Part 2 – Arduino Test Drive

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

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

<table>
<thead>
<tr>
<th>Function Call</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial.begin(9600);</td>
<td>- void setup()</td>
</tr>
<tr>
<td>Serial.print();</td>
<td>- void loop ()</td>
</tr>
<tr>
<td>Serial.println();</td>
<td>- void loop ()</td>
</tr>
<tr>
<td>pinMode(pin#, mode);</td>
<td>- void setup()</td>
</tr>
<tr>
<td>digitalWrite(pin#, value);</td>
<td>- void loop ()</td>
</tr>
<tr>
<td>delay(time);</td>
<td>- void loop ()</td>
</tr>
</tbody>
</table>
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
Digital

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

![Diagram of digital signal with bits B0 to B6, start, parity, and stop bits labeled.]
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

14 Digital Input/Outputs

External Interrupts

Serial I/O

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

USB
3.3 V
Regulator
5 V
Regulator
9V DC
Power In

3.3 V

5.0 V
GND
6 Analog Inputs

RockOn! 2016
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

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

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

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

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}

Digital

0 → 204 → 408 → 612 → 816 → 1023
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

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

0  204  408  612  816  1023
Digital:

- A state is one unique combination of bits
  - 1 bit – 0 or 1 = $2^1$
  - 2 bits – 00, 01, 10, 11 = $2^2$
  - 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

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

0°C = 0V
50°C = 5V

4.20V = 42.0 °C

10 bit ADC

5V = 1023
0V = 0

Analog vs. Digital

RockOn! 2016
42.0°C temp
Real World

Real World to
Analog Voltage

0°C = 0V
50°C = 5V

4.20V = 860

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

860 = 1101011100 binary

Storage for later use

5V = 1023

0V = 0

ADC = Analog to Digital Converter
= Voltage to Binary

RockOn! 2016
Gateway to Space Theater
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.
**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**
- 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:

- 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…

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

  // 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

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:**

- 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?

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

\[
0.00489V \times 689 = \text{Voltage}
\]

\[
3.3692 = \text{Voltage}
\]
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

```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
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

```cpp
// Potentiometer

void setup() {
  pinMode((10), OUTPUT);
  digitalWrite(10, LOW);
  pinMode((9), OUTPUT);
  pinMode((8), OUTPUT);
  pinMode((7), OUTPUT);
  pinMode((6), OUTPUT);
  pinMode((5), OUTPUT);
  pinMode((4), OUTPUT);
  pinMode((3), OUTPUT);
  digitalWrite((10), LOW);
  digitalWrite((9), LOW);
  digitalWrite((8), LOW);
  digitalWrite((7), LOW);
  digitalWrite((6), LOW);
  digitalWrite((5), LOW);
  digitalWrite((4), LOW);
  digitalWrite((3), LOW);
}

void loop() {
  int input = analogRead(sensor);
  if (input > 200) {
    digitalWrite(9, HIGH);
  } else {
    digitalWrite(9, LOW);
  }
}
```
**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

  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

```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
- Humidity Sensor
- Monitor
- PC/Mac
- LEDs
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
    sensor = analogRead(A2);
    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

    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 = (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

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

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

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

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

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

- Arduino Uno
  - Monitor
  - PC/Mac
  - LEDs
  - 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:

- 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>
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...

\[ \text{TempC} = \frac{(0.77 - 0.5)}{0.01} = 0.27 \quad \text{C} \]

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

- Build and Upload

- Test by touching your temp sensor

PLEASE SAVE YOUR SKETCH FILE
- 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