Announcements:

- CoDR Presentations – DUE 7 AM 09-18-14

- HW 3 and 4 DUE NOW – Anyone forget anything?
Announcements:

- Pick up HW #5 hardware bag today
Announcements:

Additional Class Directions...

1. All Team meetings shall be held on campus.

2. No team meeting or team function associated with this class shall involve alcohol, whether student(s) is/are of legal age or not.

*Any team and/or team member that does not follow these clear directions going forward will result in academic and/or disciplinary action in this class, in said student's department, as well as in the College of Engineering and Applied Science.*
Next Time...

**Arduinos**
Please be early to help setup

**DLC 1B70**
Bring HW #4 hardware and Laptops (2 per team)
HW #03 and #04 DUE
Monday...

Proposals Due 8:00 AM

Colorado Space Grant Consortium
Next Tuesday...

Arduinos Part II

HW #5 Due – Bring to Class
Next Time...

Conceptual Design Review (CoDR)

Proposals Due 9/30 @ 8AM
Arduino Part 1: Type and Blink
Arduino Overview:

- Last class, solder and blink

But…
- How do you change it?
- How can you really use this?
- What could you use?
Arduino Overview:
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, MS/Windows, is an integrated windowing environment. It will let personal computer users combine individual programs into a powerful, integrated system.

“When we set out to design MS/Windows, 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 5.086 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 MS/Windows really is, we give them a demonstration. On the Tandy 2000:

Engineered for Excellence!
We’ve introduced the latest in technology for under $300. The Tandy 2000 offers twice the speed, graphics resolution and disk storage of other MS-DOS systems.

Our new 1985 computer catalog is yours for the asking at any Radio Shack Computer Center or participating Radio Shack store or dealer. Check out our complete line of microcomputers—from pocket models to laptop portables, from powerful desktop computers to multi-user office systems. We have it all. That’s why we invite comparisons.

Radio Shack
COMPUTER CENTERS
A Division of Tandy Corporation
Since 1966 as repair, sold.

12
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
Today + 9/23 and 9/25 + 9/25 PM
**Arduino Overview:**

**What’s under the hood?**

- **USB**
- **3.3 V**
- **Regulator**
- **5.0 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:

Arduino (TM) UNO Rev3
- So what does all that mean?
Arduino Overview:

The Easy Stuff…
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

[Image of an Arduino board]
Arduino Overview:

Arduino (TM) UNO Rev3
Arduino Overview:

- **USB**
- **3.3 V** Regulator
- **5 V** Regulator
- **9V DC** Power In
- **14 Digital Input/Outputs**
- **Serial I/O**
- **External Interrupts**
- **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…
Arduino Overview:

Arduino Uno
Arduino Overview:

- Monitor
- PC/Mac

Arduino Uno
**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
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:

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

- Upload (aka Program the Arduino)
- Verify (aka Compile)
- Code
- Serial Monitor
- 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
- 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)
- **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

   #1

   ![Compiler Screenshot]

   Should see this at the bottom…

   - Done compiling.
   - Binary sketch size: 466 bytes (of a 32,256 byte maximum)
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:

Monitor → Arduino Uno → PC/Mac

Diagram showing the connection between Monitor, Arduino Uno, and 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.
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?

- 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.
**Arduino Communication:**

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

```c
void setup() {
    // put your setup code here
    // initialize serial communication
    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);
}
```
Modify your sketch to include the following:

```c
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:*
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

![Serial Monitor](image)
Arduino Communication:

- You should see this on your serial monitor
- 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

```
// 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?
*/
```
Arduino Communication:

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 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:

```
// Pin 13 has an LED
// give it a name:
int led = 13;
```

- Note that “int” turned orange \(\rightarrow\) 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()`

```java
void setup() {
  // put your setup code here, to initialize the hardware
  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

```c
// initialize the digi
pinMode(led, OUTPUT)
```

Pin 13 is now an output
Blink an LED:

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

```cpp
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:

```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:

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 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. Infrared Sensor
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
LED Visual Display:

Breadboard 101
- Columns connected
- Rows connected on power rails
**LED Visual Display:**

**Our Breadboard**
- Columns connected

- NO Rows connected

- Two sides

- Columns on one side not connected to columns on other side
LED Visual Display:

- Remove LED from 9 and GND
- Connect LED to breadboard **exactly** as shown

Negative ( - ) Lead
LED Visual Display:

- Connect 330 ohm resistor *exactly* as shown

- Span resistor from Side A and Side B

- Connect negative lead from Side A to Side B with jumper wire
**LED Visual Display:**

- Connect end of resistor to Pin 9 on Arduino

- Connect Negative Lead of LED to GND on Arduino
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

- Keep color order
LED Visual Display:

- Connect all grounds together with jumper wires
LED Visual Display:

- Connect all resistor ends 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

LED Visual Display:
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

```c
void loop() {
    // put your main code here, to run repetitively
    // 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