Part 1 – Arduino Test Drive
Sensors

A. LED Visual Display
B. Analog vs. Digital
C. Balloon Shield Build
D. Thermometer
Part 1 – Arduino Test Drive

Sensors

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

- Arduino Uno
- Monitor
- PC/Mac
- LEDs
**LED Visual Display:**

**Breadboard 101**
- Columns connected
- Rows connected on power rails
- Two sides
- Columns on one side not connected to columns on other side
LED Visual Display:

- Breadboard has power and ground rails
- Individual points on rails (rows) are connected
- One rail, and its points, are independent of other rails
LED Visual Display:

- Also has numbers and letters to coordinate builds
**LED Visual Display:**

- Connect negative lead of LED to **C10** and positive lead to **C11** as shown.

- Connect 330 ohm resistor to positive lead at **D11** and **F11**.

- Connect breadboard wire to negative lead **D10** to **GND Rail**.
LED Visual Display:

- Connect resistor J11 to pin 9 on Arduino

- Connect GND Rail to GND on Arduino as shown
LED Visual Display:

- Upload the same code from the end of Part 1 with led = 9

- Verify the LED blinks

- Tinker with the delay times

PLEASE SAVE YOUR SKETCH FILE
LED Visual Display:

- 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 at this point if you want

- Now that we know all the LEDs on our Display are working, let’s use the Arduino to control each LED individually
LED Visual Display:

- 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

```cpp
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

```c
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

```cpp
void setup() {
    // put your setup code here, to run once:
}
```
LED Visual Display:

- Should see Green LED turn on, then Purple, then Red, then Yellow

- Tinker with the delay times

PLEASE SAVE YOUR SKETCH FILE
Review from Arduino Part 1:

- Serial.begin(9600);
- Serial.print( );
- Serial.println( );
- pinMode(pin#, mode);
- digitalWrite(pin#, value);
- delay(time);
- void setup( )
- void loop ( )
- void loop ( )
- void setup( )
- void loop ( )
Part 1 – Arduino Test Drive

Sensors

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

- Common Interpretation
Analog:

- Voltage, continuous, real-world
Digital:

- Bits and Bytes, On/Off, 1 or 0, high or low, non-continuous
Arduino takes care of this through the ADC

**Analog vs. Digital:**

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

**Components:**
- USB
- 3.3 V Regulator
- 5.0 V Regulator
- 9V DC Power In
- 3.3 V
- 5.0 V
- GND
- 6 Analog Inputs
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
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
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?
- A 10-bit conversion has $2^{10}$ (0 to 1023) possible values

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

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

\[0.00489\text{V} \times \text{Decimal} = \text{Voltage}\]
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

4.20V = 860

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

860 = 1101011100 binary
Analog vs. Digital:

- If this seems a bit confusing – DON’T WORRY!!

- The more you use it the more sense it will make
Part 1 – Arduino Test Drive

Sensors

A. LED Visual Display
B. Analog vs. Digital
C. Potentiometer
D. Balloon Shield Build
E. Thermometer
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 pot;

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

    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...

\[
0.00489V \times \text{Decimal} = \text{Voltage}
\]

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

\[
0.00489V \times 689 = \text{Voltage} \\
3.3692 = \text{Voltage}
\]
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
What would you have to do to use the potentiometer to control the delay of LED Blink pattern

- Replace time in delay command with pot value

- Try it
Potentiometer:

- Let’s look at the code changes

- One more step…

```c
void loop() {
  // put your main code here, to run repeatedly
  pot = analogRead(A0);
potVolt = pot*(5.0/1023);
Serial.print(pot);
Serial.print("\t potVolt ");
Serial.println(potVolt);

  // 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(pot);
digitalWrite(5, HIGH);  // Green LED
delay(pot);
digitalWrite(6, HIGH);  // Purple LED
delay(pot);
digitalWrite(7, HIGH);  // Red LED
delay(pot);
digitalWrite(9, HIGH);  // Yellow LED
delay(pot);
}
```
- 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
**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
Potentiometer:

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

```c
if(potVolt > 1.24) {
    digitalWrite(5, HIGH);
}
if(potVolt > 2.49) {
    digitalWrite(6, HIGH);
}
if(potVolt > 3.74) {
    digitalWrite(7, HIGH);
}
if(potVolt > 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
Potentiometer:

- Connect USB and upload same code
- Same results?
- What happens?
Part 1 – Arduino Test Drive

Sensors

A. LED Visual Display
B. Analog vs. Digital
C. Potentiometer
D. Balloon Shield Build
E. Thermometer
The Balloon Shield will be used on your payload.

It will function the same as your breadboard but wires will not come loose.

After certain points working with the code and wires, we will add items to the balloon shield and retest.
Balloon Shield Build Part 1:
Balloon Shield Build Part 1:

- Add Headers

- Keep flush and perpendicular to board
Balloon Shield Build Part 1:

Solder from bottom of board
Balloon Shield Build Part 1:

- Add LEDs

- Notice + and -
Also notice the LEDs shape on board
Balloon Shield Build Part 1:

- Bend leads over flat so LEDs are flush with top of board
Balloon Shield Build Part 1:

- Trim after soldering – LED LEEDS ONLY!!
Balloon Shield Build Part 1:

Add resistors (no polarity)
Balloon Shield Build Part 1:

- Bend leads so resistors are flush on top of board

- Trim after soldering
Balloon Shield Build Part 1:

- Solder 2 pin headers to D4 and D3
- Short pins to into board
- Solder from the bottom

- LONG PINS SHOULD BE ON TOP
Balloon Shield Build Part 1:

- 1 - 3 pin header connector at HUM
- 2 - 4 pin header connectors at PRES
- 1 - 6 pin header connector at ACCEL
- Solder on the bottom of the board

MUST BE PERPENDICULAR AND FLUSH WITH SHIELD BOARD
Balloon Shield Build Part 1:

- 1 - 3 pin locking header at TEMP2
- Short pins into board.
- Solder from the bottom

MUST INSTALL EXACTLY AS PICTURED.

BACKWARDS WILL RESULT IN TEMP SENSOR OVERHEATING
Balloon Shield Build Part 1:

- 1– 6 pin header at OPENLOG
- Solder on bottom of board

Result should look like this
Balloon Shield Build Part 1:

- Disconnect Arduino from laptop
- Disconnect Breadboard from Arduino
- Connect SHIELD to Arduino
- Line up before squeezing
Balloon Shield Build Part 1:

- Once aligned, gently press two together
Balloon Shield Build Part 1:

Completed product
Part 1 – Arduino Test Drive

Sensors

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

- Arduino Uno
- PC/Mac
- LEDs
- Monitor
- 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 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 A0**
Temperature Sensor:

Leave Balloon Shield Connected 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 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>
**Temperature Sensor:**

- 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…

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

\[
TempF = TempC \times \frac{9}{5} + 32
\]
// 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);
}
**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*
Temperature Sensor:

- Build and Upload

- Test by touching your temp sensor

PLEASE SAVE YOUR SKETCH FILE
Balloon Shield Build Part 2:

- Disconnect your Balloon Shield and add the Temperature Sensor 1

- Note the orientation
- Solder from bottom of board and then trim leads
Balloon Shield Build Part 2:

- Reconnect your Balloon Shield to the Arduino
- Connect USB and reload code
- Verify same results
Part 1 – Arduino Test Drive
Sensors

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