪ Self-heating	▪ T<200°C ▪ Power supply required ▪ Slow ▪ Self-heating ▪ Limited configurations

Source: https://in.omega.com/prodinfo/integrated-circuit-sensors.html

Temperature Sensor Introduction

Lesson 6: Temp and Humidity

- IR Sensor: IC based temperature sensor
  - How it Works: Detect infrared radiation and convert it to a temperature
  - Thermal cameras use the same principle



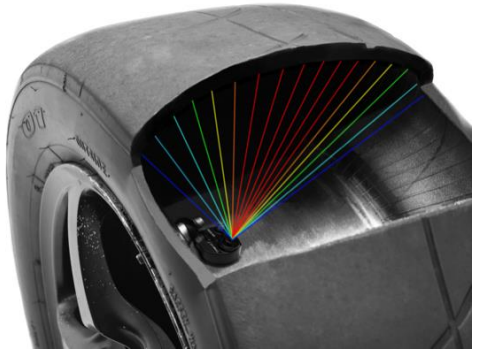
- More information: https://www.flir.com/discover/rd-science/temperature-guns-versus-thermal-imaging-technology/#:~:text=Both%20spot%20pyrometers%20and%20thermal,it%20into%20a%20temperature%20reading.&text=A%20spot%20pyrometer%20reads%20the,of%20the%20entire%20thermal%20image

Temperature Sensor Introduction

Lesson 6: Temp and Humidity

- Temperature sensors on a race car *(x47 – 131)*
  - **Fluid:** coolant *x2*, engine oil *x2*, gearbox oil *x2*, hydraulic fluid *x0-6*
  - **Gases:** ambient air, intake air *x2-4*, tire air *x4*, exhaust *x2-10*, cockpit + HVAC *(x1-5)*
  - **Object:** tire surface *x12-64*, brake rotor *x4*, brake caliper *x4*, track surface, clutch, force sensor correction *x0-12*
  - **Internal Electronics:** Device, Microcontroller and PCB *x10-12*
  - **Surface Temp Stickers + Paint:** Brakes, Electronics + More *x8-20*

Thermocouple Amplifier



Internal Tire Temp (IR)

Caliper Temp (sticker)

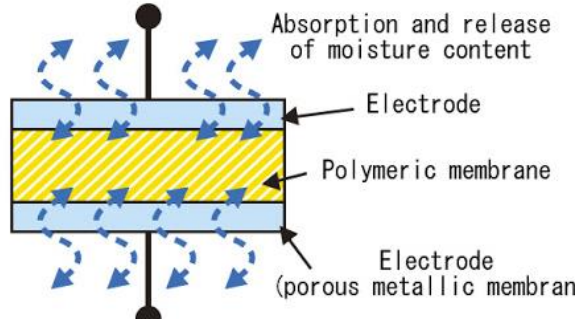
Rotor Temp (paint)

Track Surface Temp (IR)

Humidity Sensor Introduction

Lesson 6: Temp and Humidity

- Humidity Sensor: measures relative humidity (water vapor in the air)
  - To determine dew point and absolute humidity: combine relative humidity and temperature measurements
- Types
  - **Capacitive** (most common): A thin strip of non-conductive polymer film between two electrodes. The electrical capacity of the film changes with the atmosphere's relative humidity. Measure a change in the capacity of the film.
  - **Resistive:** Similar construction to the capacitive sensor. The resistance of the material between the electrodes changes with humidity.
  - **Thermal:** Two thermal sensors conduct electricity based upon the humidity of the surrounding air. One sensor is encased in dry nitrogen while the other measures ambient air. The difference between the two measures the humidity.



- Applications: HVAC, weather, environment

- http://blog.servoflo.com/humidity-sensors-capacitive-vs-resistive
- https://www.electronicsforu.com/tech-zone/electronics-components/humidity-sensor-basic-usage-parameter#:~:text=A%20humidity%20sensor%20(or%20hygrometer,both%20moisture%20and%20air%20temperature.&text=Humidity%20sensors%20work%20by%20detecting,Capacitive

Lesson 6 Hardware

Lesson 6: Temp and Humidity

- What hardware will we need for this Lesson?
  - Grove Temperature and Humidity Module on pin D3
  - Seeeduino Lotus (Arduino Uno compatible board)
    - The Arduino has the serial port hardware built into the device

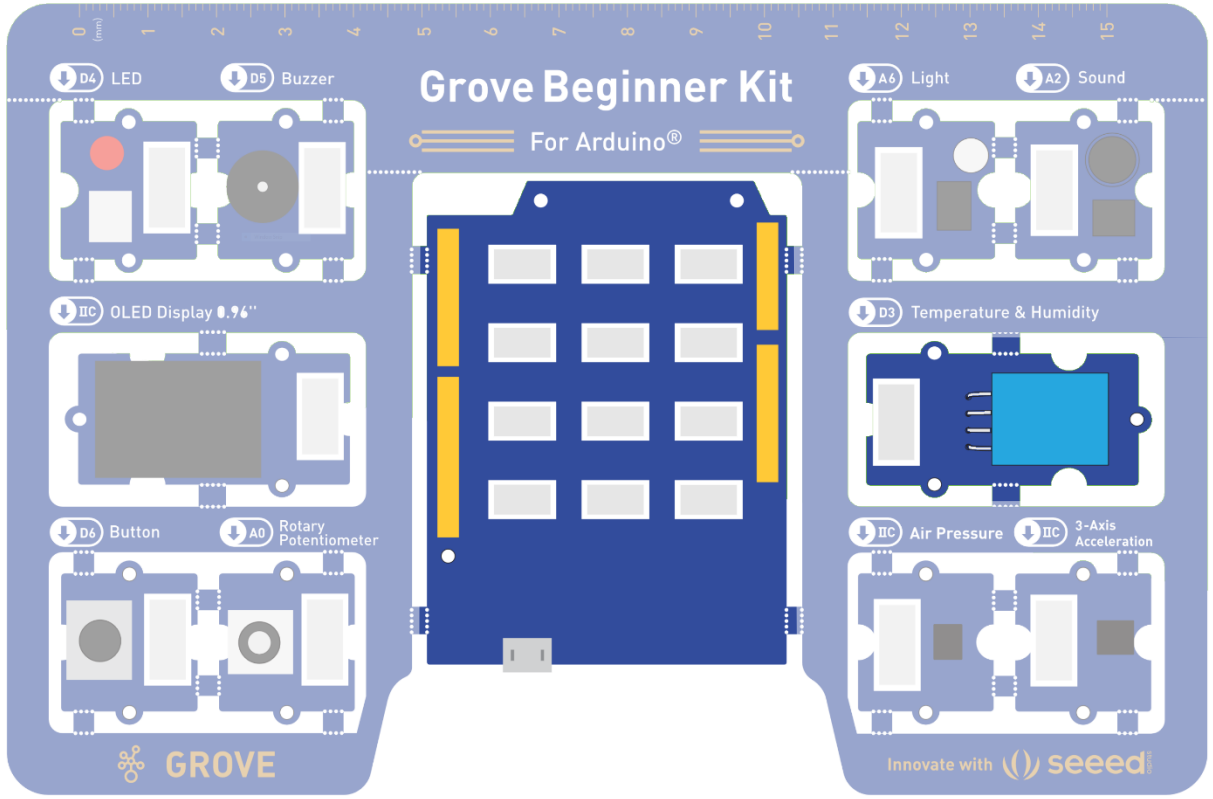
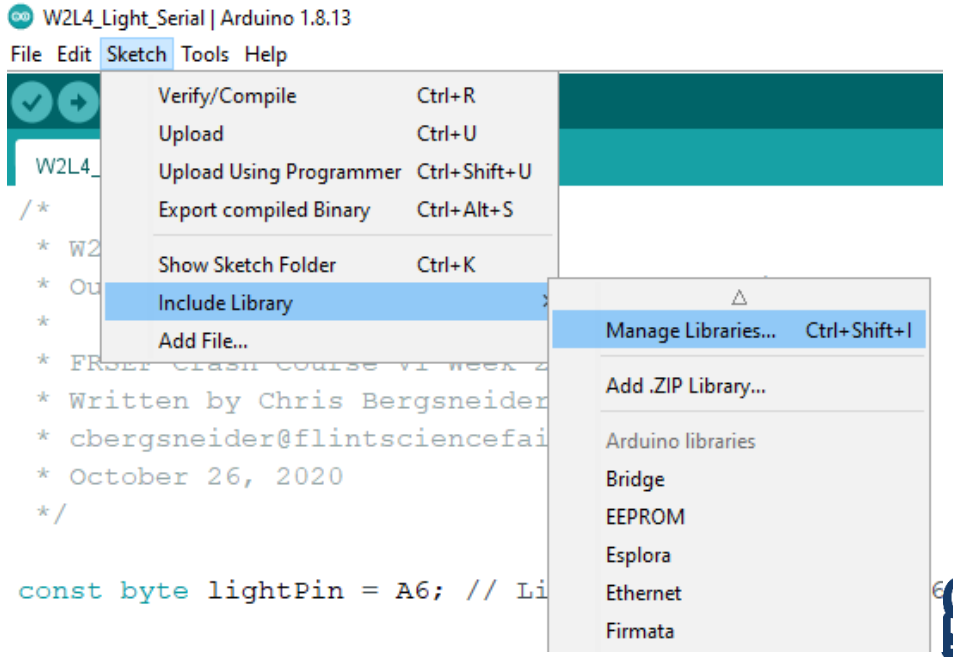


Image modified from https://files.seeedstudio.com/wiki/Grove-Beginner-Kit-For-Arduino/res/Grove-Beginner-Kit-For-ArduinoPDF.pdf

Libraries

Lesson 6: Temperature and Humidity

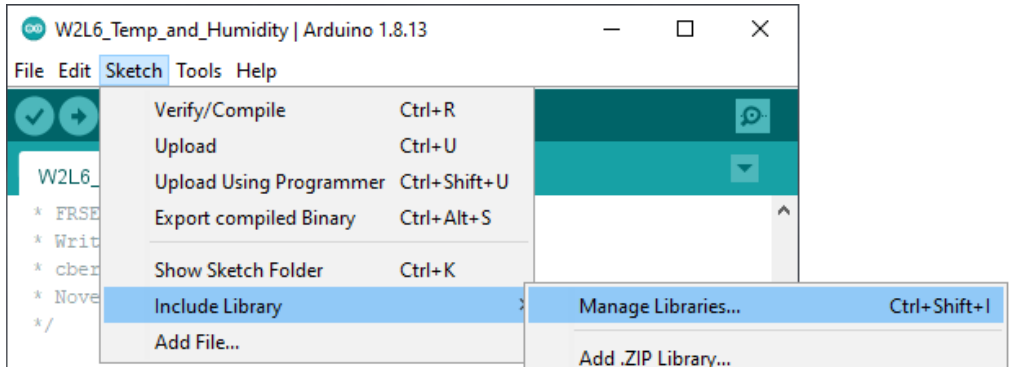
- Library in Arduino
  - What it is: Collecition of pre-defined code.
  - What is does: Provides extra functionalities for sketches, easily.
  - Why: Simplify our sketches, easily integrate components and functions (sensor, display, math, etc)
  - How: Arduino Library Manager
    - *Sketch/Include Library / Manage Libraries*

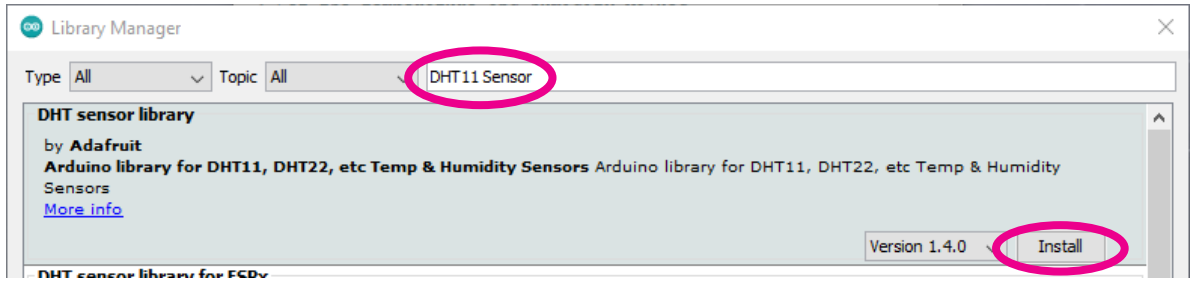


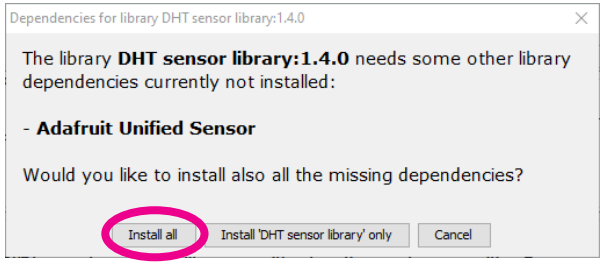
Installing a Library

Lesson 6: Temperature and Humidity

1. Open the Manage Libraries Dialog
   a. **Sketch → Include Library → Manage Libraries…**
2. Install DHT11 Library
   a. Search “**DHT11 Sensor**” in the search Box
   b. Click **Install** on **DHT sensor library** by Adafruit
   c. Click **Install all** to also install the Adafruit Unified Sensor library dependency
   d. Close out of the Library Manager







Code Analysis: Including a Library

Lesson 6: Temp and Humidity

```
#include <DHT.h>
```

  - Make the classes, methods and functions in the installed DTH library available to our sketch
- The `#include` preprocessor directive is used to include a library into your sketch.
- Libraries offer access to a large pool of functions and capabilities written by others and offered to you through the open source community
- Syntax:

```
#include <lib.h>
```

  - Installed library include. These libraries are located in the installed library folder, but the must be installed first

```
#include "lib.h"
```

  - Local library include. These libraries are searched for in the same folder as your sketch
- More Infromation:
  - https://www.arduino.cc/reference/en/language/structure/further-syntax/include/

Open and Upload Sketch

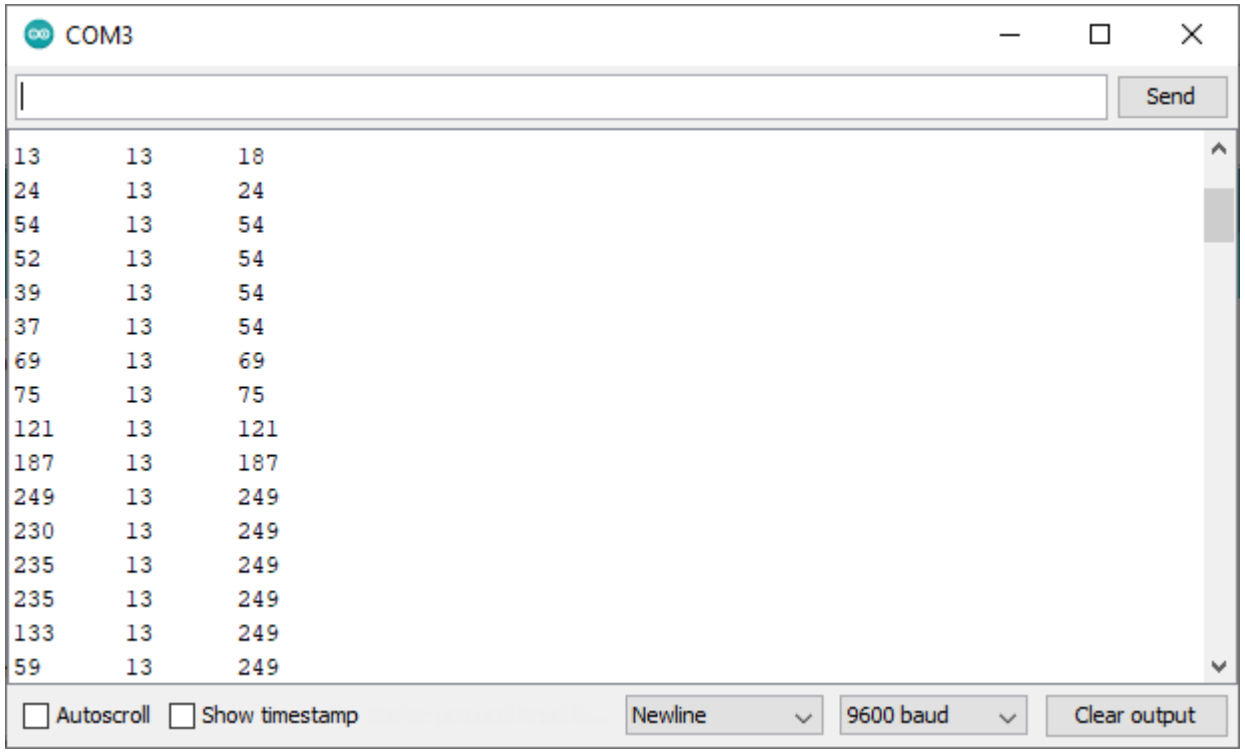
Lesson 6: Temp and Humidity

1. Open Temp and Humidity Sketch
   a. **File → Sketchbook → FRSEF_Crash_Course → Week_3 → W3L6_Temp_and_Humidity.ino**
2. Upload the sketch to your Arduino by clicking the Upload Button.
   a. The sketch should compile, and then upload to your Arduino.
3. Open the serial monitor.
   a. **Tools → Serial Plotter** (Ctrl+Shift+L)
4. Observe the output in the Serial Plotter

Serial Monitor

Lesson 6: Temp and Humidity Sensor

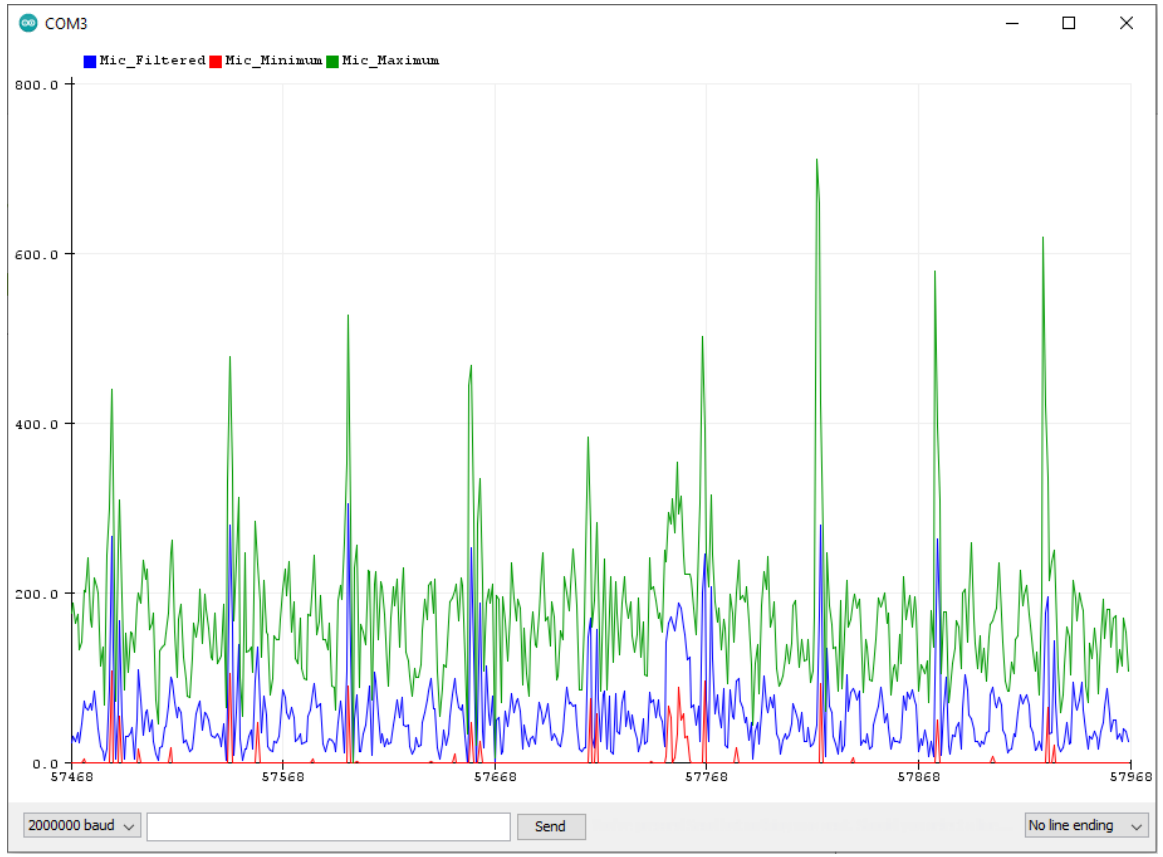
- What is the Serial Monitor?
  - The Serial Monitor is a feature of the Arduino IDE that gives you a serial terminal to see what is being sent to the COM port and allows you to send stuff out of the COM port.
  - We use this for receiving data from the Arduino.
  - We can also use this to help us debug our sketches.



Serial Plotter

Lesson 6: Temp and Humidity Sensor

- What is the Serial Plotter?
  - The Serial Plotter is a feature of the Arduino IDE that gives you a graphical representation of what is being sent to the COM port.
  - We use this for receiving data from the Arduino.
  - The serial plotter will display up to 500 consecutive sample periods.



- More Information:
  - https://arduinogetstarted.com/tutorials/arduino-serial-plotter

Serial Plotter

Lesson 6: Temp and Humdity Sensor

- How do we use the serial plotter?
  - Optionally we start off with a header using the syntax:

```
Serial.println("header_1 header_2");
```

    - We can add more headers by separating them with a space
  - To display the values, we use the **`Serial.print()`** and **`Serial.println()`** functions to send values to the Serial Plotter similar to how we sent values to the Serial Monitor.
  - Each value in a sample period should be separated by tab **`'/t'`** character. Each new sample period should be separated by a newline character or using the **`Serial.println()`** function.

Example Serial Plotter Code

```
void setup()
{
  Serial.begin(9600);
  Serial.println("header1 header2");
}

void loop()
{
  // get values to display
  int val1 = analogRead(A0);
  int val2 = analogRead(A2);
  Serial.print(val1);
  Serial.print('/t');
  Serial.println(val2);
}
```

Code Analysis: Creating a Function

Lesson 6: Temp and Humidity

```
float CtoF(float tempC)
{
  return tempC * 1.8 + 32;
}
```

- Function that takes a float representing a Celsius temperature and returns a float representing the Fahrenheit conversion
- What is a function?
  - A function is a block of code that
- Syntax:

```
return_type function_name(parameter_type parameter_name, …)
{
  // your code here
  return return_value
}
```

- More information:
  - https://www.arduino.cc/en/Reference/FunctionDeclaration

Code Analysis: `#define` macro

Lesson 6: Temp and Humidity

```
#define DHTTYPE DHT11
```

  - before compile, find all instances of “DHTTYPE” and replace with “DHT11”
- The `#define` preprocessor directive is used to name constants at the beginning of the file before compiling
- Caution: this preprocessor acts like a “find and replace all” command without regard to context. It is recommended to use all caps and be verbose when defining your constant names to minimize the possibility that the constant name was used elsewhere when using this method.
- Syntax:

```
#define CONSTANTNAME value
```

  - `CONSTANTNAME` – name of the constant or macro to define
  - `value` – value to assign to the constant or macro
- More Information:
  - https://www.arduino.cc/reference/en/language/structure/further-syntax/define/

Code Analysis: Using the DHT library, DHT Class

Lesson 6: Temp and Humidity

```
DHT dht11(dht11Pin, DHTTYPE);
```

- Create a DHT class instance called dht11 using pin dht11Pin and sensor type DHTTYPE
- What is a class?
  - A class is a collection of related variables and functions.
  - Each instance of a class is called an object
  - I like to think of classes as a super variable that has its own functions
- Syntax:

```
class_name object_name(initializing_value(s), …)
```

- More information:
  - https://www.w3schools.com/cpp/cpp_classes.asp

Activities

Lesson 6: Temp and Humidity

- Output the temperature in F and K
- Change the interval we read the sensor from 1000 msec to 100 msec

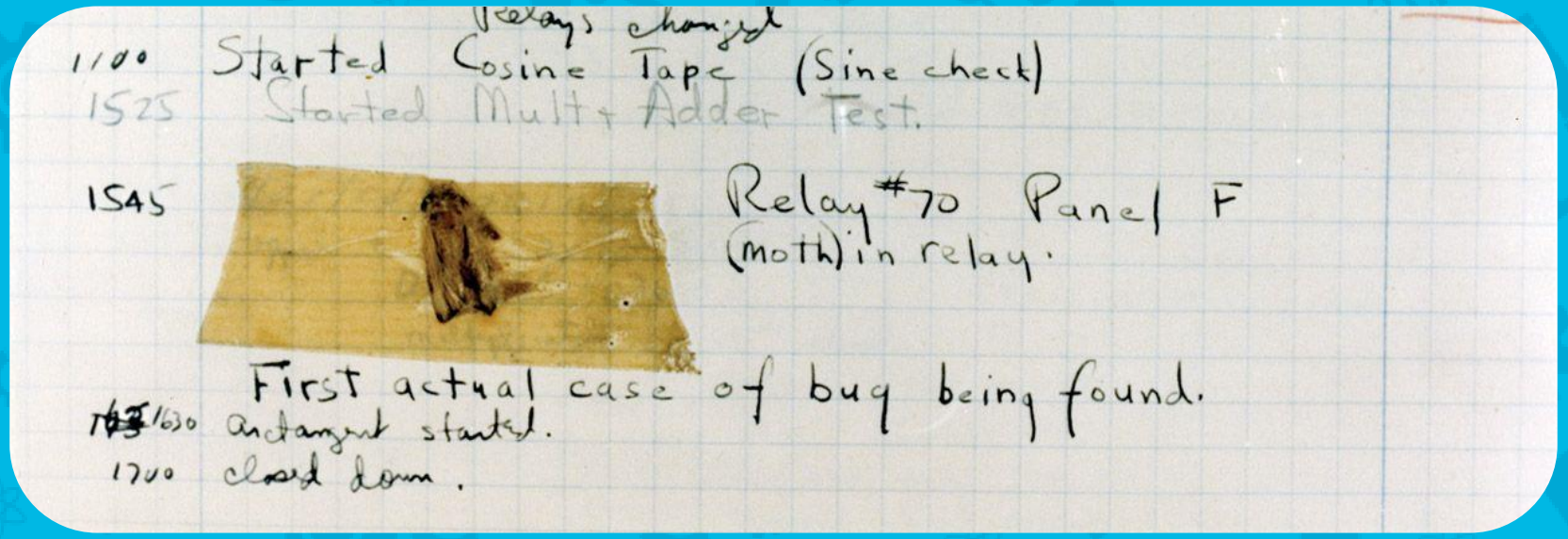
Timing

Code Analysis: A new timing method

Lesson 6: Temp and Humidity

```
Const unsigned int interval = 1000;
static unsigned long nextTime = 0;
unsigned long time = millis();
if (time >= nextTime)
{
  // your code here
  nextTime = time + interval;
}
```

- The above code shows a better way to keep more accurate timing of code execution
- It uses the `millis()` function to keep track of the actual elapsed time and calculates when it needs to run the code again.
- It is different from the `delay()` function because `delay()` waits for a period of time whereas using the `millis()` method can compensate for the time it takes to execute code.
- See timing demos for an example of the running differences.

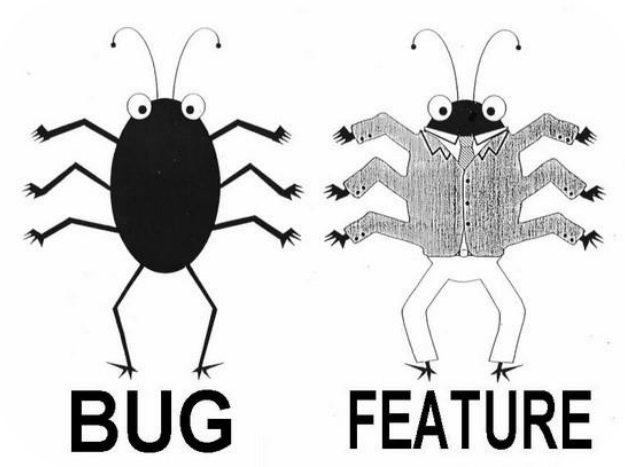

1100 Started Cosine Tape (Sine check)
1525 Started Multy Adder Test.
1545
Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1700 closed down.

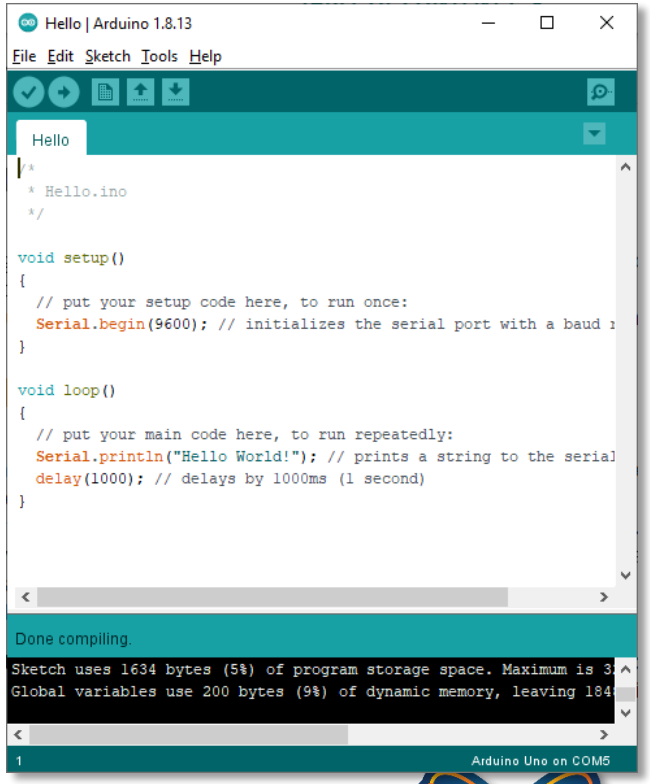
Debugging!

Debugging

- Debugging: Figuring out why something is not working as expected.

- Tools for Debugging Sketches in Arduino IDE
  - Message Area: provides basic messages "error"
  - Console: provides details





Message Area

Console

Debugging

- Steps
  - Attempt to Upload and view in serial monitor: *DebugProgram_1*

Debugging

- Solution
  - *DebugProgram_1*

`{` *missing*

Add a `{` *in line after* `void loop ()`

Debugging

- Steps
  - Attempt to Upload and view in serial monitor: *DebugProgram_2*

Debugging

- Solution
  - *DebugProgram_2*

) *missing*

Add a) to complete `if (time >= nextTime)`

Debugging

- Steps
  - Attempt to Upload and view in serial monitor: *DebugProgram_3*

Debugging

- Solution
  - *DebugProgram_3*

; missing

Add ; to complete `serial.print('\t');`

DebugProgram_3 | Arduino 1.8.13

File Edit Sketch Tools Help

DebugProgram_3

```
  return tempC * 1.8 + 32;
}
void setup()
{
  Serial.begin(9600); // Start the Serial Port
  Serial.println("Temperature_°C Temperature_°F Humidity _%RH"); // print out
  dht11.begin(); // Start the DHT11 temperature and Humidity sensor
}

void loop()
{
  static unsigned long nextTime = 0; // at what time should we re-run the cod
  unsigned long time = millis(); // get the current time

  if (time >= nextTime)
  {
    float temp = dht11.readTemperature(); // get the temperature already scal
    float humd = dht11.readHumidity();  // get the humidity already scaled in

    Serial.print(temp);  // send the temperature in °C
    Serial.print('\t') // separate the values with a TAB character
    Serial.print(CtoF(temp));  // send the temperature °F
    Serial.print('\t'); // separate the values with a TAB character
    Serial.println(humd); // send the humidity
    nextTime = time + interval;
  }
```

expected ';' before 'Serial'

Copy error messages

```
exit status 1
expected ';' before 'Serial'
```

54 Arduino Uno on COM4

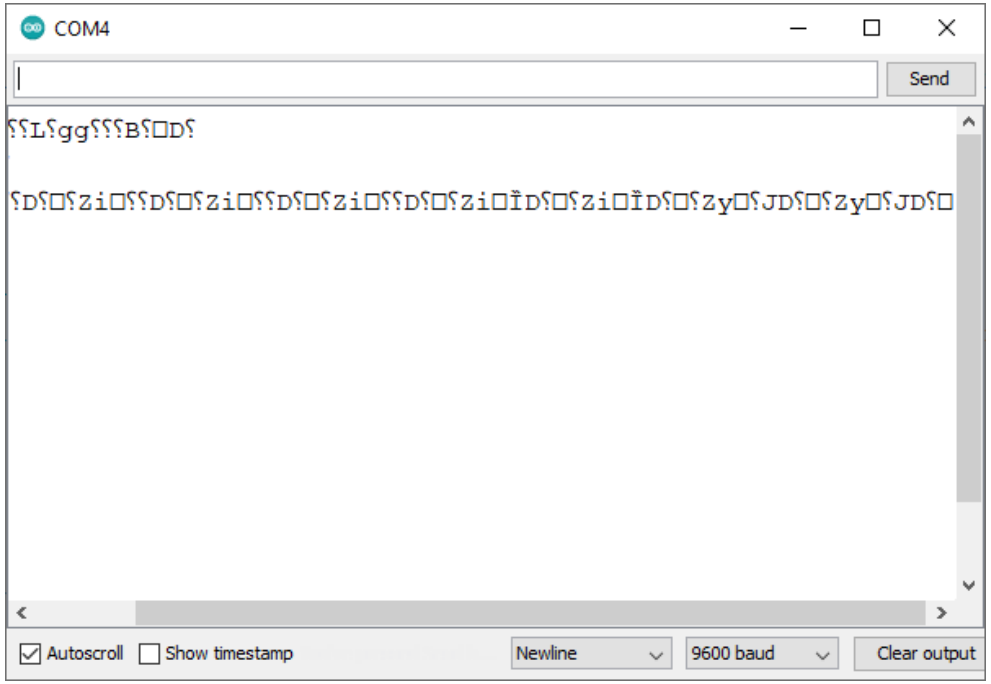
Debugging

- Steps
  - Attempt to Upload and view in serial monitor: *DebugProgram_4*

Debugging

- Solution
  - *DebugProgram_4*

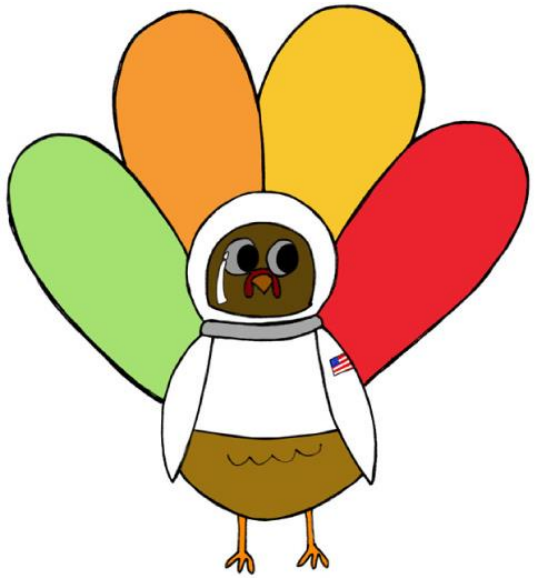
baud rate mismatch
- change in serial plotter
- change in sketch to `serial.begin(9600);`



Next Weeks

- Office Hours
  - Monday Nov. 23rd @ 7:00 PM
  - Monday Nov. 30th @ 7:00 PM

- Weekly Session
  - ~~Thursday Nov. 26~~ Thanksgiving, no session
  - Thursday Dec. 3rd @ 7:00 PM



Lesson 5: Microphone

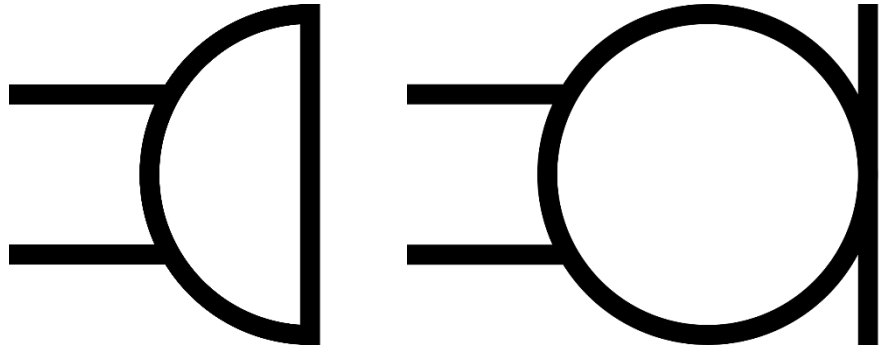
See the output of the microphone in the Serial Plotter

Microphone Introduction

Lesson 5: Microphone

- What is a microphone?
  - A microphone is a transducer that converts sound wave to an electrical signal.
  - Microphones are used to record music and voice, but also used for scientific analysis.
- Where are microphones used?
  - Audio recording, cell phones, walkie-talkie, computers, sonar, presence detection, knock detection, etc.
- How do we use the Microphone?
  - Microphones must be amplified or conditioned before we can use the signal. We can then read the analog signal with the ADC in the microcontroller.
- More Information:
  - https://en.wikipedia.org/wiki/Microphone

Microphone Electrical Symbols



By ErikBuer - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=50257723

By ErikBuer - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=50257685

Microphone

Downloaded from https://www.digikey.com/en/products/detail/cui-devices/CMA-6542PF/1869980

Lesson 5 Hardware

Lesson 5: Microphone

- What hardware will we need for this Lesson?
  - Grove Sound Module on pin A2
  - Seeeduino Lotus (Arduino Uno compatible board)
    - The Arduino has the serial port hardware built into the device

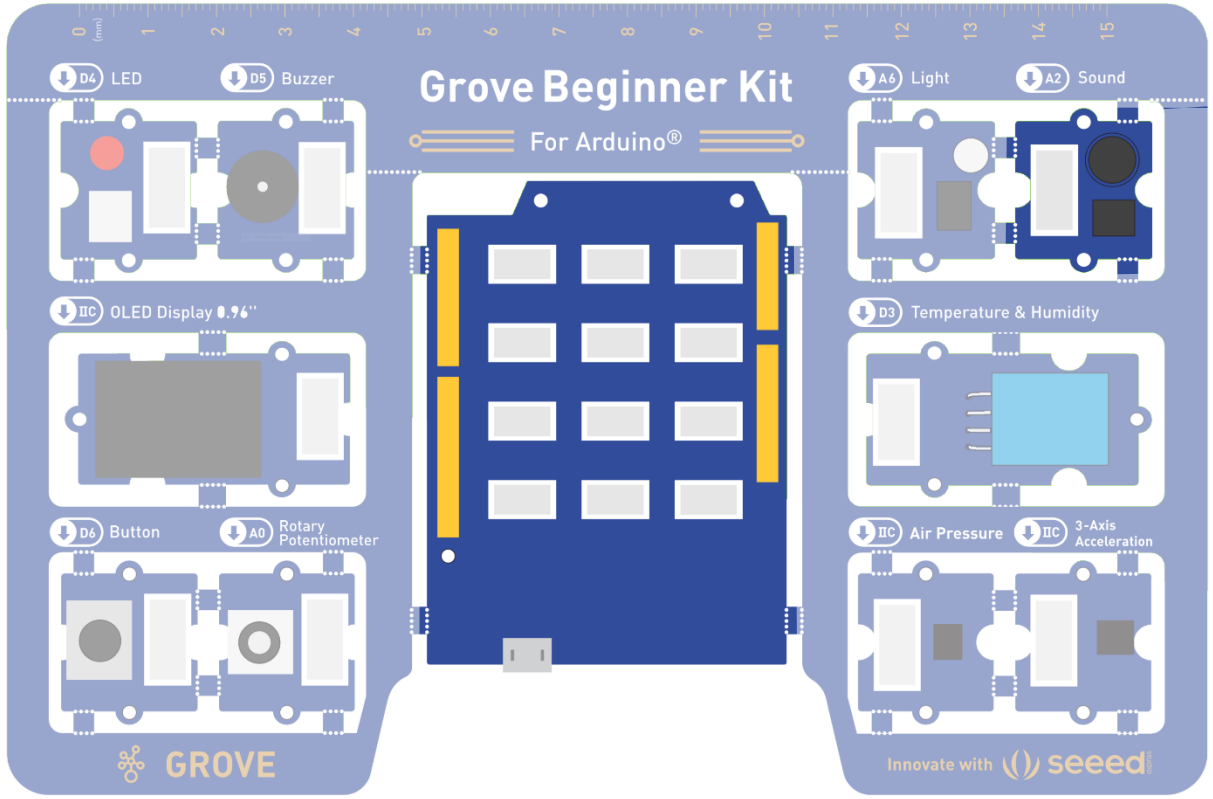


Image modified from https://files.seeedstudio.com/wiki/Grove-Beginner-Kit-For-Arduino/res/Grove-Beginner-Kit-For-ArduinoPDF.pdf

Open and Upload Sketch

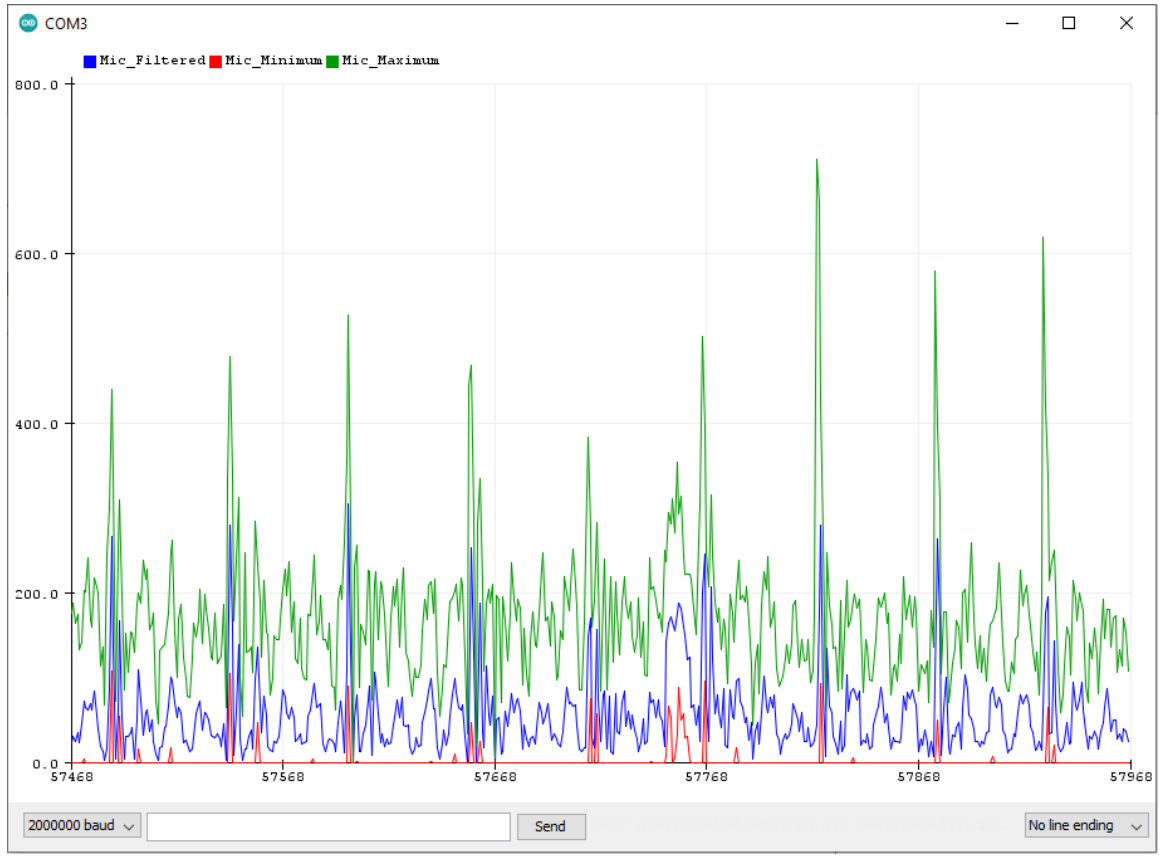
Lesson 5: Microphone

1. Open Microphone Sketch
   a. **File → Sketchbook → FRSEF_Crash_Course → Week_2 → W2L5_Microphone.ino**
2. Verify the sketch by clicking the Verify Button.
   a. The sketch should compile with no errors.
3. Upload the sketch to your Arduino by clicking the Upload Button.
   a. The sketch should re-compile, and then upload to your Arduino.
4. Open the serial monitor.
   a. **Tools → Serial Plotter** (Ctrl+Shift+L)
5. Observe the output in the Serial Plotter

Serial Plotter

Lesson 5: Microphone

- What is the Serial Plotter?
  - The Serial Plotter is a feature of the Arduino IDE that gives you a graphical representation of what is being sent to the COM port.
  - We use this for receiving data from the Arduino.
  - The serial plotter will display up to 500 consecutive sample periods.



- More Information:
  - https://arduinogetstarted.com/tutorials/arduino-serial-plotter

Serial Plotter

Lesson 5: Microphone

- How do we use the serial plotter?
  - Optionally we start off with a header using the syntax:

```
Serial.println("header_1 header_2");
```

    - We can add more headers by separating them with a space
  - To display the values, we use the **`Serial.print()`** and **`Serial.println()`** functions to send values to the Serial Plotter similar to how we sent values to the Serial Monitor.
  - Each value in a sample period should be separated by tab **`'/t'`** character. Each new sample period should be separated by a newline character or using the **`Serial.println()`** function.

Example Serial Plotter Code

```
void setup()
{
  Serial.begin(9600);
  Serial.println("header1 header2");
}

void loop()
{
  // get values to display
  int val1 = analogRead(A0);
  int val2 = analogRead(A2);
  Serial.print(val1);
  Serial.print('/t');
  Serial.println(val2);
}
```

Code Analysis – C++ Bitshift Operators

Lesson 5: Microphone

```
for(unsigned int i = 0; i < 1<<filterConstant; i++)
```

- Shift binary 1 to the left by **filterConstant** bits
  - Equivalent to $2^{\texttt{filterConstant}}$

<< Bitshift Left `0b0001 << 2 = 0b0100`

- Shift the binary value on the left by the number of bits on the right

>> Bitshift Right `0b0101 >> 1 = 0b0010`

- Shift the binary value on the right by the number of bits on the left

```
A << 1 → A * 2
B << 2 → B * 4
C << 3 → C * 8

D >> 1 → D / 2
E >> 2 → E / 4
F >> 3 → F / 8
```

- Leading or trailing digits get dropped, and new digits are 0
- Often used to efficiently multiply (<<) or divide (>>) by powers of 2
- More Information:
  - https://beginnersbook.com/2017/08/cpp-operators/
  - https://www.arduino.cc/reference/en/

Code Analysis – C++ Comparison Operators

Lesson 5: Microphone

```
for(unsigned int i = 0; i < 1<<filterConstant; i++)
```

- Is **i** less than $2^{filterConstant}$?

==	Equal To	→ TRUE if the left side is **equal to** the right side
!=	Not Equal To	→ TRUE if the left side is **not equal to** the right side
<	Less Than	→ TRUE if the left side is **less than** the right side
<=	Less Than or Equal To	→ TRUE if the left side is **less than or equal to** the right side
>	Greater Than	→ TRUE if the left side is **greater than** the right side
>=	Greater Than or Equal To	→ TRUE if the left side is **greater than or equal to** the right side

- More Information:
  - https://beginnersbook.com/2017/08/cpp-operators/
  - https://www.arduino.cc/reference/en/

Code Analysis – C++ Arithmetic Operators

Lesson 5: Microphone

+	Addition	`A = 1 + 2` →	`A = 3`
-	Subtraction	`B = 3 - 1` →	`B = 2`
*	Multiplication	`C = 2 * 4` →	`C = 8`
/	Division	`D = 6 / 3` →	`D = 2`
%	Modulo (remainder)	`E = 7 % 4` →	`E = 3`

- More Information:
  - https://beginnersbook.com/2017/08/cpp-operators/
  - https://www.arduino.cc/reference/en/

Code Analysis – C++ Auto-increment and Auto-decrement Operators

Lesson 5: Microphone

```
for(unsigned int i = 0; i < 1<<filterConstant; i++)
```

- Increment **i** by 1 at the end of the for loop.

++ Auto-increment `i++` → `i = i + 1`

- Increments the value of a variable by 1

-- Auto-decrement `j--` → `j = j - 1`

- Decrements the value of a variable by 1

- More Information:
  - https://beginnersbook.com/2017/08/cpp-operators/
  - https://www.arduino.cc/reference/en/

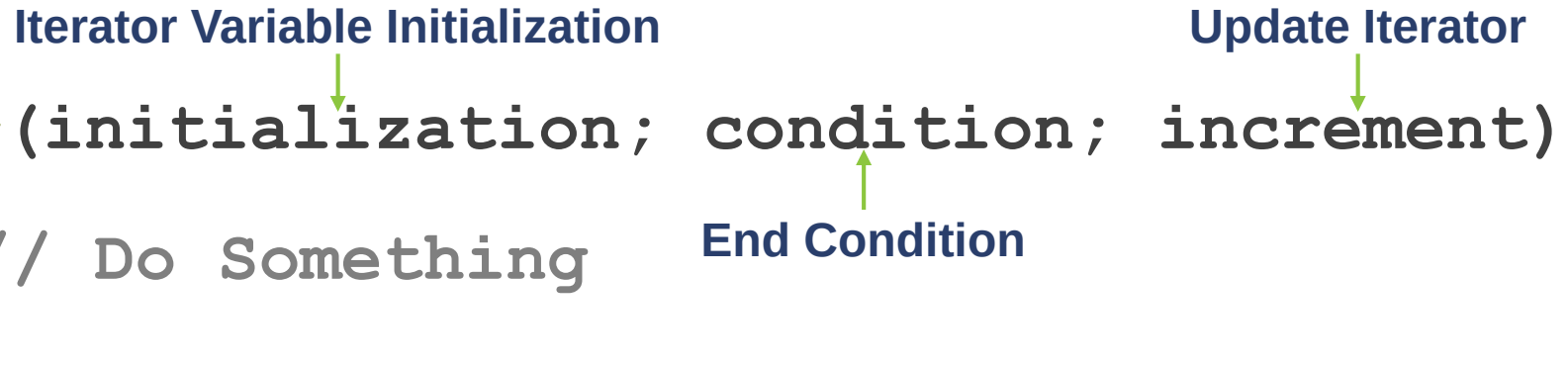
Code Analysis – `for` Loop

Lesson 5: Microphone

```
for(unsigned int i = 0; i < 1<<filterConstant; i++)
{/* Do Something */}
```

- Repeat code inside the curly braces $2^{\texttt{filterConstant}}$ times

- `for()` Loops are used to repeat code that appears between its curly braces
- Syntax:



- More Information:
  - https://www.arduino.cc/reference/en/language/structure/control-structure/for/
  - https://beginnersbook.com/2017/08/cpp-for-loop/

Code Analysis – C++ Assignment Operators

Lesson 5: Microphone

```
micValueLong += micValue;
```

- Add **micValue** and **micValueLong** then store the result in **micValueLong**

=	Equals	Assigns value of right side to the left side	
+=	Plus Equals	`A += 2` →	`A = A + 2`
-=	Minus Equals	`B -= 3` →	`B = B - 3`
*=	Multiplication	`C *= 4` →	`C = C * 4`
/=	Division	`D /= 5` →	`D = D / 5`
%=	Modulo (remainder)	`E %= 6` →	`E = E % 6`

- More Information:
  - https://beginnersbook.com/2017/08/cpp-operators/
  - https://www.arduino.cc/reference/en/

Code Analysis – Averaging Filter

Lesson 5: Microphone

- What is a filter?
  - A filter is used to remove an unwanted component of a signal.
  - For sensor measurements a **low pass filter** is often used to reduce noise or some high frequency component.
  - There are many different types of filters, and numerous ways to implement filters.
- What is averaging?
  - Averaging is taking the mean value of a signal over the sampling period.
- More Information:
  - https://en.wikipedia.org/wiki/Filter_(signal_processing)
  - https://en.wikipedia.org/wiki/Average
  - https://www.mathsisfun.com/mean.html

```
// Average filter
int average;
int sumSamples = 0;
for(int i = 0; I < numSamples; i++)
{
 sumSamples += analogRead(A2);
}
average = sumSamples / numSamples;
```