Week 8 - interactive electronics with Arduino

PREP WORK

1. Obtain or assemble enough Arduino starter kits for each student containing the following components:
    - Arduino Uno
    - USB A-to-B cable
    - High-brightness LED with 100ohm resistor
    - Breadboard-friendly tactile switch
    - Breadboard-friendly potentiometer/trimpot
    - Micro servo
2. Obtain or assemble enough simple analog sensors for class - such as this 37-in-1 sensor kit
3. Obtain enough laptops for each student to use and ensure they are operational
4. Ensure laptops have latest Arduino IDE installed
5. Bring in or acquire external LCD screen to demonstrate code to class

Outline

1. *Discussion of Curiosity Handbook work since last class session*
2. Introduction to microcontrollers and the Arduino framework
3. Introduction to Arduino board - features, concepts, pins
4. Introduction to Arduino IDE - example sketches, verifying, uploading, serial monitor
5. *Project activities*
    a. Project #1 - turning on an LED with Arduino (digitalWrite())
    b. Project #2 - flashing an LED (setup(), loop(), delay())
    c. Project #3 - turning on/off LED with pushbutton
    d. Project #4 - dimming an LED (PWM, analogWrite() and for loops)
    e. Project #5 - using a potentiometer and the serial monitor
    f. Project #6 - dimming an LED with potentiometer

    g. Project #7 *(optional)* - reading analog sensors (voltage dividers, analogRead())
    h. Project #8 *(optional)* - controlling a servo motor (using libraries)
6. Arduino board variants - what they are, why they are useful, and common variants

Introduction to microcontrollers and the Arduino framework

1. Microcontroller = a tiny computer that executes code you write in order to read inputs and control outputs connected to it.
    a. *Pin* - metal leg that can be soldered to other components or circuit boards. Connects the innards of the microcontroller (IC) to the outside world.
    b. *Inputs* - components that create or alter electrical signals based on real-world phenomena
        i. Examples include buttons and switches, potentiometers, CdS cells, humidity sensors, tilt sensors, accelerometers, GPS
    c. *Outputs* - components that do physical actions whenever they receive electrical signals
        i. Examples include motors, LEDs, speakers
2. Challenges and problems with traditional microcontrollers
    a. *High barrier to entry* - made by and for engineers, no focus on community or approachable design
    b. *Very difficult to program* - generally require learning embedded C and complex toolchains
    c. *Hard to set up* - usually requires setting up lots of extra components just to get the chip to turn on, before any work real work begins.
    d. *Tedious to upload code to* - usually no built-in programming interfaces, requiring users to build even more circuitry just to send programs to chip.
    e. *Lack of community and support* - any forums were dominated by heavily engineering-biased users who tended not to encourage non-engineers to continue in electronics.
3. Arduino framework = in response to problems listed above, created by art and design professors at a new media school in Italy in 2005 as a way to teach their students how to make creative interactive projects and art installations without spending 3+ semesters learning pure electronics
    a. *Arduino circuit board* - a board that contains a powerful microcontroller along with everything needed to make it work with minimal effort. Will be discussed much more in-depth shortly.

b. *Arduino programming language* - much more accessible and easy-to-read language built on to of C. Allows users to get up and running with as little as a single line of code, but does not prevent expert users from going well beyond the basics.
c. *Arduino IDE (*integrated development environment) - program for desktop computers that allows you to write code for your Arduino, check it for problems and send it to your board with a single mouse click.

Introduction to the Arduino board

1. Distribute Arduino Uno boards
2. Point out the various board features
    a. *Microcontroller* - stores and executes small programs for you, making use of inputs and outputs accessible via ...
    b. *Headers* - allow you to use wires to connect components to the microcontroller
        i. Digital - allows you to read and write 0 (LOW, ground, 0V) and 1 (HIGH, +5V)
            1. Avoid using pins 0 and 1 (RX/TX) until you learn more
        ii. Analog - allows you to read a range of voltages between 0-5V
            1. Also work as digital pins if you need more
        iii. Power - allows you to power external components using 5V and 3.3V, as well as connect things to ground.
            1. Ignore IOREF, RESET and Vin until you learn more
    c. *Power section* - creates a steady supply of 5V and 3.3V, though with limits on the amount of current.
        i. DC barrel jack allows for 9-12V external power supply
        ii. Otherwise power will be drawn from USB connection, which is 5V.
    d. *Programming interface* - much smaller microcontroller whose entire job is to translate messages from the USB cable into raw voltages needed for the microcontroller, and vice versa.
    e. *Reset button* - momentarily cuts power, effectively allowing you to restart the program loaded on the microcontroller

Introduction to the Arduino IDE

1. What is it - the IDE is a program you use on your desktop machine to write simple programs for your Arduino board. Also let's you upload these programs easily.
2. Sketch - Arduino's name for "program". Meant to sound less scary, but means the same thing.
3. Connect your Arduino Uno using USB cable - allow Windows to locate and install driver if necessary
4. Open and configure the IDE
    a. Open Arduino IDE
    b. Choose "Arduino Uno" from Tools > Board menu
    c. Choose the appropriate COM port from the Tools > Port menu
5. Check out an example sketch
    a. Load the "Blink" sketch under File > Examples > Basics
    b. Click the "Verify" button to check the sketch for errors
        i. Add a random character somewhere in the program and try again to see what an error looks like.
    c. Click the "Upload" button to send the sketch to the board
        i. Watch the board for flashing lights (RX/TX LEDs)

Project activities

Goal is to cover four main topics:
digital inputs, digital outputs, analog inputs and analog outputs

Set up and use external LCD with laptop to code the sketches along with students

1. Project #1 - turning on an LED with Arduino (digitalWrite())
    a. Distribute breadboards, jumper wires, LEDs and resistors
    b. Add LED and resistor to breadboard
    c. Connect end of LED to Arduino's GND
    d. Connect resistor to Arduino pin 10
    e. Make new sketch
    f. Type in digitalWrite(10, HIGH);
    g. Upload sketch to board

2. Project #2 - flashing an LED (setup(), loop(), delay())
    a. Add setup() and loop() functions
    b. Add pinMode(10, OUTPUT); to setup()
    c. Use digitalRead() and delay() to make LED turn on and off for 1 second each time
    d. Upload sketch to board
3. Project #3 - using a push-button (pull-ups, digitalRead() and variables)
    a. Add switch to breadboard
    b. Connect one side of switch to GND
    c. Connect other side of switch to Arduino pin 9
    d. Add variable named buttonState
    e. Add pinMode(9, INPUT) to setup()
    f. Add digitalWrite(9, HIGH) to setup() to enable internal pull-up
    g. Use digitalRead() in loop() to read state of button and store it in buttonState variable
    h. Use digitalWrite to turn on/off LED using buttonState
    i. Upload sketch to board
4. Project #4 - dimming an LED (PWM, analogWrite() and for loops())
    a. Remove everything inside setup() and loop(), along with the buttonState variable
    b. Add pinMode(10, OUTPUT) to setup()
    c. Add analogWrite(10, 255) to loop() and explain PWM
    d. Upload sketch to board and observe LED brightness
    e. Change the second number inside analogWrite between 0-255 and upload sketch each time and observe change in LED brightness.
    f. Add a for loop to loop(), to increment i from 0-255, with a short delay() in each cycle.
    g. Upload sketch to board and observe LED brightness
    h. Ask students to figure out how to make LED also fade out in brightness (duplicate for loop and make i get smaller)

5. Project #5 - using a potentiometer and the Serial Monitor (analogRead, Serial)
    a. Add potentiometer to breadboard
    b. Connect one outer lug to 5V
    c. Connect other outer lug to GND
    d. Connect center lug to Arduino analog pin 0
    e. Create variable named potValue
    f. Add pinMode(0, INPUT) to setup()
    g. Add Serial.begin(9600); to setup()
    h. Use analogRead() to read value of pot and store it in potValue;
    i. Use Serial.println() to output potValue to serial connection
    j. Upload sketch to board
    k. Open Serial Monitor under Tools > Serial Monitor and observe incoming data
6. Project #6 - dimming an LED using potentiometer (analogWrite(), PWM, map())
    a. Use analogWrite() to write a value between 0-255 to the LED's pin
    b. Use map() to convert potValue range (0-1023) to PWM range (0-255)
    c. Upload sketch to board
    d. Rotate potentiometer and observe change in LED brightness
7. Project #7 - reading analog sensors
    a. Choose a simple analog sensor
    b. Hook it up in the place of the potentiometer, making sure to get +V and GND correct.
    c. If you want, change "potValue" variable to "sensorValue"
8. Project #8 - controlling a servo motor (using libraries)
    a. Connect servo to board using +5V, GND and signal (on pin 10, in place of LED)
    b. Import Servo library using Sketch > Include library > Servo
    c. Create Servo object named myServo
    d. Add myServo.attach(10) to setup() to set up the servo
    e. Use myServo.write() to change position of servo based on potValue
        i. Make sure to use map() to change potValue range from 0-1023 to 0-180
    f. Upload sketch to board

Arduino board variants

1. What is a variant - because the Arduino framework is entirely open-source, many people have remixed and adapted its technology to solve various specific needs.
2. Notable variants
    a. Teensy family (3.2, LC and 2.0 ++) - low-cost, small-footprint boards with extremely powerful microcontrollers on-board. Creators are responsible for thousands of important updates to Arduino core libraries and IDE.
    b. Lilypad, FLORA and GEMMA - circular boards designed for wearables and e-textiles projects, optimized for being sewn into clothing and fabrics.
    c. Trinket - ultra-low-cost, small-footprint boards suitable for small, permanent projects and wearables
    d. Tiny Circuits - low-cost, ultra-small-footprint boards suitable for extremely small projects
    e. Particle (Photon and Electron) - WiFi or 2G/3G-enabled boards that allow for wireless programming and connectivity. Surprisingly low-cost and powerful microcontrollers.
    f. ArduPilot - autopilot board used for controlling RC vehicles and drones