RAMPED – Summer 2016

Robot/Alarm Lesson Plan

Dean Cyphers

Pretest Questions	1. Describe in one sentence below what you understand by the term 'robot' 2. What are the main parts of a robot? 3. What do people do to make a robot move?
Objectives	● Define what a robot is. ● Describe the main components of a robot. ● Explain how the LEGO MINDSTORMS NXT robot can be programmed to move. ● Explain how engineers apply robotics to solve real-world problems. ● Build a simple Arduino alarm system
Catch	What is a robot? (Presentation) Today we will be talking about robots and learning how to work with robots. Clear popular culture-driven misconceptions about what robots are and what they do. Then, define what a robot is. Emphasize how engineers have used robots to make our lives easier.
Activity	1) Understanding communication with a robot. Student teams act out robot instructions and then program a LEGO NXT taskbot to go through a simple maze. Through the human and robot examples, students see that a robot's computer simply follows instructions as given, thus one must be logical and precise with programming instruction. They also see how robot sensors are used to perform movement tasks. 2) Using the Arduino microproccesorand sensors build a simple alarm system
Review	An automatic door at a grocery store is an everyday example of a robot that engineers have designed to make our lives easier. What are some ways the automatic door makes peoples' lives easier?
Assessments	● What are three things that an automatic door would need to have in order to be a robot? ● What is the output of the automatic door? ● What does the computer do? ● Complete a working Arduino alarm system

RAMPED – Summer 2016

Posttest Questions (same as pretest questions)	1. Describe in one sentence below what you understand by the term 'robot' 2. What are the main parts of a robot? 3. What do people do to make a robot move?
Standards	**Wyoming State Career & Technical Education Standards** ● **3) Critical Thinking and Problem Solving**: Students use critical thinking skills to plan and conduct research, manage projects, solve problems, and make informed decisions using appropriate technology, tools, and resources. ● **5) Technical Proficiency and Productivity:** Students safely, ethically, and productively use existing and new technologies and systems. **International Technology and Engineering Educators Association: Technology** ● The development of technology is a human activity and is the result of individual and collective needs and the ability to be creative.
Crosscutting Concepts from NGSS	**Next Generation Science Standards: Science** ● Generate and compare multiple possible solutions to a problem based on how well each is likely to meet the criteria and constraints of the problem

Image 1, Ref - see slide 17-18

What is a Robot?

A look at characteristics of robots using the LEGO NXT as a specific example

(50 minutes)

PRE/POST-ASSESSMENT SHEET - What is a robot?

1. Describe in one sentence below what you understand by the term ‘robot’

2. What are the main parts of a robot?

3. What do people do to make a robot move?

PRE/POST-ASSESSMENT SHEET - What is a robot?

1. Describe in one sentence below what you understand by the term ‘robot’

A robot is a machine that gathers information about its environment (senses) and uses that information (thinks) to follow instructions to do work (acts).

2. What are the main parts of a robot?

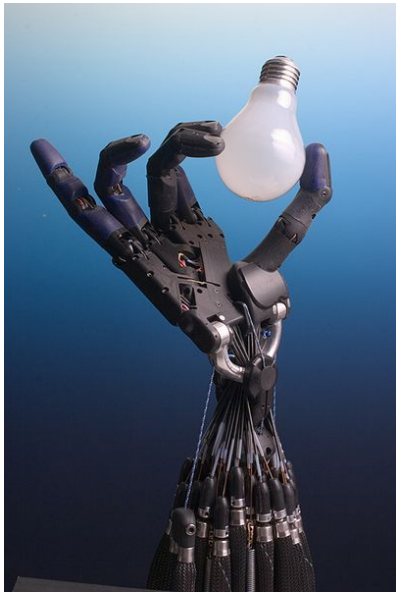
Computer (to make decisions), Input ports (connected to sensors), and Outputs (connected to motors, for example).

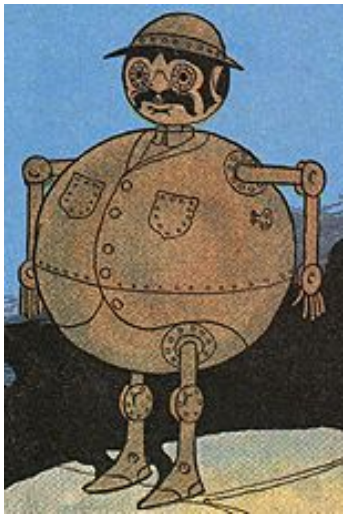
3. What do people do to make a robot move?

They program it using software telling it precisely what to do, step by step.

Robots in the World

- We always think of Robots in movies
  - Human Characteristics
  - Learning
  - Emotions
- While there have been advances is Artificial Intelligence (AI)….





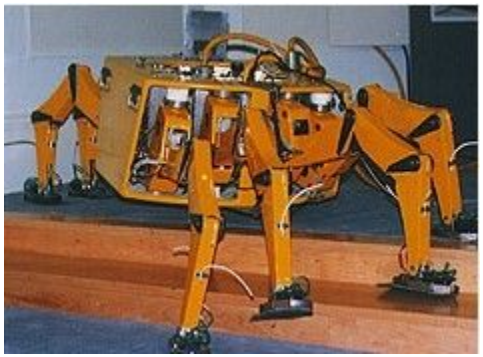


Image 2-5, Ref - see slide 17-18

Robots in the World

- …most robots look and act nothing like what we imagine in movies.
- In fact, many everyday objects are robots, even though you may not have ever noticed.
- This doesn't mean that what they do is any less amazing!

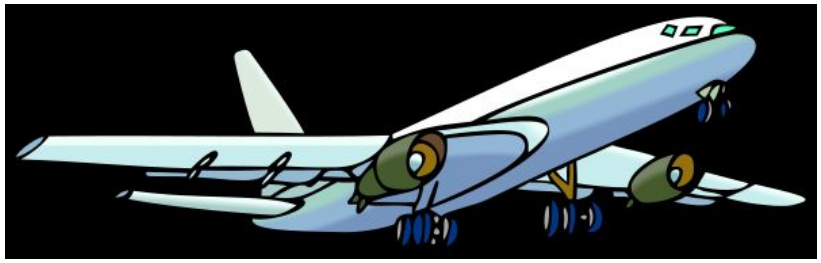


Image 6-8, Ref - see slide 17-18

How do you define a robot?

- Definition: "A robot is a machine that gathers information about its environment (senses) and uses that information (thinks) to follow instructions to do work (acts)."[1]
- Senses – using SENSORS; thinks – using COMPUTER; acts – using MOTORS, for example.
- Engineers design robots to perform complex tasks more easily and with greater accuracy. Some everyday examples of robots include: automatic car washes, vending machines, automatic doors, robotic arms used in manufacturing, remote control cars and trucks, automatic teller machines (ATMs)
- Can you name some devices that are robots?
- …and some that are not?

[1]The Tech Museum. "Robotics: Classroom Activities." http://www.thetech.org/robotics/activities/index.html. 2005.

Some robot videos

- LEGO® Robots

http://www.teachersdomain.org/resource/eng06.sci.engin.design.legorobot/

- Engineering for the Red Planet

http://www.teachersdomain.org/resource/eng06.sci.engin.design.ayanna/

- Anatomy of a Rover

http://www.teachersdomain.org/resource/eng06.sci.engin.design.rover/

- Kismet

http://www.teachersdomain.org/resource/eng06.sci.engin.design.kismet/>

- RoboSnail

http://www.teachersdomain.org/resource/eng06.sci.engin.systems.robosnail/

- Robofly

http://www.teachersdomain.org/resource/eng06.sci.engin.systems.robofly/

- Design Inspired by Nature

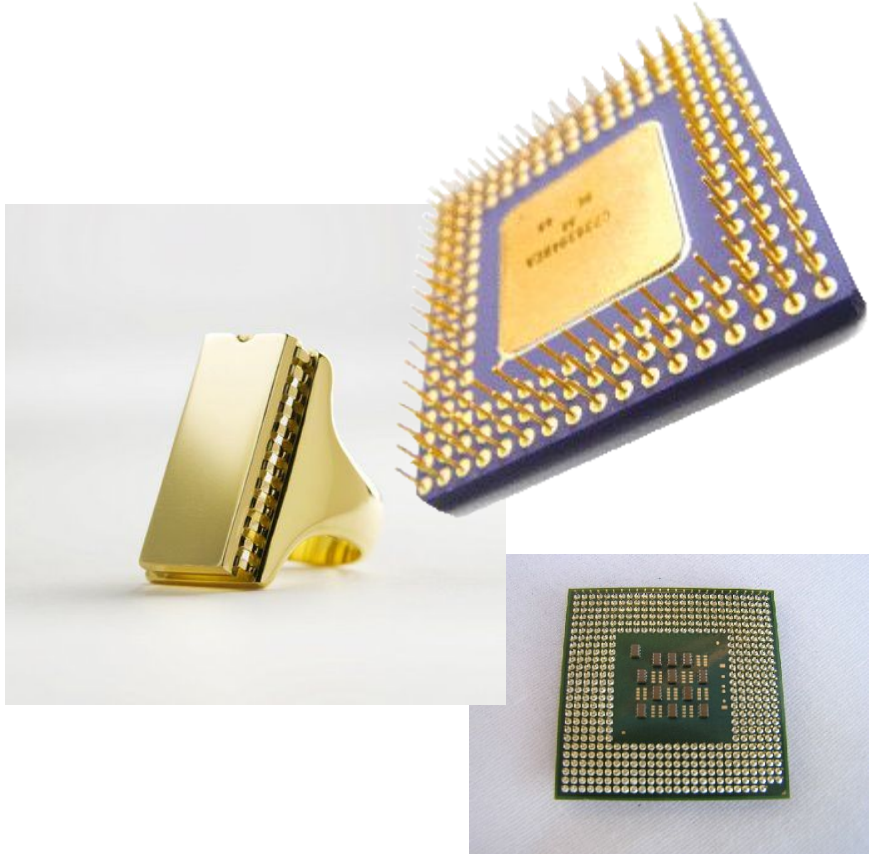
http://www.teachersdomain.org/resource/eng06.sci.engin.design.biomimicry/

The NXT

- **The specific robot that we will be working with is the LEGO NXT.**
- **The first robot you will make with the NXT is called a Taskbot.**

What Makes Something a Robot? #1:
A Computer (to help it 'make decisions')

- There is one characteristic that ALL robots have:
  - They are ALL controlled by a computer
    - Not necessarily a computer like the one you have at home!
    - Computers come in all shapes and sizes.



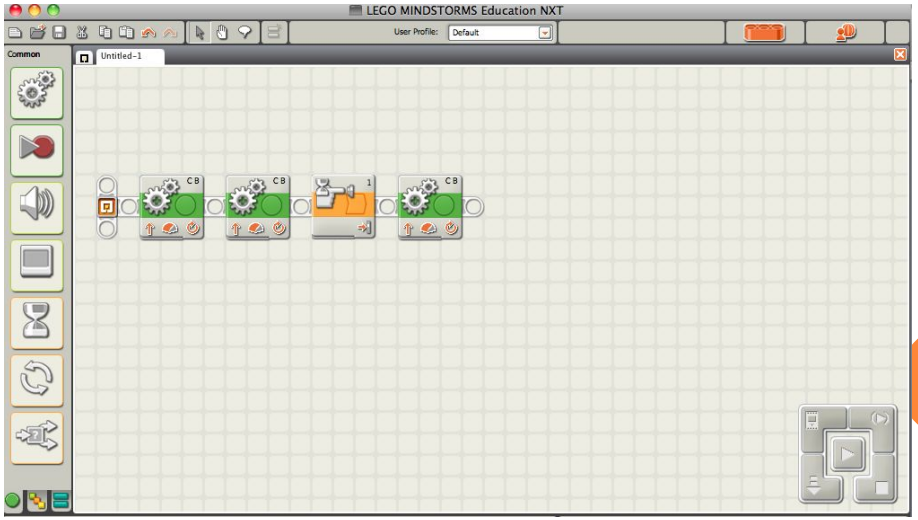
Believe it or not, these all are computers!

What Makes Something a Robot? #1: A Computer (to help it 'make decisions')....Contd.

- **Since robots cannot think <u>on their own</u> like us, they have to follow specific instructions <u>to make decisions</u>.**
- **These instructions are given to their computer in the form of a *program*.**
- **The computer can then read the program and act like the robot's brain, controlling the robot based on just running each instruction in the program in sequence.**

Computer of the NXT

- The computer of the NXT is called the NXT computer brick.
- We can program the NXT using a programming software that LEGO provides with the NXT.



What Makes Something a Robot? #2: Inputs to 'SENSE' (via sensors)

- Robots need inputs, which provide information to the its computer to make decisions.
  - Think of it as a signal going "in" to the computer
- Example
  - The mouse and keyboard on your computer
  - A camera on a robot that can act as its eye
- Note: The important thing about inputs is that they have no effect if the program doesn't tell the computer to look for them!

Inputs of the NXT: (sensors)

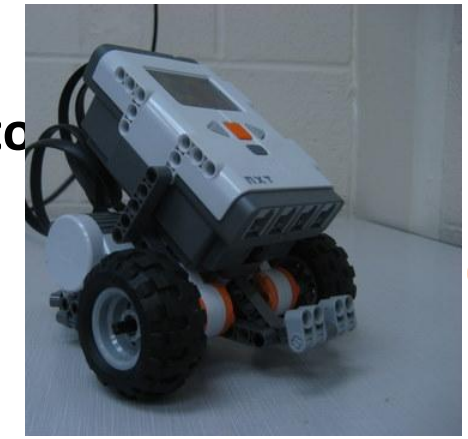
- Four sensors come with the NXT
  - Light sensor, touch sensor,
  - Sound sensor, ultrasonic sensor

- The sensors are connected to the INPUT ports of the NXT – 1, 2, 3, 4

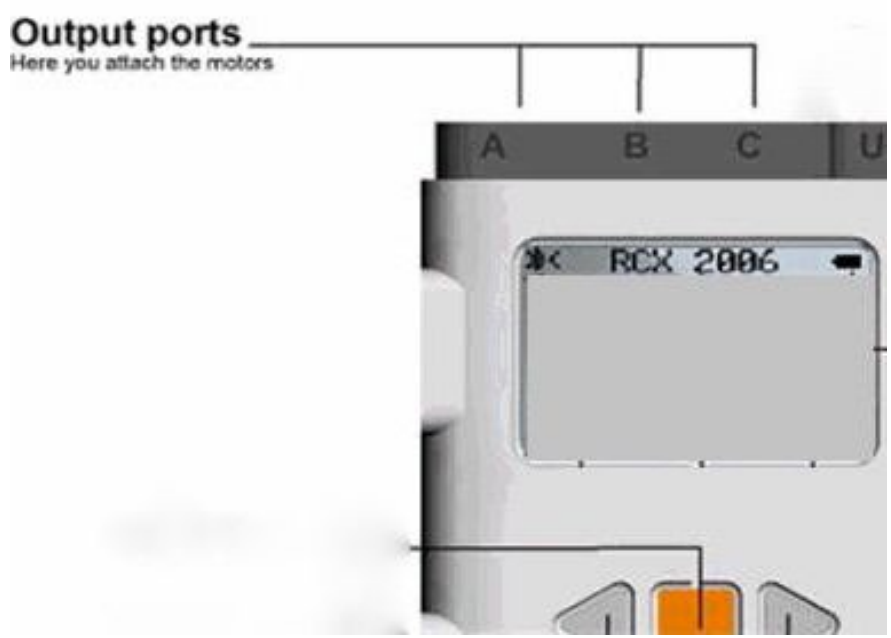
- These are shown alongside and you will be using them activities.

What Makes Something a Robot? #3: Outputs to 'ACT' (via motors, for example)

- An output is a command sent out by the robot computer to some other part of the robot
  - You can think of it as a signal going "out" of the computer.
- Examples:
  - Your computer monitor
    - Your computer sends an image to your computer screen so that you can see what the computer is doing and use it easily.
  - Motors on a robot
    - The computer sends a signal to the moto of a robot to make it move.



Outputs of the NXT



- **The output ports are located at the top of the NXT computer bricks. These ports don't take in any information, they only send it out.**

- **We can attach motors or lights to the output ports of the NXT and then program the computer to activate them.**

Summary

- **The parts of a robot are**
  - **A computer that needs to be programmed (to make decisions)**
  - **Inputs (to 'sense' via sensors)**
  - **Outputs (to 'act', e.g., via motors)**

- **Our NXT has all three of these, and in the following activity, we'll see how exactly we can put all these together and make the NXT robot move based on your commands.**

Image source/rights

Image 1: ADA Description: Asimov robot

Image file name: Robot_asimo_cropped.jpg

Source/Rights: http://commons.wikimedia.org/wiki/File:Robot_asimo_cropped.jpg

Image 2: ADA Description: Shadow hand bulb

Image file name: Shadow_Hand_Bulb_large_Alpha.png

Source/Rights: http://meta.wikimedia.org/wiki/File:Shadow_Hand_Bulb_large_Alpha.png

Image 3: ADA Description: Tiktok of Oz

Image file name: Tiktok.png

Source/Rights: http://meta.wikimedia.org/wiki/File:Tiktok.png

Image 4: ADA Description: Sony Orio robot

Image file name: Sony_Qrio_Robot.jpg

Source/Rights: http://meta.wikimedia.org/wiki/File:Sony_Qrio_Robot.jpg

Image 5: ADA Description: Robug

Image file name: Robug-3.jpeg

Source/Rights: http://meta.wikimedia.org/wiki/File:Robug-3.jpeg

Image source/rights

Image 6: ADA Description: Airplane

Image file name: Airplane_clipart.svg

Source/Rights: http://meta.wikimedia.org/wiki/File:Airplane_clipart.svg

Image 7: ADA Description: Truck

Image file name:
OS_ANGELES_WESTERN_AVENUE_FIRE_DEPARTMENT_TRUCK_IMAGE_PATRICE_RAUNET_HOLLYWOOD.jpg

Source/Rights: http://meta.wikimedia.org/wiki/File:
LOS_ANGELES_WESTERN_AVENUE_FIRE_DEPARTMENT_TRUCK_IMAGE_PATRICE_RAUNET_HOLLYWOOD.jpg

Image 8: ADA Description: TV

Image file name: TV-icon-2.svg

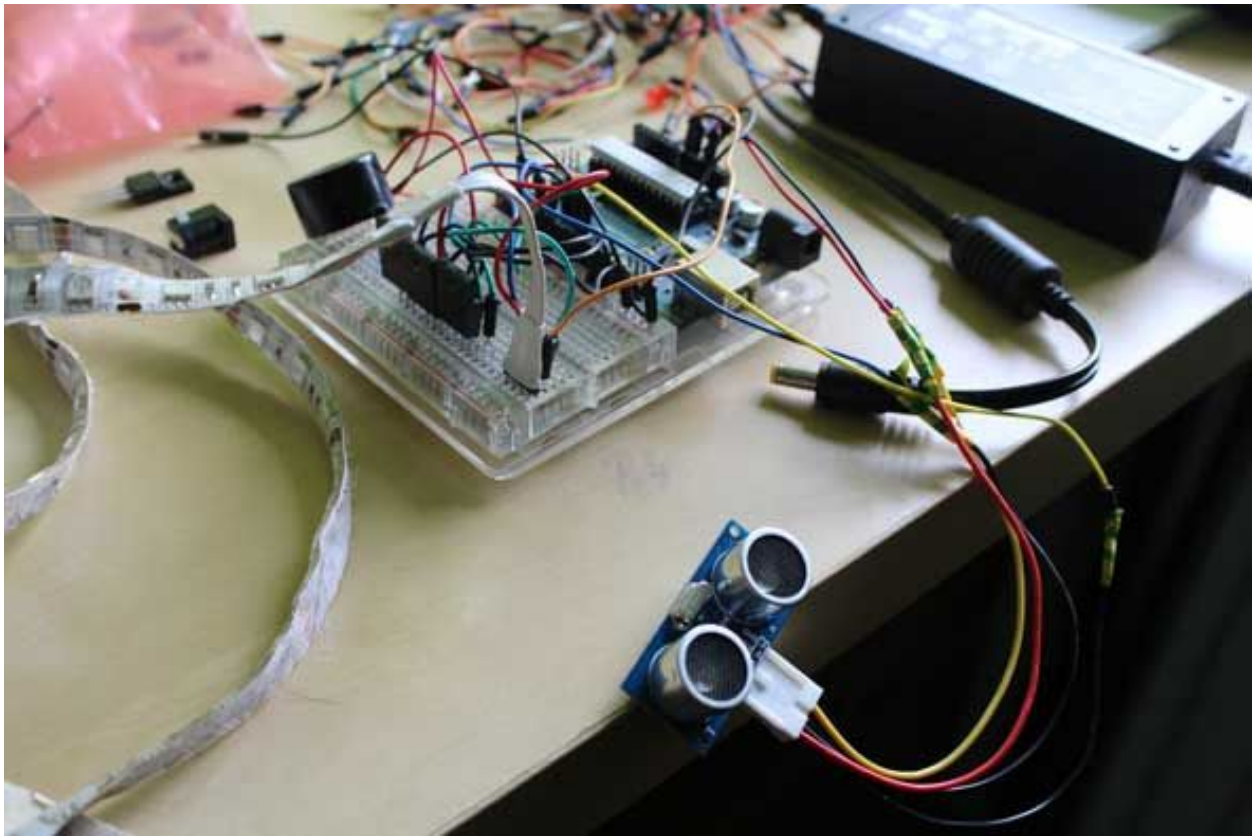
Source/Rights: http://meta.wikimedia.org/wiki/File:TV-icon-2.svg

How To Make a Simple Arduino Alarm System

You'll need:

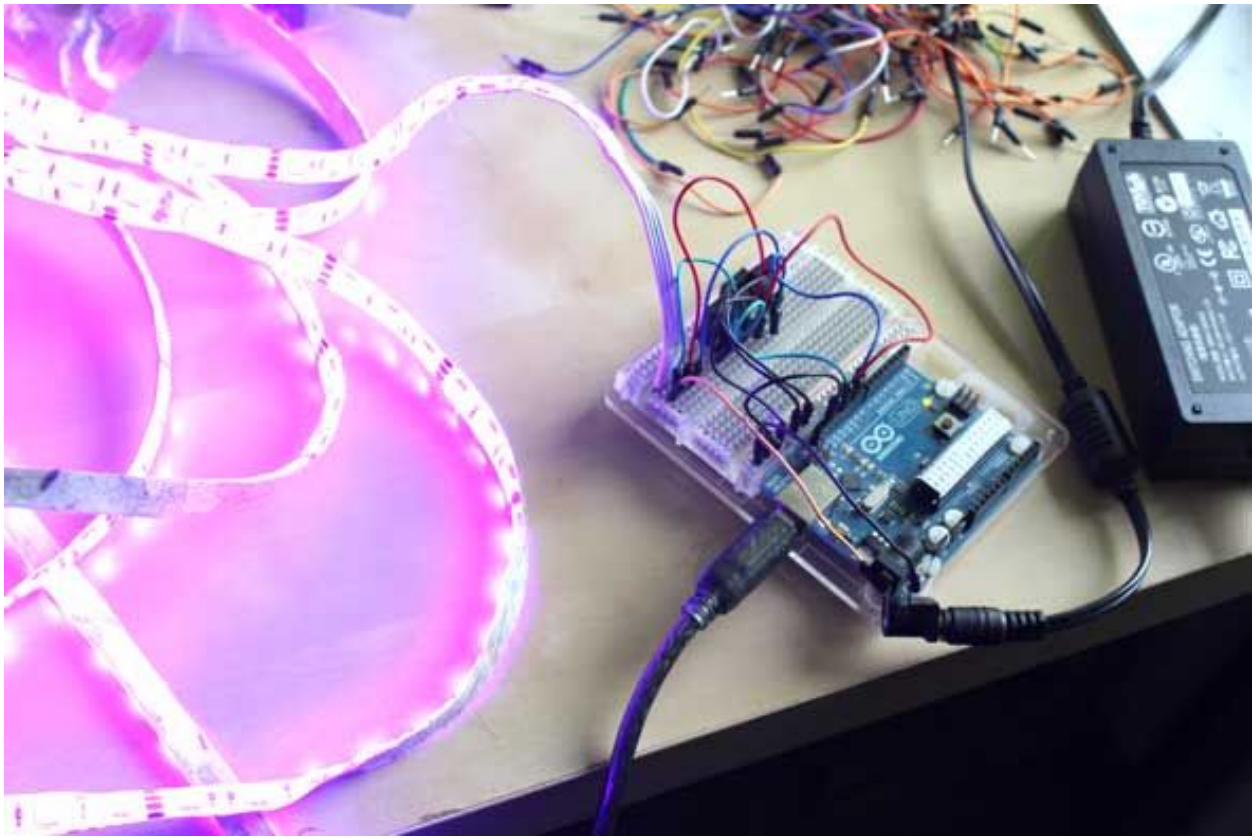
- An Arduino
- Ultrasonic "ping" sensor, I'm using HC-SR04 A PIR would be better, but those are expensive. A ping sensor can be placed surreptitiously in a doorway and still serve the same basic job, and is only $5
- A piezo buzzer
- LED strip light, with the same wiring we used back in this project.

As you're wiring up this project, don't remove everything each time — just keep building on the last block. By the time you get to "Coding The Alarm System" section, you should have all the bits and pieces wired up, looking something like this:



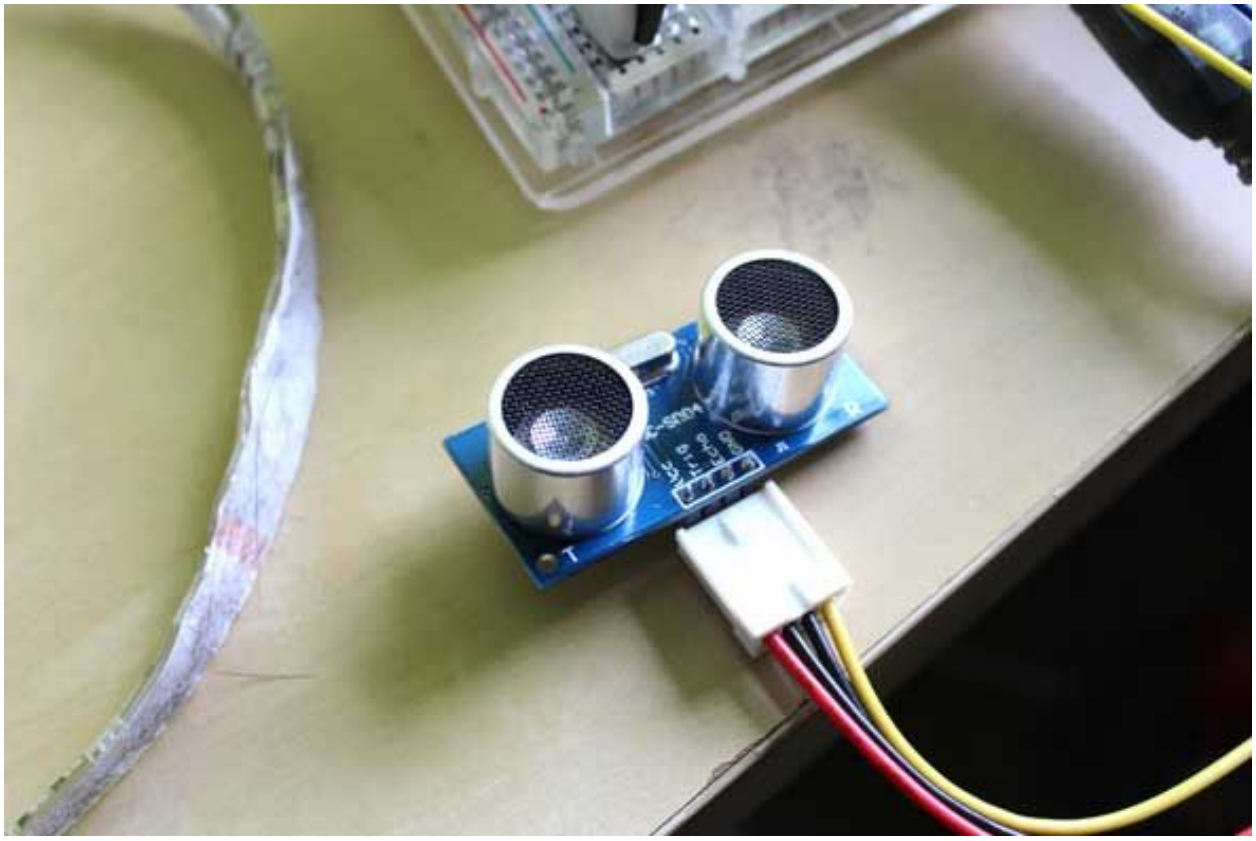
Flashing Lights

Use the wiring diagram from this project to hook up your LED strip; don't change the pins, as we need PWM output. Use this code to quickly test your wiring. If all goes well, you should have this:

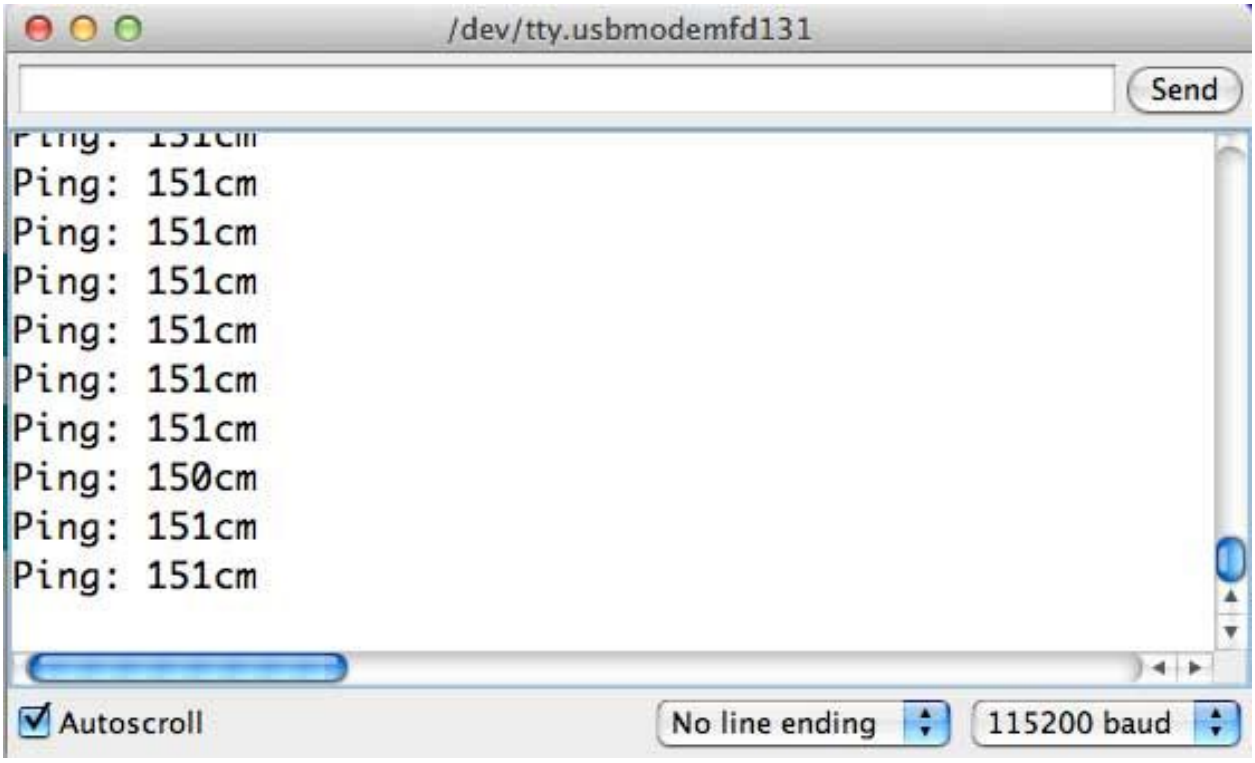


Distance Sensor

On the SR04 module, you'll find 4 pins. **VCC** and **GND** go to +5V rail and ground respectively; **TRIG** is the pin used to send a sonar signal, put this on pin 6; **ECHO** is used to read the signal back (and therefore calculate the distance) — put this on 7.



To make things incredibly simple, there's a library we can use called NewPing. Download and place in your Arduino's **Library** folder and restart the IDE before continuing. Test using this code; open up the serial monitor and make sure the speed is set to 115200 baud. With any luck, you should see some distance measurements being send back to you at a pretty high speed. You may find a variance of 1 or 2 centimeters, but this is fine. Try running your hand in front of the sensor, moving it up and down to observe the changing readings.



The code should be fairly simply to understand. There are a few declaration of relevant pins at the start, including a maximum distance – this may vary according to the exact sensor you have, but as long as you're able to get less than 1 meter readings accurately, you should be fine.
In the loop of this test app, we use the **ping()** function to send out a sonar ping, getting back a value in milliseconds of how long it took for the value to return. To make sense of this, we use the NewPing libraries built in constant of **US_ROUNDTRIP_CM**, which defines how many microseconds it takes to go a single centimeter. There's also a 50 ms delay between pings to avoid overloading the sensor.

Piezo Alarm

The Piezo crystal sensor is a simple and cheap buzzer, and we can use a PWM pin 3 to make different tones. Connect one wire to pin 3, one to ground rail – it doesn't matter which.
Use this code to test.
The only way to kill the rather obnoxious and loud alarm is to pull the plugs. The code is a little complex to explain, but it involves using sine waves to generate a distinctive sound. Tweak the numbers to play with different tones.

Coding The Alarm System

Now that we have all the pieces of this puzzle, let's combine them together.
Go ahead and make a new sketch, called **Alarm**. Start by combining all the variables and pin definitions we've in the test examples until now.

```
#include <NewPing.h>

// Select which PWM-capable pins are to be used.
#define RED_PIN    10
#define GREEN_PIN   11
#define BLUE_PIN  9

#define TRIGGER_PIN  6 // Arduino pin tied to trigger pin on the ultrasonic sensor.
#define ECHO_PIN     7 // Arduino pin tied to echo pin on the ultrasonic sensor.
#define MAX_DISTANCE 100 // Maximum distance we want to ping for (in centimeters).

#define ALARM 3

float sinVal;
int toneVal;
```

Begin by writing a basic **setup()** function – we'll only deal with the lights for now. I've added a 5 second delay before the main loop is started to give us some time to get out of the way if needed.

```
void setup(){

  //set pinModes for RGB strip
   pinMode(RED_PIN,OUTPUT);
   pinMode(BLUE_PIN,OUTPUT);
   pinMode(GREEN_PIN,OUTPUT);

  //reset lights
   analogWrite(RED_PIN,0);
   analogWrite(BLUE_PIN,0);
   analogWrite(RED_PIN,0);

  delay(5000);
}
```

Let's use a helper function that allows us to quickly write a single RGB value out to the lights.

```
//helper function enabling us to send a colour in one command
void color (unsigned char red, unsigned char green, unsigned char blue)    // the color
generating function
{
    analogWrite(RED_PIN, red);
    analogWrite(BLUE_PIN, blue);
    analogWrite(GREEN_PIN, green);
}
```

Finally, our loop for now is going to consist of a simple color flash between red and yellow (or, whatever you want your alarm to be — just change the RGB values).

```
void loop(){
  color(255,0,0); //red
  delay(100);
  color(255,255,0); //yellow
  delay(100);
}
```

Upload and test that to ensure you're on the right track.

Now, let's integrate the distance sensor to trigger those lights only when something comes within, say, 50 cm (just less than the width of a door frame). We've already defined the right pins and imported the library, so before your **setup()** function add the following line to instantiate it:

```
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup of pins and maximum distance.
```

Underneath that, add a variable to store the state of the alarm being triggered or not, defaulting to false, of course.

```
boolean triggered = false;
```

Add a line to the **setup()** function so we can monitor the output on serial and debug.

```
Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
```

Next, let's rename the current loop to **alarm()** – this is what's will be called if the alarm has been tripped.

```
void alarm(){
  color(255,0,0); //red
  delay(100);
  color(255,255,0); //yelow
  delay(100);
}
```

Now create a new **loop()** function, one in which we fetch a new ping, read the results, and trigger the alarm if something is detected within the meter range.

```
void loop(){
   if(triggered == true){
     alarm();
   }
   else{
     delay(50);// Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.
     unsigned int uS = sonar.ping(); // Send ping, get ping time in microseconds (uS).
     unsigned int distance = uS / US_ROUNDTRIP_CM;
     Serial.println(distance);
     if(distance < 100){
```

```
      triggered = true;
    }
  }
}
```

Let me explain the code briefly:

- Start by checking to see if the alarm has been triggered, and if so, fire off the alarm function (just flashing the lights at the moment).
- If it's not triggered yet, get the current reading from the sensor.
- If the sensor is reading <100 cm, something has padded the beam (adjust this value if it's triggering too early for you, obviously).

Give it a trial run now, before we add the annoying piezo buzzer.
Working? Great. Now let's add that buzzer back. Add **pinMode** to the **setup()** routine.

```
pinMode(ALARM, OUTPUT);
```

Then add the piezo buzzer loop to the alarm() function:

```
for (int x=0; x<180; x++) {
  // convert degrees to radians then obtain sin value
   sinVal = (sin(x*(3.1412/180)));
  // generate a frequency from the sin value
   toneVal = 2000+(int(sinVal*1000));
   tone(ALARM, toneVal);
 }
```

If you try to compile at this point, you're going to run into an error — I've left this in deliberately so you can see some common issues. In this case, both the NewPing and standard tone library use the same interrupts — they are conflicting basically, and there's not a lot you can do to fix it. Oh dear.
No worries though. It's a common problem, and someone has a solution already — download and add this NewTone to your Arduino Libraries folder. Adjust the beginning of your program to include this:

```
#include <NewTone.h>
```

And adjust the line:

```
  tone(ALARM, toneVal);
```

to

```
   NewTone(ALARM, toneVal);
```

instead.

That’s it. Set your alarm up in the doorway of your bedroom for the next hapless would-be burglar.
Or, a dopey dog, which seemed completely unfazed by the alarm.
Having trouble with the code? Here’s the complete app. If you’re getting random errors, try pasting them below and I’ll see if I can help.