Today:

- Announcements
- Day in Space
- Next classes
- One Minute Reports
- Bit of Catch up
- Arduino Part 2
Rules for class today:

- All stay through Analog vs. Digital

- Team members not interested in learning more about Arduino after that may leave to work on team project

- 2 members need to remain in class (more preferred)

- Those that leave shall return by end of class to meet up with team and turn in one minute report

- Questions?
Announcements:

- Great job everyone on Arduino Part #1

- Reached our goal for that class RIGHT?

- Based on 1 Minute Reports, many of you enjoyed the class

- Wish I could have an Arduino for each of you to use but...
Announcements:

- By next Tuesday night, we will finish everything else

- Proposals DUE Monday at 8 AM

- HW #5 Due today

- Another CA

- ITLL Quick Launch

- CoDR presentations
Day in Space: Kounotori (HTV-6)
Day in Space: Kounotori (HTV-6)
Next Tuesday...

Arduinos – Part 3

Bring HW #4 and #5 hardware and Laptops (2 per team)
Tuesday Night...

**Arduinos – 4 DLC 1B70**
Please be early to help setup

**Bring HW #4 and #5 hardware and Laptops (2 per team)**
6:00 – 9:00 PM
One Minute Report Questions:

http://www.goatslive.com
Part 2 – Arduino Test Drive

A. LED Visual Display
B. Analog vs. Digital
C. Potentiometer
D. Balloon Shield Build
- Duplicate the LED circuit three more times

- Note negative leads and connect to **GND Rail**

- Keep color order (Except Blue is purple)

- **Tie all resistors together**
LED Visual Display:

- GND should still be connected to Arduino GND
- Red wire should still be connected to Arduino Pin 9
LED Visual Display:

- Upload same code again and verify all LEDs blink

- Tinker until all are at this point

- Now that we know all the LEDs on our Display are working, let’s use the Arduino to control each LED individually
- Remove wires connecting resistors and Pin 9 from Arduino

- Now what?
LED Visual Display:

- Connect Yellow LED resistor to Pin 9
- Connect Red LED resistor to Pin 7
- Connect Purple LED resistor to Pin 6
- Connect Green LED resistor to Pin 5
**LED Visual Display:**

- Time to modify your sketch

- “Comment out” `int LED = 9;`

- `pinMode` for pins 5, 6, 7, and 9 as OUTPUTs

```java
void setup() {
    // put your setup code here, to run once

    Serial.begin(9600);

    // setup the LED Visual Display
    pinMode(5, OUTPUT);  // Green LED
    pinMode(6, OUTPUT);  // Purple LED
    pinMode(7, OUTPUT);  // Red LED
    pinMode(9, OUTPUT);  // Yellow LED
}
```
**LED Visual Display:**

- **Comment out** `Serial.println`
- **Turn off** LEDs at start of loop
- **Turn on** individual LEDs as shown

```cpp
void loop() {
    // put your main code here, to run repeatedly
    // Turn script running LEDs OFF at beginning
    digitalWrite(5, LOW); // Green LED
    digitalWrite(6, LOW); // Purple LED
    digitalWrite(7, LOW); // Red LED
    digitalWrite(9, LOW); // Yellow LED
    delay(1000);

    digitalWrite(5, HIGH); // Green LED
    delay(500);
    digitalWrite(6, HIGH); // Purple LED
    delay(500);
    digitalWrite(7, HIGH); // Red LED
    delay(500);
    digitalWrite(9, HIGH); // Yellow LED
    delay(500);
}
```
**Blink an LED:**

1. Compile code and check for messages

2. Upload code to Arduino
LED Visual Display:

- Should see Green LED turn on, then Purple, then Red, then Yellow
- Tinker with the delay times until all teams here

PLEASE SAVE YOUR SKETCH FILE

ADC (next slide) or Balloon Shield Testing 142)
**LED Visual Display:**

- Same circuit as bread board but embedded in PCB from HW #4
LED Visual Display:

- Disconnect Arduino from laptop
- Disconnect Breadboard from Arduino
- Connect SHIELD to Arduino
- Line up before squeezing
LED Visual Display:

- Once aligned, gently press two together
LED Visual Display:

- What changes do you have to make to the software?
- None, reconnect to laptop and blink pattern should be same

- Verify then your done!
Review from Arduino Part 1:

- Serial.begin(9600);  - void setup()

- Serial.print( );     - void loop ( )

- Serial.println( );   - void loop ( )

- pinMode(pin#, mode); - void setup ( )

- digitalWrite(pin#, value); - void loop ( )

- delay(time);         - void loop ( )
Part 2 – Arduino Test Drive

A. LED Visual Display
B. Analog vs. Digital
C. Potentiometer
D. Balloon Shield Build
Analog vs. Digital

- Common Interpretation
Digital

- Bits and Bytes, On/Off, 1 or 0, high or low, non-continuous
Analog:

- Voltage, continuous, real-world
Analog vs. Digital

- Low resolution conversion (1 bit or 2 states)
Analog vs. Digital

- More bits, better resolution

Red line – 2 states (1 Bit) = less info
Green line – 16 states (4 Bit) = more info
Analog vs. Digital:

Arduino takes care of this through the ADC

ATmega2560
- 10 Bit ADC
- 16 MHz
- 256 KB Flash
- I^2C & SPI
- -40 to +85C

54 Digital Input/Outputs

USB
5.0 V
Regulator
9V DC
Power In

3.3 V
5.0 V
GND
16 Analog Inputs
Analog vs. Digital

- What it really is…

Analog

0.0 V  1.0 V  2.0 V  3.0 V  4.0 V  5.0 V

Digital

0  204  408  612  816  1023
Analog vs. Digital

- 10-bit conversion has $2^{10}$ (0 to 1023) possible values

\[
\frac{5V}{1023} = 0.00489V
\]

0.00489V * Decimal = Voltage

\[
\text{Decimal} = \frac{\text{Voltage}}{0.00489V}
\]
Analog vs. Digital

- What it really is...

Analog

0.0 V
1.0 V
2.0 V
3.0 V
4.0 V
5.0 V

Digital

0
204
408
612
816
1023

0 = \frac{0.0}{0.00489V}
204 = \frac{1.0}{0.00489V}
408 = \frac{2.0}{0.00489V}
612 = \frac{3.0}{0.00489V}
816 = \frac{5.0}{0.00489V}
1023 = \frac{5.0}{0.00489V}
Analog vs. Digital

- What it really is…

Analog

0.0 V
1.0 V
2.0 V
3.0 V
4.0 V
5.0 V

Digital

0
204
408
612
816
1023

0 = 0.00489V * 0
1 = 0.00489V * 204
2 = 0.00489V * 408
3 = 0.00489V * 612
4 = 0.00489V * 816
5 = 0.00489V * 1023
Digital:

- A state is one unique combination of bits
  - 1 bit – 0 or 1 = 2 states = \(2^1\)
  - 2 bits – 00, 01, 10, 11 = 4 states = \(2^2\)
  - 4 bits – 0000, 0001….1111 = 16 States = \(2^4\)
  - 8 bits = 00000000….11111111 = 256 states = \(2^8\)
  - 10 bits = 0000000000….11111111 = 1024 states = \(2^{10}\)
  - 16 bits = 0000000000000000…111111111111 = 65,536 states = \(2^{16}\)

- More bits provides more precision over a given voltage range

- If it is necessary to record small changes, more precision (bits), is required

- 8 bits is a byte

- 10 bits is how many bytes?
Analog vs. Digital

Level of Precision… Figuring out what you NEED to know

Say you want to hit a barn from 10 feet away with a rock. What do you need to know to do that?
Analog vs. Digital

Hit the barn Yes or No = one bit -> two states

0 = Miss
1 = Hit
Analog vs. Digital

Say you want to know if you hit specific part of the barn…
00 = Right Barn Door
01 = Left Barn Door
10 = Roof
11 = Side barn

Two bits -> Four States
Analog vs. Digital

How many bits (states) does this knowledge require?

4 bits -> 16 States

More resolution costs more memory/storage/bandwidth
Digital:

- A state is one unique combination of bits
  - 1 bit – 0 or 1 = 2 states = 2^1
  - 2 bits – 00, 01, 10, 11 = 4 states = 2^2
  - 4 bits – 0000, 0001….1111 = 16 States = 2^4
  - 8 bits = 00000000….11111111 = 256 states = 2^8
  - 10 bits = 0000000000….1111111111 = 1024 states = 2^{10}
  - 16 bits = 0000000000000000…1111111111111111
    = 65,536 states = 2^{16}

- More bits provides more precision over a given voltage range

- If it is necessary to record small changes, more precision (bits), is required
  - 8 bits is a byte
  - 10 bits is how many bytes?
Analog vs. Digital

- A 10-bit conversion has $2^{10}$ (0 to 1023) possible values

- Resolution is $1/(2^{10} - 1) \times 5V = 1/1023 \times 5V = 0.00489 V$

\[
\frac{5V}{1023} = 0.00489V
\]

\[
0.00489V \times Decimal = Voltage
\]

\[
Decimal = \frac{Voltage}{0.00489V}
\]
Analog vs. Digital

42.0 C temp
Real World

Real World to
Analog Voltage

0C = 0V
50C = 5V

4.20V = 42.0 C

10 bit ADC

5V = 1023
0V = 0

Analog vs. Digital

RockOn! 2016
Analog vs. Digital

42.0°C temp
Real World

Real World to
Analog Voltage

0°C = 0V
50°C = 5V

4.20V = 42.0°C

10 bit ADC

5V = 1023

0V = 0

860 = 1101011100 binary

4.20V = 860

(4.20V / 5.0V * 1023)
= 860.16
= 860

Storage for
later use

ADC = Analog to Digital Converter
= Voltage to Binary
Analog vs. Digital

- Clear as...

- Don’t worry, the more you use it the more sense it will make
Part 2 – Arduino Test Drive

A. LED Visual Display
B. Analog vs. Digital
C. Potentiometer
D. Balloon Shield Build
Potentiometer:

- Arduino Uno
- Monitor
- PC/Mac
- LEDs
- Sensor
- We are now going to add our first sensor – a potentiometer or variable resistor

- It can sweep its output between two voltages it is supplied.
Potentiometer:

- It can sweep its output between two voltages it is supplied.
**Potentiometer:**

- Connect the Red wire from POT to **5V on Arduino**

- Connect Black wire from POT to **GND on Arduino**
**Potentiometer:**

- Connect the Red wire from POT to **5V** on Arduino

- Connect Black wire from POT to **GND** on Arduino
Potentiometer:

- Connect the White wire from POT to A0 on the Arduino
**Potentiometer:**

- Modify your sketch to add the following variable

```cpp
// Definitions
int sensor;

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);

    // setup the LED Visual Display
    pinMode(5, OUTPUT);  // Green LED
```
Potentiometer:

- Read value on pin A0 by using `analogRead`
- `Serial.println` the value on A0
- Change delay to 50 ms
Potentiometer:

- Compile and Upload
- Start Serial Monitor
- LEDs should be blinking fast
- What does the value mean/represent?
Potentiometer:

- Value is digital (integer – whole number) equivalent of analog value

- When the voltage is 0.0V we see “0”

- When the voltage is 5.0V we see “1023”

- What resolution?
**Potentiometer:**

- 10-bit conversion has $2^{10}$ (0 to 1023) possible values

- Resolution is...

$$\frac{1}{(2^{10} - 1)} \times 5V = \frac{5V}{1023} = 0.00489V$$

0.00489V * Decimal = Voltage

- What is the voltage output of the potentiometer if value is 689?

$$0.00489V \times 689 = Voltage$$

3.3692 = Voltage
Potentiometer:

- Modify the sketch to calculate the voltage based on the `analogRead` value and print to the screen

- Will need to create a new variable (float) and use some math

- Printing more than two items to the screen, use...
  > `Serial.print("   ")`  // to print to same line
  > `Serial.print("\t _____")`  // to create tab
  > `Serial.println("   ")`  // to create a new line
Potentiometer:

- Let’s look at the code changes

- **float** because it’s not a whole number

- Verify and Upload
**Potentiometer:**

- Launch Serial Monitor

- Turn potentiometer until you see 689 and verify same value we calculated

- Tinker
Potentiometer:

- What would you have to do to use the potentiometer to control the delay of LED Blink pattern

- Replace time in delay command with sensor value

- Try it
- Let’s look at the code changes
- Everyone here?
- Questions?
- One more step…

```cpp
void loop() {
  // put your main code here, to run repeatedly

  sensor = analogRead(A0);
  sensorVolt = sensor*(5.0/1023);
  Serial.print(sensor);
  Serial.print("\t voltage ");
  Serial.println(sensorVolt);

  // Turn script running leds OFF at beginning
  digitalWrite(5, LOW);  //Green LED
  digitalWrite(6, LOW);  //Purple LED
  digitalWrite(7, LOW);  //Red LED
  digitalWrite(9, LOW);  //Yellow LED

delay(sensor);

digitalWrite(5, HIGH);  //Green LED
  digitalWrite(6, HIGH);  //Purple LED
  digitalWrite(7, HIGH);  //Red LED
  digitalWrite(9, HIGH);  //Yellow LED

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);

delay(sensor);
Potentiometer:

- Let’s look at the code changes
- Everyone here?
- Questions?
- One more step...

```c
void setup()
{
  pinMode(5, OUTPUT); // Ground
  pinMode(6, OUTPUT); // Pulsing
  pinMode(7, OUTPUT); // Reading
  pinMode(9, OUTPUT); // Yellow

  digitalWrite(5, HIGH); // Ground
  delay(sensor);
  digitalWrite(6, HIGH); // Pulsing
  delay(sensor);
  digitalWrite(7, HIGH); // Reading
  delay(sensor);
  digitalWrite(9, HIGH); // Yellow
  delay(sensor);
}
```
Potentiometer:

- Modify the sketch so we can use our LED Visual Display instead of the serial monitor to know what the sensor value / voltage is

- Use a series of if statements to turn LEDs for different values

0.00V to 1.25V  = Turn on Green LED
1.26V to 2.50V  = Turn on Green/Purple LED
2.51V to 3.75V  = Turn on Green/Purple/Red LED
3.75V to 5.00V  = Turn on Green/Purple/Red/Yellow LED
Potentiometer:

- Let’s look at the Sketch

- Comment out previous `digitalWrite` commands

```c
/*
delay(sensor);
digitalWrite(5, HIGH); //Green
delay(sensor);
digitalWrite(6, HIGH); //Purple
delay(sensor);
digitalWrite(7, HIGH); //Red
delay(sensor);
digitalWrite(9, HIGH); //Yellow
delay(sensor);
*/
```
**Potentiometer:**

- Add the following **if statements** to your `void` loop

- Compile and Upload

- Verify LED Display is working by comparing with Serial Monitor and Potentiometer reading

- Tinker until everyone is at this point

```cpp
void loop() {
  // put your main code here, to run repeatedly

  sensor = analogRead(A0);
  sensorVolt = sensor*(5.0/1023);
  Serial.print(sensor);
  Serial.print("\t voltage ");
  Serial.println(sensorVolt);

  // Turn script running leds OFF at begin
  digitalWrite(5, LOW);  //Green LED
  digitalWrite(6, LOW);  //Purple LED
  digitalWrite(7, LOW);  //Red LED
  digitalWrite(9, LOW);  //Yellow LED

  if(sensorVolt > 1.24) {
    digitalWrite(5, HIGH);
  }
  if(sensorVolt > 2.49) {
    digitalWrite(6, HIGH);
  }
  if(sensorVolt > 3.74) {
    digitalWrite(7, HIGH);
  }
  if(sensorVolt > 4.99) {
    digitalWrite(9, HIGH);
  }
  delay(100);
}
/*
  delay(sensor);
*/

digitalWrite(5, HIGH);  //Green LED
delay(sensor);
```
Potentiometer:

- Add the following `if` statements to your `void` loop

```cpp
if(sensorVolt > 1.24) {
  digitalWrite(5, HIGH);
}
if(sensorVolt > 2.49) {
  digitalWrite(6, HIGH);
}
if(sensorVolt > 3.74) {
  digitalWrite(7, HIGH);
}
if(sensorVolt > 4.99) {
  digitalWrite(9, HIGH);
}
delay(100);
```

- Compile and Upload

- Verify LED Display is working by comparing with Serial Monitor and Potentiometer reading

- Tinker until everyone is at this point
Congratulations…

- You have completed your first sensor integration

- They get easier now

- Why?

PLEASE SAVE YOUR SKETCH FILE

bwavo
(clappity clappity)
Part 2 – Arduino Test Drive

A. LED Visual Display
B. Analog vs. Digital
C. Potentiometer
D. Balloon Shield Build
Part 3 – Arduino Road Trip

A. Humidity Sensor
B. Temperature Sensor
C. Pressure Sensor
D. Accelerometers
E. External Temp Sensor
Part 3 – Arduino Road Trip

A. Humidity Sensor
B. Temperature Sensor
C. Pressure Sensor
D. Accelerometers
E. External Temp Sensor
Humidity Sensor:
Humidity Sensor:

- Humidity sensor (or the Darth Vader Sensor)

- It measures moisture in the air, which is great for balloon flights (condensation failures)
**Humidity Sensor:**

*Leave your Balloon Shield attached to Arduino*

- Wire Arduino 5V to Breadboard (BB) 5V PWR Rail

- Wire Arduino GND to BB GND Rail

- Wire Sensor 5V to BB 5V Rail

- Wire Sensor GND to BB GND Rail

- Wire Sensor OUT to Arduino A2
**Humidity Sensor:**

*Leave your Balloon Shield attached to Arduino*

- Wire **Arduino 5V** to Breadboard (BB) 5V PWR Rail
- Wire **Arduino GND** to BB GND Rail
- Wire **Sensor 5V** to BB 5V Rail
- Wire **Sensor GND** to BB GND Rail
- Wire **Sensor OUT** to Arduino A2
Humidity Sensor:

Leave your Balloon Shield attached to Arduino

- Wire Arduino 5V to Breadboard (BB) 5V PWR Rail

- Wire Arduino GND to BB GND Rail

- Wire Sensor 5V to BB 5V Rail

- Wire Sensor GND to BB GND Rail

- Wire Sensor OUT to Arduino A2
Humidity Sensor:
Humidity Sensor:

- Modify sketch to read new sensor on A2

```cpp
// Definitions
int sensor;
float sensorVolt;

void loop() {
    // put your main code here, to run repeatedly.
    sensor = analogRead(A2);
    sensorVolt = sensor*(5.0/1023);
    Serial.print(sensor);
    Serial.print("\t voltage ");
    Serial.println(sensorVolt);

    // Turn script running leds OFF at beginning of loop
    digitalWrite(5, LOW);  //Green LED
}
Humidity Sensor:

- Compile and Upload
- Start Serial Monitor
- Breathe on humidity sensor like Darth Vader
- Watch LEDs on Shield

- Next, let’s convert volts to % humidity
### Humidity Sensor:

- Look at the data sheet to understand output of the sensor

- We know Vout and Vsupply so using algebra

<table>
<thead>
<tr>
<th>Voltage output (1&lt;sup&gt;st&lt;/sup&gt; order curve fit)</th>
<th>$V_{OUT} = (V_{SUPPLY})(0.0062(\text{sensor RH}) + 0.16)$, typical at 25 °C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature compensation</td>
<td>$\text{True RH} = (\text{Sensor RH})/(1.0546 - 0.00216T)$, T in °C</td>
</tr>
</tbody>
</table>
Humidity Sensor:

- % RH is linear as function of voltage

- 100% RH looks like \( \sim 3.7 \) V
Humidity Sensor:

- Here’s the algebra and the equation to code

\[
V_{OUT} = (V_{SUPPLY}) \left( 0.0062 \left( sensorRH \right) + 0.16 \right)
\]

\[
\frac{ \frac{V_{OUT}}{V_{SUPPLY}} - 0.16 }{ 0.0062 } = sensorRH
\]

\[
sensorRH = \frac{ \left( \frac{V_{OUT}}{5.0 \text{ V}} \right) - 0.16 }{ 0.0062 }
\]
**Humidity Sensor:**

```c
// Definitions
int sensor;
float sensorVolt;
float sensorUnits;

// put your main code here, to run repetitively

sensor = analogRead(A2);
sensorVolt = sensor*(5.0/1023);
sensorUnits = (((sensorVolt/5.0)-0.16)/0.0062);
Serial.print(sensor);
Serial.print("\t voltage ");
Serial.print(sensorVolt);
Serial.print("\t units ");
Serial.println(sensorUnits);
```

```c
if(sensorUnits > 10) {
    digitalWrite(5, HIGH);
}
if(sensorUnits > 20) {
    digitalWrite(6, HIGH);
}
if(sensorUnits > 30) {
    digitalWrite(7, HIGH);
}
if(sensorUnits > 40) {
    digitalWrite(9, HIGH);
}
delay(100);
```
**Humidity Sensor:**

- Verify and upload your code
- Launch serial monitor
- Breathe on humidity sensor like Darth Vader
- Watch LEDs on Shield
**Humidity Sensor:**

- While waiting for the rest of the group, play with your new sensor

- Also, look at the data sheet and determine the voltage at maximum humidity

**PLEASE SAVE YOUR SKETCH FILE**
Balloon Shield Build Part 2:

- Disconnect your Balloon Shield and add the Humidity Sensor
Balloon Shield Build Part 2:

- Reconnect your Balloon Shield to the Arduino

- Connect USB and reload code

- Verify same results
Part 3 – Arduino Road Trip

A. Humidity Sensor
B. Temperature Sensor
C. Pressure Sensor
D. Accelerometers
E. External Temp Sensor
Temperature Sensor:
Temperature Sensor:

Temperature sensor is the TMP36 - Temperature Sensor

Will use two on balloon flight
- One internal
- One external

Only working with internal now
Temperature Sensor:

- Leave Balloon Shield Connected to Arduino

- Use same wiring as humidity sensor except middle wire goes to A0
**Temperature Sensor:**

- Leave Balloon Shield Connected to Arduino

- Use same wiring as humidity sensor except middle wire goes to A0
Temperature Sensor:

- Leave Balloon Shield Connected to Arduino

- Use same wiring as humidity sensor except middle wire goes to A0
Temperature Sensor:
- Let’s consult the data sheet for the sensor
- 10 mV/C (0.010V/C)

The TMP35 is functionally compatible with the LM35/LM45 and provides a 250 mV output at 25°C. The TMP35 reads temperatures from 10°C to 125°C. The TMP36 is specified from -40°C to +125°C, provides a 750 mV output at 25°C, and operates to 125°C from a single 2.7 V supply. The TMP36 is functionally compatible with the LM50. Both the TMP35 and TMP36 have an output scale factor of 10 mV/°C.
Temperature Sensor:

- Data sheet also says there is an offset

- For TMP36, Offset = 0.5 Volts

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Offset Voltage (V)</th>
<th>Output Voltage Scaling (mV/°C)</th>
<th>Output Voltage @ 25°C (mV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMP35</td>
<td>0</td>
<td>10</td>
<td>250</td>
</tr>
<tr>
<td>TMP36</td>
<td>0.5</td>
<td>10</td>
<td>750</td>
</tr>
<tr>
<td>TMP37</td>
<td>0</td>
<td>20</td>
<td>500</td>
</tr>
</tbody>
</table>
- So to understand the data, we need to do some math to convert voltage to C

\[
TempC = \frac{(\text{tempVoltage} - 0.5)}{0.01}
\]

Using what we are seeing from our serial monitor, 0.77 Volts, we would get...

\[
TempC = \frac{(0.77 - 0.5)}{0.01} = 27 \, ^\circ C
\]

\[
TempF = TempC \times \frac{9}{5} + 32
\]
// Definitions
    int sensor;
    float sensorVolt;
    float sensorUnits;
    float sensorUnitsC;

void loop() {
    // put your main code here, to run rep

    sensor = analogRead(A0);
    sensorVolt = sensor*(5.0/1023);
    sensorUnitsC = (sensorVolt - 0.5)/(0.01);
    sensorUnits = (sensorUnitsC*(9.0/5.0) + 32);
    Serial.print(sensor);
    Serial.print("\t voltage ");
    Serial.print(sensorVolt);
    Serial.print("\t units ");
    Serial.println(sensorUnits);
}

if(sensorUnits > 78.0) {
    digitalWrite(5, HIGH);
}
if(sensorUnits > 79.0) {
    digitalWrite(6, HIGH);
}
if(sensorUnits > 80.0) {
    digitalWrite(7, HIGH);
}
if(sensorUnits > 81.0) {
    digitalWrite(9, HIGH);
}
delay(100);
Temperature Sensor:

- Build and Upload the code and look at serial monitor

- Should see ~0.77 V

- Put your fingers on temp sensor and lightly squeeze

- Look at monitor and LEDs for change

PLEASE SAVE YOUR SKETCH FILE
Balloon Shield Build Part 3:

- Disconnect your Balloon Shield and add the Temperature Sensor 1

- Note the orientation

Homework
Balloon Shield Build Part 3:

- Solder from bottom of board and then trim leads
Balloon Shield Build Part 3:

- Reconnect your Balloon Shield to the Arduino
- Connect USB and reload code
- Verify same results

<table>
<thead>
<tr>
<th>voltage</th>
<th>units</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>76.60</td>
</tr>
<tr>
<td>0.75</td>
<td>76.60</td>
</tr>
<tr>
<td>0.75</td>
<td>76.60</td>
</tr>
<tr>
<td>0.75</td>
<td>76.60</td>
</tr>
<tr>
<td>0.75</td>
<td>76.60</td>
</tr>
<tr>
<td>0.75</td>
<td>76.60</td>
</tr>
<tr>
<td>0.75</td>
<td>76.60</td>
</tr>
<tr>
<td>0.75</td>
<td>76.60</td>
</tr>
<tr>
<td>0.75</td>
<td>76.60</td>
</tr>
<tr>
<td>0.75</td>
<td>76.60</td>
</tr>
<tr>
<td>0.75</td>
<td>76.60</td>
</tr>
</tbody>
</table>
Part 3 – Arduino Road Trip

A. Humidity Sensor
B. Temperature Sensor
C. Pressure Sensor
D. Accelerometers
E. External Temp Sensor
Pressure Sensor:

Arduino Uno

Monitor ↔ PC/Mac

LEDs

Pressure

Humidity

Temp1
Pressure Sensor:

- Pressure Sensors is fragile and $$$

- A bit tricky to see the markings to install correctly

- Can use it to determine pressure/altitude of payload

- To be safe, please disconnect power from your Arduino
**Pressure Sensor:**

- Pressure sensor orientation
**Pressure Sensor:**

- Pressure sensor orientation

<table>
<thead>
<tr>
<th>Pin 1</th>
<th>Pin 2</th>
<th>Pin 3</th>
<th>Pin 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>$V_{supply}$</td>
<td>OUTPUT+</td>
<td>GND</td>
</tr>
</tbody>
</table>
Pressure Sensor:

- Connect GND to Pin 4, 5V to Pin 2, and Pin 2 to A4 on the Arduino

<table>
<thead>
<tr>
<th>Pin 1</th>
<th>Pin 2</th>
<th>Pin 3</th>
<th>Pin 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>$V_{\text{supply}}$</td>
<td>OUTPUT+</td>
<td>GND</td>
</tr>
</tbody>
</table>
Pressure Sensor:

- Connect GND to Pin 4, 5V to Pin 2, and Pin 3 to A4 on the Arduino

<table>
<thead>
<tr>
<th>Pin 1</th>
<th>Pin 2</th>
<th>Pin 3</th>
<th>Pin 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>V_{supply}</td>
<td>OUTPUT+</td>
<td>GND</td>
</tr>
</tbody>
</table>
**Pressure Sensor:**

- Connect GND to Pin 4, 5V to Pin 2, and Pin 3 to A3 on the Arduino
Pressure Sensor:

- Look at the data sheet to understand output of the sensor

- Known:
  V\text{supply} = 5.0 \text{ V}
  P_{\text{max}} = 15.0 \text{ psi}
  P_{\text{min}} = 0.0 \text{ psi}
  \text{Output}(V) = \text{measured}
  \text{Pressure applied} = \text{solve}

\text{Output} (V) = \frac{0.8 \times V_{\text{supply}}}{P_{\text{max}} - P_{\text{min}}} \times (\text{Pressure}_{\text{applied}} - P_{\text{min}}) + 0.10 \times V_{\text{supply}}
Pressure Sensor:

- Here’s the algebra and the equation to code

\[
\text{Output}(V) = \left( \frac{0.8*V_{\text{SUPPLY}}}{(P_{\text{max}}-P_{\text{min}})} \right) \ast (\text{pressure}_{\text{applied}} - P_{\text{min}}) + 0.10*V_{\text{supply}}
\]

\[
\text{Output}(V) = \left( \frac{0.8*5.0}{15.0-0.0} \right) \ast (\text{pressure}_{\text{applied}} - 0.0) + 0.10*5.0
\]

\[
\text{Output}(V) = \left( \frac{4.0}{15.0} \right) \ast (\text{pressure}_{\text{applied}}) + 0.5
\]

\[
\frac{15.0}{4.0} \ast (-0.5 + \text{Output}(V)) = \text{pressure}_{\text{applied}}
\]
// Definitions
int sensor;
float sensorVolt;
float sensorUnits;

void loop() {
    // put your main code here, to run
    sensor = analogRead(A3);
sensorVolt = sensor*(5.0/1023);
sensorUnits = (sensorVolt-0.5)*(15.0/4.0);
    Serial.print(sensor);
    Serial.print("\t voltage ");
    Serial.print(sensorVolt);
    Serial.print("\t units ");
    Serial.println(sensorUnits);

    if(sensorUnits < 12.20) {
digitalWrite(5, HIGH);
    }
    if(sensorUnits < 10.10) {
digitalWrite(6, HIGH);
    }
    if(sensorUnits < 8.10) {
digitalWrite(7, HIGH);
    }
    if(sensorUnits < 3.10) {
digitalWrite(9, HIGH);
    }
    delay(100);
}
Pressure Sensor:

- Build and Upload

- **DO NOT BLOW** or **DO NOT APPLY PRESSURE**; it will break the sensor

- Use solder sucker

- Also use mouth but be careful not to **spit**

**PLEASE SAVE YOUR SKETCH FILE**
Pressure Sensor:

- While waiting for the rest of the group, play with your new sensor

- Try to get your sensor to zero

PLEASE SAVE YOUR SKETCH FILE
Balloon Shield Build Part 4:

- Disconnect your Balloon Shield and add the Pressure Sensor split header

- Bend leads to hold in place

- Solder from the bottom

- Trim leads
Balloon Shield Build Part 4:

- Install Pressure Sensor into headers
Balloon Shield Build Part 4:

- Reconnect your Balloon Shield to the Arduino

- Connect USB and reload code

- Verify same results