

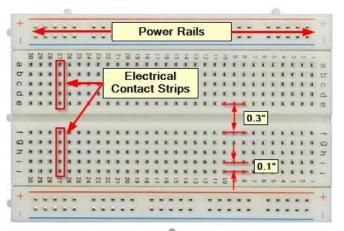
TODAS - Electricity, Arduinos, Fun

- Introduction
- Water Flow a way to think about Electricity
- Arduino (computer on a chip), buttons, and LEDs
- Sonar Control and big LED ring
- Bananas and Sounds
- Resources

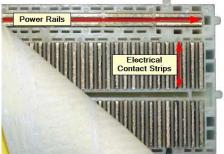
Introduction

- We have one hour it is just an introduction
 - The idea is to do a quick overview
 - Questions OK but this is not a deep course
 - Lots of resources on the web for further investigation
- 4 projects, 3 using a small computer called Arduino
 - Do some wiring
 - Customize some software
 - Have some fun

01 - Breadboards - how do they work?

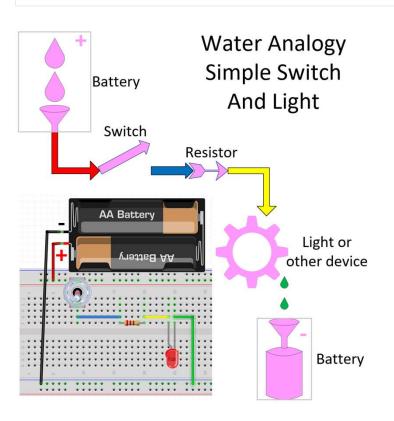


- Sides have power rails length
- Center has contact strips width



• Images from protosupplies.com

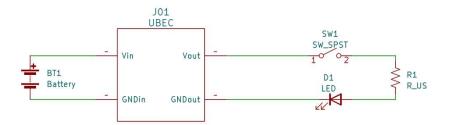
01 - Water Analogy - Simple Circuit



- 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!
- Positive to negative flow was good enough for Benjamin Franklin and for me!
- "Pictorial" circuit is "Fritzing Diagram"
 - https://fritzing.org/ 8 Euros

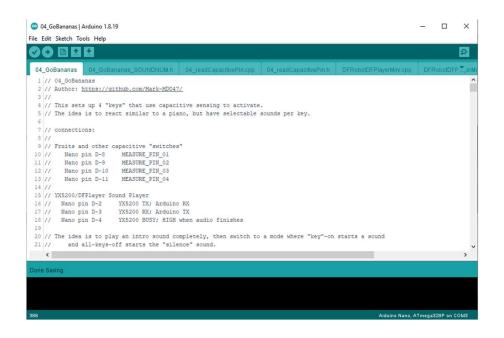
01 Simple Circuit Schematic





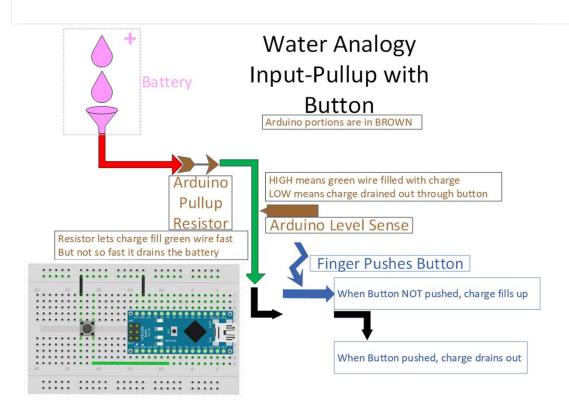
- The Universal Battery Eliminator Circuit (photo of type I used) produces 5 volts if the input is from 5.5 Volts to 26 Volts.
- The resistor prevents sending too much current through the LED; that would burn it up.
- Schematic diagram is from KiCad
 - https://www.kicad.org/ KiCad is free!

02 - Arduino IDE



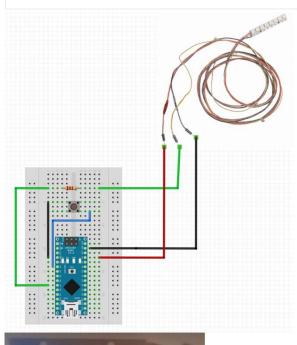
- IDE means Integrated
 Development Environment
- https://www.arduino.cc/en/software Arduino IDE is free!

02 - Water Analogy Input-Pullup & Button



 Remember: electrons in wires actually flow from negative to positive, but often easier to understand by pretending positive to negative

02 - Arduino Moving Color Lights

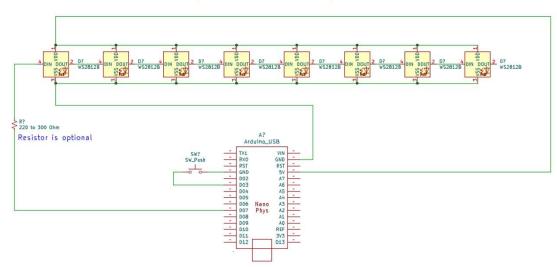




- "Fritzing diagram" of color lights in motion!
 - LED is Light Emitting Diode
- Configure software for color choices
- Press button to stop, release to continue
- How does it work? SOFTWARE!
 - Pins can be input or output you tell it which
 - Read the button
 - If not pressed display next light
 - Short 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

- When button is pushed, Arduino input is LOW
- When button is not pushed, Arduino input is HIGH
- 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

02 - Arduino Code Structure - Empty Sketch

- Empty "sketch" has
 - setup() called once at start
 - Initialize hardware
 - Initialize software
 - loop() called repeatedly
 - Process events
 - Generate outputs

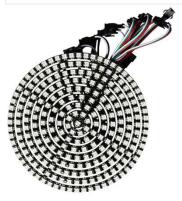
```
void setup() {
2   // put your setup code here, to run once:
3 }
void loop() [[
5   // put your main code here, to run repeatedly:
6 }
```

02 - Arduino Button Input Software

- this defines a name for pin D3 for the button
- Inside "setup()" pinMode sets that pin to be an input with "pullup"; HIGH unless connected to ground by pushing button
- ptrn_phase() is called from code inside "loop()"
 - Because current_phase is "static", it remembers previous value from last time ptrn_phase() was called
 - The "if(HIGH" block does the motion if button NOT pushed
 - If button WAS pushed we return current_phase of -1 to signify stopping LED motion

```
#define BUTTON PIN STOP 3 // press to stop action; release to restart
    pinMode (BUTTON PIN STOP, INPUT PULLUP); // digital INPUT PULLUP means voltage HIGH unless grounded
85 // ptrn phase() - determine the state of the phase of pattern generation
        returns: long int with either value >= 0 phase to blink or value < 0 STOP
89 long int ptrn_phase() {
90 static long int current phase = -1;
92 wif (HIGH == digitalRead(BUTTON PIN STOP)) {
      current phase += 1;
      current phase %= gPatternsRepeat; // loop through the number of calls before repeat
95
      if (0 == current phase) {
96
        step color value();
97
    → current phase = -1; // STOP
100
101
    return (current phase);
103 } // end ptrn phase()
```

03 - Go Big with Ultrasonic Sonar Control

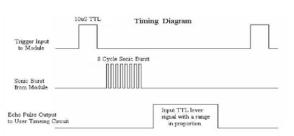




- 8 LEDs is fun but 241 LEDs is MORE fun!
 - TLDR WS2812B individually addressable color LEDs using FastLED Library
- Displays moving patterns we choose
- Disk is mostly wired; less soldering needed
- Needs more power than Arduino can give
- Video: https://youtube.com/shorts/0KehSIJmKcs
- Pictures from amazon.com
 - TLDR https://www.amazon.com/WESIRI-WS2812B-Individually-Addressable-Controller/dp/8083VWVP3J

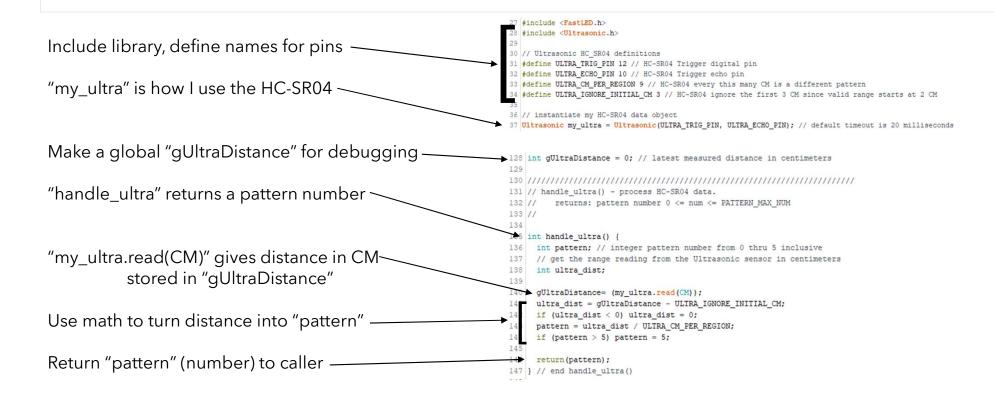
03 - Ultrasonic Sonar Control



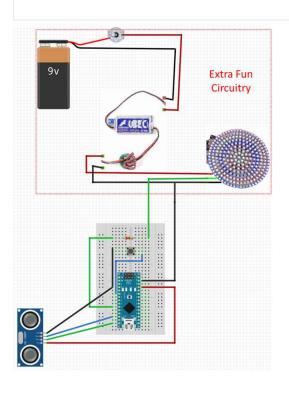


- Ultrasonic Sonar sensor detects distance
- Bounces ultrasonic sound off objects
- Set "Trig" HIGH then LOW to start sonic burst
- Measure time until "Echo" is HIGH
- Divide time by speed of sound to get distance
- TLDR diagram from the sparkfun.com HC-SR04 spec.

03 - Ultrasonic Sonar Detector Software

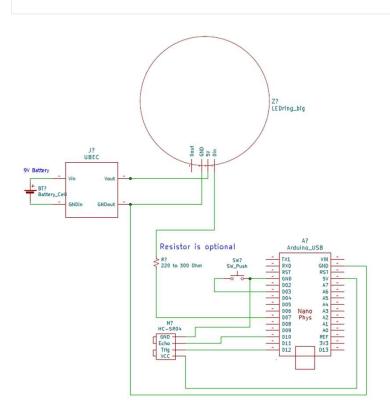


03 - Fritzing Diagram



- 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.

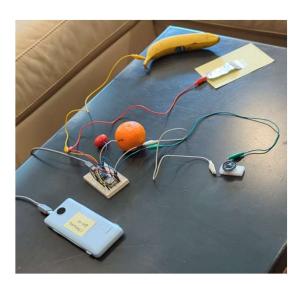
We leave it on the breadboard; no need to remove it.

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

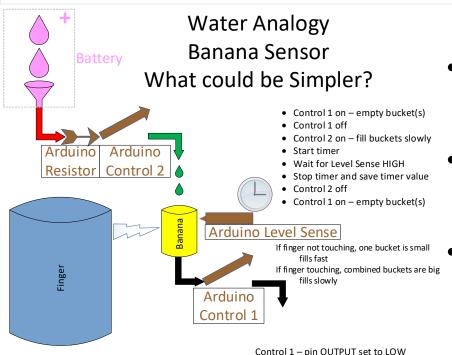
04 - Go Bananas!

- 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/EC1qHbE89Jl
- TLDR Image to the left by krakenimages.com on Freepik



04 - Water Analogy - Banana Input

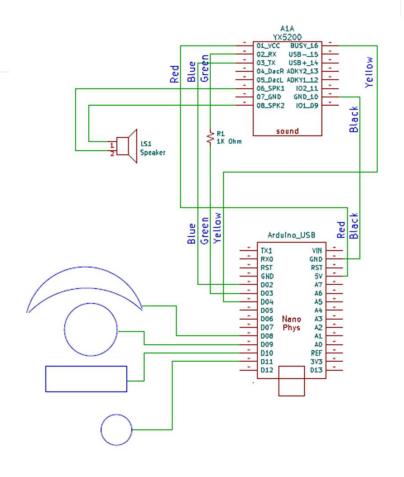
Control 2 - pin INPUT with pull-up resistor



- Note that this is an analog of how it works
- The diagram shows the flow of the code
- The code uses tricks to be fast makes code harder to understand
 - But that is all hidden in 04_readCapacitivePin.*

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



04 Bananas Software - customization

```
// "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_ORGAN_ORGAN_ORGAN_ORGAN_ORGAN_ORGAN_ORGAN_ORGAN_ORGAN_ORGAN_ORGAN_ORGAN_ORGAN_ORGAN_ORGAN_ORGAN_OR
```

- 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) {
  uintl6_t soundNum = SOUNDNUM_INVALID;
 if ((pinIndex >= -1) and (pinIndex < NUM MEASURE PINS)) {
  soundNum = gPinIndex2SoundNum[pinIndex+1];
   else {
   Serial.print("ERROR pin2soundnum() - pinIndex="); Serial.pri
    soundNum = SOUNDNUM_INVALID; // not really needed
 return (soundNum);
} // end pin2soundnum()
```

04 Bananas Software - loop() - repeats

```
    Executes block every 50 milliseconds

                                                                         // loop()

    Gets active PinIndex —

                                                                         void loop() {
                                                                           EVERY N MILLISECONDS ( 50 ) {
                                                                             gCurrentPinIndex = handle_capacitive(); }
                                                                             if (gPrevPinIndex != gCurrentPinIndex) {
                                                                              // "key" (PinIndex) is different than before so start a new sound

    If this is change in PinIndex: start sound -

                                                                                    -1 will start the silent sound, otherwise the chosen key sound will start
      · Could be "silence" sound
                                                                               gPrevPinIndex = gCurrentPinIndex;
                                                                              DFstartSound(pin2soundnum(gCurrentPinIndex), SOUND DEFAULT VOL);
• Not a change, still holding: repeat sound •
                                                                             } 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);

    No key pressed: repeat silence —

                                                                                // PinIndex is -1 so no key is held down - start silence sound
                                                                                gPrevPinIndex = -1;
                                                                               DFstartSound(pin2soundnum(gCurrentPinIndex), SOUND DEFAULT VOL);
• Extra credit: can you simplify the last if/else/endif?
                                                                           } // end EVERY N MILLISECONDS
• Extra credit: can you write simple code to replace
                                                                         } // end loop()
   EVERY N MILLISECONDS(50)? Do we even need this?
```

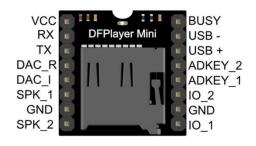
04 Bananas Software - setup() - one time

```
• Initialize serial debug interface
                                                       void setup() {
                                                         Serial.begin(115200);
                                                                                      // this serial communication is for general debug; set the USB serial por
                                                         while (!Serial) {
                                                           ; // wait for serial port to connect. Needed for native USB port only

    Do capacitive sensing setup

                                                         Serial.println(""); // print a blank line in case there is some junk from power-on
                                                         CapacitiveSetup();
· Initialize sound card interface.
                                                         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();
• Initialization complete
                                                         Serial.println("TODAS init complete...");
                                                         // play the INTRO sound to completion, then allow normal loop() processing
                                                         DFstartSound(SOUNDNUM_introduction, SOUND_DEFAULT_VOL);
• Play intro sound then start •
                                                         while (!DFcheckSoundDone()) {
                                                           delay(10); // wait for the INTRO sound to finish
                                                         Serial.println("Intro Sound Complete");
                                                         gCurrentPinIndex = gPrevPinIndex = -1;
                                                         DFstartSound(SOUNDNUM silence, SOUND DEFAULT VOL);
                                                       } // end setup()
```

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 https://github.com/Mark-MDO47/AudioPlayer-YX5200
- Espeak for robotic voice <u>https://github.com/Mark-MDO47/RubberBandGun/tree/master/sounds</u>
- Audacity for sound processing <u>https://www.audacityteam.org/</u>

Resources - Arduino



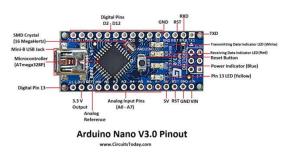


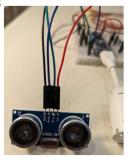


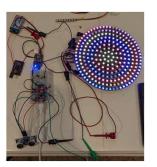
If you know a bit of programming...

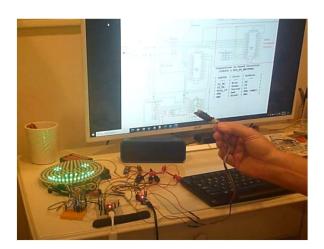








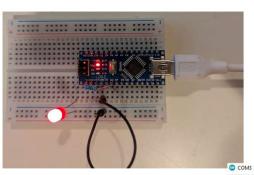




Resources - C for Arduino







If you can use a programming refresher...

