Gateway To Space

ASEN 1400

Class #07 x1

Colorado Space Grant Consortium
Today:

- Announcements

- Next classes

- Arduino Part 1
Announcements:

- HW #4 and HW #5 due now

- Any one **not** bring items involved with HW #4 HW #5?
Questions?
Part 1 – Arduino Driving Lessons

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
Part 3 – Arduino Road Trip

A. Humidity Sensor
B. Temperature Sensor
C. Pressure Sensor
D. Accelerometers
E. External Temp Sensor
Part 4 – Arduino Race Track

A. SHIELD Integration
B. SD Card Code Integration
C. Data Retrieval
Arduino Overview:

Goals…

1. **Teach you** what I know about Arduinos

2. Make you **comfortable** working with code and **sensors** connected to an Arduino

3. **Record and Retrieve data** on a microSD card

4. Have **fun** doing 1 – 3
Arduino Overview:

- How many of you have ever used a laptop?
- How many of you have ever used MS office?
- How many of you have ever worked with Arduinos?
- How many of you have ever worked with code?
- How many of you have ever worked with sensors?
Part 1 – Arduino Driving Lessons

A. Arduino Overview
B. Arduino Communication
C. Blink an Led, Change the World
Arduino Overview:
Arduino Overview:

“Designing a revolutionary concept in software demanded a computer with extraordinary performance. The Tandy 2000 delivered.” —Bill Gates, Chairman of the Board, Microsoft

Bill Gates has been at the leading edge of personal computing from the very beginning. His company is a leading producer of microcomputer software.

“MOS-Windows is an integrated environment. It will let personal computer users combine individual programs into a powerful, integrated system.

“...When we set out to design MOS-Windows in color, we knew that the Tandy 2000 computer would let us turn an extraordinary product into a work of art. The graphics are sharp and crisp, and gave us a degree of creativity like nothing before.

“Our engineers were quite impressed with the processing speed of the Tandy 2000.5840 microprocessor, too. And while the finished product will utilize the 2000’s Digio Mouse, the well laid out keyboard has helped us speed through the design stage.

“We’re proud of our work. So when we want to show someone how great MOS-Windows really is, we give them a demonstration. On the Tandy 2000:

“...personal computer! Go ahead, watch how much faster today’s most sophisticated program run on the high-technology Tandy 200.

“You can choose from the hottest programs around, too, with our exclusive Express Order Software service.

“Tandy 2000 systems start at $299, and can be leased for only $275 per month*. Come in today and see what you’ve been missing.

*Conditions apply. Check with your local Radio Shack store or dealer. Check out our complete line of microcomputers—from pocket models to laptop Portable, from powerful desktop computers to multi user office systems. We have it all. That’s why we invite comparison!”

Engineered for Excellence! We’re introducing the best in technology for over 40 years. The Tandy 2000 offers twice the speed, graphics resolution, and disk storage of other MOS-DOS systems.

Radio Shack COMPUTER CENTERS
Arduino Overview:

**General Purpose computer**
- Usually has a human in the loop
- Can be reconfigured to do any number of tasks (excel, email, music)

**Embedded Systems**
- Human input not required all the time
  - Takes specific inputs and computes outputs for a very specific application
- Meets real-time goals
  - Heart monitor
  - Automatic braking systems (ABS)
Arduino Overview:

- Arduino is an embedded system
- Board supports an open source environment, lots of assistance available online
- Extremely modular
- Types of Arduinos: Uno, Due, Mega
- Each version has different capabilities
- Lots of analog and digital I/O
Arduino Overview:

Arduino Uno Rev 3
Arduino Overview:
MicroSD Card Shield:

- Arduino Uno
- Micro SD Card OpenLog
- LEDs
- Monitor
- PC/Mac
- LEDs
- Temp 1
- Temp 2
- Pressure
- Humidity
- AccelX
- AccelZ
Arduino Overview:

What’s under the hood?

- USB
- 3.3 V Regulator
- 5.0 V Regulator
- 9V DC Power In
- External Interrupts
- 14 Digital Input/Outputs
- Serial I/O
- ATmega328
  - 10 Bit ADC
  - 16 MHz
  - 32 KB Flash
  - I2C, Serial & SPI
  - 40 to +85C
- 3.3 V
- 5.0 V
- GND
- 6 Analog Inputs
Arduino Overview:
Arduino Overview:

- So what does all that mean?
Arduino Overview:

The Easy Stuff…

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

3.3 V
5.0 V
GND
Arduino Overview:

Arduino (TM) Uno Rev3
Arduino Overview:

The Chips…

ATmega16U
- Handles the USB interface to the computer
- We don’t program this one

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

Arduino(TM) UNO Rev3
Arduino Overview:

Other...

14 Digital Input/Outputs
6 Analog Inputs
External Interrupts
Serial I/O
Arduino Overview:

Arduino(TM) UNO Rev3
Arduino Overview:

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

**Input/Output Features:**
- 14 Digital Input/Outputs
- 6 Analog Inputs
- GND

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

**Serial I/O:**
- External Interrupts

**Power Supply:**
- 3.3 V
- 5.0 V
- GND
Arduino Overview:

Let’s take it for a drive...
MicroSD Card Shield:

- LEDS
- Arduino Uno
- Micro SD Card OpenLog
- Monitor
- PC/Mac
- LEDs
- AccelZ
- AccelX
- Pressure
- Humidity
- Temp 2
- Temp 1
Arduino Overview:

Arduino Uno
Arduino Overview:

Should Have Done this For HW #4

Arduino Uno

Monitor ↔ PC/Mac
**Arduino Overview:**

- Launch the Arduino Software
- A new Sketch opens

![Arduino IDE Screenshot](image-url)
Arduino Overview:

- Select the right board = Tools > Board > Arduino Uno
**Arduino Overview:**

- Select the right board =
  Tools > Board > Arduino Uno
**Arduino Overview:**

- Select a serial port
- For Mac use **Tools > Serial Port > /dev/tty.usbmodemxxx**

- Note: the ‘xxx’ and ‘xx’ above can be any number – it does not matter which number you choose as long as one is selected
**Arduino Overview:**

- Select a serial port

- For PC use **Tools > Serial Port > COMxx**

- Note: the ‘xxx’ and ‘xx’ above can be any number – it does not matter which number you choose as long as one is selected
**Arduino Overview:**

**Port** is big source of frustration for Windows users
- The dreaded “grayed out” port

When it happens…
- Unplug Arduino from laptop
- Close Arduino software
- Plug Arduino back into laptop
- Restart Arduino software
Arduino Overview:

Review the Sketch

Upload

Compile

Serial Monitor

Code

Message Box
Arduino Overview:

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

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

There are three main sections of code in an Arduino sketch:

- Definitions
- Void Setup
- Void Loop
Arduino Overview:

- Definitions are declared **prior to void setup** and can include pin definitions, libraries to include in the sketch, functions, and global variables.

- Most programs declare something, but this is not required. Examples later on…
**Arduino Overview:**

- **void setup** is the first code block in the sketch
- It is run only once
- Used for setup of pin modes, communication initialization, and any code we only want to run one time (i.e. we prepare an SD card by formatting it in the beginning)
**Arduino Overview:**

- **void loop** is the second code block in the Arduino sketch and it continuously repeats itself.

- For code that needs to repeat such as sampling a sensor every couple of seconds.

- Where the primary tasks of the code are carried out.
- Even though this Sketch is not doing anything, it has all the necessary ingredients to be compiled and uploaded
**Arduino Overview:**

1. Compile code and check for messages
   
   ![Screenshot of Arduino IDE showing compiled code]
   
   Should see this at the bottom…
   
   ```
   void setup()
   
   // put your setup code here, to run once:
   ```

   ```
   Done compiling.
   ```

   ```
   Binary sketch size: 466 bytes (of a 32,256 byte maximum)
   ```

   ```
   Arduino Uno on /dev/tty.usbmodemf121
   ```
Arduino Overview:

What is Compiling?

- It checks your code for syntax errors and returns error messages

- Converts human-readable code into machine language (zeroes and ones)

- When you tell the Arduino to upload, it first compiles then uploads (programs) your code (communicating with laptop and Arduino)
Part 1 – Arduino Driving Lessons

A. Arduino Overview
B. Arduino Communication
C. Blink an Led, Change the World
Arduino Overview:
**Arduino Communication:**

Arduino uses **serial communication** to communicate with your laptop.

**Serial communication** is a widely used protocol for transmitting (Tx) and receiving (Rx) binary data and requires a few easy functions to get it started with Arduino.
Arduino Communication:

2. Upload code to Arduino

*If successfully uploaded, you will know that your PC/MAC can communicate with your Arduino*
Arduino Communication:

- Any problems?

- Please wait until this has been completed by each team
Arduino Communication:

What is code?
What is code?
- It is a language to talk with your computer
- Programming languages are like foreign languages
- We say “Hello,” Arduino says
  Serial.begin(9600);
  Serial.print(“Hello”);
- Arduino language is based on C/C++
## 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.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.println(value to print);</code></td>
<td><code>Serial.println(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><code>/* blah blah a comment block blah blah */</code></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_name = initial value;</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>
**Arduino Communication:**

- Modify the sketch to add the following to the `void setup()`

```
void setup() {
    // put your setup code here,
    Serial.begin(9600);
}
```
Arduino Communication:

`Serial.begin()`

- `Serial.begin()` needs us to specify a communication rate (baud rate)
- We use 9600 bits per second, so put 9600 in the parentheses
- `Serial.begin()` is in `setup` because this rate needs to be set only once

```cpp
void setup() {
    // put your setup code here
    Serial.begin(9600);
}
```
**Arduino Communication:**

Modify your sketch to include the following:

```c
void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("hello");
}
```

- **Serial.print()** will just print to the monitor

- **Serial.println()** will print to the monitor and then go to the next line (essentially pushes ‘return’)
**Arduino Communication:**

1. **Compile code and check for messages**

2. **Upload code to Arduino (will check communication with Arduino too)**

![Arduino IDE screenshot showing setup code]

```cpp
void setup() {
    // put your setup code here, to run once:
}
```
Arduino Communication:

- To open the serial monitor, click here

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

void loop() {
    // put your main code here, to run repeatedly:
    Serial.println("hello");
}
```
Arduino Communication:

- You should see this on your serial monitor

![Serial Monitor Output](image.png)
Arduino Communication:

- Any problems?

- Try different text in the print command while waiting for everyone to complete this step
- Congratulations!

- You have now successfully programmed your Arduino – You are a computer programmer
Arduino Preferences:

Other features about Arduino IDE

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

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

Line Number of cursor in code: 6

Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on /dev/uart0modem145121
Arduino Preferences:

Other features about Arduino IDE
Arduino Preferences:

Other features about Arduino IDE

Display line numbers
Arduino Preferences:

Other features about Arduino IDE

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

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

**Other features about Arduino IDE**

![Preferences Window]

**Editor font size:** 14 (requires restart of Arduino)
Ada Lovelace

First computer programmer in 1842

Saw that it was more than number crunching and was a tool capable of great problem solving

“Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent”
Arduino Communication:

Commenting

- Arduino ignores comments but humans read them
- Words become light gray if they commented out
- Put `//` in front of a line to comment out whole line
- To comment out an entire section, put `/*` at the beginning and `*/` at the end

```cpp
//you can type anything you want here!
this is NOT a comment!!  //uh oh!
/*
I can type whatever I want here.
Notice how it's gray?
*/
```
Commenting – MOST IMPORTANT THING!!

It makes your code readable, provides context, helps draft out what you want to next.

- Click to the top of the sketch hit enter to create a new line above void setup()

- Try out your own comment. Insert your name at the top of the sketch. Try out both methods.

Your name here!
Part 1 – Arduino Driving Lessons

A. Arduino Overview
B. Arduino Communication
C. Blink an Led, Change the World
Blink an LED:

- Arduino Uno
- Monitor
- PC/Mac
- LEDs
**Blink an LED:**

- Connect an LED (+ lead) to pin 13 and (- lead) to GND

- Negative lead is usually the shorter lead
Blink an LED:

- Add the following to the definitions area of your sketch - *above*

```cpp
void setup()
{
    // Pin 13 is where the LED is connected
    int led = 13;
}
```

- Add some comments too
Blink an LED:

```cpp
// Pin 13 is where
int led = 13;
```

- Note that "int" turned blue ➔ we are defining a data type
- Arduino knows variable "led" represents an integer
- Anytime "led" is used in the code, Arduino sees 13
  - Ex. 12 + led = 25
- There are many more data types
Blink an LED:

- Add the following to the your sketch in `void setup()`

```cpp
void setup() {
  // put your setup code here, to run once:
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}
```
Blink an LED:

- `pinMode(pin#, mode)`

  - “`pin#`” refers to a specific pin on the Arduino you are wanting to use (in our case pin 13 aka “led”)
  
  - “`mode`” is either INPUT or OUTPUT

  - **OUTPUT** sets up the pin so it can give outputs
  - **INPUT** sets up the pin so it can receive inputs

```cpp
// initialize the led
pinMode(led, OUTPUT)
```

Pin 13 is now an output
Blink an LED:

- Add the following to your sketch in **void loop()**

```c
void loop() {
  // put your main code here,
  Serial.println("hello");
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

- **void loop ()** ...
- Runs once **void setup** is finished
- Loops through the code within **forever**
**Blink an LED:**

```c
Serial.println("Hello");
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
```

digitalWrite(pin#, value)

- **pin#** is whichever pin you are writing to
- **value** can be either HIGH or LOW
  - **HIGH** means the pin is at 5V – “on”
  - **LOW** means the pin is at 0V – “off”
Blink an LED:

```
Serial.println("Hello");
digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
```

delay(time)

- Arduino will wait a specific amount of time (in milliseconds) before going to the next line of code
**Blink an LED:**

1. Compile code and check for messages

2. Upload code to Arduino
Blink an LED:

- Does LED blink?

- Change the delay in the sketch and try again

- Do you see a change?
Blink an LED:

- Remove the LED from PIN 13 and GND
- Another LED on the board should start blinking
- The “L” on the Uno stands for LED
- Do you see this?
Blink an LED:

- Say you wanted to blink an LED on Pin 9, what would you change in the code?

- int LED = 9;
Blink an LED:

- Could you connect LED directly to Pin 9 and GND like for Pin 13?

- No (OK for a few seconds) but why?

- LED requires some current limiting (resistor)
Blink an LED:

- Let’s look at Pin 13 on the schematic

- Follow the line and find a built in 1K resistor
Blink an LED:

- If you can Blink an LED, you can change the world

- Why?

PLEASE SAVE YOUR SKETCH FILE
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
Part 2 – Arduino Test Drive

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

- Where you should have left off from HW #4

- Yellow LED should be blinking with previously sketch

- How do you make the others blink?
**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 until all are at this point

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

```c
void setup() {
  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**

```cpp
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
LED Visual Display:

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

- Tinker with the delay times until all teams here

Please save your sketch file

ADC (next slide) or Balloon Shield Testing (142)
LED Visual Display:

- Same circuit as bread board but embedded in PCB from HW #4 --- Let’s test it
**LED Visual Display:**

- Disconnect Arduino from laptop
- Disconnect Breadboard from Arduino
- Connect SHIELD to Arduino
- Line up before squeezing
LED Visual Display:

- Once aligned, gently press two together
LED Visual Display:

- What changes do you have to make to the software?
- None, reconnect to laptop and blink pattern should be same

- Verify then your done!
Review from Arduino Part 1:

- Serial.begin(9600);
- Serial.print();
- Serial.println();
- pinMode(pin#, mode);
- digitalWrite(pin#, value);
- delay(time);
- void setup();
- void loop();
Part 2 – Arduino Test Drive

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