

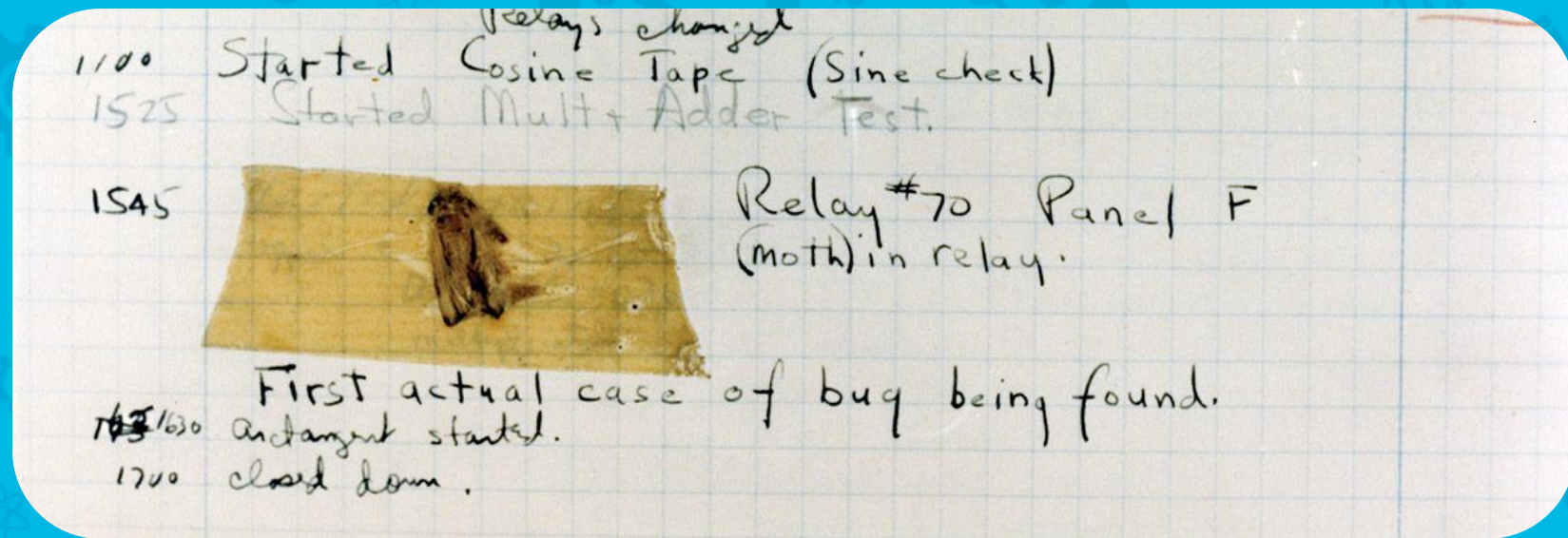


Measurements, Sensors and Data Logging Course

Week 4

Upcoming Weeks

- Office Hours
 - Monday Dec. 7th @ 7:00 PM
 - Monday Dec. 14th @ 7:00 PM
- Weekly Sessions
 - Thursday Dec. 10th @ 7:00 PM
 - Thursday Dec. 17th @ 7:00 PM

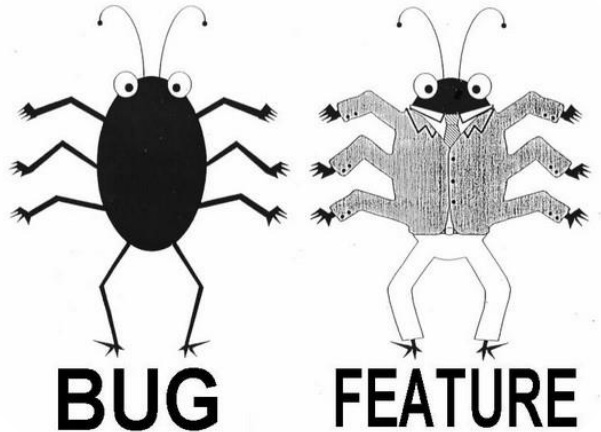


Debugging!

Debugging



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



Debugging

- Solution

- *DebugProgram_1*

{ *missing*

Add a { in line after void loop ()



```
DebugProgram_1 | Arduino 1.8.13
File Edit Sketch Tools Help

DebugProgram_1
* November 15, 2020
*/

// include Adafruit's DHT libraries. This must be installed first (DHT sensor
#include <DHT.h>
#include <DHT_U.h>

#define DHTTYPE DHT11 // DHT11 sensor type (the blue one)

const byte dht11Pin = 3; // DHT Sensor is on pin D3
const unsigned int interval = 1000; // interval between readings in ms DHT11

DHT dht11(dht11Pin, DHTTYPE); // create an instance of the DHT class to interact

// Convert from Celsius to Fahrenheit
float CtoF(float tempC)
{
    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
}

expected initializer before '/' token
Not used: C:\Users\MSDMikko\Documents\Arduino\libraries\Grove_Temperature_And_Humidity\Grove_Temperature_And_Humidity.cpp
exit status 1
expected initializer before '/' token

16
Arduino Uno on COM4
```

Debugging



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

Debugging

- Solution

- *DebugProgram_2*

-) *missing*

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



```
DebugProgram_2 | Arduino 1.8.13
File Edit Sketch Tools Help

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 code
  unsigned long time = millis(); // get the current time

  if (time >= nextTime
  {
    float temp = dht11.readTemperature(); // get the temperature already scaled
    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 '{' token
exit status 1
expected ')' before '{' token

48 Arduino Uno on COM4
```

Simple Datalogger

Lesson 8

Arduino Shields

Lesson 8: Simple Datalogger

- What is an Arduino Shield?
 - An Arduino shield is a circuit board that follows the basic shape and pinout of the Arduino board. It plugs into the top of the Arduino Board to extend the capabilities of the Arduino with the circuit on the shield.
- How do we use an Arduino Shield?
 - Arduino shields are plugged into the top of the Arduino board and can be stacked together.
 - Libraries and code can then be used to access the additional capabilities of the shield.
- More Information:
 - <https://learn.sparkfun.com/tutorials/arduino-shields>
 - <https://www.arduino.cc/en/Main/arduinoShields>

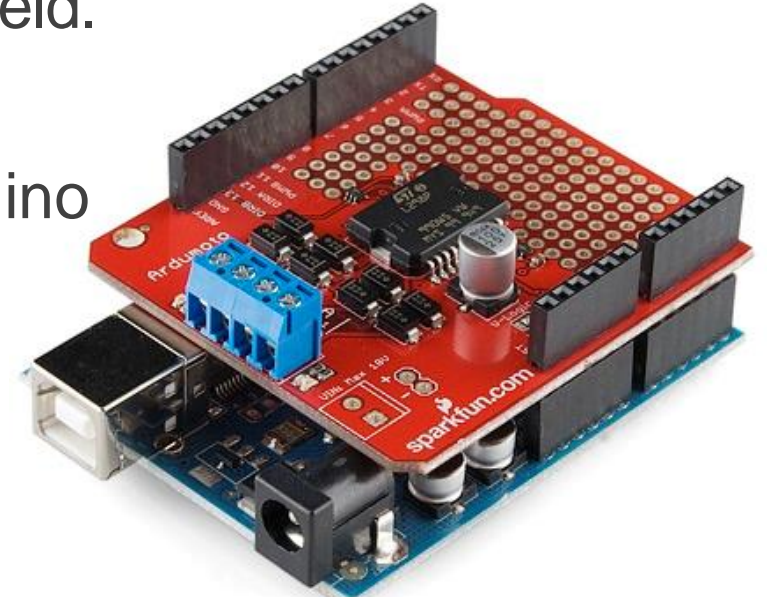
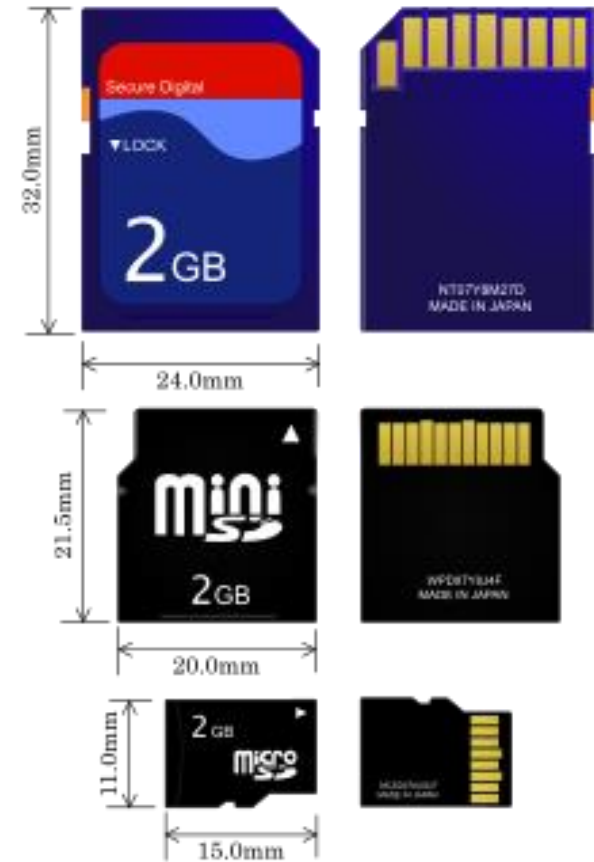


Image copied from <https://learn.sparkfun.com/tutorials/arduino-shields>

SD Card

Lesson 8: Simple Datalogger

- What is an SD Card?
 - Memory storage device
 - Stands for Secure Digital
 - Multiple Sizes: physical form factors and memory capacity
 - Where is Flash data storage used?
 - How do we use an SD card with an Arduino?
 - SD card shield!!
 - SD card library
 - SPI communication (digital communication)
- <https://www.arduino.cc/en/reference/SD>



CSV File

Lesson 8: Simple Datalogger

- What is a CSV file?
 - Comma Separated Value
 - Typically used as a generic spreadsheet, can be opened in many programs.
 - Excel, Google Sheets
 - Also used for exporting and importing data with specialized programs.
- How do we use a CSV file?
 - Can be opened in Excel and used similar to a regular excel file
 - Excel features (math, etc) cannot be saved
 - File can be saved as a regular .xls / .xlsx
 - Can also be opened in google sheets, notepad++, etc

Digital Communications

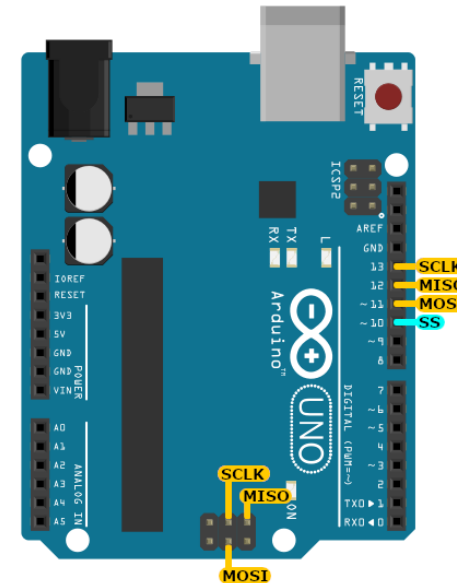
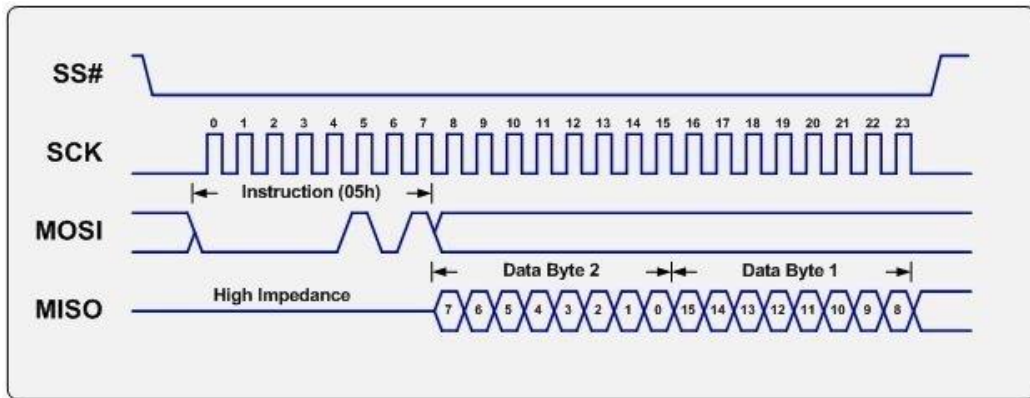
Lesson 8: Simple Datalogger

- Digital Communications
 - Digital signals used to communicate between devices
 - Ability to combine multiple signals together
 - Many different protocols
- Example protocols
 - CAN (2 wires)
 - Ability to have multiple devices communicating to each other (3+)
 - RS232 (3 wires)
 - Only 2 devices can communicate together

SPI

Lesson 8: Simple Datalogger

- SPI (Serial Peripheral Interface), multiple devices, short distances
 - Synchronous Communication: clock signal shared between master and slave
 - MISO: Master In Slave Out, *sends data to the master from the slave device*
 - MOSI: Master Out Slave In, *data from the master to the slave device*
 - SCK: Serial Clock
 - CS: Slave Select (individual for each device)
- <https://www.arduino.cc/en/reference/SPI>
- <https://www.corelis.com/education/tutorials/spi-tutorial/>



Master Device



Slave Device

Lesson 8 Hardware

Lesson 8: Simple Datalogger

- What hardware will we need for this Lesson?
 - Potentiometer on pin A0
 - Grove Light Sensor on pin A6
 - Grove Sound Sensor A2
 - Seeeduino Lotus (Arduino Uno compatible board)
 - SD Card Shield (HiLetGo)
 - Multiple options are available
 - SD Card
 - Formatted as FAT16

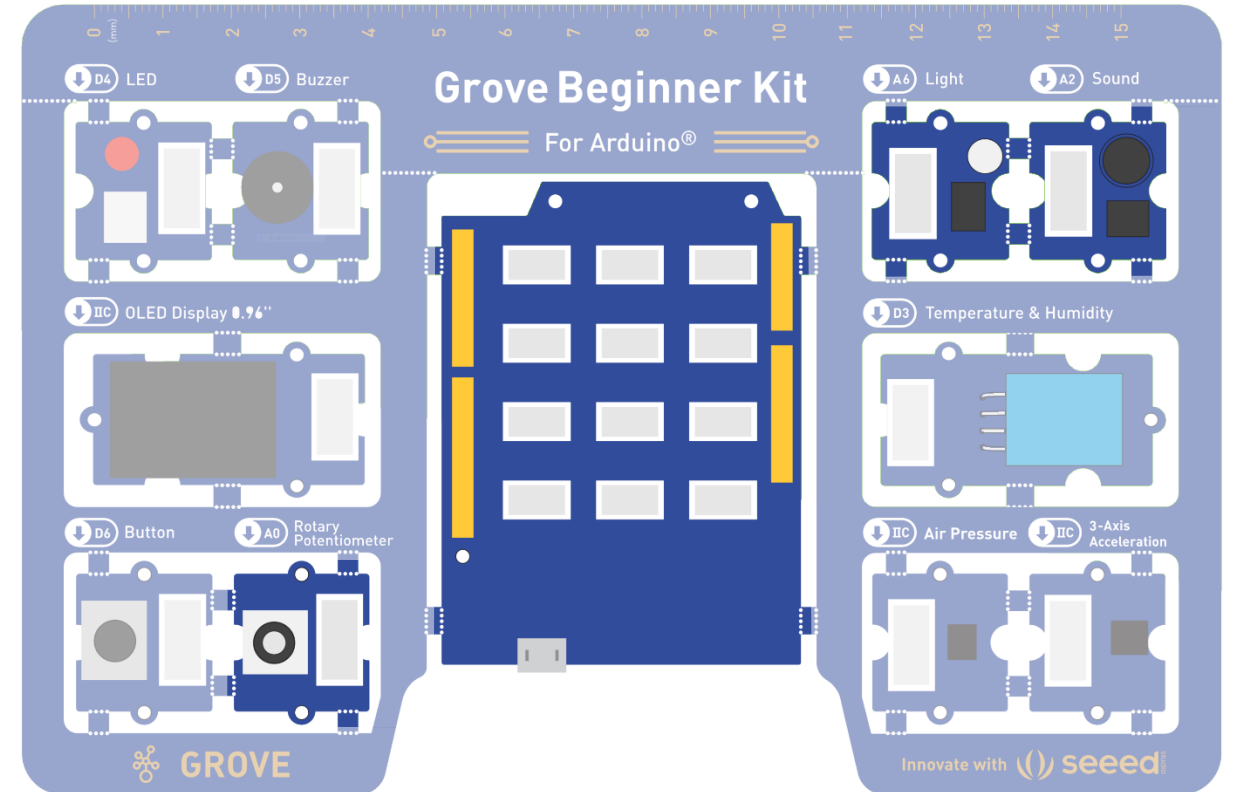


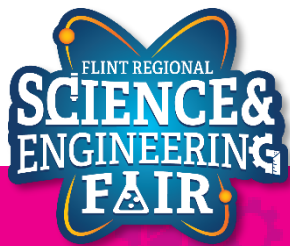
Image modified from <https://files.seeedstudio.com/wiki/Grove-Beginner-Kit-For-Arduino/res/Grove-Beginner-Kit-For-ArduinoPDF.pdf>



Image copied from <https://www.amazon.com/HiLetGo-Logging-Recorder-Logger-Arduino/dp/B00PI6TQWO/>



Image copied from <https://www.microcenter.com/product/485234/micro-center-64gb-microsdxc-class-10-uhs-1-flash-memory-card>



Assemble the Datalogging Hardware

Lesson 8: Simple Datalogger

1. Carefully align the pins of the shield with the sockets in the Seeeduino Lotus and press down until seated
2. Place micro-SD card into SD card adaptor and insert into SD Card Shield
3. Plug in USB cable between Seeeduino Lotus and Computer

Open and Upload Sketch

Lesson 8: Simple Datalogger

4. Open Simple Datalogger Sketch
 - **File → Sketchbook → FRSEF_Crash_Course → Week_4 → L8_Simple_Datalogger.ino**
5. Upload the sketch to your Arduino by clicking the Upload Button.
 - The sketch should compile, and then upload to your Arduino.
6. Open the serial monitor.
 - **Tools → Serial Monitor (Ctrl+Shift+M)**
7. Observe the output in the Serial Monitor for a few seconds
 - Cover the light sensor, rotate the potentiometer, and make some noise to change the value of the sensor readings
8. Unplug the USB connector

Open Logged Data on Computer

Lesson 8: Simple Datalogger

9. Remove SD card from Shield
10. Insert into computer
11. Open “datalog.csv” in excel or other spreadsheet program
12. Bonus Activity: Graph the data!

SD Library

Lesson 8: Simple Datalogger

```
#include <SD.h>
```

- The SD Library allows us to communicate with SD cards formatted with FAT16 and FAT32 filesystems.
- It is built into the Arduino IDE, so there is no need to install it.
- It takes advantage of the SPI interface of the SD card, so the SPI library must also be included (`#include <SPI.h>`) before the SD library.
- It requires the chip select (sometimes called slave select) pin from the SD card to be connected to a digital output of the Arduino.
 - The SD Card Shield we chose uses pin D10 for the chip select
- More information:
 - <https://www.arduino.cc/en/Reference/SD>

SD Library – Appending data to a file

Lesson 8: Simple Datalogger

- We can append data to a file by:
 - Opening the file for writing
 - Checking that the file was opened
 - Writing a String to the file (we cover creating this String on the next slides)
 - Closing the file

```
File dataFile = SD.open("datalog.csv", FILE_WRITE);  
if (dataFile)  
{  
    dataFile.println(dataString);  
    dataFile.close();  
}
```

Strings

Lesson 8: Simple Datalogger

```
String example = "This is an example of a string.";
```

- Create a String class instance named `example` containing the text "This is an example of a string."
- Strings are representations of **text** instead of numbers, and as such they do not behave in the same way as our previous datatypes.
- We have used string constants before in our sketches. Look for text encased in quotes, usually inside our `print()` and `println()` statements.
- More information:
 - <https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/>
 - <https://www.arduino.cc/en/Tutorial/BuiltInExamples#strings>

Converting other datatypes to Strings

Lesson 8: Simple Datalogger

- Syntax:

```
String(val)
```

```
String(val, base)
```

```
String(val, decimalPlaces)
```

- `val` – variable to format as a string
- `base` – (optional) if `val` is an integer, what base should be used for the string representation. Options – DEC (default), BIN, or HEX
- `decimalPlaces` – (optional) if `val` is a float, how many decimal places should be used. 2 is default

- For example:

- | | | |
|-----------------------------------|-------------|---|
| – <code>String(42)</code> | → “42” | // convert integer value to DEC |
| – <code>String(42, BIN)</code> | → “101010” | // convert integer value to BIN |
| – <code>String(42, HEX)</code> | → “2A” | // convert integer value to HEX |
| – <code>String('x')</code> | → “x” | // convert character value to String |
| – <code>String(0.12345)</code> | → “0.12” | // convert float value to DEC to 2 places |
| – <code>String(0.12345, 5)</code> | → “0.12345” | // convert float value to DEC to 5 places |

- This operation is called type casting

String Concatenation

Lesson 8: Simple Datalogger

- Concatenation – combining 2 strings into one larger string
- There are several way to concatenate strings in Arduino
- Syntax:

```
string1 += string2;
```

– Appends `string2` to the end of `string1`

- Example:

```
String string1 = "1";
```

```
String string2 = "2";
```

```
string1 += string2; // string 1 now equals "12"
```

```
Serial.print(string1); // prints 12
```

- More information and more ways to concatenate strings:


- <https://www.arduino.cc/reference/en/language/variables/data-types/string/functions/concat/>
- <https://www.arduino.cc/reference/en/language/variables/data-types/string/operators/concatenation/>
- <https://www.arduino.cc/reference/en/language/variables/data-types/string/operators/append/>

Creating a CSV Row with a String

Lesson 8: Simple Datalogger

```
String dataString = "";  
dataString += String(val1);  
dataString += ",";  
dataString += String(val2);  
dataString += ",";  
dataString += String(val3);
```

val1, val2, val3



	A	B	C
1	val1	val2	val3

Activities

Lesson 8: Simple Datalogger

- Read the temperature and humidity and light sensors overnight
 - Read the data and open it in a program (excel) to analyze and plot

Real Time Clock + Time Library

Lesson 10

Real Time Clock

Real Time Clock + Time Library

- What is a Real Time Clock (RTC)?
 - A clock that keeps track of the current time, independent from the Arduino.
 - On-board crystal oscillator.
 - Battery keeps the RTC powered even when the Arduino is powered down.
 - Arduino can be used to read this time and program.
- How do we use the RTC?
 - Time and DS1307RTC Libraries
 - I2C communication
 - Need to set the time on the RTC the first time it is powered (already set on your hardware)
- More Information:
 - https://www.pjrc.com/teensy/td_libs_Time.html
 - https://www.pjrc.com/teensy/td_libs_DS1307RTC.html

Hz + Logging Rates

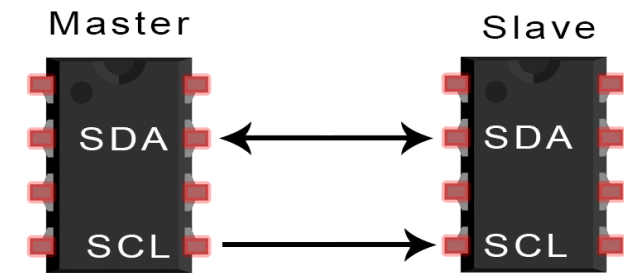
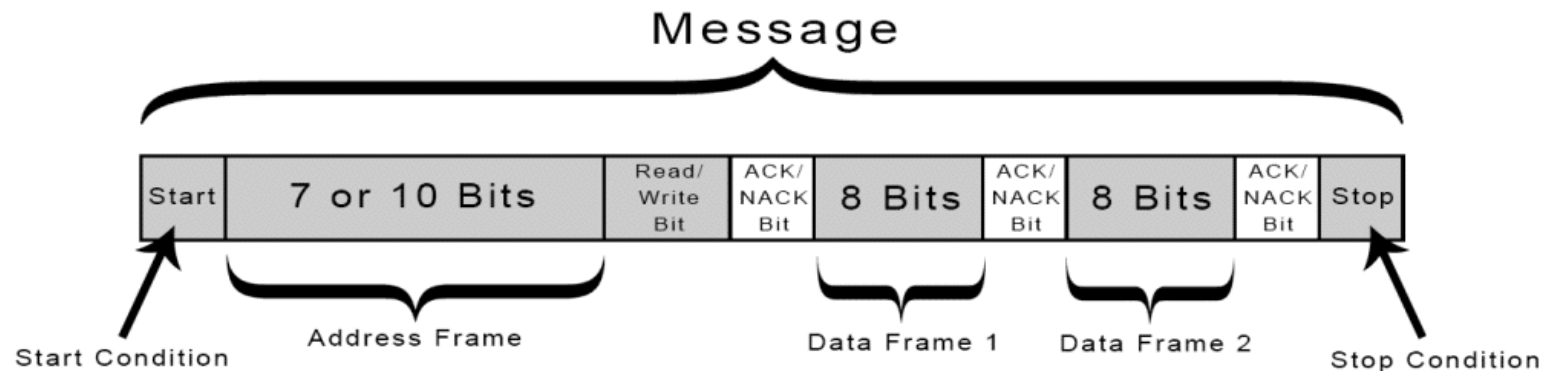
Real Time Clock + Time Library

- What is Hz?
 - A unit of frequency. aka: how often something is happening each second
- How do we calculate Hz?
 - # of occurrences / second or 1 / time event takes (sec)
 - $\frac{\text{\# of occurrences}}{1 \text{ (sec)}}$ ex. 5 occurrences / sec = 5 Hz
 - $\frac{1}{\text{time event takes (sec)}}$ ex. 5 seconds / event $\Rightarrow 1 / 5 = 0.2$ Hz
- What rate should we log our data at?
 - Depends!
 - Ideal: 10 times faster than the signal changes
 - Minimum: 2 times faster than the signal changes

I²C (Inter-Integrated Circuit)

Real Time Clock + Time Library

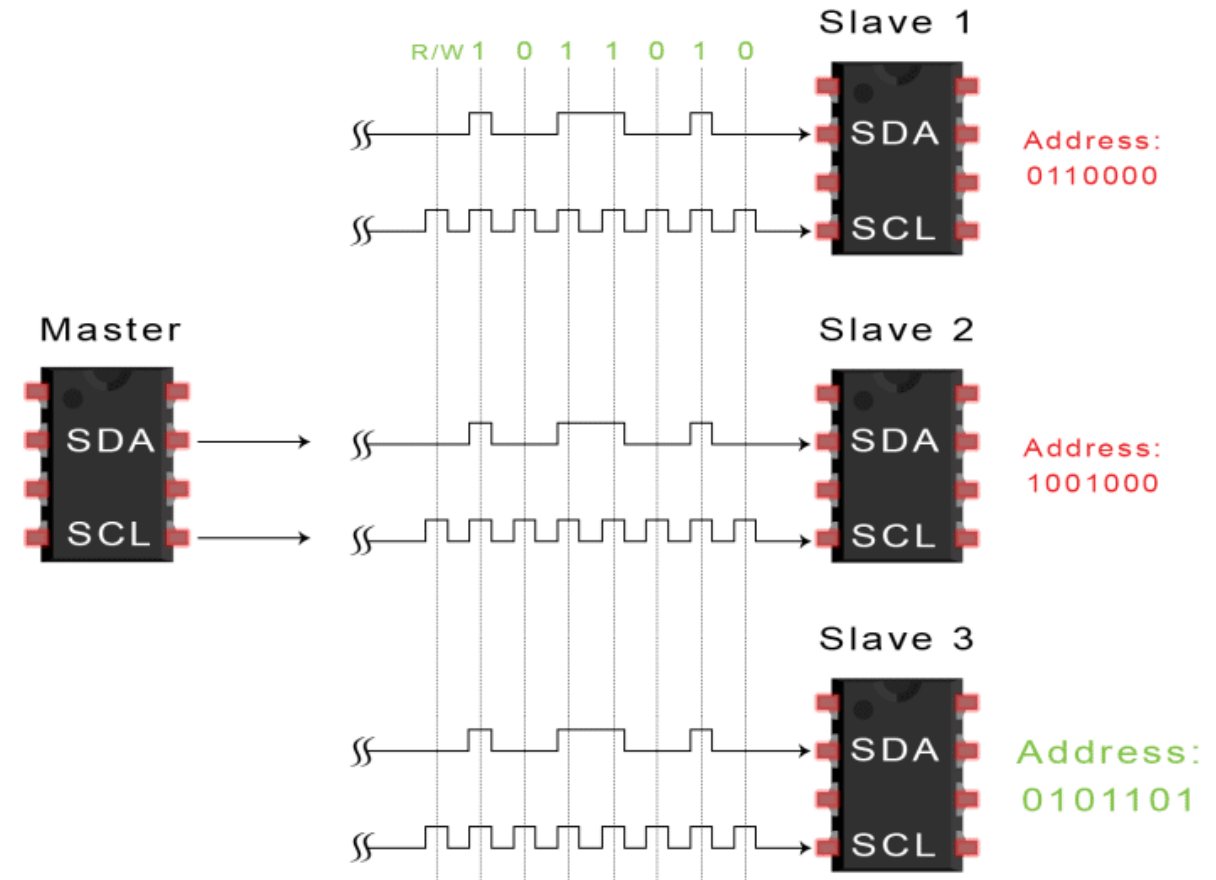
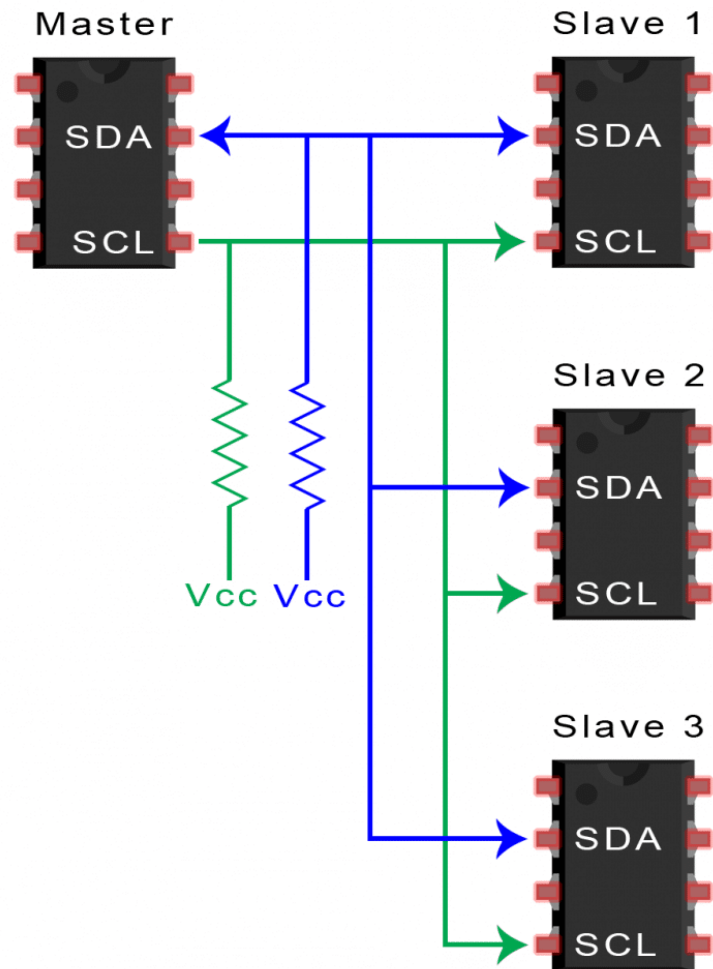
- I²C, sometimes I2C or IIC
 - Two wires and robust, but slower transfer rate than SPI
 - Library: Wire.h
 - SDA (serial data): The line for the master and slave to send and receive data.
 - SCL (serial clock): The line that carries the clock signal.



- <https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/#:~:text=I2C%20is%20a%20serial%20communication,always%20controlled%20by%20the%20master>

I²C (Inter-Integrated Circuit)

Real Time Clock + Time Library



Lesson 9 Hardware

Real Time Clock + Time Library

- What hardware will we need for this Lesson?
 - Potentiometer on pin A0
 - Grove Light Sensor on pin A6
 - Grove Sound Sensor A2
 - Seeeduino Lotus (Arduino Uno compatible board)
 - SD Card Shield (HiLetGo) w/ battery
 - Multiple options are available
 - SD Card
 - Formatted as FAT16

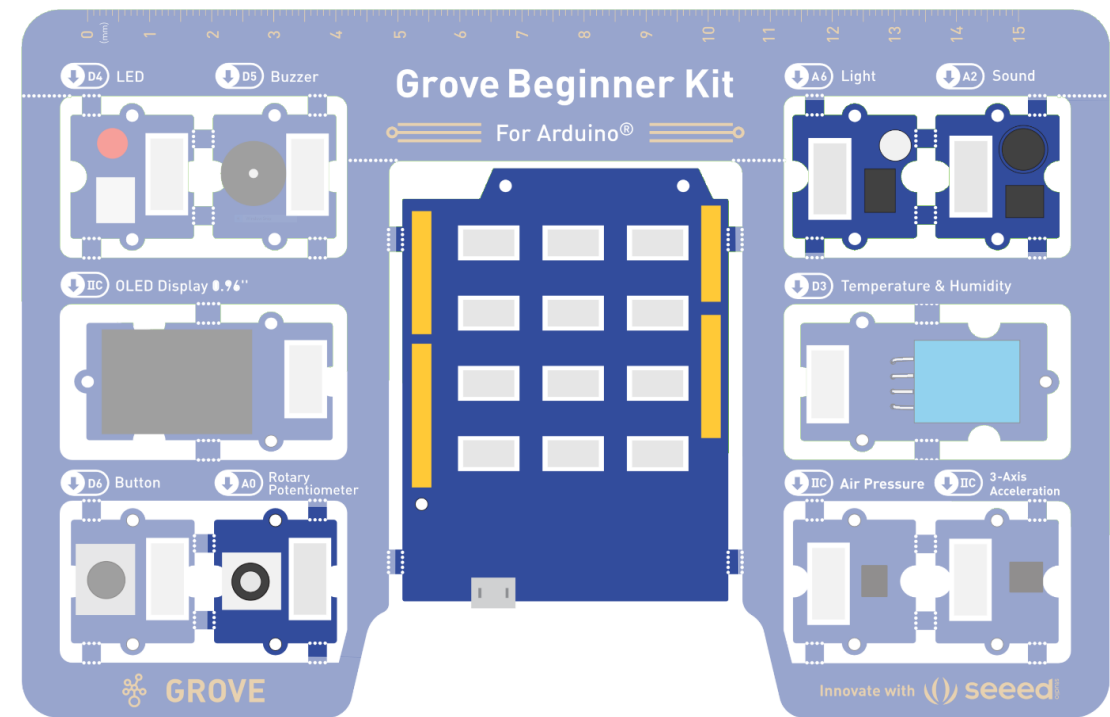


Image modified from <https://files.seeedstudio.com/wiki/Grove-Beginner-Kit-For-Arduino/res/Grove-Beginner-Kit-For-ArduinoPDF.pdf>

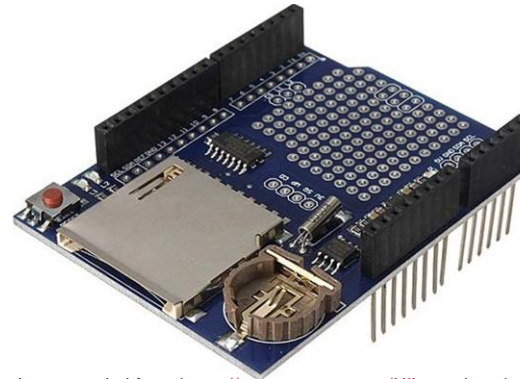
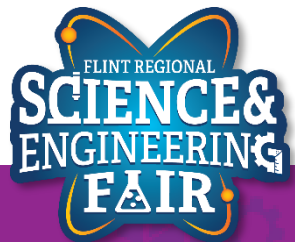


Image copied from <https://www.amazon.com/HiLetGo-Logging-Recorder-Logger-Arduino/dp/B00PI6TQWO/>



Image copied from <https://www.microcenter.com/product/485234/micro-center-64gb-microsdxc-class-10-uhs-1-flash-memory-card>



Install the Time Libraries

Real Time Clock + Time Library

1. Install the Time library from the Library Manager
 - a. Sketch → Include Library → Manage Libraries...
 - b. Search for “**DS1307RTC**”
 - c. Install **Time** by Michael Margolis
 - d. Install **DS1307RTC** by Michael Margolis
 - e. Search for “**TimeAlarms**”
 - f. Install **TimeAlarms** by Michael Margolis

Open and Upload Sketch

Real Time Clock + Time Library

2. Open Simple Datalogger Sketch
 - **File → Sketchbook → FRSEF_Crash_Course → Week_4 → L9_Timed_Datalogger.ino**
3. Upload the sketch to your Arduino by clicking the Upload Button.
 - The sketch should compile, and then upload to your Arduino.
4. Open the serial monitor.
 - **Tools → Serial Monitor (Ctrl+Shift+M)**
5. Observe the output in the Serial Monitor for a few seconds
 - Cover the light sensor, rotate the potentiometer, and make some noise to change the value of the sensor readings
6. Unplug the USB connector

Open Logged Data on Computer

Lesson 8: Simple Datalogger

7. Remove SD card from Shield
8. Insert into computer
9. Open “datetime.csv” in excel or other spreadsheet program. Note the date and time recognized by the spreadsheet
10. Bonus Activity: Graph the data!

Time Library

Real Time Clock + Time Library

- What is the Time library?
 - An Arduino library that helps in keeping the time and provides functions to deal with seconds, minutes, hours, days, months and years.
 - The Time library can be synced with a RTC or other time and date services (GPS, NTP, Serial or other service that provides a standard Unix `time_t` time)

```
#include <TimeLib.h>
```

- Syntax:

`year()` – return the current year
`year(t)` – return the year of `t`
`month()` – return the current month (1-12)
`month(t)` – return the month of `t` (1-12)
`day()` – return the current day of the month (1-31)
`day(t)` – return the day of `t` (1-31)

`hour()` – return the current hour (0-23)
`hour(t)` – return the hour of `t` (0-23)
`minute()` – return the current minute (0-59)
`minute(t)` – return the minute of `t` (0-59)
`second()` – return the current second (0-59)
`second(t)` – return the second of `t` (0-59)
`now()` – return the current time as a `time_t` number

`t` is a `time_t` number

`setSyncProvider(getTimeFunction)` – configure Time library to periodically call a user specified function to sync the clock.
`getTimeFunction` is the name of the function that gets called.

- More Information

- https://www.pjrc.com/teensy/td_libs_Time.html
- <https://github.com/PaulStoffregen/Time>

DS1307RTC Library

Real Time Clock + Time Library

```
setSyncProvider (RTC.get) ;
```

- What is the DS1307RTC library?

- An Arduino library to interface with the DS1307 RTC over I²C (I2C or IIC).
- It provides lower level access to the get (read) and set (write) the time to and from the RTC chip.

```
#include <DS1307RTC.h>
```

- Syntax:

`RTC.get()` – reads the current date and time from the RTC and returns it as a `time_t` number. (often used inside the `setSyncProvider()` function from the time library.)

- More Information

- https://www.pjrc.com/teensy/td_libs_DS1307RTC.html
- <https://github.com/PaulStoffregen/DS1307RTC>

DS1307RTC Library Setting the Time

Real Time Clock + Time Library

- How to set the time of the RTC (we have already done this for you)
 1. Open the SetTime example:
 - a. File → Examples → DS1307RTC → SetTime
 2. Upload to the Arduino
- This example utilizes the compile time to determine the current time and set it into the RTC.
- When would you have to do this:
 - After turning on the RTC after any power loss (ex battery died, first power on)
 - If the time is significantly off.
 - Adjusting to a different time zone

TimeAlarms Library

Real Time Clock + Time Library

- What is the TimeAlarms library?
 - An Arduino library designed to work with the Time library to run functions at specific times.
 - Can work similar to an alarm clock or a timer and can trigger these alarms once or repeatedly.

```
#include <TimeAlarms.h>
```

- Syntax:

`Alarm.delay`(milliseconds) – check if alarm or timer should run and then delay for specified milliseconds

`Alarm.alarmRepeat`(dayofweek, hours, minutes, seconds, function) – create repeating alarm that calls function at a particular time. Optional dayofweek can be used to only repeat on a certain day.

`Alarm.alarmOnce`(dayofweek, hours, minutes, seconds, function) – create one off alarm to call function at a particular time. Optional dayofweek can be used to only trigger on a certain day.

`Alarm.timerRepeat`(seconds, function) – create a timer that calls function at seconds interval.

`Alarm.timerOnce`(seconds, function) – create one off timer to call function at a particular time once.

- More Information

- https://www.pjrc.com/teensy/td_libs_TimeAlarms.html
- <https://github.com/PaulStoffregen/TimeAlarms>

Activities

Real Time Clock + Time Library

- Read the temperature and humidity and light sensors at 1 minute intervals overnight
 - Read the data and open it in a program (excel) to analyze and plot
- Turn on the LED for 2 minutes at 8:00 AM and 8:00 PM each day

Operators

Lesson 10

Code Analysis – C++ Comparison Operators

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

– Is `i` less than $2^{\text{filterConstant}}$?

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

- More Information:

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



Code Analysis – C++ Arithmetic Operators

Operators

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

- More Information:

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

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

Operators

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

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

++ Auto-increment $i++ \rightarrow i = i + 1$

- Increments the value of a variable by 1

-- Auto-decrement $j-- \rightarrow j = j - 1$

- 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

Operators

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

– Repeat code inside the curly braces $2^{\text{filterConstant}}$ times

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

• Syntax:

Iterator Variable Initialization
↓

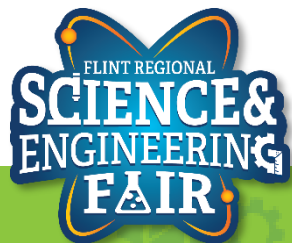
Update Iterator
↓

```
for(initialization; condition; increment)  
{  
    // Do Something  
}
```

↑
End Condition

- 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

Operators

```
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
*=	Multiply Equals	C *= 4	→	C = C * 4
/=	Divide Equals	D /= 5	→	D = D / 5
%=	Modulo Equals	E %= 6	→	E = E % 6

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

Code Analysis – Averaging Filter

Operators

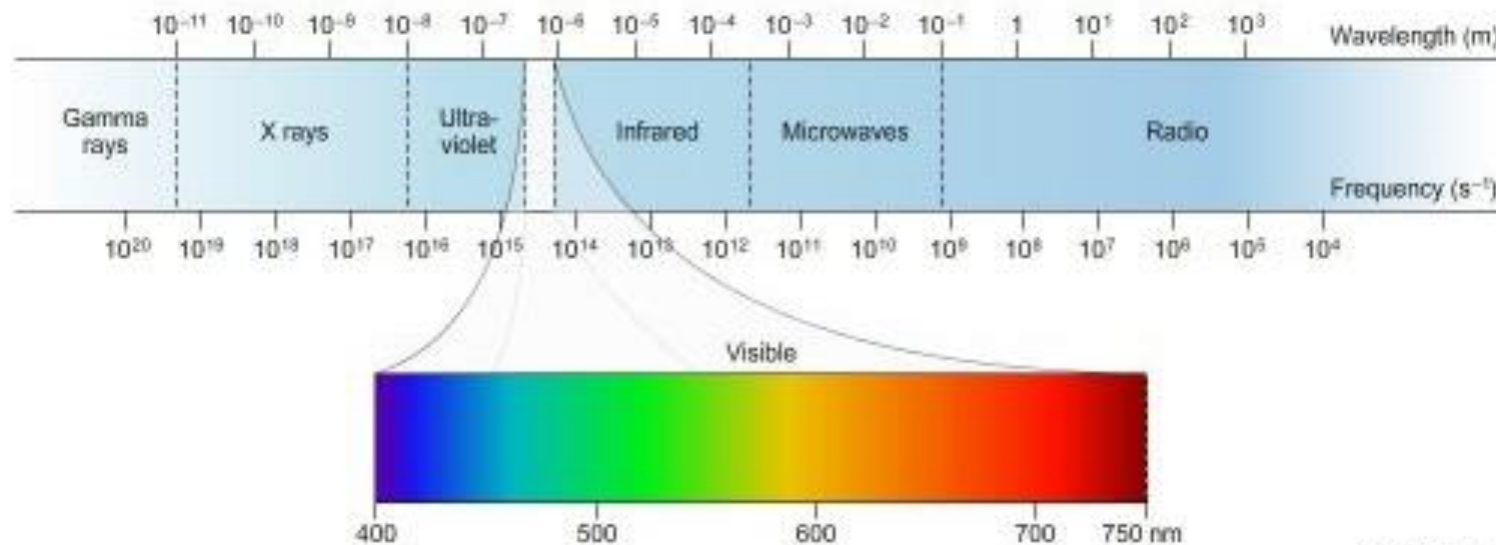
- What is a filter?
 - A filter is used to remove an unwanted component of a signal.
 - For sensor measurements a **low pass filter** is often used to reduce noise or some high frequency component.
 - There are many different types of filters, and numerous ways to implement filters.
- What is averaging?
 - 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/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;
```

Sensors & Applications

Sensors & Applications: Color Sensors

- Think of how Red, Green and Blue combine to make colors
 - Sensors have individual photodiodes that are sensitive to a frequency band of light (color) and measure the intensity of that frequency.
 - Filters are used to make the photodiodes sensitive to limited frequency bands
 - Data from the individual diodes is combined to create a color measurement.

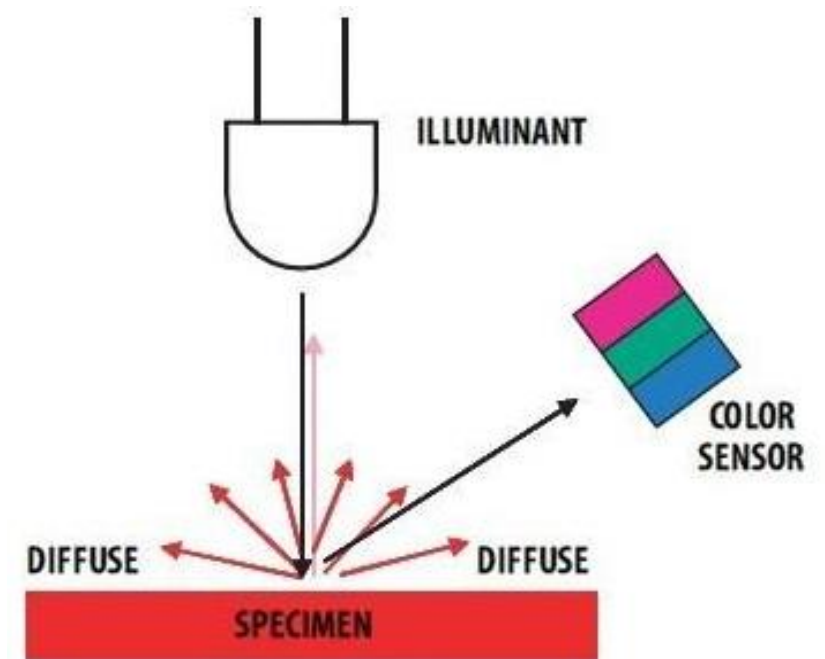


© Seping Learning

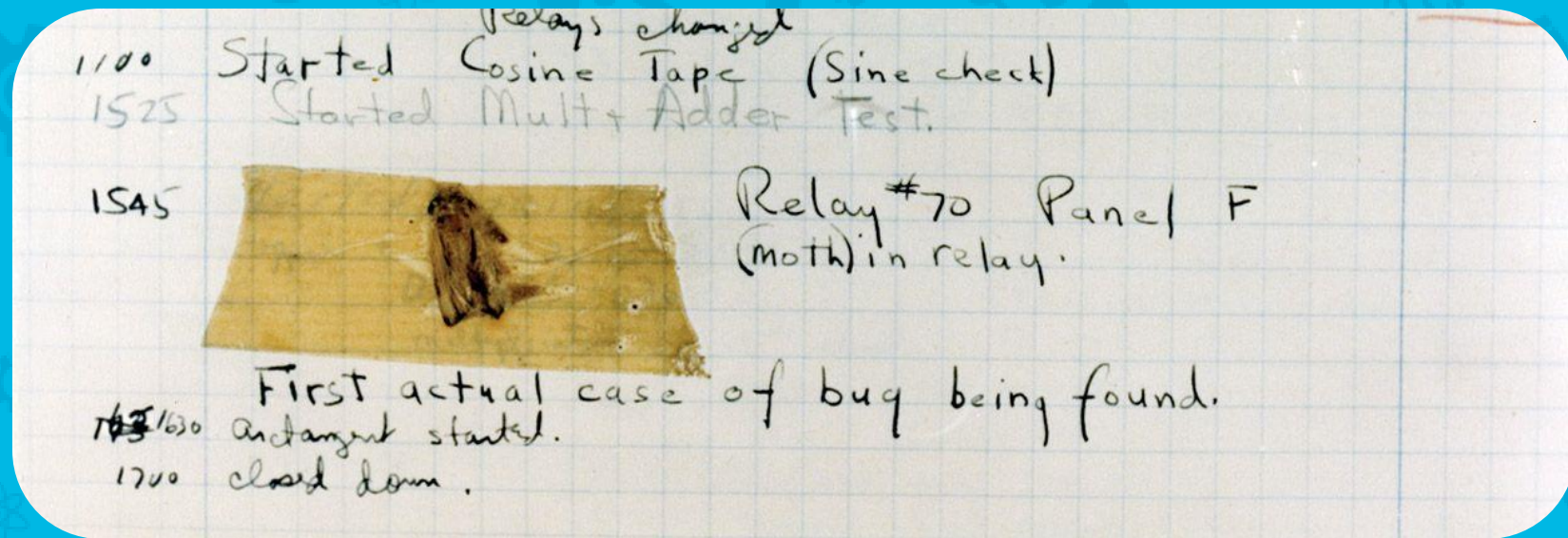
Sensors & Applications: Color Sensors

- Object Color Detection

- Typical Application: light that shines out, reflected wavelengths are measured
- Able to measure intensity of ambient light



- https://wiki.seeedstudio.com/Grove-I2C_Color_Sensor/



Debugging!

Debugging



- Steps
 - Attempt to Upload and view in serial monitor: *DebugProgram_5*

Debugging

- Solution
 - *DebugProgram_5*

incorrect pin for the temperature sensor in sketch
- D3 is the correct pin for the Temp & Humidity sensor

~~const byte dht11Pin = 6;~~ (incorrect line)

const byte dht11Pin = 3;



```
DebugProgram_5 | Arduino 1.8.13
File Edit Sketch Tools Help
DebugProgram_5
/*
 * W3L6_Temp_and_Humidity.ino
 * Get the temperature and humidity values
 * from the DHT11 and send them to the
 * serial port.
 * Temperature 0°C - 50°C
 * Humidity 20%RH - 80%RH
 *
 * Learning Objectives:
 * - installing a library
 * - using a function
 *
 * FRSEF Crash Course V1 Week 3 Lesson 6
 * Written by Chris Bergsneider
 * cbergsneider@flintsciencefair.org
 */
Done compiling.
Sketch uses 5170 bytes (16%) of program storage space.
Global variables use 269 bytes (13%) of dynamic memory.
22 Arduino Uno on COM4
```

Debugging



- Steps
 - Attempt to Upload and view in serial monitor: *DebugProgram_6*

Debugging

- Solution
 - *DebugProgram_6*

Variable not declared in sketch

- temp *was meant to be* tempC

~~Serial.print(temp);~~ (incorrect line)

serial.print(tempC);



```
DebugProgram_6 | Arduino 1.8.13
File Edit Sketch Tools Help
DebugProgram_6
Serial.println("Temperature_°C Temperature_°F Hum:");
dht11.begin(); // Start the DHT11 temperature and
}

void loop()
{
    static unsigned long nextTime = 0; // at what time
    unsigned long time = millis(); // get the current

    if (time >= nextTime)
    {
        float tempC = dht11.readTemperature(); // get the
        float humd = dht11.readHumidity(); // get the h

        Serial.print(temp); // send the temperature in
        Serial.print('\t'); // separate the values with
        Serial.print(CtoF(tempC)); // send the temperat

        'temp' was not declared in this scope
exit status 1
'temp' was not declared in this scope
52 Arduino Uno on COM4
```