Gateway To Space
ASEN 1400 / ASTR 2500

Class #08

Colorado Space Grant Consortium
Today:

- Announcements
- CoDR Feedback
- One Minute Report Questions
- Next classes
- Arduino Part 1
Announcements:

- HW #4 due now

- Any one **not** bring items involved with HW#4?
Announcements:

- HW #5 assigned today

- Each team should take one HW#5 bag home today

- Due Next Tuesday
Announcements:

- HW #5 assigned today
Announcements:

- HW #5 assigned today
Announcements:

- HW #5 assigned today
Announcements:

- HW #5 assigned today
Announcements:

- HW #5 assigned today
Announcements:

- HW #5 assigned today
Announcements:

- HW #5 assigned today
Announcements:

- HW #5 assigned today
Announcements:

- HW #5 assigned today
CoDR Feedback:

- Very impressed with every team (for first time speaking)
- No missions really were headed in the wrong direction
- More research on the HOW is recommended for all of you
- Detailed schedule in proposals (roadmap)
- Whiteboard/hand drawings OK but not for proposal
- Introductions (few teams did it)
- Picard (How I know)
- No cue cards (high school)
- Seats in FLEM 104 (ugh)
Proposals...

DUE 2/09 @ 8 AM

Colorado Space Grant Consortium
Next Time...

**Arduinos – Part 2**

- Bring everything from HW #4 and HW#5 to class
- Bring USB cables and same laptops
- FLEM 104

Colorado Space Grant Consortium
Next Thursday...

Arduinos – Part 3

- Bring everything from HW #4 and HW#5 to class
- Bring USB cables and same laptops
- DLC 1B70
Next Thursday PM...

**Arduinos – Part 4 (6 – 8ish)**

- Bring everything from HW #4 and HW#5 to class
- Bring USB cables and same laptops
- DLC 1B70
- Team Representation (2 each)
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 data on to a microSD card
4. Have fun doing 1 – 3
Arduino Overview:

- 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?

- How many of you have ever used a data logger?

- How many of you have ever used a laptop?

- How many of you have ever used MS office?
Part 1 – Arduino Driving Lessons

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

IT'S A SMALL MIRACLE HOW HEWLETT-PACKARD PUT 656K OF MEMORY, LOTUS 1-2-3, WORD PROCESSING, A TELECOMMUNICATIONS MODEM AND COMPLETE IBM CONNECTABILITY INTO

A 9-POUND COMPUTER.

THE PORTABLE.

For your business people had to choose between the power of a desktop computer and the limited capabilities of the first portables. That problem was solved when Hewlett-Packard introduced The Portable.

The Portable is designed with more total memory than most leading desktop personal computers...656K...in fact, that includes 372K of user memory. So, The Portable's built-in business software can work with enormous amounts of data.

1-2-3 from Lotus, America's most popular spreadsheet, file management and business graphics program, is permanently built into The Portable. So is Hewlett-Packard's 1-2-3 word processing program, MemoMaker. Just press the key and you're ready to work.

The Portable even has a built-in modem and easy-to-use telecommunications software to send or receive data using a standard telephone jack. If you use a Hewlett-Packard Touchscreen PC, IBM PC, XT or an IBM compatible, you'll be glad to know that your desktop and The Portable can talk to each other with the simple addition of the Hewlett-Packard Portable-Desktop Link.

The Portable's rechargeable battery gives you 16 hours of continuous usage on every charge. Finally, you can work comfortably on a full-size keyboard and in an easy-to-read 16-line by 80-column screen. And it all fits...The Portable into a simple nine-pound box.

A small miracle...perhaps. But then consider where it came from.

See The Portable and the entire family of personal computers, software and peripherals at your authorized Hewlett-Packard dealer. Call (800) FOR-HFPC for the dealer nearest you.

Setting You Free

HEWLETT-PACKARD

“TI’s Home Computer. This is the one.”

Texas Instruments

1984 AD

Easy to use, the Portable is a fully-featured portable computer with a powerful processor, full-screen graphics, and a built-in modem. It can be used for various applications such as word processing, spreadsheet, and communication.

*HEWLETT-PACKARD is a registered trademark of Hewlett-Packard Company. 1983 and Lotus are trademarks of Lotus Development Corporation.
Arduino Overview:

“Designing a revolutionary concept in software demanded a computer with extraordinary performance. The Tandy 2000 delivered.”

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

“Our newest software product, MSWindows, is in an integrated windowing environment. It will let personal computer users combine individual programs into a powerful, integrated system.

“When we set out to design MSWindows 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 give us a degree of creativity like nothing before.

“Our engineers were quite impressed with the processing speed of the Tandy 2000’s 8086 microprocessor, too. And while the finished product will utilize the 2000’s DigiMouse, the full-sized keyboard has helped us speed through the design stage.

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

“Isn’t it time you enjoyed peak performance from a personal computer? Go ahead, watch how much faster today’s most sophisticated programs run on the high technology Tandy 2000.

“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 $35 per month. Come in today and see what you’ve been missing.

Engineered for Excellence!

“We introduced the best in technology for over 50 years. The Tandy 2000 offers twice the speed, graphics resolution and disk storage of other MS-DOS systems.

Radio Shack
COMPUTER CENTERS
A DIVISION OF TANDY CORPORATION

28
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 a 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:

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

What’s under the hood?

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

GND
14 Digital Input/Outputs
External Interrupts
Serial I/O

3.3 V
5.0 V
GND
6 Analog Inputs

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

Arduino(TM) UNO Rev3
Arduino Overview:

- So what does all that mean?
Arduino Overview:

The Easy Stuff…

- **USB**
- **3.3 V**
- **5.0 V**
- **Regulator**
- **9V DC**
- **Power In**
- **3.3 V**
- **5.0 V**
- **GND**
Arduino Overview:
**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 Overview:

Other…

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

Arduino (TM) UNO Rev3
Arduino Overview:

- **USB**
- **3.3 V Regulator**
- **5 V Regulator**
- **9V DC Power In**
- **14 Digital Input/Outputs**
- **GND**
- **External Interrupts**
- **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:

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

- Arduino Uno
- LEDs
- Monitor
- PC/Mac
- MicroSD Card Shield
  - AccelZ
  - AccelX
  - Pressure
  - Humidity
  - Temp 2
  - Temp 1
Arduino Overview:

Arduino Uno
Arduino Overview:

Arduino Uno

Monitor ↔ PC/Mac
Arduino Overview:

- Launch the Arduino Software

- A new Sketch opens
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
- 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:

- Navigate to: File > Examples > 01.Basics > BareMinimum
Arduino Overview:

- Verify (aka Compile)
- Upload (aka Program the Arduino)
- Serial Monitor
- Code
- Message Box
Arduino Overview:

```c
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.
Arduino Overview:

- 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

Should see this at the bottom…
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 Uno

Monitor ↔ PC/Mac
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*
- Any problems?

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

What is code?
What is code?

- Something you want the computer to do
- There are different programming languages in the same way that we have different languages
- We say “Hello,” Arduino says
  
  ```
  Serial.begin(9600);
  Serial.print(“Hello”);
  ```
- Arduino language is based on C/C++
- Just like with spoken human languages, once you know one learning others is easier.
- Modify the sketch to add the following to the `void setup()`:

```cpp
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
  // initialize serial communication
  Serial.begin(9600);
}
```
Arduino Communication:

Modify your sketch to include the following:

```cpp
void loop() {
  // put your main code here, to run repeatedly:
  // print "Hello" to the Serial Monitor:
  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 Communication:

- To open the serial monitor, click here
**Arduino Communication:**

- You should see this on your serial monitor

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

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

- Any problems?

- Try different text in the print command while waiting for everyone to complete this step
Arduino Communication:

- Congratulations!

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

```plaintext
//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.

### Arduino Communication:

Your name here!
Part 1 – Arduino Driving Lessons

A. Arduino Overview
B. Arduino Communication
C. Blink an Led, Change the World
Blink an LED:
**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* `void setup()`

- Remember, the comments help you and others understand the code

```c
// Pin 13 has an LED connected on most Arduino boards.
// Give it a name:
int led = 13;
```
Blink an LED:

- Note that “int” turned orange → we are defining a data type

- We are telling the Arduino to let a variable called “led” represent 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 your sketch in `void setup()`

```cpp
void setup() {
    // put your setup code here, to initialize digital pins
    pinMode(led, OUTPUT);
    // initialize serial communication
    Serial.begin(9600);
}
```
Blink an LED:

- One use of `void setup()` is to assign pins

- We do this using the function `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
    // print "Hello" to the Serial
    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:

```cpp
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:

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

delay(time)

- Tells the Arduino to 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:

- So if we use any other pin to light up an LED, we need to add a resistor

- We will do that in Part 2

- But first…
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:**

- Remove LED from 9 and GND

- 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 until all teams here

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 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
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() {
  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 repeat

    // 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(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)
Balloon Shield Build Part 1:

- Carefully connect to Arduino

- Line up before squeezing
Balloon Shield Build Part 1:

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

Retest Code…