Gateway To Space

ASEN 1400/ASTR 2500

Class #10

Colorado Space Grant Consortium
Today:

- Announcements

- Next classes

- Arduino Part 2
Announcements:

- Great job everyone during last class
- Reached our goal for that class
- 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 tonight, we will finish everything else

- We will start at 5:30 in the DLC

- Pizza for those who RSVP on attendance sheet

- Special diet requests on attendance sheet

- Please arrive around 5:20 so we are ready to start at 5:30
Announcements:

- **HW 8** assigned today due next Thursday
- Submit via email and bring a copy to class
- There is a trick but there is a solution
- You must figure it out

- HW 7 will be handed out at ATP Meetings

- **Design Document Rev A/B** due March 3rd
Announcements:

**Proposals**
71, 71, 72.2, 73, 77.8, 81, 81.5, 84.9, 88.8, 89.9
Last semester…
46, 65, 68.5, 75.5, 76.5, 77.5, 78, 81.8, 82.5, 85.8

- RFP Requirements
- WHY and HOW sections
- Testing Plan
- Schedule

- **Foolish** not to incorporate feedback or to ignore it
Announcements:

- Working with other (Dan and Tim), you must bring this sheet to them filled out.

- Be sure to thank others helping you.

Consultant Information Sheet

Experienced talented consultants are typically a bit costly (think $100/hour), but are worth it if you use their time well. This paper is intended to highlight key project areas and help to ensure that everyone's time is well used. You should fill out one of these sheets before you consult with key people (e.g. Tim May, Dan Godrick). Note, that a dimensioned drawing fulfills this need for the manufacturing center. Have the TAs and/or professor review the competed sheet if time permits.

Consultant Name:

Team Name:

Date:

Project Goal (short statement):

The approach we envision to achieve our project goals:

What suggestions do you have about our plan-of-attack?

What resources do you recommend that we need?

Our current skill sets are:

What skills do you recommend we acquire?
**Announcements:**

- ATP meetings today and Friday *(missing Team 7)*

<table>
<thead>
<tr>
<th>10 participants</th>
<th>Thu 11</th>
<th>Fri 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team 3</td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>Josh Kirby team#4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team 5</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Team 1</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Team 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team 9</td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>Team 6</td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>No longer available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*No availability entry for Team 2.*
Announcements:

- Bring questions on proposal
- Bring HW 6 so we can order parts
- Be prepared to take hardware home
- Including HW 7 bags
Part 1 – Arduino Driving

A. Arduino Overview
B. Arduino Communication
C. Blink an Led, Change the World
Part 2 – Arduino Test Drive

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

**Arduinos – Part 3 AND 4** DLC 1B70

Please be early to help setup

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

5:30 – 9:00 PM
Next Tuesday...

Requirements and Foam Core
Next Thursday

Guest Lecture
Systems Engineering
HW 8 Due

Colorado Space Grant Consortium
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?
Code Checklist:

<table>
<thead>
<tr>
<th>Code Structure</th>
<th>Example</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>void setup();{}</td>
<td>void setup();</td>
<td></td>
</tr>
<tr>
<td>void loop();{}</td>
<td>void loop();</td>
<td></td>
</tr>
<tr>
<td>Serial.begin(baud rate);</td>
<td>Serial.begin(9600);</td>
<td>Setup</td>
</tr>
<tr>
<td>Serial.println();</td>
<td>Serial.println(&quot;hello world&quot;);</td>
<td>Loop</td>
</tr>
<tr>
<td>Serial.println(&quot;\t&quot;);</td>
<td>Serial.println(&quot;\t Tabs are fun&quot;);</td>
<td>Loop</td>
</tr>
<tr>
<td>Serial.print(value to print);</td>
<td>Serial.print(sensorValue);</td>
<td>Loop</td>
</tr>
<tr>
<td>Serial.print(value to print, # of digits);</td>
<td>Serial.print(sensorValue, 2);</td>
<td>Loop</td>
</tr>
<tr>
<td>//</td>
<td>// This is a comment</td>
<td>Anywhere</td>
</tr>
<tr>
<td>/* */</td>
<td>/* blah blah */</td>
<td>Anywhere</td>
</tr>
<tr>
<td>int integer_name = initial value;</td>
<td>int led = 13;</td>
<td>Definitions</td>
</tr>
<tr>
<td>float decimal_number_name;</td>
<td>float sensorValue;</td>
<td>Definitions</td>
</tr>
<tr>
<td>pinMode(pin, mode);</td>
<td>pinMode(13, OUTPUT);</td>
<td>Setup</td>
</tr>
<tr>
<td>digitalWrite(pin, value);</td>
<td>digitalWrite(13, HIGH);</td>
<td>Loop</td>
</tr>
<tr>
<td>delay(time in millisec);</td>
<td>delay(1000);</td>
<td>Loop</td>
</tr>
<tr>
<td>analogRead(pin);</td>
<td>analogRead(A0);</td>
<td>Loop</td>
</tr>
</tbody>
</table>
**Review from Arduino Part 1:**

<table>
<thead>
<tr>
<th>Function</th>
<th>Setup Code</th>
<th>Loop Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial.begin(9600);</td>
<td>void setup()</td>
<td></td>
</tr>
<tr>
<td>Serial.print();</td>
<td>void loop ()</td>
<td></td>
</tr>
<tr>
<td>Serial.println();</td>
<td>void loop ()</td>
<td></td>
</tr>
<tr>
<td>pinMode(pin#, mode);</td>
<td>void setup()</td>
<td></td>
</tr>
<tr>
<td>digitalWrite(pin#, value);</td>
<td>void loop ()</td>
<td></td>
</tr>
<tr>
<td>delay(time);</td>
<td>void loop ()</td>
<td></td>
</tr>
</tbody>
</table>
Questions?

Colorado Space Grant Consortium
Balloon Shield Build Part 1:

Same circuit as bread board

Did everyone’s sketch work with Balloon Shield?
Balloon Shield Build Part 1:

While connected to Arduino, all connections are same to 5V and Ground, etc through Shield
Part 2 – Arduino Test Drive

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

- Common Interpretation
Analog:

- Voltage, continuous, real-world
Digital:

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

- Arduino takes care of this through the ADC

**ATmega328**
- 10 Bit ADC
- 16 MHz
- 32 KB Flash
- I2C & SPI
- 40 to +85°C

**Features**
- Serial I/O
- External Interrupts
- 14 Digital Input/Outputs
- 6 Analog Inputs
- 3.3 V
- 5.0 V
- GND

**Power Sources**
- USB 3.3 V
- Regulator 5.0 V
- 9V DC Power In
Analog vs. Digital:

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

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

Red line – 2 states (1 Bit) = less info
Green line – 16 states (4 Bit) = more info
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…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:

Do you need to know: Is something there or is it a circle?

10dpi 72dpi 300dpi
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
Analog vs. 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 $\frac{1}{(2^{10} - 1)} \times 5V = \frac{1}{1023} \times 5V = 0.00489$ V

- For a device that is very precise, a 10-bit conversion allows for a higher resolution on the data (high-range accelerometers)

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

$$0.00489V \times Decimal = Voltage$$
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
**Analog vs. Digital:**

42.0 C temp
Real World

→

Real World to
Analog Voltage

0C = 0V
50C = 5V

4.20V = 860
1101011100 binary

(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
**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 potentiometer connected to the analog input of the Arduino](image-url)
**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
}
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…

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

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

    \[
    0.00489V \times 689 = Voltage = 3.3692V
    \]
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
- 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

- 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
Potentiometer:

- Let’s look at the code changes

- Everyone here?

- Questions?

- One more step...

```c
void loop() {
    // put your main code here, to run repeatly
    sensor = analogRead(A0);
    Serial.println(sensor);

    // 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
delay(sensor);

digitalWrite(6, HIGH);  // Purple LED
delay(sensor);

digitalWrite(7, HIGH);  // Red LED
delay(sensor);

digitalWrite(9, HIGH);  // Yellow LED

delay(sensor);
}
```
- 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
Let’s look at the Sketch

- Comment out previous `digitalWrite` commands

```cpp
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
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() {
  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);
  /*
  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

```c
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);
```
Congratulations…

- You have completed your first sensor integration

- They get easier now

- Why?

**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:

- Arduino Uno
- PC/Mac
- Monitor
- LEDs
- 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
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>$V_{out} = (V_{supply})(0.0062(\text{sensor RH}) + 0.16)$, 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 a function of voltage
- 100% RH looks like ~3.7 V
Humidity Sensor:

- Here’s the algebra and the equation to code

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

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

\[
sensorRH = \frac{\left( \frac{V_{OUT}}{\left( 5.0 \ V \right)} - 0.16 \right)}{0.0062}
\]
**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("\t Sensor Voltage ");
Serial.print(sensorVolt);
Serial.print("\t RH% ");
Serial.println(RH);
```

```cpp
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
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:

- 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

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

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

\[
\begin{align*}
\text{TempC} &= \frac{(0.77 - 0.5)}{0.01} \\
&= \frac{0.27}{0.01} \\
&= 27 \degree C
\end{align*}
\]

\[
\text{TempF} = \text{TempC} \times \frac{9}{5} + 32
\]
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
i (TempF > 72.0) {
digitalWrite(2, HIGH);
}
i (TempF > 73.0) {
digitalWrite(3, HIGH);
}
i (TempF > 74.0) {
digitalWrite(4, HIGH);
}
i (TempF > 75.0) {
digitalWrite(5, HIGH);
}
delay(100);
```
Temperature Sensor:

- Build and Upload

- Test by touching your temp sensor

PLEASE SAVE YOUR SKETCH FILE