Today:

- Announcements
- Next classes
- Arduino Part 3
Rules for class today:

- All stay through announcements

- 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:

- HW #7 Assigned today

- CA Office Hours – Check your emails

- First round of grades next week

- Proposal grading in process
  - I have graded 5 of the 9
  - Rest graded by tomorrow morning

- Authority to Proceed meetings Wednesday and Thursday please
### Announcements:

<table>
<thead>
<tr>
<th>Wednesday</th>
<th>Thursday</th>
</tr>
</thead>
<tbody>
<tr>
<td>- 8:30</td>
<td>- 11:00 TEAM 1</td>
</tr>
<tr>
<td>- 9:00</td>
<td>- 11:30 TEAM 2</td>
</tr>
<tr>
<td>- 9:30</td>
<td>- 12:00</td>
</tr>
<tr>
<td>- 10:30</td>
<td>- 12:30</td>
</tr>
<tr>
<td>- 11:00 TEAM 6</td>
<td>- 1:00 TEAM 5</td>
</tr>
<tr>
<td>- 11:30 TEAM 3</td>
<td>- 1:30</td>
</tr>
<tr>
<td>- 12:00 TEAM 8</td>
<td>- 2:00 TEAM 9</td>
</tr>
<tr>
<td>- 2:00</td>
<td>- 2:30</td>
</tr>
<tr>
<td>- 2:30</td>
<td>- 3:00 TEAM 7</td>
</tr>
<tr>
<td></td>
<td>- 3:30 TEAM 4</td>
</tr>
</tbody>
</table>
One Minute Report Questions:

http://www.goatslive.com
Day In Space:

- OSIRIS-REx Launch
Day In Space:

- Asteroid in 2018 and return in 2023
Tonight...

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

Bring HW #4 and #5 hardware and Laptops (2 per team)
6 – 8:30 PM

**RSVP on Attendance Sheet – BYOB Pizza at 5:45**
Thursday...

Requirements and HELIOS V

Colorado Space Grant Consortium
Questions?

Colorado Space Grant Consortium
Potentiometer:

- Arduino Uno
- Monitor
- PC/Mac
- LEDs
- Sensor
**Potentiometer:**

- 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.

![Diagram of a potentiometer connected to the analog input of the Arduino](image.png)
Potentiometer:

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

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

- Connect Black wire from POT to GND on Arduino/Shield
Balloon Shield Build Part 1:

Re-connect potentiometer
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
```
- 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...

\[
\frac{1}{2^{10} - 1} * 5V = \frac{5V}{1023} = 0.00489V
\]

\[
0.00489V * Decimal = Voltage
\]

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

\[
0.00489V * 689 = Voltage
\]

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

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

void setup() {
    sensor = analogRead(A0);
    sensorVolt = sensor*(5.0/1023);
    Serial.print(sensor);
    Serial.print("\t Sensor Voltage ");
    Serial.println(sensorVolt);
}
```
**Potentiometer:**

- Launch Serial Monitor

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

- Try turning OFF the LED Switch on the board

- What happens?
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() {
  sensor = analogRead(A0);
  Serial.println(sensor);

  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
  delay(sensor);
  digitalWrite(6, HIGH); // Purple LED
  delay(sensor);
  digitalWrite(7, HIGH); // Red LED
  delay(sensor);
  digitalWrite(9, HIGH); // Yellow LED
}
```
- 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

<table>
<thead>
<tr>
<th>Voltage Range</th>
<th>LED Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00V to 1.25V</td>
<td>Turn on Green LED</td>
</tr>
<tr>
<td>1.26V to 2.50V</td>
<td>Turn on Green/Purple LED</td>
</tr>
<tr>
<td>2.51V to 3.75V</td>
<td>Turn on Green/Purple/Red LED</td>
</tr>
<tr>
<td>3.75V to 5.00V</td>
<td>Turn on Green/Purple/Red/Yellow LED</td>
</tr>
</tbody>
</table>
Potentiometer:

- Let’s look at the Sketch

- Comment out previous `digitalWrite` commands
**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() {
    sensor = analogread(A0);
    sensorVolt = sensor*(5.0/1023);
    Serial.print(sensor);
    Serial.print("\t Sensor Voltage \\
    Serial.println(sensorVolt);

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

    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);
}
```
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
Congratulations...
- You have completed your first sensor integration
- They get easier now
- Why?

**Potentiometer:**

**bwavo**

**(clappity clappity)**

PLEASE SAVE YOUR SKETCH FILE
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:
Humidity Sensor:

- Modify sketch to read new sensor on A2

```cpp
sensor = analogRead(A2);
sensorVolt = sensor * (5.0/1023);
Serial.print(sensor);
Serial.print("\t Sensor Voltage ");
Serial.println(sensorVolt);

if(sensorVolt > 1.25) {
    digitalWrite(5, HIGH);
}
if(sensorVolt > 1.75) {
    digitalWrite(6, HIGH);
}
if(sensorVolt > 2.25) {
    digitalWrite(7, HIGH);
}
if(sensorVolt > 2.75) {
    digitalWrite(9, HIGH);
}
delay(100);
```
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 $V_{out}$ and $V_{supply}$ so using algebra

<table>
<thead>
<tr>
<th>Voltage output (1st order curve fit)</th>
<th>$V_{out} = (V_{supply})(0.0062 \times \text{sensor RH} + 0.16)$, typical at 25 °C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature compensation</td>
<td>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 ~3.7 V
Humidity Sensor:

- Here’s the algebra and the equation to code

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

\[ \left( \frac{V_{OUT}}{\left( V_{SUPPLY} \right)} - 0.16 \right) \]

\[ \frac{0.0062}{0.0062} = sensorRH \]

\[ sensorRH = \left( \frac{V_{OUT}}{\left( 5.0 \ V \right)} - 0.16 \right) \]

\[ \frac{0.0062}{0.0062} \]
Humidity Sensor:

- Modify your code like before

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

sensor = analogRead(A2);
sensorVolt = sensor*(5.0/1023);
RH = (((sensorVolt/5.0)-0.16)/0.0062);
Serial.print(sensor);
Serial.print(" \\	 Sensor Voltage ");
Serial.print(sensorVolt);
Serial.print(" \\	 RH% ");
Serial.println(RH);
if(RH > 10) {
digitalWrite(5, HIGH);
}
if(RH > 25) {
digitalWrite(6, HIGH);
}
if(RH > 40) {
digitalWrite(7, HIGH);
}
if(RH > 55) {
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
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:

- Arduino Uno
- PC/Mac
- LEDS
- Monitor
- Humidity
- Temp1
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:

- Code Changes... A0
- Comment out RH

```c
sensor = analogRead(A0);
sensorVolt = sensor*(5.0/1023);
// RH = (((sensorVolt/5.0)-0.16)/0.16); 
//Serial.print(sensor);

if(sensorVolt > 0.76) {
    digitalWrite(5, HIGH);
}
if(sensorVolt > 0.77) {
    digitalWrite(6, HIGH);
}
if(sensorVolt > 0.78) {
    digitalWrite(7, HIGH);
}
if(sensorVolt > 0.79) {
    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

- Next, let’s convert volts to Celsius
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
Temperature Sensor:

- So to understand the data, we need to do some math to convert voltage to C

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

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

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

\[
TempF = TempC \times \frac{9}{5} + 32
\]
Temperature Sensor:  - Modify your sketch

```cpp
sensor = analogRead(A0);
sensorVolt = sensor*(5.0/1023);
// RH = (((sensorVolt/5.0)-0.16)/0.00
TempC = (sensorVolt - 0.5)/(0.01);
TempF = (TempC*(9.0/5.0) + 32);
Serial.print(sensor);
Serial.print("\t Sensor Voltage ");
Serial.print(sensorVolt);
Serial.print("\t Temp C ");
Serial.print(TempC);
Serial.print("\t Temp F ");
Serial.println(TempF);
```

```cpp
(TempF > 72.0) {
digitalWrite(2, HIGH);
}
(TempF > 73.0) {
digitalWrite(3, HIGH);
}
(TempF > 74.0) {
digitalWrite(4, HIGH);
}
(TempF > 75.0) {
digitalWrite(5, HIGH);
}
delay(100);
```
Temperature Sensor:

- Build and Upload

- Test by touching your temp sensor

Save this sketch as TEMP 2 as we will use this later for Temp 2

PLEASE SAVE YOUR SKETCH FILE
Balloon Shield Build Part 3:

- Disconnect your Balloon Shield and add the Temperature Sensor 1

- Note the orientation
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
Part 3 – Arduino Road Trip

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

Arduino Uno

Monitor ↔ PC/Mac

LEDs

AccelZ
AccelX
Pressure
Humidity
Temp1
Accelerometer:

- Accelerometers are used to detect forces acting on a payload

- This is a 3 axis accelerometer

- Measures g forces in X, Y, and Z directions

- Only have two analog channels left so X and Z
Accelerometer:

- Wire accelerometer as shown

  Vcc is to 3.3V  
  GND is to GND  
  X is to A4  
  Z is to A5
**Accelerometer:**

- Wire accelerometer as shown

Vcc is to **3.3V**
GND is to GND
X is to A4
Z is to A5
Accelerometer:

- Wire accelerometer as shown

Vcc is to 3.3V
GND is to GND
X is to A4
Z is to A5
Accelerometer:

- Looking at the data sheet...

The ADXL335 output is ratiometric, therefore, the output sensitivity (or scale factor) varies proportionally to the supply voltage. At $V_s = 3.6$ V, the output sensitivity is typically 360 mV/g. At $V_s = 2$ V, the output sensitivity is typically 195 mV/g.

The zero g bias output is also ratiometric, thus the zero g output is nominally equal to $V_s/2$ at all supply voltages.

<table>
<thead>
<tr>
<th>SENSITIVITY (RATIOMETRIC)²</th>
<th>Each axis</th>
<th>$V_s = 3$ V</th>
<th>$V_s = 3$ V</th>
<th>$V_s = 3$ V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensitivity at $X_{OUT}$, $Y_{OUT}$, $Z_{OUT}$</td>
<td>270</td>
<td>300</td>
<td>330</td>
<td>mV/g</td>
</tr>
<tr>
<td>Sensitivity Change Due to Temperature³</td>
<td>±0.01</td>
<td></td>
<td></td>
<td>%/°C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ZERO g BIAS LEVEL (RATIOMETRIC)</th>
<th>$V_s = 3$ V</th>
<th>$V_s = 3$ V</th>
<th>$V_s = 3$ V</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 g Voltage at $X_{OUT}$, $Y_{OUT}$</td>
<td>1.35</td>
<td>1.5</td>
<td>1.65</td>
</tr>
<tr>
<td>0 g Voltage at $Z_{OUT}$</td>
<td>1.2</td>
<td>1.5</td>
<td>1.8</td>
</tr>
<tr>
<td>0 g Offset vs. Temperature</td>
<td>±1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOISE PERFORMANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
**Accelerometer:**

- 3.3V/2 is what it should read at “zero G” orientation or 1.65V

- Then 330 mV for every G so…

\[
Gs = \frac{(\text{Accelvoltage} - 1.65 \text{ V})}{(0.330 \text{ V})}
\]

ADXL335

The ADXL335 output is ratiometric, therefore, the output sensitivity (or scale factor) varies proportionally to the supply voltage. At \( V_s = 3.6 \text{ V} \), the output sensitivity is typically 360 mV/g. At \( V_s = 2 \text{ V} \), the output sensitivity is typically 195 mV/g.

The zero g bias output is also ratiometric, thus the zero g output is nominally equal to \( V_s/2 \) at all supply voltages.
Accelerometer:

- Code Changes... *A4, A5*
  - change print statements and
  comment out LED ifs

```c
/*
  if(psi < 12.20) {
    digitalWrite(5, HIGH);
  }
  if(psi < 10.10) {
    digitalWrite(6, HIGH);
  }
  if(psi < 8.10) {
    digitalWrite(7, HIGH);
  }
  if(psi < 3.10) {
    digitalWrite(9, HIGH);
  }
  delay(100);
*/
```

```
// Definitions
int sensorX;
int sensorZ;
float sensorXVolt;
float sensorZVolt;
float Xg;
float Zg;
sensorX = analogRead(A4);
sensorZ = analogRead(A5);
sensorXVolt = sensorX*(5.0/1023);
sensorZVolt = sensorZ*(5.0/1023);
Xg = (sensorXVolt - (3.3/2))/(0.330);
Zg = (sensorZVolt - (3.3/2))/(0.330);
Serial.print("Xg ");
Serial.print(Xg);
Serial.print("\t Zg ");
Serial.println(Zg);
```
Accelerometer:

- Upload your code and launch your serial monitor (no LEDs this time)

- Rotate your breadboard and look for changes in both X and Z

- X up and X down
- Z up and Z down

- Next, let’s convert volts to Gs
Accelerometer:

- Upload your code and launch your serial monitor

- When Z up ~ 1.0G
- When Z down ~ -1.0G
- When X up ~ 1.0G
- When X down ~ -1.0G
- Disconnect your Balloon Shield and add the Accelerometer

- YES, humidity sensor is very close to accel board

- Solder from bottom of board
Balloon Shield Build Part 5:

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

Homework
Part 3 – Arduino Road Trip

A. Humidity Sensor
B. Temperature Sensor
C. Pressure Sensor
D. Accelerometers
E. External Temp Sensor
External Temperature Sensor:
Balloon Shield Build Part 6:

- Add Orange LED to D4
- Red wire to + and Black wire to -
Balloon Shield Build Part 6:

- Add Blue LED to D3
- Red wire to + and Black wire to -
Balloon Shield Build Part 6:

- Add Temp2 to Temp2
- Note wire colors
Balloon Shield Build Part 6:

- Open **Temp2** Sketch; save as Temp2

```cpp
// Definitions
int sensor;
float sensorVolt;
float RH;
float TempC2;
float TempF2;

pinMode(3, OUTPUT); //Blue LED
pinMode(4, OUTPUT); //Orange
pinMode(5, OUTPUT); //Green LED
pinMode(6, OUTPUT); //Purple
pinMode(7, OUTPUT); //Red LED
pinMode(9, OUTPUT); //Yellow
```

```cpp
if (tempF2 > 78.0) {
digitalWrite(3, HIGH);
}
if (tempF2 > 79.0) {
digitalWrite(4, HIGH);
}
if (tempF2 > 80.0) {
digitalWrite(5, HIGH);
}
if (tempF2 > 81.0) {
digitalWrite(6, HIGH);
}
delay(100);
```

```cpp
sensor = analogRead(A1);
sensorVolt = sensor*(5.0/1023);
tempC2 = (sensorVolt - 0.5)/(0.01);
tempF2 = (tempC2*9.0/5.0 + 32);
Serial.print(sensor);
Serial.print("\t Sensor Voltage ");
Serial.print(sensorVolt);
Serial.print("\t TempC2 ");
Serial.print(tempC2);
Serial.println("\t TempF2 ");
Serial.println(tempF2);
```
Balloon Shield Build Part 6:

- Build and upload your sketch
- Temp2 will stick outside your BalloonSat
- LED 3 and 4, will also stick outside your BalloonSat
Balloon Shield Build Part 6:

- Build and upload your sketch

- Temp2 will stick outside your BalloonSat

- LED 3 and 4, will also stick outside your BalloonSat
Part 3 – Arduino Road Trip

A. Humidity Sensor
B. Temperature Sensor
C. Pressure Sensor
D. Accelerometers
E. External Temp Sensor
Great Job!

My fist.... bump it