ARDUINO ON RAMP

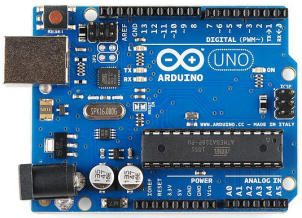
Go from ZERO to ONE learning to use the Arduino environment to build electronics projects. Class will review development boards, software and basic electronic components while walking the group through a project.

The goal of this class is to introduce a beginner to the world of micro-electronics' development boards, get you started with the software, a discuss how to control a few components, put together a straight forward project, and highlight a few available resources so that each participant has some direction on next actions they can take to learn more, build more, connect more when they leave.

Prerequisites: 1) Bring Own laptop. No Chromebooks.
2) Download Arduino IDE:
go to https://www.arduino.cc/en/Guide/HomePage
Scroll down to "Install the Arduino IDE"
Click on whichever operating system you have: Windows, Linux, Mac
Follow step by step directions
3) download files for this class:
https://github.com/haleyma/Arduino-On-Ramp-Class
Make a new folder on your computer(Maybe ArduinoOnRamp?). Go to the link above on github and click the green button to the right, choose download .zip, and extract into your folder.
4. These Arduinos use the CH340 USB serial driver. You need to install that driver on your computer. Follow these instructions from Instructables.Com.

Video examples of project - Intruder Alarm!
Blue: https://youtu.be/giVNOM73hOc
Red: https://youtu.be/Gdnnvital-8

INTRODUCTION

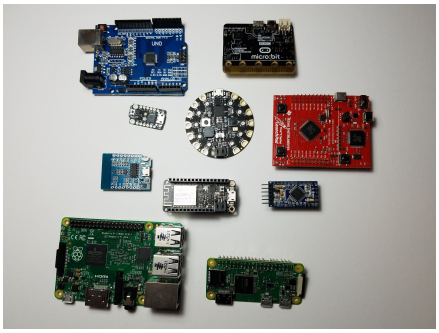


Welcome to the Arduino On Ramp class. We're going to work on the example project from the video where we use a microcontroller, an Ultrasound sensor to detect something close to the sensor and then respond with an alarm sound and flashing lights. The focus is going to be on the basics of using the Arduino environment, so everyone will get the software installed, we'll review how to get the programs into the software, how to download libraries, how to load the program on your board. Each of three components of this project will be reviewed a little and you'll be provided a basic program for each so that you can play with it later on your own. We will review some basic electronics concepts, talk about the components available for microcontrollers, talk about breadboards and discuss your Uno in a little more detail. Finally, we'll review your Intruder Alarm as a project for it's strengths and weaknesses, discuss other projects you might be interested in and make available some resources so that you can further your learning.

What do you know about Arduino? Have you worked with one before?

We will be using an Arduino Uno board “clone”. It is the basic, standard Arduino board and it’s good introduction to the environment. What you are really being introduced to is the world of development boards and there are hundreds. A development board is going to be a electronic circuit board with a collection of components that serve the purpose you want them to. The board will have at least one microprocessor chip, a small amount of memory to store a program and some values, inputs, outputs, a way to program it, a power source. Many companies produce them: Intel had an Edison board, Texas Instruments makes a Launchpad, Lilypad makes small round wearables, Beagle Bone makes boards, Micro:bit makes boards, Adafruit makes boards, there are mega boards with more memory and more pins, there are tiny boards with smaller power efficient chips and their are ones which are teeny enough you could make a little watch to wear on your wrist. Many of these are manufactured and sold to hobbyists, as are the sensors and outputs and the whole ‘industry’ has really grown in the last 10-20 years, along with 3-D printing and the whole Maker movement. They are also used by professionals for prototyping and sometimes in finished commercial products. Arduino has been a big part of that.

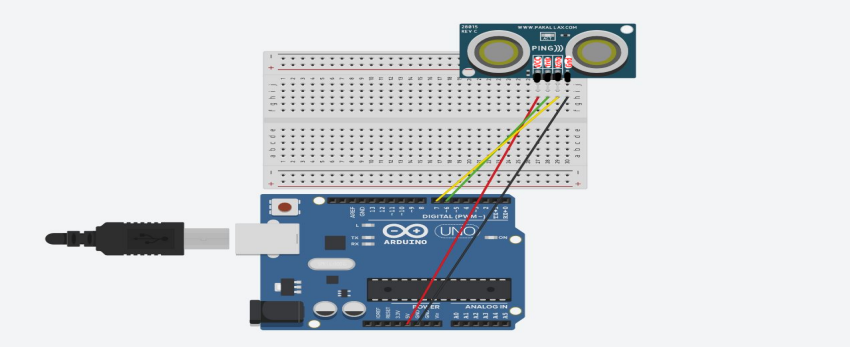
The Arduino website describes itself as an electronics platform. The platform has three major aspects. The first one is the board. As noted above there are lots of similar and some better boards. A group of developers got together, selected a chip that was pretty fast but cost efficient, decided what components and parts a general board might need, added lots of connectors and an uncomplicated way to upload programs - the usb cable. The second aspect was the Arduino software, called the Arduino IDE - Integrated Development Environment. They created some software and made it available to anyone who wants to download it for free. In the IDE you can write programs, edit programs, upload programs, monitor programs that are running. The software runs on Windows, Macintosh, and Linux operating systems. You can import special libraries and access lots of example code - all free and all within the IDE. The Arduino code is their own version of C++ which they call Arduino code. Traditionally, code for embedded circuits was written in the C or C++ programming languages. The Arduino IDE will compile code written in C++, but it also does some things behind the scenes specific to the Arduino modifications. This class isn’t going to teach you programming, but there are lots of free online resources to learn programming. Most of them teach the basics of computational thinking. Most do not teach C++, but you can learn the fundamentals. The third aspect about the Arduino that really established it is that it is open source hardware and software. The term ‘open source’ packs a lot of meaning, but basically all of the source code for the software and the plans of the actual boards are published online. Hackers can hack them, hobbyists can even build their own, developers can modify them for their projects and products. They’ve managed to grow a huge community of Makers and learners who contribute to online resources. So even though there are lots of boards and languages and IDE’s to choose from, this is a good one for a beginner because there are so many resources available to get help and assistance. Lots of other people posting their projects and making the code available. Our project today is a modification of one that someone posted online.



Many people have heard about a little computer called a Raspberry Pi. For a while, there were some limitations controlling some of the microprocessor sensors with Raspberry Pi, but people are figuring out how to do that now. A Raspberry Pi is actually a full computer - you can attach a monitor, keyboard, mouse and upload a full operating system that will run a web browser and graphic interface. The Arduino doesn’t work like that...here, we’ll upload one program and the board will just keep running that one program. We can change the program

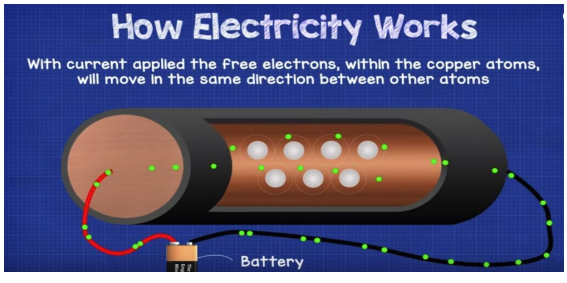
and upload the changes. We can upload a completely different program, but it's one at a time, one program running over and over. The Raspberry Pi computer runs many programs, lots of software, the programs communicate with each other, and it's a whole system with memory. Big difference, but since the Raspberry PI's are small and cheap and can run all this other software...more and more people are using them in their projects.
So, that's our plan. Use some sensors, set up an circuit, control it with a program...and along the way learn how we can use the Arduino to create our own projects.

BREAK OUT: CIRCUITS AND COMPONENTS

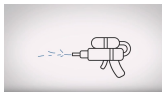


For reference, here are a couple really good videos which go over the basic electrical concepts we'll be covering:
https://www.youtube.com/watch?v=mvuHsu8S6v8&t=211s
https://www.youtube.com/watch?v=mc979OhitAg&t=163s

You don't have to know everything about electrical engineering to build Arduino projects and you will pick up more and more as you go along, but a little knowledge of the basics will help out a lot. Briefly, all materials are made of molecules (hopefully this is not news to you!) and molecules are made of atoms. Atoms have a central nucleus with neutrons and positively charged particles called protons. To conceptualize negatively charged particles called **ELECTRONS**, we often use models that look like little solar systems. There's a central nucleus and outer rings of electrons circling it. It's a fair model for our purposes, and the key factor is that the *OUTER LAYER OF ELECTRONS* can be far enough away from the center that they are not all that closely held. In conductive materials like copper wire, a string of atoms can actually share their outside electrons. An electron might move from one atom to another down the line - all the way down the wire. This is what they do. We create a charge DIFFERENCE from one end to the other with a battery and the ELECTRONS move *TOWARD* the positive charge (opposites attract). Don't think TOO MUCH about this now, but this movement of electrons is call **CURRENT** and is by convention written with an arrow pointing *AWAY* from the positive terminal (although we now know the electrons are actually moving toward the +). No worries - this will always be confusing to you. Kinda wish I hadn't mentioned it.

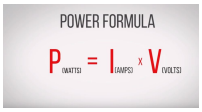


Current is measured in **AMPS** and, with smaller circuits, milliamps(1/1000 amp). Current is the amount of electrons moving in a circuit, and **VOLTAGE** is the force at which the current is moving. A frequently used analogy is comparing electrons to water: Voltage is the pressure and current is the amount of water. Together, voltage and current account for the amount of **POWER** in our circuits. Using the water analogy, consider a squirt gun: squeezing the trigger creates a lot of pressure (VOLTS) and pushes out a thin stream of water (CURRENT) but it would have enough power to turn a water wheel. On the other hand, a garden hose has less pressure (VOLTS) but a bigger stream (CURRENT) and also could turn the



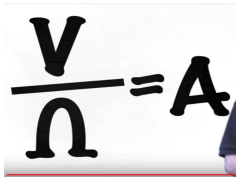
water wheel.

Voltage(Volts) and Current(Amps) are used to determine Power(Watts)

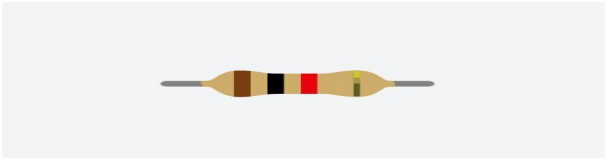


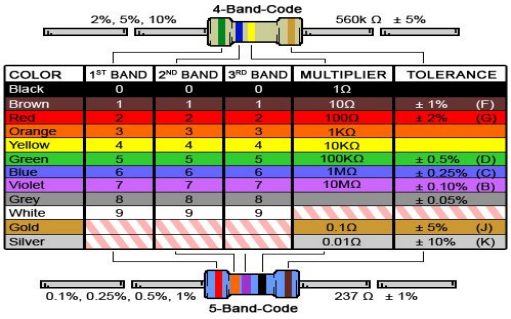
Probably more important to us Arduino users is the relationship between current, voltage and **RESISTANCE**. Resistance is a restriction to the flow of electrons and is measured in Ohms (upside down horseshoe symbol/greek letter). Some resistance can be thought of as "friction" like the natural resistance of wires, different conduction abilities of different materials, length and thickness of wires.

So, what we will often work with is Ohm's Law which you will see written as V=IR where "I" is the symbol for current. Voltage equals Current times Resistance. Voltage is often already known to us, so it can be helpful to think of ***Ohm's Law*** as I=V/R. Current equals Voltage divided by Resistance.



A BATTERY stores energy in chemical form and makes it available for electrical circuits. We'll be getting our electrical power (our +/- differential) from the wall sockets. We also use components called **RESISTORS** in our circuits to control the current. Resistors (a way of restricting current) come in a multitude of sizes. They are marked with colored bands to identify them.





COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)

A particular type of resistor is a potentiometer which also known as a 'variable resistor'. The potentiometer often has a dial that can be turned to vary the resistance. We will use the Arduino 5V output to power our ultrasound sensor.

Your HC-SR04 Ultrasound Sensor: The github repository contains a .pdf for the Ultrasound sensor datasheet. This sensor works by sending a sound/sonar pulse (at 40,000 Hertz) and if the sound wave strikes an object, it will be reflected back and detected by the sensor. Based upon the TIME it takes for the signal to return, the distance between the sensor and the object can be calculated. These sensors are supposed to be accurate between 2cm - 400cm.

Here's a good instructional video:
https://www.youtube.com/watch?v=ZejQOX69K5M

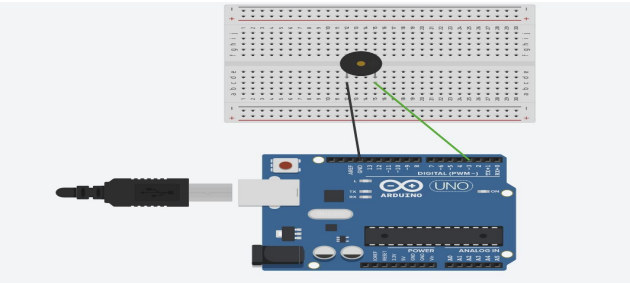
The sensor requires a 5V input (VCC) and we complete the circuit with the ground (Gnd) wire. Your Arduino utilizes a TRIGGER(Trig) pin which transmits the signal and an ECHO pin which is the receiver. The github repository for this class contains a sample program that will allow you to use the SERIAL MONITOR in the Arduino IDE and measure distance. (ArduinoOnRampUStrial.ino) Plus, we'll be installing a LIBRARY in you IDE called NewPing that our Intruder code utilizes.

You will also find a basic example sketch in your Arduino IDE under File -> example -> Built In Examples ->Sensors -> ping.

Open the serial monitor by clicking Tools -> Serial Monitor. Make sure the "baud rate" in the lower right hand corner of your serial monitor is the same as the one in the sketch you are using.

In our Intruder Alarm program, try adjusting the distance at which the sensor will trigger the alert. It is set at 5cm so that we won't have the alarms going off all the time during class.

BREAK OUT: IDE AND SOFTWARE



The Arduino website has all the instruction needed to get your software installed. Go to https://www.arduino.cc/
and on the top horizontal index hover over "RESOURCES" and click on "Getting Started". Scroll down to "Install the Arduino Desktop IDE".
Choose your operating system. This software works with Windows, MAc's, and Linux but not with Chromebooks. Each section gives instructions for downloading and installing the IDE.

Next, return to the getting started page and you will see a menu on the right side of the page. Under "Instructions for Boards", find the "UNO" and click on that. We are going to be using an UNO clone in this class.

Depending on your operating system, you may need to install drivers (Windows) or enable serial ports (Linux). The instruction on the Arduino site walk you through all of this.

Fixing Driver Issues

Some OSX users had trouble. Seemed to be fixed by

- Info on this blog post
- Downloading these drivers (original manufacturer?)
- Signed driver — try this next
- Here are locally downloaded links of these drivers: Windows Mac Linux

Install libraries:

Libraries are pre-written files that help you control how your program runs. We are going to be using libraries that contain helper files/functions for our sensors. The libraries we'll be using are:
NewPing by Tim Eckel
NewTone
SPI
Adafruit_VS1053
SD
Adafruit_NeoPixel
AVR

To install them, go to the horizontal menu in the IDE, click on Sketch -> Include Library -> Manage Libraries. Sometimes you have to scroll up the pop up menu in "include Library" to get to the top where it says "Manage Libraries".

Once the Library Manager has loaded, make sure the "Type" and "Topic" say "All" and then type the name of one of the libraries in the "filter your search" bar.

To Install the NewPing library go to this link:
https://bitbucket.org/teckel12/arduino-new-ping/wiki/Home

Download the NewPing library and install as a .zip file from your Library Manager:
In the Arduino IDE: Sketch -> Include Library -> Manage Libraries -> add .ZIP library ->browse to where you downloaded the file.
Close and re-open your Arduino IDE after installing a library.

If you are interested in learning more about git, here's a getting started video:
https://git-scm.com/video/get-going
And here's a good Udacity course on version control:
https://classroom.udacity.com/courses/ud775

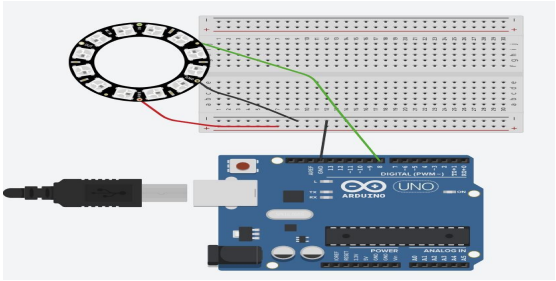
Piezo Buzzer:

Your piezo buzzer has a little disc of piezoceramic material in it. This disc will stretch or compress when an electric signal is passed through it. This motion produces a sound wave which is augmented by the plastic enclosure. The amount they stretch/compress is dependent upon the FREQUENCY of the signal passed through them and that determines the TONE of the sound. Piezoelectric materials possess the characteristic of generating a mechanical change when electricity is passed through them and ALSO can generate an electrical signal when mechanical pressure is applied...for this reason, piezo buzzers can also be used as sensors - detecting movement of the piezo disc by the electrical signal that it generates.

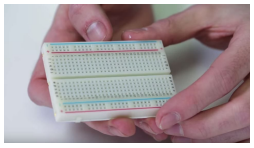
The piezo has a lead that goes to ground and one lead that goes to a digital pin on the Arduino. The microcontroller sends the electrical signal to the piezo via pin 3 in our Intruder Code. The correct pin needs to be identified when you use sample code. The github repository has two sample programs: one plays a scale (PiezoSimpleSounds.ino) and the other plays a siren sound by using the values from a sin wave to generate the change in frequency (ArduinoOnRampPiezotrial.ino). The Arduino File-> examples -> built in examples -> Digital -> tone.. Also has four "tone...." sketches as examples that you can trial.

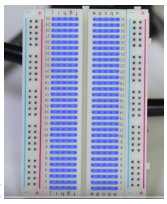
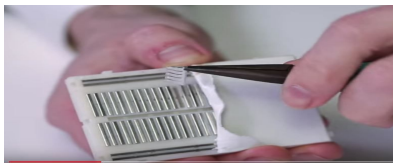
The Simple Sounds sketch came from this really good Adafruit lesson - be sure and get your pin number the same in your sketch and on your board.
https://learn.adafruit.com/adafruit-arduino-lesson-10-making-sounds/playing-a-scale

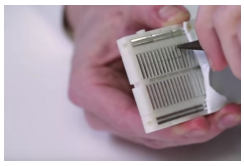
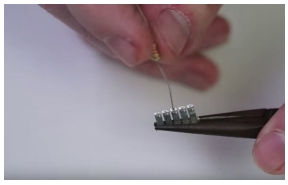
BREAK OUT: ARDUINO UNO AND BREADBOARDS

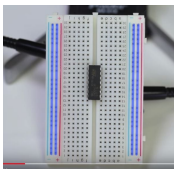


I clipped many of these breadboard images from Colin Cunningham's excellent video on Breadboards and Perfboards:
https://www.youtube.com/watch?v=w0c3t0fJhXU
Check out his other videos, too. He does a great job of explaining electronic concepts.

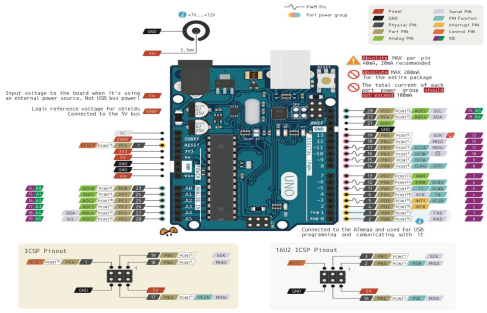


The breadboard included in your kit is a great tool for prototyping a project. Breadboards are reusable and require no soldering. Our breadboards have 400 holes in them. The two central rows of horizontal holes have metallic clips in them as seen in the photo:  

Each row of five holes has a clip with five connections and any wires placed in this five hole row be electrically connected.  

Along each side of the breadboard, there are two vertical columns: 

The long column of vertical holes are connected longways. One has is marked with a red line and one with a blue or black line. These are referred to as the “power rails”. One column of 25 holes is connected but is separate from the column beside it (opposite from the center part of the board.) The columns on the right are not physically connected to the left side, but you can run a jumper wire from one to the other and connect them.



Your Arduino Uno Clone board has been designed to make prototyping projects easy, too. The board uses a cord with a standard issue USB 2.0 cable. This a common A to B Male/Male type peripheral cables that’s usually used for printers. It has become less common over time and most newer and smaller boards will use the micro USB cable that a lot of people of lying around because they are used a lot in cell phones.

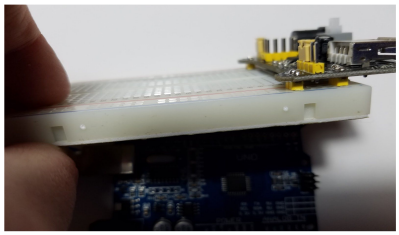
The cable is how we connect our board to our computer so we can upload programs. The board can be powered from this cord, but it can also be powered from the DC jack once the program is already

loaded. Near the USB connector, you will find a small reset button that can be used to reset the program you are running.

Along both sides of the board, there are rows of headers - headers are the black plastic raised parts. Headers have a hole and metallic inside connector so we can, again, plug and/or remove wires for connections. Each header is located at what we call a PIN which is really a connection point. You may see the term GPIO pin which stands for “general purpose input/output pin”. One section of the pins is marked for digital input/output and another is marked for analog input/output. For our build, the I/O’s are digital, but when you are dealing with sound or temperature or other inputs you would use the analog pins. There is also a pin section for power which has ground pins and 5V and 3.3V pins. We are powering our Ultrasound sensor from the 5V Arduino pin - the board is powered from the USB cable and putting out 5V to our sensor.

The surface mounted components on the board consist of the microprocessor, resistors and capacitors and various other chips that manage power and signals.
REview neopixels, power requirements, libraries - sample code

Addressable RGB LED strips: Our Intruder Alarm includes a flashing strip of lights when the alert() is triggered. The lights are LED’s - light emitting diodes. In the simplest terms, a light-emitting diode (LED) is a semiconductor device that emits light when an electric current is passed through it. Diodes are components that allows current to flow in one direction and not in the other. Light emitting diodes (LED’s) give off photons as their electrons move across the space between two plates of semiconductor material. The energy of the photons is different for different materials. Photons are light, so different materials will give off different colors of light. We’ll use light emitting diodes or LED’s in our project. Actually, we will use THREE led’s for each light - a red, a green, and a blue one. That’s 3 * 22 LED’s in your build - 66!. 66 LED’s and their wire connections create enough resistance that we will power them separately using the DC jack on our power converter board and the plug in ‘wall wart’.

This build will use a **power converter board** connected to the positive and negative rails of the breadboard. The github repository contains a datasheet on this board. This will run 5V (or 3.3V) of power down the red side and the blue/black column will be our ground. We will use the 9V/1AMP DC jack power cord to power these LED’s separately from the arduino board. Later, once our code is loaded, we can also power the arduino from this board. 

An RGB “neopixel” has three LED’s in it. The combination of the red, green, and blue signals give us a rainbow of colors. Our strip of LED’s actually has a little circuit chip at each pixel - these are called addressable RGB LED strips. Through our data line on pin 8, we can send instructions to the strip of lights and the chips tell which LED’s to light up.

Our program uses a library from Adafruit.com to make programming the strip of lights less complicated for us. We will need to designate the PIN and the number of pixels in our strip in the code. The Intruder Alarm code uses PIN 8 and 22 pixels.

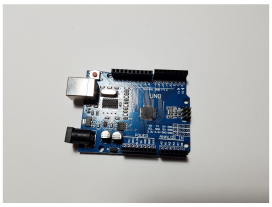
Adafruit.com has a thorough guide for their LED strips called neopixels:
https://learn.adafruit.com/adafruit-neopixel-uberguide/the-magic-of-neopixels

GROUP WRAP-UP

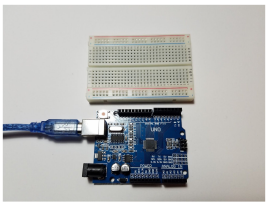
Complete/troubleshoot installation of Intruder code
Discuss areas to modify (US, LED's, PIEZO)
How can we mod this project? Motion sensor alarm with flashing LEDs?https://www.makeuseof.com/tag/how-to-make-a-simple-arduino-alarm-systemWhat other inputs could we use? What other outputs? How about design/case? What projects are you interested in building? Do you have any ideas
for a different project for this level of class? Would you be interested in an intermediate level class?
Demonstrate Charles' Projects!!!
Links to a couple of my projects:
Halloween Lights (uses one Arduino Uno and a custom made circuit for the lights plus another Uno and an Adafruit Music Maker Shield to deliver the sound files - and our Ultrasound sensor!):
https://www.youtube.com/watch?v=Gv1IGuTeVFQ
https://www.youtube.com/watch?v=pb-eQfNiYQU
Wreath (uses an Adafruit Gemma board):
https://www.youtube.com/watch?v=4JLM3clIm_g
ThugCat (uses a Adafruit Circuit Python Express board)
https://www.youtube.com/watch?v=iNxwlK5BFUc

THE BUILD

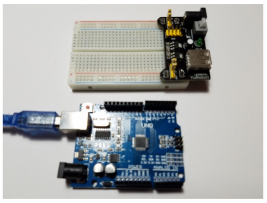
The Uno Board



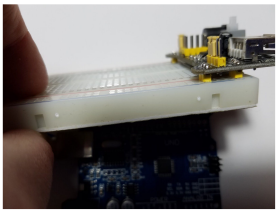
Uno and Breadboard



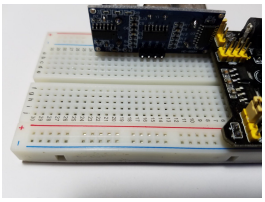
Power Converter Board on breadboard



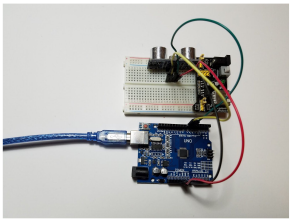
Side view of Power Converter



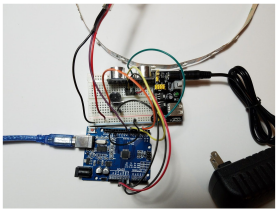
Ultrasound Pins attached to breadboard



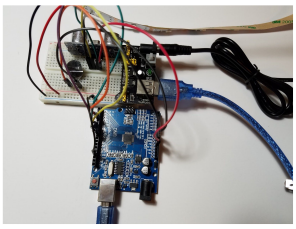
Ultrasound Sensor Wired up



Completed Build



Arduino Powered by Power Converter



MANY THANKS to the original link to this alarm project:
https://www.makeuseof.com/tag/how-to-make-a-simple-arduino-alarm-system/

RESOURCES

Local:

Decatur Makers - https://decaturmakers.org/
Decatur Makers Microcontrollers/Raspberry Pi Meetup - second Monday monthly.
https://www.meetup.com/Decatur-Makers/
Decatur Makers Slack channel: microcontrollers

Online Tutorials/Learning:

Arduino.cc
Jeremy Blum: https://www.jeremyblum.com/2011/01/02/arduino-tutorial-series-it-begins/
Adafruit.com: www.adafruit.com
Sparkfun: https://learn.sparkfun.com/
Instructables: https://www.instructables.com/
Hackaday: https://hackaday.com/
Hackster.io: www.hackster.io
Element 14: https://www.element14.com/community/welcome

Shopping Local:

Microcenter - Duluth and N. Atlanta
RadioShack?

Shopping Online:

Ebay
Amazon
Adafruit
Sparkfun
Element14/Newark: http://www.newark.com/?COM=element14
Google

ONLINE SIMULATORS:

Tinkercad.com: https://tinkercad.com
MakeCode: https://makecode.microbit.org/
Adafruit: https://makecode.adafruit.com/

DOWNLOADABLE software simulator:

Fritzing.org

ONLINE RESOURCES TO LEARN C++ PROGRAMMING LANGUAGE:

https://www.udemy.com/free-learn-c-tutorial-beginners/
http://www.learncpp.com/
https://www.udacity.com/course/c-for-programmers--ud210

ONLINE GENERAL LEARN TO CODE IN GENERAL:

EDx: https://www.edx.org/

Good Python courses from MIT and GA Tech

Coursera: https://www.coursera.org/

Anything by Charles Severance from Michigan State, great Python course from Rice University that teaches by having you code games.

Codeschool
Codecademy: https://www.codecademy.com/learn
Khan Academy
Udacity
Udemy
FreeCodeCamp.com (mostly uses Web Development languages)