Gateway To Space

ASEN 1400 / ASTR 2500

Class #9

Colorado Space Grant Consortium
Announcements:

- Hardware from last Tuesday, HW 4 and HW 5 needed today
Announcements:

- All proposals were received on time
- I have reviewed 4 of the 8
- Feedback this week, meetings Friday
- HW #6 Due Friday at one on one meetings
- HW #7 Hardware Kits today
- Attendance (emails) day of class not weeks after
- Grades posted next class and passwords given
Announcements:

- Homework should be submitted as directed on the Syllabus and the assignment

- Microsoft Word, name on top, etc

- Read the instructions and follow them (example 1 page resume)

- Failure to do this, results in more work for Crawford and a lower grade

- HELP emails need to have HELP in subject line

- Write clearly on your One minute reports
Announcements:

- Pick up HW #7 hardware bag today (one per team)
Announcements:

- CoDR Presentation General Feedback

- A few very good and a few very bad

- Practice was evident

- Everyone must speak (beyond their name)

- Presentation is more than the slides (formal)

- Cards are for High School

- Introductions (Team 7)
Next Time...

Arduinos Part 3 (Digital Sensors)

Bring all hardware

Colorado Space Grant Consortium
Thursday Night

Arduinos Part 4 (MicroSD cards)

DLC 1B70

Bring all hardware

Food provided for RSVPs

Colorado Space Grant Consortium
Friday...

One on One meetings

Hardware ordering

Chris’s office

Colorado Space Grant Consortium
Next Tuesday...

Requirements and HASP Student presentation

Colorado Space Grant Consortium
Arduino Part 2: Analog Sensors

Colorado Space Grant Consortium
Rules for class today/Thursday:

- All stay through Analog vs. Digital

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

- A minimum of 2 members need to remain in class

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

- Questions?
LED Visual Display:

- Connect Yellow LED resistor to Pin 5
- Connect Red LED resistor to Pin 4
- Connect Blue LED resistor to Pin 3
- Connect Green LED resistor to Pin 2
LED Visual Display:

- Time to modify your sketch

- “Comment out”
  int LED = 9;

- pinMode for pins 2, 3, 4, and 5 as OUTPUTs
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 rep
  // Serial.println("Hello");
  digitalWrite(2, LOW);  // Yellow LED
  digitalWrite(3, LOW);  // Red LED
  digitalWrite(4, LOW);  // Blue LED
  digitalWrite(5, LOW);  // Green LED
  delay(1000);
  digitalWrite(2, HIGH);  // Yellow LED
  delay(500);
  digitalWrite(3, HIGH);  // Red LED
  delay(500);
  digitalWrite(4, HIGH);  // Blue LED
  delay(500);
  digitalWrite(5, HIGH);  // Green 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 Blue, then Red, then Yellow

- Tinker with the delay times until all teams here

PLEASE SAVE YOUR SKETCH FILE
Part 2 – Arduino Test Drive

A. LED Visual Display
B. Analog vs. Digital
C. Potentiometer
D. Infrared Sensor
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

**Power In**
- 9V DC

**Regulator**
- 5.0 V
- 3.3 V

**USB**
- 5.0 V

**Serial I/O**
- External Interrupts

**Input/Outputs**
- 14 Digital
- 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
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:

How many dots per inch do you need to communicate your message?

240p  1080p
Analog vs. Digital:

Do you need to know: Is something there or is it a circle?
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?
Analog vs. Digital:

- What is the difference between 8-bit and 10-bit conversions?

- An 8-bit conversion has $2^8$ (0 to 255) possible values,

- 8-Bit Resolution is $\frac{1}{(2^8 - 1)} * 5V = \frac{1}{256} * 5V = 0.0196\ V$

\[
\frac{1}{(2^8 - 1)} * 5V = \frac{5V}{256} = 0.0195V
\]

0.0195V * Decimal = Voltage
Analog vs. Digital:

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

- Resolution is $1/(2^{10} - 1) \times 5V = 1/1024 \times 5V = 0.00488\, V$

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

$$\frac{1}{(2^{10} - 1)} \times 5V = \frac{5V}{1024} = 0.00488V$$

$0.00488V \times \text{Decimal} = \text{Voltage}$
Analog vs. Digital:

42.0°C temp
Real World

Real World to
Analog Voltage

4.20V = 42.0°C

0°C = 0V
50°C = 5V

10-bit ADC

5V = 1023

0V = 0
Analog vs. Digital:

42.0 C temp Real World

Real World to Analog Voltage

4.20V = 42.0 C

0C = 0V

50C = 5V

5V = 1023

0V = 0

860 = 1101011100 binary

4.20V = 860

(4.20V / 5.0V * 1024)

= 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. Infrared Sensor
LED Visual Display:

Arduino Uno

Monitor

PC/Mac

LEDs

Pot/Other
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](attachment:potentiometer_diagram.png)
Potentiometer:

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

- Connect the Red wire to +5V, Black to GND, and White (middle) to A0
Potentiometer:

- Connect the Red wire to +5V, Black to GND, and White (middle) to A0
Potentiometer:

- Modify your sketch to add the following variable

```cpp
int sensorValue;
```
**Potentiometer:**

- Read and **Serial.println** the value on A0
- Change delay to 50 ms

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

digitalWrite(2, LOW);  // Yellow LED
digitalWrite(3, LOW);  // Red LED
digitalWrite(4, LOW);  // Blue LED
digitalWrite(5, LOW);  // Green LED

delay(50);
digitalWrite(2, HIGH);  // Yellow LED
delay(50);
digitalWrite(3, HIGH);  // Red LED
delay(50);
digitalWrite(4, HIGH);  // Blue LED
delay(50);
digitalWrite(5, HIGH);  // Green LED
delay(50);
}
```
**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...

\[
\frac{1}{(2^{10} - 1)} \times 5V = \frac{5V}{1024} = 0.00488V
\]

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

\[
0.00488V \times 689 = Voltage
\]

\[
3.36232 = Voltage
\]
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 `sensorValue` to control the delay of LED Blink pattern?

- Replace time in delay command with `sensorValue`.

- 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 repeatedly

    sensorValue = analogRead(A0);
    sensorVoltage = sensorValue*(5.0/1023);

    Serial.print(sensorValue);
    Serial.print("\t Voltage ");
    Serial.println(sensorVoltage);

    digitalWrite(2, LOW); //Yellow LED
    digitalWrite(3, LOW); //Red LED
    digitalWrite(4, LOW); //Blue LED
    digitalWrite(5, LOW); //Green LED

    delay(sensorValue);
    digitalWrite(2, HIGH); //Yellow LED
    delay(sensorValue);
    digitalWrite(3, HIGH); //Red LED
    delay(sensorValue);
    digitalWrite(4, HIGH); //Blue LED
    delay(sensorValue);
    digitalWrite(5, HIGH); //Green LED
    delay(sensorValue);
}
```
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/Blue LED
  2.51V to 3.75V  = Turn on Green/Blue/Red LED
  3.75V to 5.00V  = Turn on Green/Blue/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

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

  sensorValue = analogRead(A0);
  sensorVoltage = sensorValue*(5.0/1023);

  Serial.print(sensorValue);
  Serial.print("\n Voltage ");
  Serial.println(sensorVoltage);

  digitalWrite(2, LOW); //Yellow LED
  digitalWrite(3, LOW); //Red LED
  digitalWrite(4, LOW); //Blue LED
  digitalWrite(5, LOW); //Green LED

  //* delay(sensorValue);
  digitalWrite(2, HIGH);
  delay(sensorValue);
  digitalWrite(3, HIGH);
  delay(sensorValue);
  digitalWrite(4, HIGH);
  delay(sensorValue);
  digitalWrite(5, HIGH);
  delay(sensorValue);

  if(sensorVoltage > 1.24) {
    digitalWrite(2, HIGH); //Yellow LED
  }
  if(sensorVoltage > 2.49) {
    digitalWrite(3, HIGH); //Red LED
  }
  if(sensorVoltage > 3.74) {
    digitalWrite(4, HIGH); //Blue LED
  }
  if(sensorVoltage > 4.99) {
    digitalWrite(5, HIGH); //Green LED
  }
  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
if(sensorVoltage > 1.24) {
  digitalWrite(2, HIGH);  //Yellow LED
}
if(sensorVoltage > 2.49) {
  digitalWrite(3, HIGH);  //Red LED
}
if(sensorVoltage > 3.74) {
  digitalWrite(4, HIGH);  //Blue LED
}
if(sensorVoltage > 4.99) {
  digitalWrite(5, HIGH);  //Green LED
}
delay(100);
```
Potentiometer:

Challenge…
- Come up with a way to see 0.5 volt increments with your LED Visual Display
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. Infrared Sensor
Infrared Sensor:

- Arduino Uno
- Monitor
- PC/Mac
- LEDs
- Pot/Other
- Infrared
Infrared Sensor:

- Let’s replace the potentiometer with a different type of sensor that will detect motion and use our 4 LED to tell us how close an object is to our sensor
**Infrared Sensor:**

- This is an Infrared Proximity Sensor by Sharp (GP2Y0A21YK)

- Emits and detects IR similar to a RADAR
**Infrared Sensor:**

- Connect the 3 wire harness to the IR sensor as shown.

- Connector is “keyed” so it will only install one way.
Infrared Sensor:

- Connect the 3 wire harness to the IR sensor as shown

- Connector is “keyed” so it will only install one way
**Infrared Sensor:**

- Remove the potentiometer from the Arduino and replace with IR Sensor

- Yellow wire goes to A0, Red to 5V and Black to GND
Infrared Sensor:

- At a distance of 5 cm from the “eyes” the detector will reach its max response voltage of ~3.2 volts
Infrared Sensor:

- What do you have to change in the code?

- Just the levels since ~3.2V is the max value

- Using the graph, set levels for 5 cm, 10 cm, 20 cm and 30 cm

- Like the Potentiometer, the IR device is just another analog sensor
**Infrared Sensor:**

- Use the following values for the IR sensor

- Compile and Upload and test!

- Verify LED Visual Display is working by comparing with Serial Monitor

```c
if(sensorVoltage > 0.9) {
    digitalWrite(2, HIGH);
}
if(sensorVoltage > 1.25) {
    digitalWrite(3, HIGH);
}
if(sensorVoltage > 2.25) {
    digitalWrite(4, HIGH);
}
if(sensorVoltage > 3.2) {
    digitalWrite(5, HIGH);
}
delay(100);
```
Infrared Sensor:

- Tinker with the distances until everyone is at the same place

Challenge…
- Come up with a way to see 0.5 volt increments with your LED Visual Display

PLEASE SAVE YOUR SKETCH FILE
Part 2 – Arduino Test Drive

A. LED Visual Display
B. Analog vs. Digital
C. Potentiometer
D. Infrared Sensor
Part 3 – Arduino Road Trip

A. Temperature Sensor
B. Accelerometers
C. Humidity Sensor
D. Pressure Sensor
E. Compass