Announcements:

- CoDR Presentations – DUE 7 AM 09-26-13

- HW 3, 4, and 5 DUE NOW – Anyone forget?
Announcements:

- Need 2 laptops used in HW 4 today with USB cables

- Proposal questions – I am available during office hours or by appointment

- HW 6 – only RED stuff is due 10/01
Announcements:

- Heater stuff is now due 10/08
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...

Conceptual Design Review (CoDR)

Proposals Due 9/30 @ 8AM
Functional Block Diagrams
AVR
2MB Flash, Temp, Pressure, XY Accel Onboard

Inputs
- Analog #1
- Analog #2
- Analog #3
- Analog #4

Outputs
- 3.3V
- 5.0V
- 9.0V

Heater

Switch

Data Line

Camera

2 GB

Switch

Data Line

9V

9V

1.5V
Functional Block Diagram
Switch

(3) 9 volts

Heater

LED (w/ resistor)

Battery (internal) → Go Pro → SD Memory

Switch → Battery (internal) → Camera → SD Memory

9 volts

Switch

Servo 1

Servo 2

Arduino

9 volts

Switch

Anemometer

Accelerometer

Pressure

Digital temp

Digital temp (outside)

Humidity

SD Memory
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:
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.

“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 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’s 80486 microprocessor, too. And while the finished product will utilize the 2000’s Day-Mouse, the well laid out keyboard has helped us develop the design.

“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.”

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 $29.95 per month*. Come in today and see what you’ve been missing.

Engineered for Excellence!

We’re introduced the best in technology for over 30 years. The Tandy 2001 offers twice the speed, graphics resolution and disk storage of other MS-DOS systems.

Our new 1980s 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

Copyright © 1990 Radio Shack Computer Centers and/or participating Radio Shack stores and dealers.
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
**Arduino Overview:**

- The Arduino 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
Announcements:

- CoDR Template is on the Website – DUE 7 AM 10-02-12
- New HW #5 on Website – DUE in class 10-04-12
CoDR Template is on the Website – DUE 7 AM 10-02-12

New HW #5 on Website – DUE in class 10-04-12

Announcements:
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 & SPI
- 40 to +85C

3.3 V
5.0 V
GND
6 Analog Inputs
Arduino Overview:

Arduino (TM) UNO Rev3
Arduino Overview:

- So what does all that mean?
Arduino Overview:

The Easy Stuff…
Arduino Overview:

Arduino (TM) Uno Rev3
**Arduino Overview:**

The Chips...

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

Other…

14 Digital Input/Outputs

External Interrupts

Serial I/O

6 Analog Inputs
Arduino Overview:

Arduino (TM) UNO Rev3
Analog vs. Digital
Analog:
- Voltage, continuous, real-world
Digital:
- Bits and Bytes, On/Off, 1 or 0, high or low, non-continuous
**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 = $2^8$
  - 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?
Digital:

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

Red line – 2 bits = less info
Green line – 4 bits = more info
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,

  - Resolution is $1/(2^8 - 1) \times 5\text{V} = 1/255 \times 5\text{V} = 0.0196 \text{V}$
Analog vs. Digital

- A 10-bit conversion has $2^{10}$ (0 to 1024) possible values

  - Resolution is $1/(2^{10} - 1) * 5V = 1/1023 * 5V = 0.00489\ V$

- For a device that is very precise, a 10-bit conversion allows for a higher resolution on the data (high-range accelerometers)
Analog vs. Digital (2 bit ADC, 4 values)

42.0 C temp
Real World

Real World to
Analog Voltage

4.20V = 42.0 C

0C = 0V
50C = 5V

3 = 11 binary

(4.20V / 5.0V * 4) = 3.36 = 3

0V = 0

5V = 3

Storage for
later use

ADC = Analog to Digital Converter
= Voltage to Binary
Analog vs. Digital (10 bit ADC, 1024 values)

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

860 =
1101011100 binary

4.20V = 860

(4.20V / 5.0V * 1024)
= 860.14
= 860

Storage for later use

ADC = Analog to Digital Converter
= Voltage to Binary
Arduino Overview:

Let’s take it for a drive…

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

- Compile
- Serial Monitor
- Upload
- 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:

1. Compile code and check for messages

2. Upload code to Arduino (checking communication with board)
Arduino Overview:

- Any problems?
- Everyone, please wait until this has been completed
**Arduino Overview:**

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

- Definitions
- Void Setup
- Void Loop
Arduino Overview:

**Definitions**

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

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

**Definitions**

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

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

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

- void setup is the first main function in the Arduino sketch  
- It is run only once  
- We use it 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:**

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

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

`void loop()`

- void loop is the second main function in the Arduino sketch
- It continuously repeats itself
- We use it to run code that we wish to repeat such as sampling a sensor every couple of seconds
- This is the heart of the code where the primary tasks are carried out
Arduino Overview:

What is code?

- Code is a way to communicate with a computer
- 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++
Arduino Overview:

Arduino uses serial communication to communicate with your computer.

Serial communication is a widely used protocol for sending and receiving data and requires a few easy functions to get it started with Arduino.
Arduino Overview:

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

```c
void setup() {
  // put your setup code here, to run once:
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}
```
Arduino Overview:

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, to run once:
// initialize serial communication at 9600 bits per second:
    Serial.begin(9600);
}
```
Arduino Overview:

void loop()

- 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’)

```c
void loop() {
  // put your main code here, to run repeatedly:
  // print "Hello" to the Serial Monitor:
  Serial.println("Hello");
}
```
**Arduino Overview:**

What is Compiling?

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

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

- When you tell the Arduino to upload, it automatically compiles before uploading
Arduino Overview:

1. Compile code and check for messages

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

- To open the serial monitor, click here
Arduino Overview:

- You should see this on your serial monitor
Arduino Overview:

- Any problems?

- Everyone, please wait until this has been completed
Arduino Overview:

Commenting

- Comments are for reader’s sake only – Arduino ignores
- Words become light gray if they have been commented out
- Put `//` in front of a line to comment out everything in that line
- If you want to comment out an entire section, put `/*` at the beginning and `*/` at the end

```c
// 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 Overview:

Commenting

- 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.

```cpp
// RockOn Workshop 2013
/* RockOn Workshop 2013 */

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

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

- Add the following to your sketch above `void setup()`
- Remember, the comments help you and others understand the code
Arduino Overview:

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

- 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 we use “led” in the code, we really mean 13
  - Ex. 12 + led = 25
- There are data types other than “int” that we will get into later
Arduino Overview:

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

```cpp
void setup() {
  // put your setup code here, to run once:
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}
```
Arduino Overview:

- Remember that 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
void setup() {
  // put your setup code here
  // initialize the digital pin
  pinMode(led, OUTPUT);
  // initialize serial communication
  Serial.begin(9600);
}
```

Pin 13 is now an output
Arduino Overview:

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

```c
void loop() {
    // put your main code here, to run repeatedly:
    // print "Hello" to the Serial Monitor:
    Serial.println("Hello");
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
    delay(1000);
}
```

- Remember what **void loop** is?
  - Runs once **void setup** is finished
  - Loops through the code within
Arduino Overview:

```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”
**Arduino Overview:**

```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
Arduino Overview:

1. Compile code and check for messages

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

- Does LED blink?
- Change the delay in the sketch and try again
- Do you see a change?
Arduino Overview:

- While powered, remove the LED from PIN 13 and GND

- Another LED on the board blinking should start blinking

- Do you see this?
Arduino Overview:

- Modify sketch to have an LED blink when connected on pin 9
  - Connect the LED (+ lead) to pin 9 and (- lead) to GND

- Compile and Upload the code

- Does the LED blink?


**Arduino Overview:**

- If you can Blink an LED, you can do anything

- Why?
Arduino Overview:
Arduino Overview:
Arduino Overview:

- Remove LED from 9 and GND
- Bring out your breadboard from HW#4
Arduino Overview:

- Connect resistor row to GND on your Arduino
Arduino Overview:

- Connect + lead of LED to Arduino pin 9
Arduino Overview:

- Modify sketch to have LED blink on pin 9 through your breadboard

- Compile and Upload the code

- Do the LEDs blink?

- Switch pin number in sketch and hardware

- Tinker until all teams here
Arduino Overview:

- Understanding what is happening on pin 9
- High ( = 1 ) and Low ( = 0 ) state
- Let’s use the Arduino Serial Monitor to verify
- Change delay to 3 seconds
Arduino Overview:

- Modify the sketch to add the following variable

```cpp
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 9;
int ledState;
```
Arduino Overview:

- Modify the sketch to add the following to the setup

```cpp
void setup() {
  // put your setup code here, to run once:
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}
```
Arduino Overview:

- Modify the sketch to add the following to the loop

```c
void loop() {
    // put your main code here, to run repeatedly:
    digitalWrite(led, HIGH);  // turn the LED on (HIGH is the voltage level)
    // print out the state of the LED:
    ledState = digitalRead(led);
    Serial.print("LED is ");
    Serial.println(ledState);
    delay(3000); // wait for 3 seconds

    digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
    ledState = digitalRead(led);
    Serial.print("LED is ");
    Serial.println(ledState);
    delay(3000); // wait for 3 seconds
}
```
**Arduino Overview:**

- Compile and Upload your code

- Verify blink is every 3 seconds then…

- Click Serial Monitor icon

```c
void loop() {
    // put your main code here, to run repeatedly:
    digitalWrite(led, HIGH);  // turn the LED on (HIGH is the voltage level)
    // print out the state of the LED:
}```
Arduino Overview:

- “LED is 1 or 0” correspond with on or off of LED?
Arduino Overview:

- Now let’s modify our sketch to add an analog input
- Let’s use the potentiometer from HW #4
**Arduino Overview:**

- Connect the Red wire to +5V, Black to GND, and White to A0
Arduino Overview:

- Leave the LED connected to pin 9 but it won’t blink
Arduino Overview:

- Modify your sketch to add the following variable

```cpp
int led = 9;
int ledState;
int sensorValue;
```
Arduino Overview:

- Modify your sketch to add the following to your loop
- Comment out all LED blink and Serial Print for LED

```c
void loop() {
    // put your main code here, to run repeatedly:
    sensorValue = analogRead(A0);
    Serial.print("A0 is ");
    Serial.println(sensorValue);
    // digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
    // print out the state of the LED:
    // ledState = digitalRead(led);
    // Serial.print("LED is ");
    // Serial.println(ledState);
    // delay(3000); // wait for 3000 milliseconds

    digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
    // ledState = digitalRead(led);
    // Serial.print("LED is ");
    // Serial.println(ledState);
    // delay(3000); // wait for 3 seconds
}
```
Arduino Overview:

- Compile and Upload your code
- Click Serial Monitor icon

```cpp
void loop() {
    // put your main code here, to run repeatedly:
    digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
    // print out the state of the LED:
}```
- “A0 is ” should change as you turn the potentiometer
Arduino Overview:

- What does this number mean?

- What resolution is it? 8 bit, 10 bit, 16 bit?

- Modify sketch to Serial Print both 10 bit value and corresponding voltage value

- Recompile and upload

- Tinker until everyone is at this point
Arduino Overview:

- Let’s connect the A0 to the LED with software
- Turn your potentiometer fully counterclockwise
- What do you think will happen?
**Arduino Overview:**

- Modify sketch to add the following line:

```java
void loop() {
    // put your main code here, to run repeatedly:
    sensorValue = analogRead(A0);
    // map it to the range of the analog out:
    Serial.print("A0 is ");
    Serial.println(sensorValue);

    analogWrite(led, sensorValue);  // turn the LED on (HIGH is the voltage level)
```
**Arduino Overview:**

- Compile and Upload
- Activate Serial Monitor
- Begin turning potentiometer clockwise
- What happens?
- Why?
- 10 bit vs. 8 bit? 255 vs 1023? Look at the monitor…
Arduino Overview:

- When 255 is hit, resets

- Let’s map our 10 bit analog to a 8 bit digital

- Add the following variable to your sketch:

```c
int led = 9;
int ledState;
int sensorValue;
int outputValue;
```
**Arduino Overview:**

- Add the following lines to your loop

```c
void loop() {
    // put your main code here, to run repeatedly:
    sensorValue = analogRead(A0);
    // map it to the range of the analog out:
    outputValue = map(sensorValue, 0, 1023, 0, 255);

    Serial.print("A0 is ");
    Serial.print(sensorValue);
    Serial.print(" 	 outputValue ");
    Serial.println(outputValue);

    analogWrite(led, outputValue);  // turn the LED on (HIGH is the voltage level)
}
```
**Arduino Overview:**

- Compile and Upload
- Activate Serial Monitor
- Begin turning potentiometer clockwise
- What happens?
- $255 = 1023$?
**Arduino Overview:**

- Potentiometer is acting like an analog sensor

- We can now hook up any analog sensor and see its value (at least through serial)

- What other sensors can we hook up?

- Let’s add some…

- But first let’s add the Protoshield