

Take Our Daughters And Sons to Work Day

https://github.com/Mark-MDO47/TODAS_DaughtersAndSons.git

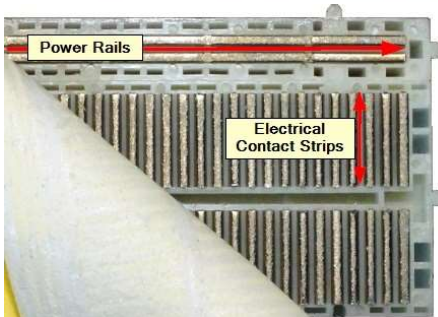
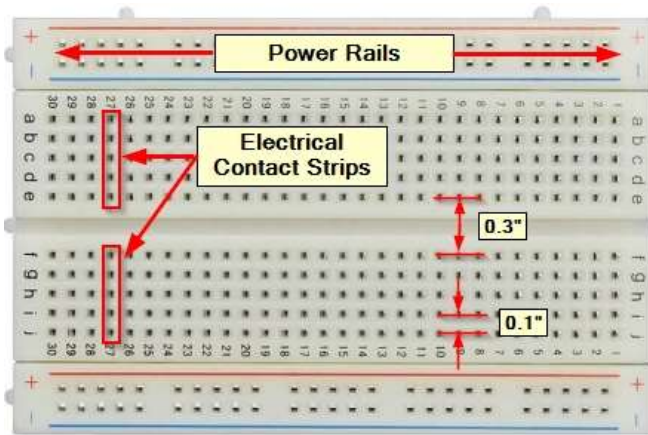
June 2024



TODAS - Electricity, Arduinos, Fun

1. [Water Flow - a way to think about Electricity](#)
 2. [Arduino \(computer on a chip\), buttons, and LEDs](#)
 3. [Sonar Control and big LED ring](#)
 4. [Bananas and Sounds](#)
- [Resources](#)

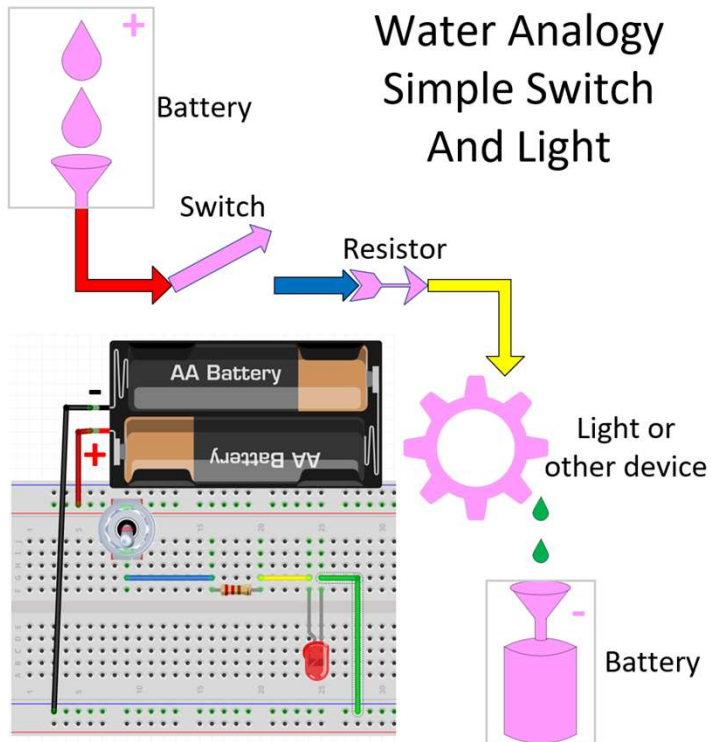
01 - Breadboards - how do they work?



- Sides have power rails - length
- Center has contact strips - width

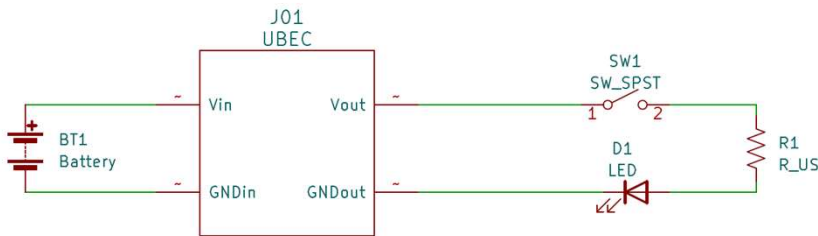
- Images from protosupplies.com

01 – Water Analogy – Simple Circuit



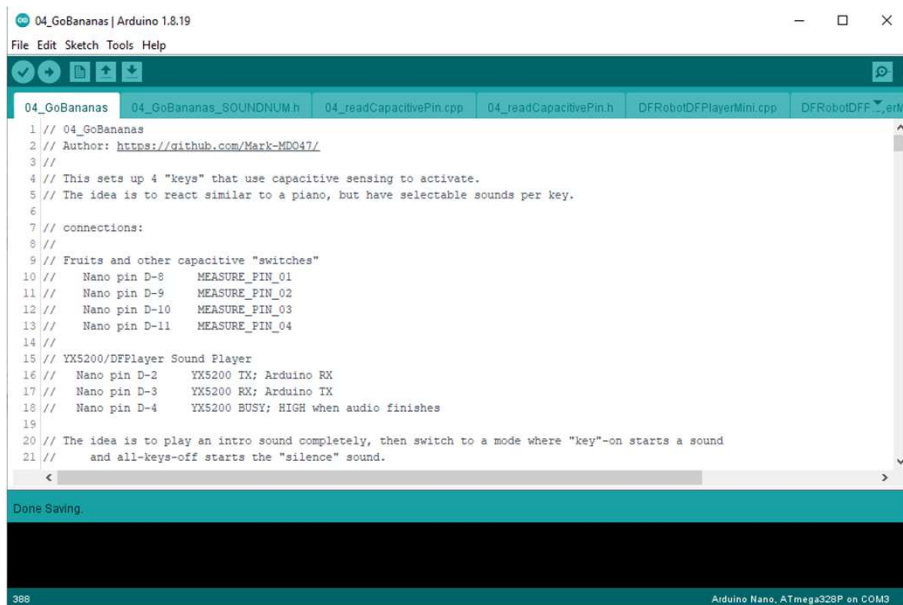
- Electrons in wires actually flow from negative to positive, but many electrical symbols are drawn as if the current flow is from positive to negative so let's get used to it!
- "Pictorial" circuit is "Fritzing Diagram"
 - <https://fritzing.org/> - 8 Euros

01 Simple Circuit Schematic



- The Universal Battery Eliminator Circuit produces 5 volts if the input is greater than or equal to about 6 volts.
- The resistor prevents sending too much current through the LED; that would burn it up.
- Schematic diagram is from KiCad
 - <https://www.kicad.org/> - it's free!

02 – Arduino IDE



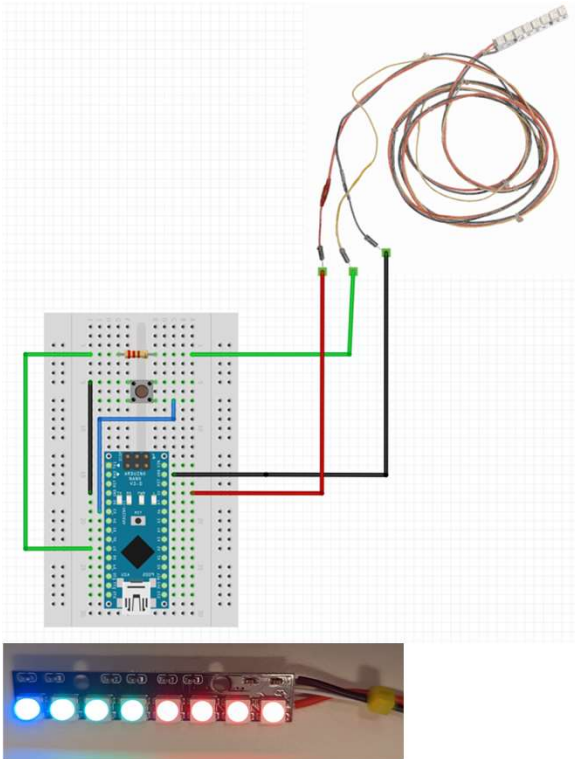
The screenshot shows the Arduino IDE interface. The title bar reads '04_GoBananas | Arduino 1.8.19'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The toolbar contains icons for opening, saving, and running. The file explorer shows several files: '04_GoBananas', '04_GoBananas_SOUNDNUM.h', '04_readCapacitivePin.cpp', '04_readCapacitivePin.h', 'DFRobotDFPlayerMini.cpp', and 'DFRobotDFPlayerMini.h'. The main editor displays the following code:

```
1 // 04_GoBananas
2 // Author: https://github.com/Mark-MD047/
3 //
4 // This sets up 4 "keys" that use capacitive sensing to activate.
5 // The idea is to react similar to a piano, but have selectable sounds per key.
6
7 // connections:
8 //
9 // Fruits and other capacitive "switches"
10 // Nano pin D-8 MEASURE_PIN_01
11 // Nano pin D-9 MEASURE_PIN_02
12 // Nano pin D-10 MEASURE_PIN_03
13 // Nano pin D-11 MEASURE_PIN_04
14 //
15 // YX5200/DFPlayer Sound Player
16 // Nano pin D-2 YX5200 TX; Arduino RX
17 // Nano pin D-3 YX5200 RX; Arduino TX
18 // Nano pin D-4 YX5200 BUSY; HIGH when audio finishes
19
20 // The idea is to play an intro sound completely, then switch to a mode where "key"-on starts a sound
21 // and all-keys-off starts the "silence" sound.
```

The status bar at the bottom indicates 'Done Saving.' and 'Arduino Nano, ATmega328P on COM3'.

- IDE means Integrated Development Environment
- <https://www.arduino.cc/en/software> - it's free!

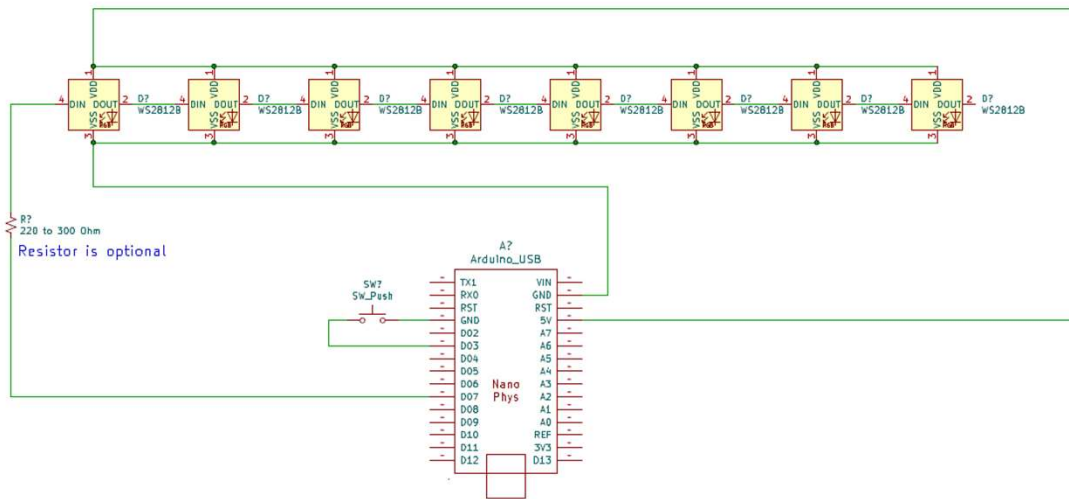
02 – Arduino Moving Color Lights



- “Fritzing diagram” of color lights in motion!
 - LED is Light Emitting Diode
- Configure software for color choices
- Press button to stop, release to continue
- How does it work? SOFTWARE!
 - Pins can be input or output – you tell it which
 - Read the button
 - If not pressed – display next light
 - Short delay
- TLDR – WS2812B individually addressable color LEDs using FastLED library

02 Arduino Moving Color Lights Schematic

NOTE: connections needed only to leftmost LED; daisy chain is within the LED Stick



- The LED stick has eight small chips that control 3 LEDs each: Red, Green, and Blue.
- 3 LEDs are close together so they look like one color

02 - Arduino Inputs

- For now we will talk about how to USE inputs
 - We will talk more about how it works when we GO BANANAS!
- We want to do LED motion when button is NOT pushed
 - Assign name to the pin we use
 - Make the pin an input
 - If the state is HIGH (button not pushed) do the motion
 - Delay a little bit

02 – Arduino Button Input Software

- The first line defines a name for pin D3 for the button
- “pinMode” sets that pin to be an input
- The “if(HIGH)” block does the motion if button NOT pushed
- “delay(500)” delays for ½ second

```
#define DPIN_BTN_IN 3 // this pin is used to sense the external button
```

```
pinMode(DPIN_BTN_IN, INPUT_PULLUP); // digital INPUT_PULLUP means voltage HIGH unless grounded
```

```
if (HIGH == digitalRead(DPIN_BTN_IN)) { // push button to STOP  
  do_motion(); // do motion if button NOT pushed  
} // end if button NOT pushed
```

```
delay(500); // wait half a second
```

02 - Arduino Software Structure

```
1 #define DPIN_BTN_IN 3 // pin D3 is used to sense the external button
2
3 // the setup function runs once when you press reset or power the board on
4 void setup() {
5     pinMode(DPIN_BTN_IN, INPUT_PULLUP); // digital INPUT_PULLUP means voltage HIGH unless grounded
6 } // end setup()
7
8 // the loop function runs over and over again forever
9 void loop() {
10     if (HIGH == digitalRead(DPIN_BTN_IN)) { // push button to STOP
11         do_motion(); // do motion if button is NOT pushed
12     } // end if button NOT pushed
13     delay(500); // wait half a second
14 } // end loop()
```

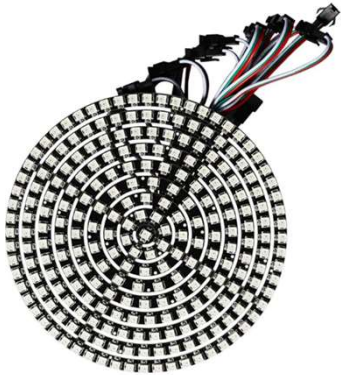
Start with stuff used everywhere

setup() for things done
once at start

loop() for things done
repeatedly after setup()

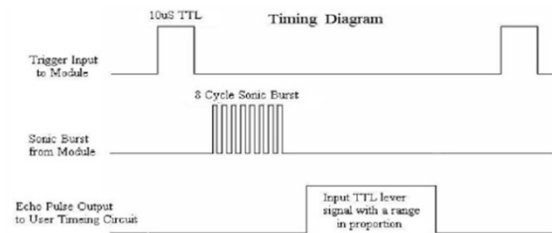
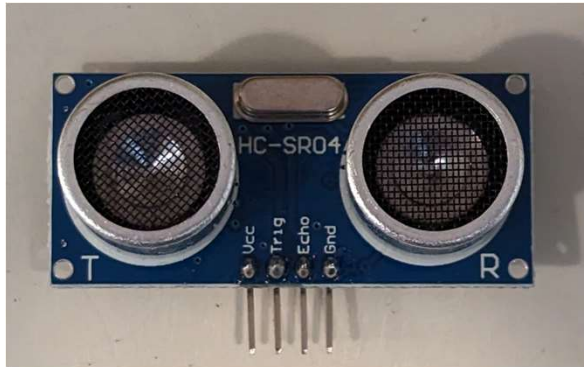
The "if" statement is magical!
Only do inside {} if it is TRUE

03 – Go Big with Ultrasonic Sonar Control



- 8 LEDs is fun – but 241 LEDs is MORE fun!
 - TLDR – WS2812B individually addressable color LEDs using FastLED Library
 - Displays moving patterns we choose
 - Disk is mostly wired; less soldering needed
 - Needs more power than Arduino can give
-
- Pictures from amazon.com
 - TLDR - <https://www.amazon.com/WESIRI-WS2812B-Individually-Addressable-Controller/dp/B083VWVP3J>

03 – Ultrasonic Sonar Control



- Sonar sensor detects distance
- Bounces ultrasonic sound off objects
- Set "Trig" HIGH then LOW to start sonic burst
- Measure time until "Echo" is HIGH
- Divide time by speed of sound to get distance
- TLDR diagram from the sparkfun.com HC-SR04 spec.

03 – Ultrasonic Sonar Detector Software

Include library, define names for pins

"my_ultra" is how I use the HC-SR04

Make a global "gUltraDistance" for debugging

"handle_ultra" returns a pattern number

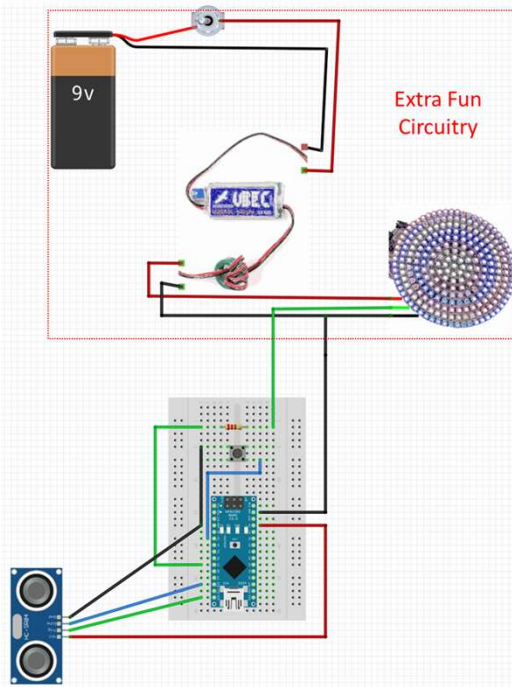
"my_ultra.read(CM)" gives distance in CM
stored in "gUltraDistance"

Use math to turn distance into "pattern"

Return "pattern" (number) to caller

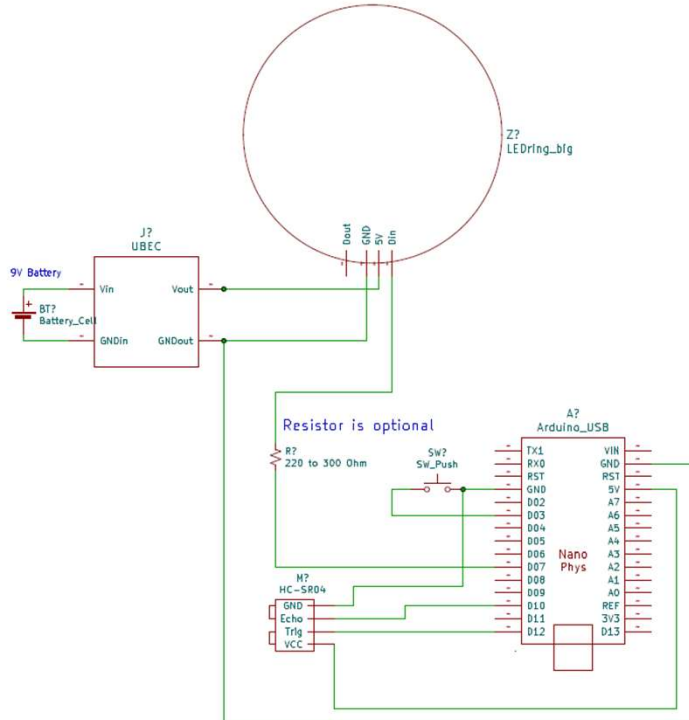
```
27 #include <FastLED.h>
28 #include <Ultrasonic.h>
29
30 // Ultrasonic HC-SR04 definitions
31 #define ULTRA_TRIG_PIN 12 // HC-SR04 Trigger digital pin
32 #define ULTRA_ECHO_PIN 10 // HC-SR04 Trigger echo pin
33 #define ULTRA_CM_PER_REGION 9 // HC-SR04 every this many CM is a different pattern
34 #define ULTRA_IGNORE_INITIAL_CM 3 // HC-SR04 ignore the first 3 CM since valid range starts at 2 CM
35
36 // instantiate my HC-SR04 data object
37 Ultrasonic my_ultra = Ultrasonic(ULTRA_TRIG_PIN, ULTRA_ECHO_PIN); // default timeout is 20 milliseconds
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128 int gUltraDistance = 0; // latest measured distance in centimeters
129
130 ///////////////////////////////////////////////////
131 // handle_ultra() - process HC-SR04 data.
132 // returns: pattern number 0 <= num <= PATTERN_MAX_NUM
133 //
134
135 int handle_ultra() {
136     int pattern; // integer pattern number from 0 thru 5 inclusive
137     // get the range reading from the Ultrasonic sensor in centimeters
138     int ultra_dist;
139
140     gUltraDistance = (my_ultra.read(CM));
141     ultra_dist = gUltraDistance - ULTRA_IGNORE_INITIAL_CM;
142     if (ultra_dist < 0) ultra_dist = 0;
143     pattern = ultra_dist / ULTRA_CM_PER_REGION;
144     if (pattern > 5) pattern = 5;
145
146     return(pattern);
147 } // end handle_ultra()
148 ...
```

03 – Fritzing Diagram



- Replace 8-LED stick with 241-LED Disk
 - Separate power for 241-LED Disk
- Add new HC-SR04 Ultrasonic Sensor
- Button unused – can leave it
- We add the battery and UBEC because the 241 LEDs take a lot of power!

03 – Schematic Diagram



The button connected to pin D03 is not used for this project.

We leave it on the breadboard; no need to remove it.

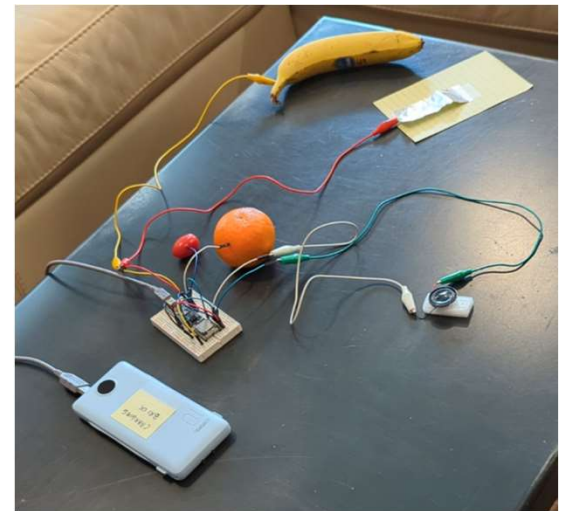
We connect the UBEC ground to Arduino, LEDs, and HC-SR04.

That allows both LED and HC-SR04 to recognize the voltages from the Arduino

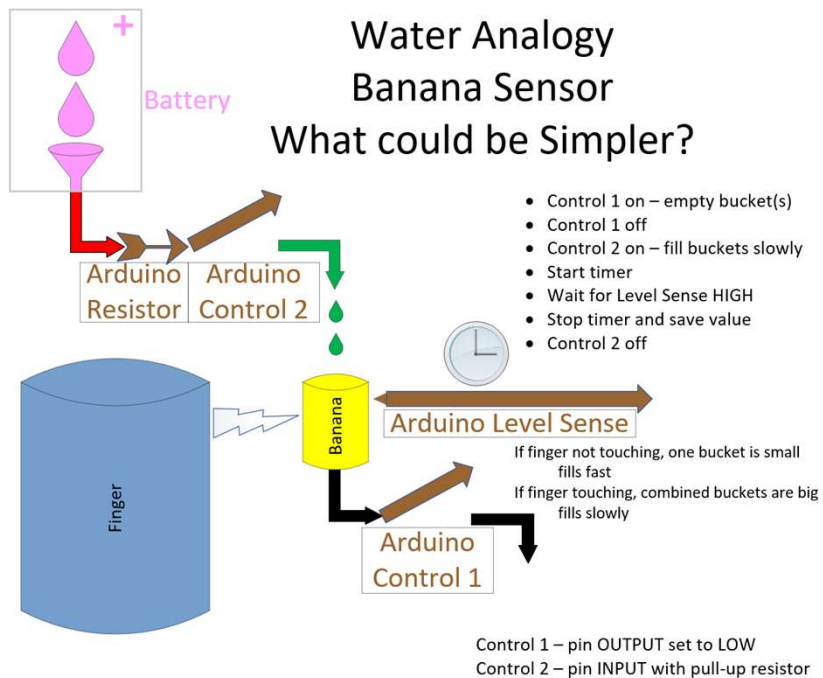
04 – Go Bananas!



- A Banana Piano! Several “key” fruit types...
 - Uses “capacitive sensing” – depends on how fast Arduino can charge it up
 - Touching banana makes it take longer to charge
- Digitized sounds are played when the “key” is pressed
- Only one sound at a time is played
- Sounds are “sampled” – digital storage format
- Video: <https://youtu.be/EC1qHbE89JI>
- TLDR - Image by krakenimages.com on Freepik

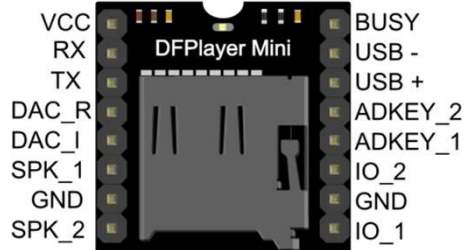


04 – Banana Input



- Note that this is an analogy of how it works
- The diagram shows the flow of the code
- The code uses tricks to be fast – makes it harder to understand

04 - Digitized Sound Output



- DFPlayer (YX5200) accepts SD card with digitized audio
 - Plays SD card audio files by number
 - UART interface to Arduino
 - Direct mono speaker output
 - Line-out stereo output, usable for BlueTooth
- Many tricks to using YX5200
 - Check out my github page on it (below)

<https://github.com/Mark-MDO47/AudioPlayer-YX5200>

```
// "pin index" state:  
//   -1 means nothing selected  
//   0-3 (3 = NUM_MEASURE_PINS-1) represent pins MEASURE_PIN_01 to MEASURE_PIN_03,  
//   any other value is invalid  
  
int8_t gCurrentPinIndex = -1; // Index number of which PinIndex is current - nothing selected  
int8_t gPrevPinIndex = -1; // previous PinIndex - nothing selected  
  
// "pin index" to sound mapping  
//   pin index goes from -1 to 3 (3 = NUM_MEASURE_PINS-1)  
//   we add one to that number to go from 0 to 4  
//       0 = Silence sound  
//       1 through 4 = MEASURE_PIN_01 through MEASURE_PIN_04  
  
uint16_t gPinIndex2SoundNum[1+NUM_MEASURE_PINS] = { SOUNDNUM_silence, SOUNDNUM_electric_piano_C, SOUNDNUM_organ_D, SOUNDNUM_organ_E,
```

- Place sound numbers into array gPinIndex2SoundNum[]
- Calls to pin2soundnum() will convert PinIndex to sound number
- Most of routine is error checking (this is normal)

Only one line is needed to actually do translation

could just use `gPinIndex2SoundNum[pinIndex+1]` if wanted to skip error checking

```

// //////////////////////////////////////
// pin2soundnum(pinIndex) - convert pin index to sound number
//
// pinIndex to sound mapping
//   pinIndex goes from -1 to 3 (3 = NUM_MEASURE_PINS-1)
//   we add one to that number to go from 0 to 4
//   0 = Silence sound
//   1 through 4 = MEASURE_PIN_01 through MEASURE_PIN_04

uint16_t pin2soundnum(int8_t pinIndex) {
    uint16_t soundNum = SOUNDNUM_INVALID;
    if ((pinIndex >= -1) and (pinIndex < NUM_MEASURE_PINS)) {
        soundNum = gPinIndex2SoundNum[pinIndex+1];
    } else {
        Serial.print("ERROR pin2soundnum() - pinIndex="); Serial.print(pinIndex);
        soundNum = SOUNDNUM_INVALID; // not really needed
    }
    return(soundNum);
} // end pin2soundnum()

```

04 Bananas Software - loop() - repeats

- Executes block every 50 milliseconds
- Gets active PinIndex
- If this is change in PinIndex: start sound
 - Could be "silence" sound
- Not a change, still holding: repeat sound
- No key pressed: repeat silence
- Extra credit: can you simplify the last if/else/endif ?

```
/////////////////////////////////////////////////////////////////
// loop()
void loop() {
  EVERY_N_MILLISECONDS( 50 ) {
    gCurrentPinIndex = handle_capacitive(); }
    if (gPrevPinIndex != gCurrentPinIndex) {
      // "key" (PinIndex) is different than before so start a new sound
      // -1 will start the silent sound, otherwise the chosen key sound will start
      gPrevPinIndex = gCurrentPinIndex;
      DFstartSound(pin2soundnum(gCurrentPinIndex), SOUND_DEFAULT_VOL);
    } else if (DFcheckSoundDone()) {
      if (gCurrentPinIndex >= 0) {
        // PinIndex is not -1 so we are still holding a key down - restart sound
        gPrevPinIndex = gCurrentPinIndex;
        DFstartSound(pin2soundnum(gCurrentPinIndex), SOUND_DEFAULT_VOL);
      } else {
        // PinIndex is -1 so no key is held down - start silence sound
        gPrevPinIndex = -1;
        DFstartSound(pin2soundnum(gCurrentPinIndex), SOUND_DEFAULT_VOL);
      } // end if
    } // end EVERY_N_MILLISECONDS
  } // end loop()
}
```

04 Bananas Software - setup() - one time

- Initialize serial debug interface
 - Do capacitive sensing setup
 - Initialize sound card interface
 - Initialization complete
 - Play intro sound then start
- ```
////////////////////////////////////
// setup()
void setup() {
 Serial.begin(115200); // this serial communication is for general debug; set the USB serial port
 while (!Serial) {
 ; // wait for serial port to connect. Needed for native USB port only
 }
 Serial.println(""); // print a blank line in case there is some junk from power-on

 CapacitiveSetup();

 pinMode(DPIN_AUDIO_BUSY, INPUT_PULLUP); // HIGH when audio stops
 mySoftwareSerial.begin(9600); // this is control to DFPlayer audio player
 // initialize the YX5200 DFPlayer audio player
 DFsetup();

 Serial.println("TODAS init complete...");

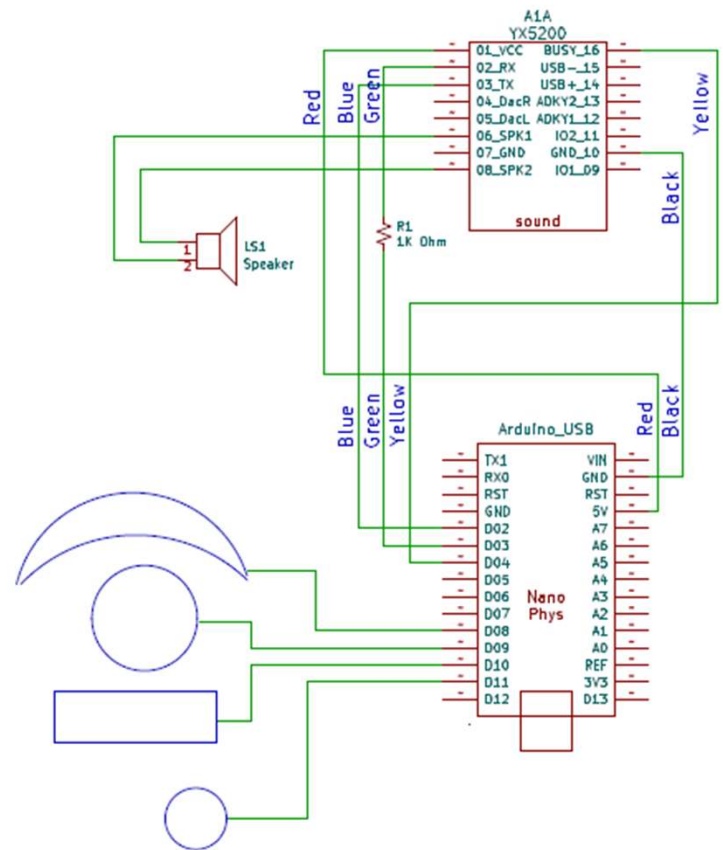
 // play the INTRO sound to completion, then allow normal loop() processing
 DFstartSound(SOUNDNUM_introduction, SOUND_DEFAULT_VOL);
 while (!DFcheckSoundDone()) {
 delay(10); // wait for the INTRO sound to finish
 } // end while
 Serial.println("Intro Sound Complete");
 gCurrentPinIndex = gPrevPinIndex = -1;
 DFstartSound(SOUNDNUM_silence, SOUND_DEFAULT_VOL);

} // end setup()
```



## 04 Bananas Schematic

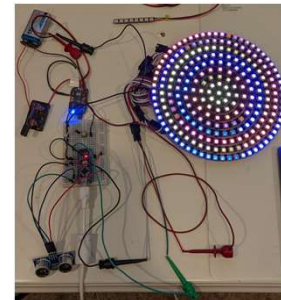
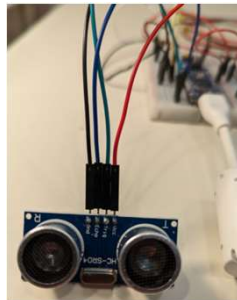
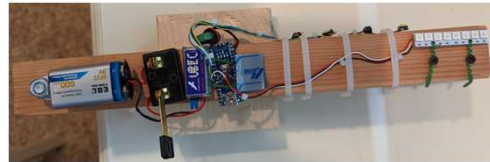
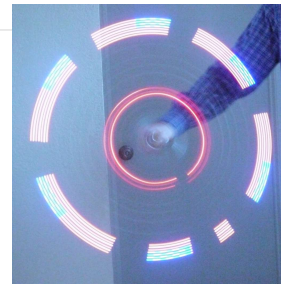
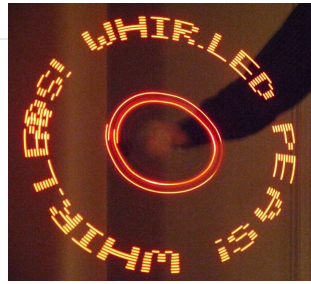
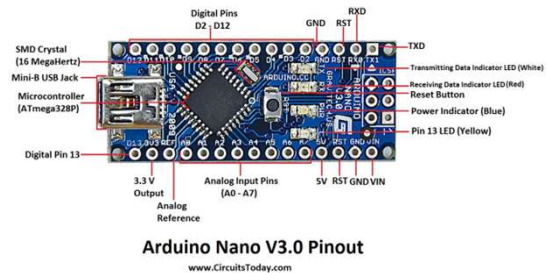
- Bottom half has Arduino and fruits
  - Capacitive sensing of fruits
- Top half has sound module and speaker
  - UART serial interface
  - Universal Asynchronous Receiver / Transmitter



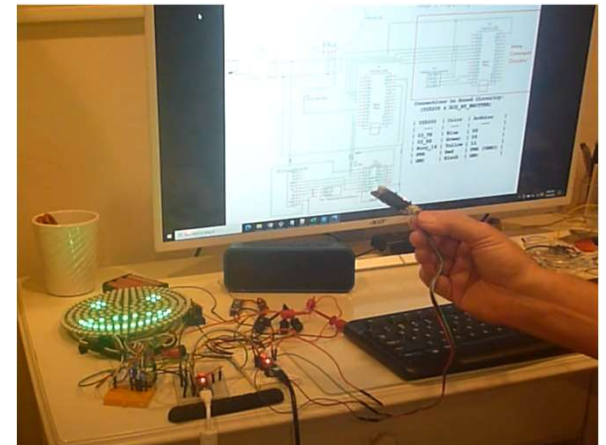
# Resources - Arduino



<https://github.com/Mark-MDO47/ArduinoClass>



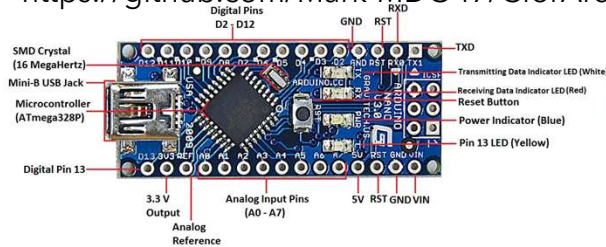
If you know a bit of programming...



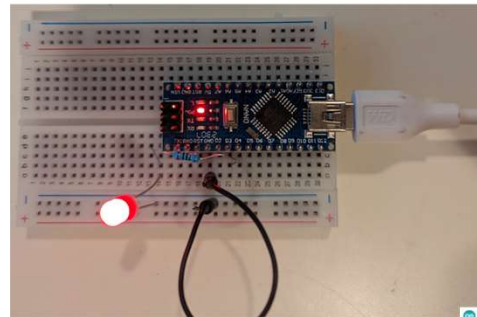
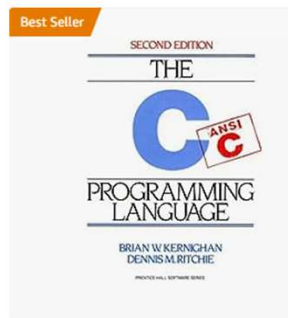
# Resources - C for Arduino



<https://github.com/Mark-MDO47/CforArduinoClass>



Arduino Nano V3.0 Pinout  
www.CircuitsToday.com



If you can use a programming refresher...

