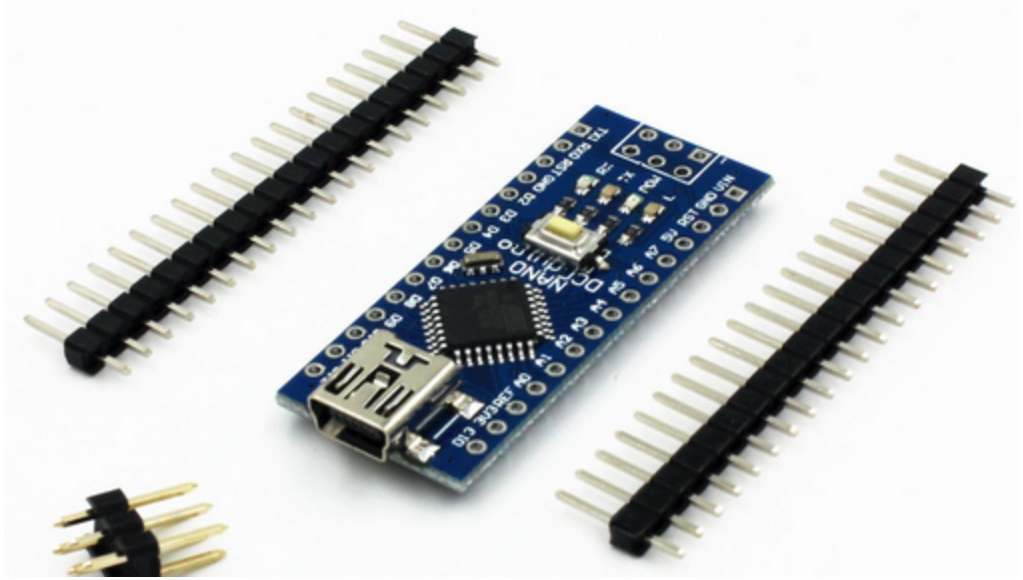
...RRRduino...

Sound by Sensing…

(for absolute beginners, like me)



by Speed Muller

v0.02

If you don't care, don't like electronics and don't want to be bothered, write this down:

www.TxNamib.com

and

blog.RRRduino.com

and

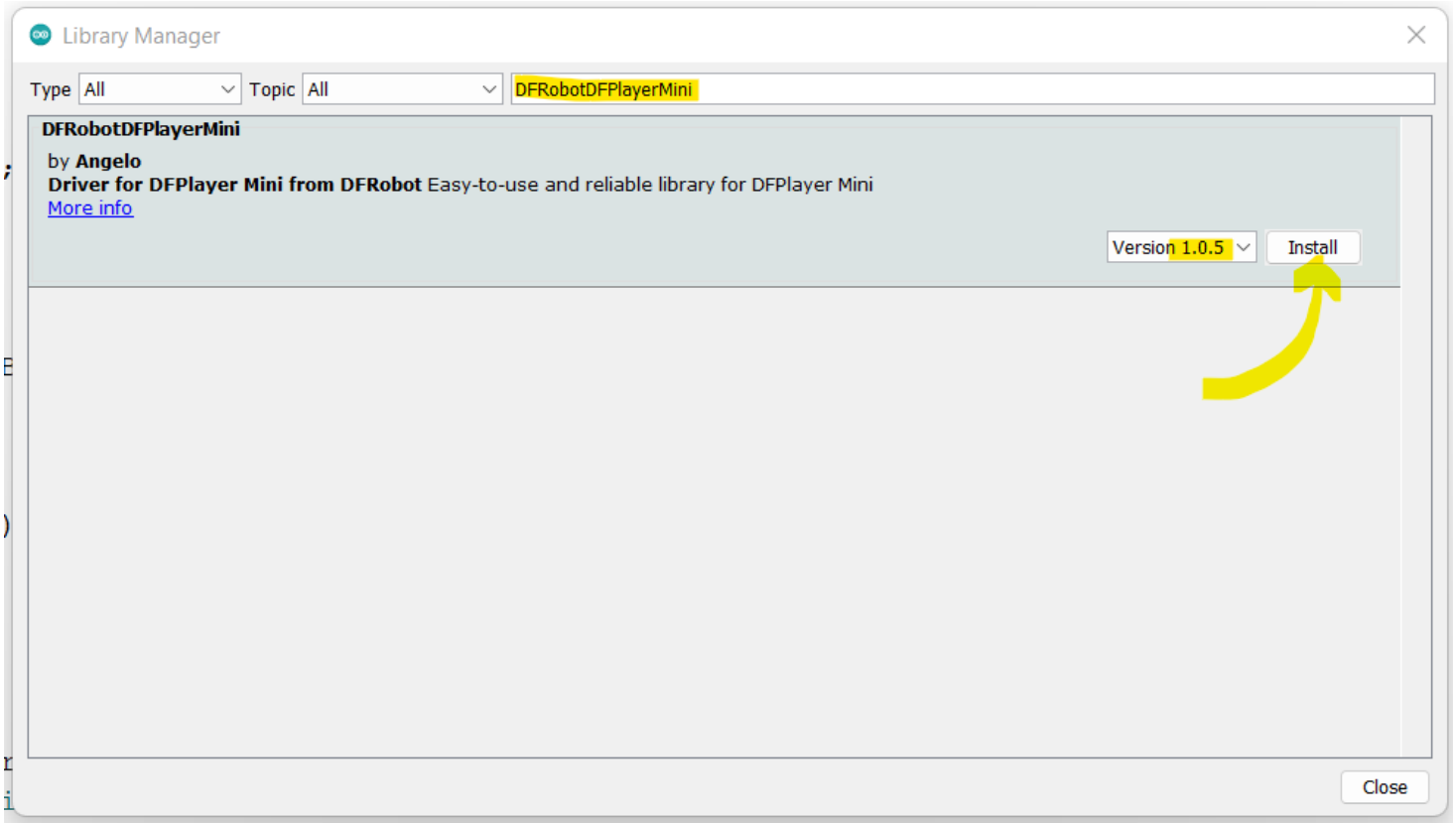
mqTrains.com

and then go ahead, take that nap!

Parts? I think we should be ready for a clinic!

Before we start, DFPlayerMini Library (i)...

- If you have an internet connection, launch the Arduino IDE
- Go to Sketch → Include Library → Manage Libraries… *(or press Ctrl+Shift+i)*
- Search for "**DFRobotDFPlayerMini**" in the top row
- Select the one by Angelo, and **Version 1.0.5**, *(run newer versions with your own dev ops support system)*
- Click **Install** and then **Close** when done installing.



Before we start, DFPlayerMini Library (ii)…

- If you don't have an internet connection, launch the Arduino IDE
- Go to Sketch → Include Library → Manage Libraries… *(or press Ctrl+Shift+i)*
- Search for "**DFRobotDFPlayerMini**" in the top row
- Select the one by Angelo, and **Version 1.0.5**, *(run newer versions with your own dev ops support system)*
- Click **Install** and then **Close** when done installing.

Before we start, Preferences…

- Please open the File -> Preferences (Ctrl+comma) and set and unset the following settings:

Check:

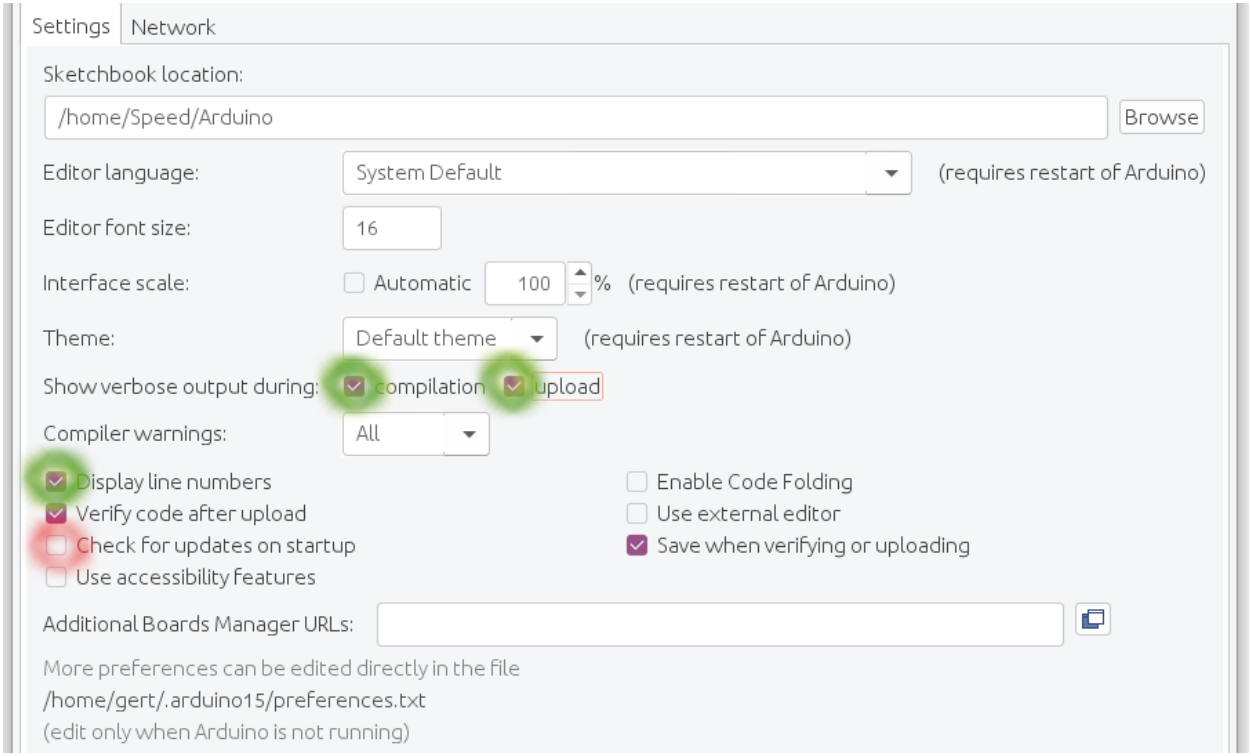
1. Show verbose output during: [x] **compilation** and [x] **upload**
2. [x] **Display line numbers**

Uncheck:

3. [ ] Check for **updates on startup**

After the clinic, you can check this one again, but the lack of internet is going to slow us down in the clinic room!

- Then click **OK** at the bottom to make it real.



What is an Arduino?

Quoted: (http://www.circuitstoday.com/story-and-history-of-development-of-arduino)

The new prototype board, the Arduino, created by Massimo Banzi and other founders, is a **low cost microcontroller board** that allows even a **novice** to do great things in electronics. An Arduino can be connected to all kinds of **lights**, **motors**, **sensors** and **other devices**; an easy-to-learn programming language can be used to program how the new creation behaves. Using the Arduino, you can build an **interactive display** or a **mobile robot** or anything that you can imagine.

You can purchase an Arduino board for **just about US $30** or **build your own** board from scratch. Consequently, Arduino has become the most powerful open source hardware movement of its time.

Today, there are Arduino-based **LED cubes**, **Twitter displays**, **DNA analysis kits**, **breathalyzers** and so much more. There are Arduino parties and Arduino clubs. As a feather to its crown, Google has recently released an Arduino-based development kit for its Android Smartphone!

FOR US? **Flickering lights, fire trucks, ambulances, police cars, crossing gates (even bringing the gates down WITH the bell ringing), signals, semaphores, turnout control, airfield lights, animation with servos, steppers and DC motors. Sensors, counting and reporting axles, and randomly nagging about a hot wheel! Also LCC, BlueTooth, WiFi, CAN Bus, transmitting data across your layout.**

How did it come about?

Quoted (http://www.circuitstoday.com/story-and-history-of-development-of-arduino):
It was in the Interactive Design Institute that a **hardware thesis** was contributed for a wiring design by a Colombian student named **Hernando Barragan**. The title of the thesis was "Arduino–La rivoluzione dell'open hardware" ("Arduino – The Revolution of Open Hardware"). Yes, it sounded a little different from the usual thesis but none would have imagined that it would carve a niche in the field of electronics. A team of **five developers** worked on this thesis and when the new wiring platform was complete, they worked to make it much lighter, less expensive, and available to the open source community.

...the Story in more Detail…

As mentioned earlier, it all started in **Ivrea, Italy**. To begin with, let's have a look at how the name Arduino, which sounds quite strange for an electronic device, was chosen. This beautiful town of Ivrea, situated in Northern Italy, is quite famous for its underdog kings. In the year 1002 AD, **King Arduin** (you got it right!) ruled the country; two years later, he was dethroned by King Henry II of Germany. In the memoir of this King Arduin, there is this 'Bar Di Re Arduino', a **pub on the cobble stoned street** in the town. Well, this place is where a new era in electronics had its roots!

This bar was frequently visited by **Massimo Banzi**, one of the founders of Arduino, who taught at Ivrea. He was the one who gave the name Arduino to this low-cost microcontroller board in honor of the place!
Before getting into how the Arduino was developed and used, let's know who the core members of the Arduino developer team are: **Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis**.

The First Prototype Board

Well, Banzi succeeded in creating the first prototype board in the year **2005**; it was a simple design and at that time, it wasn't called Arduino. *Of course, by now, you would know how he had coined the name later that year.*

Open Source Model – A Big Decision

Banzi and his collaborators strongly believed in **open-source software.** As the purpose was to develop a quick and easily accessible platform, they thought it would be better to open up the project to as many people as possible instead of keeping it closed. Another crucial factor that contributed to that big decision was that after operating for nearly five years, IDII had no more funds left and was in fact going to shut its doors. All the faculty members feared that their projects might not survive or would be embezzled. It was at this crucial point of time that Banzi decided to go ahead and make it open source!

How Banzi and team managed to create Arduino and make it available for public

Pretty obviously, the open source model had always been used to fuel innovation for software and never **hardware**. If they had to make it work, they had to find a suitable licensing solution that could apply to the board. After a little investigation, Banzi and team looked at the whole thing from a different angle and decided to use a license from **Creative Commons**, a nonprofit group whose agreements were normally used for cultural works like writing and music. *According to Banzi, hardware is a piece of culture that must be shared with other people!*

Well, the next step was to make the board. The group decided to fix a specific, student-friendly price of **$30** as their **goal**. Banzi felt that the Arduino should be affordable for all students. However, they also wanted to make it really quirky, something that would stand out and look cool as well. While other boards were green, they wanted to make theirs **blue**. While a few manufacturers saved on input and output pins, they added a lot to their board. Quite weirdly, they added a little **map of Italy** on the back of the Arduino board!

Key to the Slides:

This is some information

Here is some more information

You have to do something…

- And each step is likely in blue too!

Where to Start?

- Buy one ... and plug it in!

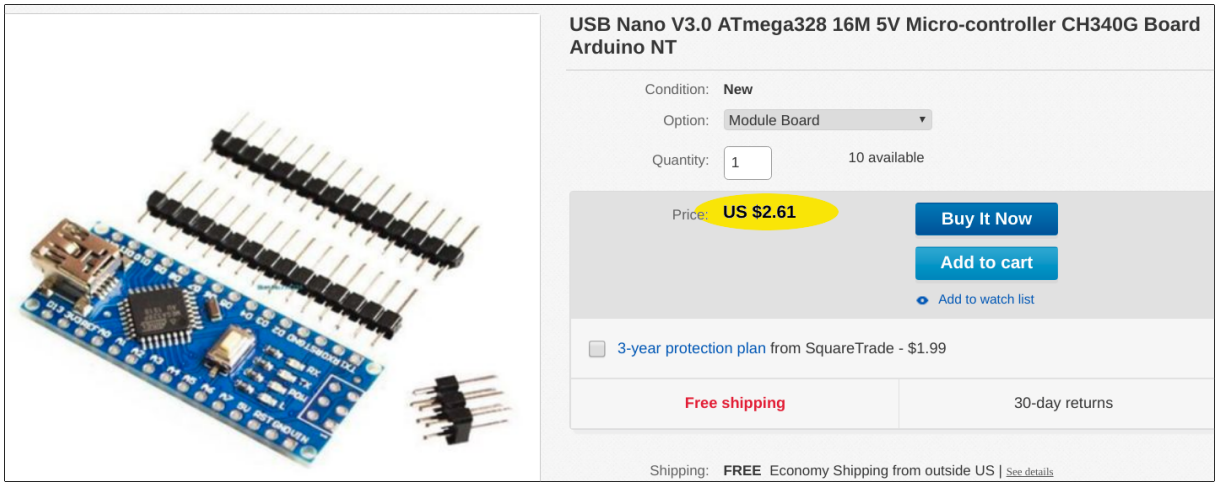
[PC → USB Cable → Nano]

See a **solid** red light and a **blinking** red* light?

Where did we get it?

Search Ebay for “**Arduino Nano USB”** … I tend to buy the ones from Hong Kong, and it feels like they ship faster *(<-- this sounds so 2016!!!)*.
Version 3.0 is newer, and we like 5Vdc too.
The CH340G USB/Serial chip requires a device driver, but Google can help you disable the device driver signing in Windows 8+, just one reboot required.

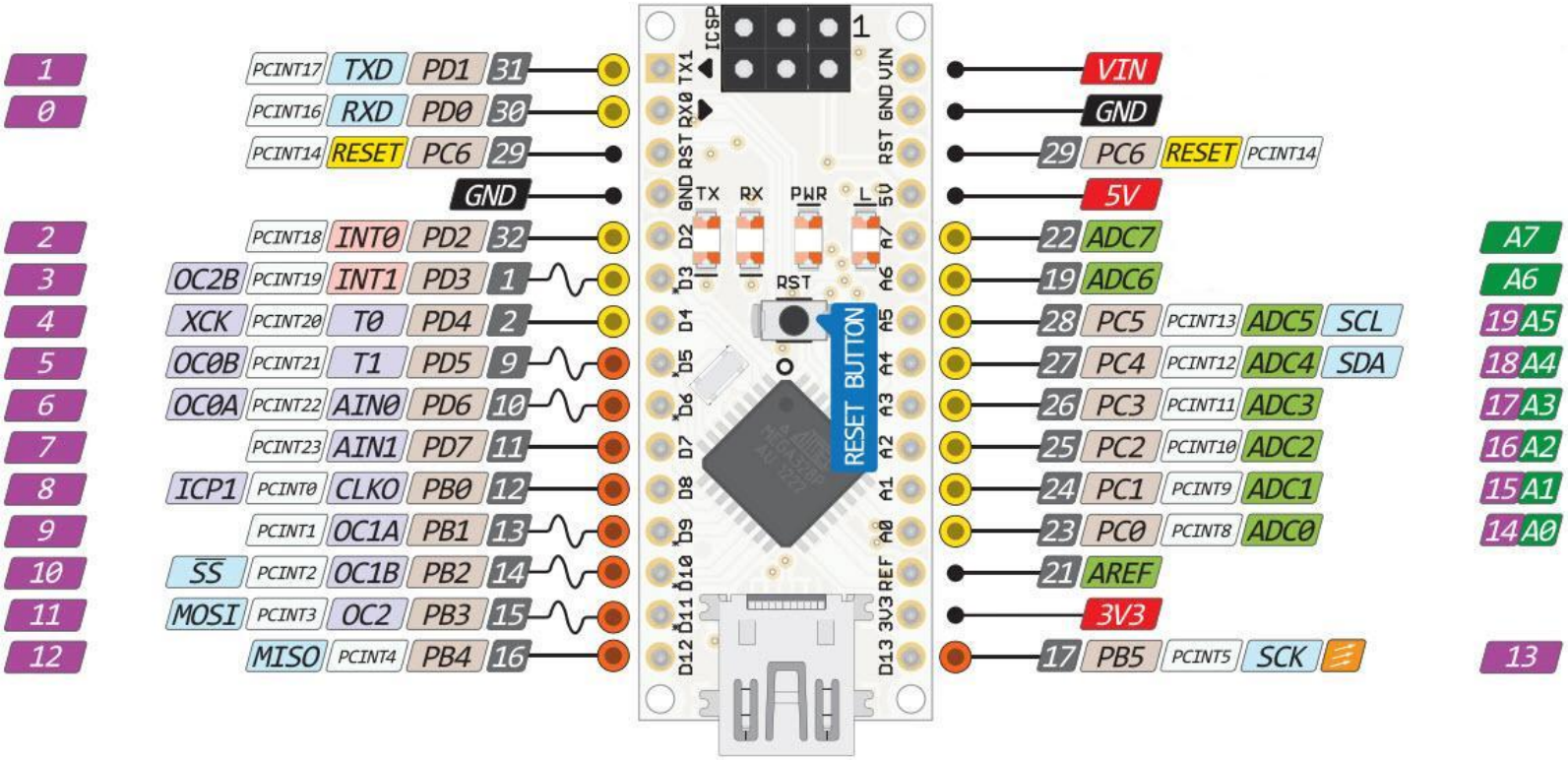
(Linux guys, sorry for that time waster, you were good without installing anything)



Of course, Adafruit, AliExpress, Amazon and Sparkfun would help you too.
Even MicroCenter has them in stock.

Here is what we just plugged in:

(Open Source, Open Hardware, so you get the schematic too)

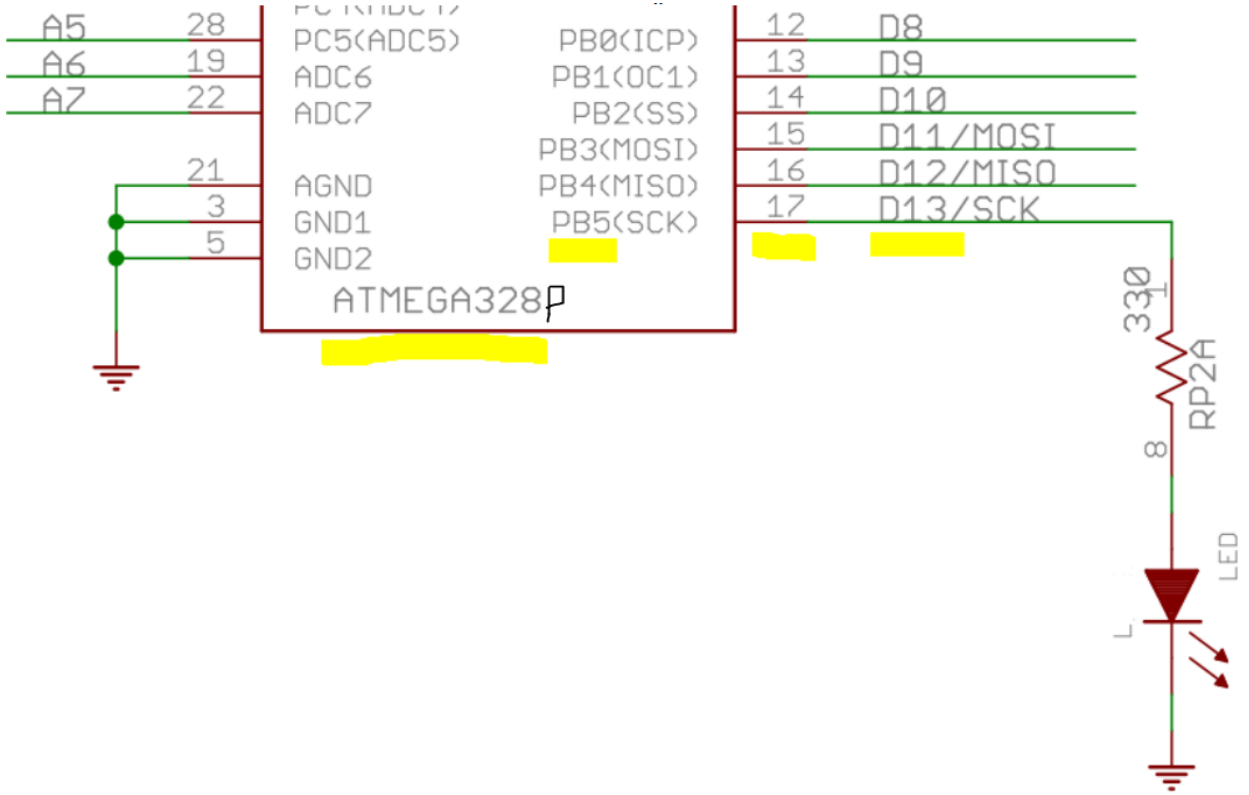


What you care about, is the **purple** and **green** numbers on the far left and right, since those are the numbers the software needs.

Green indicated pins needed for Analog **inputs**, but all the pins can do digital things.

Also note the lines on D3, D5, D6, D9, D10 and D11 have a wiggle, those can be **PWM** outputs...in simple terms, they could look like an Analog **output** with a real resistor and capacitor as filter.

Pins Ports and Numbers:



- The Atmel ATMEGA328P-PU chip has pin 17 (labeled Port.B bit 5 and Serial Clock) connected to the Arduino pin D13 … and hooked to an LED on the Arduino board. So if you toggle D13, the LED toggles on and off
- Microcontrollers can usually also sink more current than source current, so when we become advanced designers we will put the LED and resistor between the power source and the pin…and then make the pin

OK, Back up for 1 minute:

What is **Arduino** again?

- It certainly has **Hardware**

- yes we know (and you get to see the schematic and board layout, like in the real files containing it, I mean)
- it has **shields** and **plug-ins** with interfaces to almost everything else in the world, and you can add the missing ones!

- It has a **Software** development platform

- IDE, C/C++ language, library extensions, compiler, upload tool and serial monitor, oops Bootloader too.

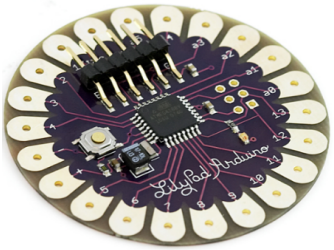
- It has a **following**, even today's kids know about it

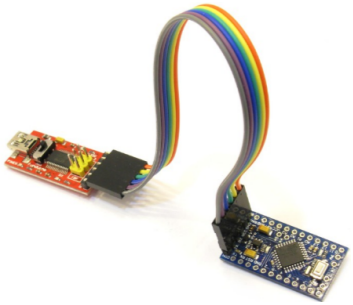
- It is world wide! It has websites, blogs, examples, even facebook and twitter, don't you?

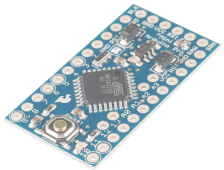
...It is a whole platform with tools...

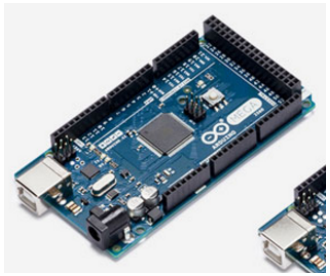
Hardware?

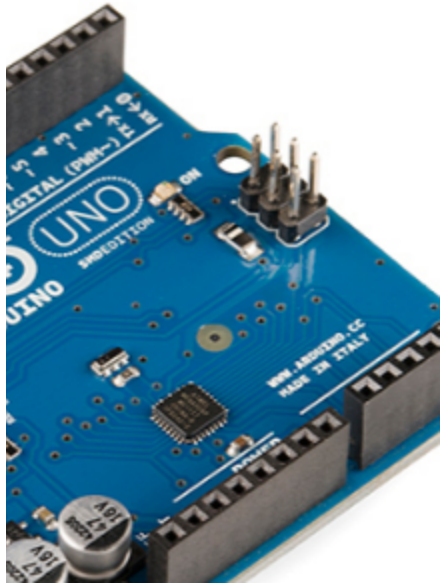
UNO was first: Atmel ATMEGA328P in a PU package, with an “NFL” size USB plug:









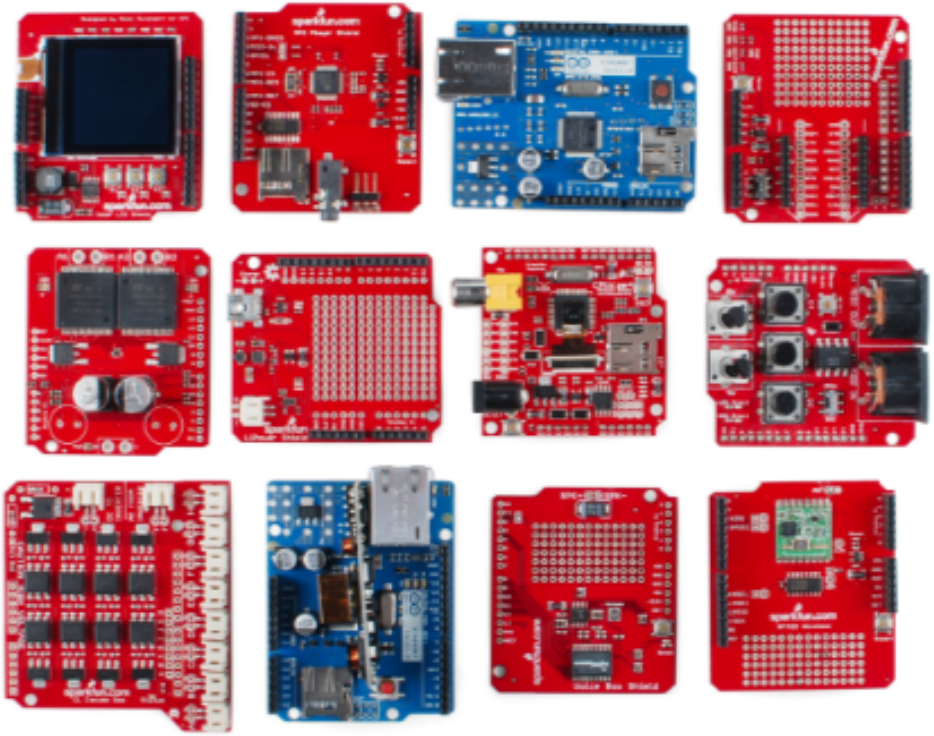


Then an SMD UNO version was made.

Wait, my Nano has that chip too!!!
...and a mini USB connector
...which the Micro does not have.

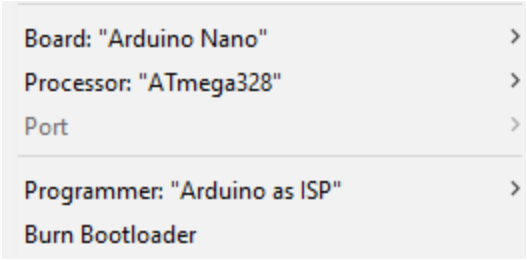
A short word on Shields

- That is how Arduinos connect to the world, if the functions is not already present
  - Simple “breadboard” Shield, so you could solder a wire to something
  - Motor Shield: DC, Stepper, Servo Motor
  - LCD Shield
  - Audio Amplifier
  - SD Card reader
  - Ethernet or WiFi Shield
  - CAN Bus
- Of course, different Shields needed for different Arduinos, Nano vs Uno
  - Plan ahead!
  - Or make your own, Fritzing, KiCAD and EasyEDA.com is free



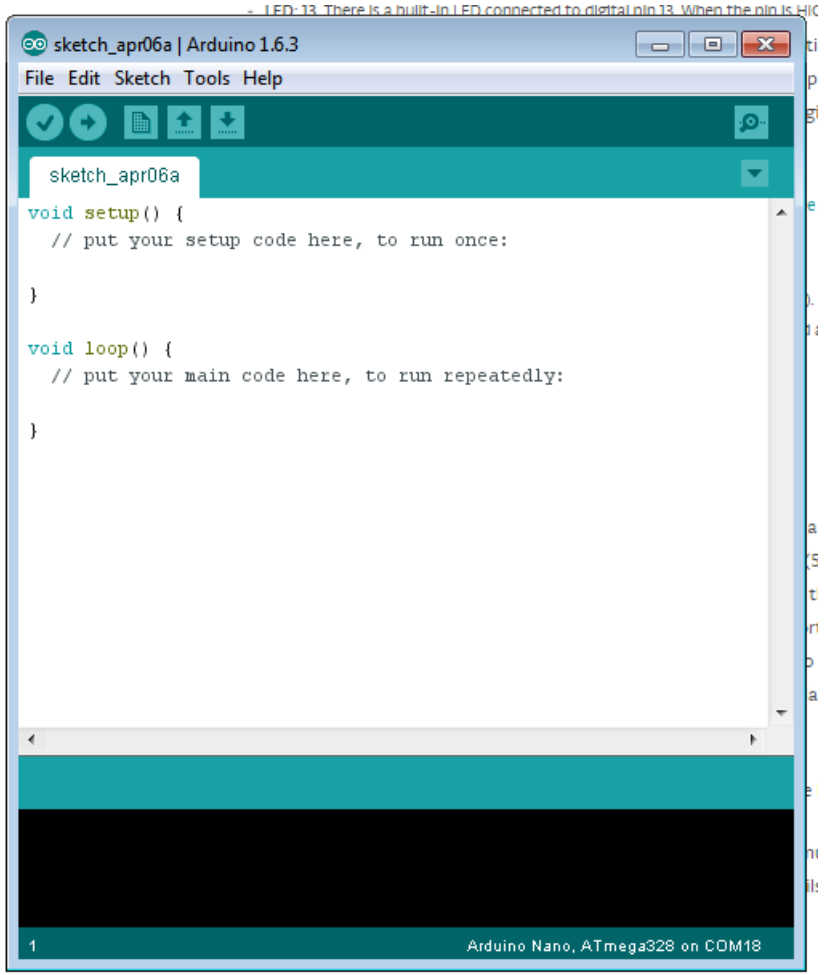
Programmers

- Bootloader
- FTDI most common, CH340G needs a device driver. Read about FTDI-gate.
- An Arduino can become a programmer to program another
- ATMEGA328P-xx vs no’P’-xx
  - Low power version vs lower cost version



And then you said Software!

- **IDE** (integrated development environment), free download
  - Windows, Mac and Linux (last one: sudo apt-get arduino, same on Raspberry Pi)
  - 1.8.19 is latest during this St Louis Clinic in August 2022
  - Has a version “Future Version” 2.0 RC (Release Candidate) (2.0.0-rc9.1)
- **Fancy editor** colors and calling things in the background!
- **Select** your board, processor, programmer and serial port from the Tools menu
- Type your **code**, **Verify** (compile) and **Upload**!
- Tip: the upload button does it all, wait…it does not type your code, ;)
- When you save a file, it will be under the Files->Skecthbook menu
- **Serial Port Monitor** (Ctrl+Shift+M) and also a **Serial Plotter** (Ctrl+Shift+L)



Programming

icons for: VERIFY, UPLOAD, NEW, OPEN, SAVE, SERIAL MONITOR.

First example on the Nano

Nano still plugged in?

- Start the software 
- Tools → Board → Arduino Nano // done only once
- Tools → Processor → ATmega328P [(Old Bootloader)] // if needed
- Tools → Port → COMx (or /dev/ttyUSB0) // done only once
- File → Examples → 01.Basics → **Blink**

```
void setup( ) {
  pinMode( 13, OUTPUT );
} // setup( )

void loop( ) {
  digitalWrite( 13, HIGH );
  delay( 1000 );
  digitalWrite( 13, LOW );
  delay( 1000 );
} // loop( )
```

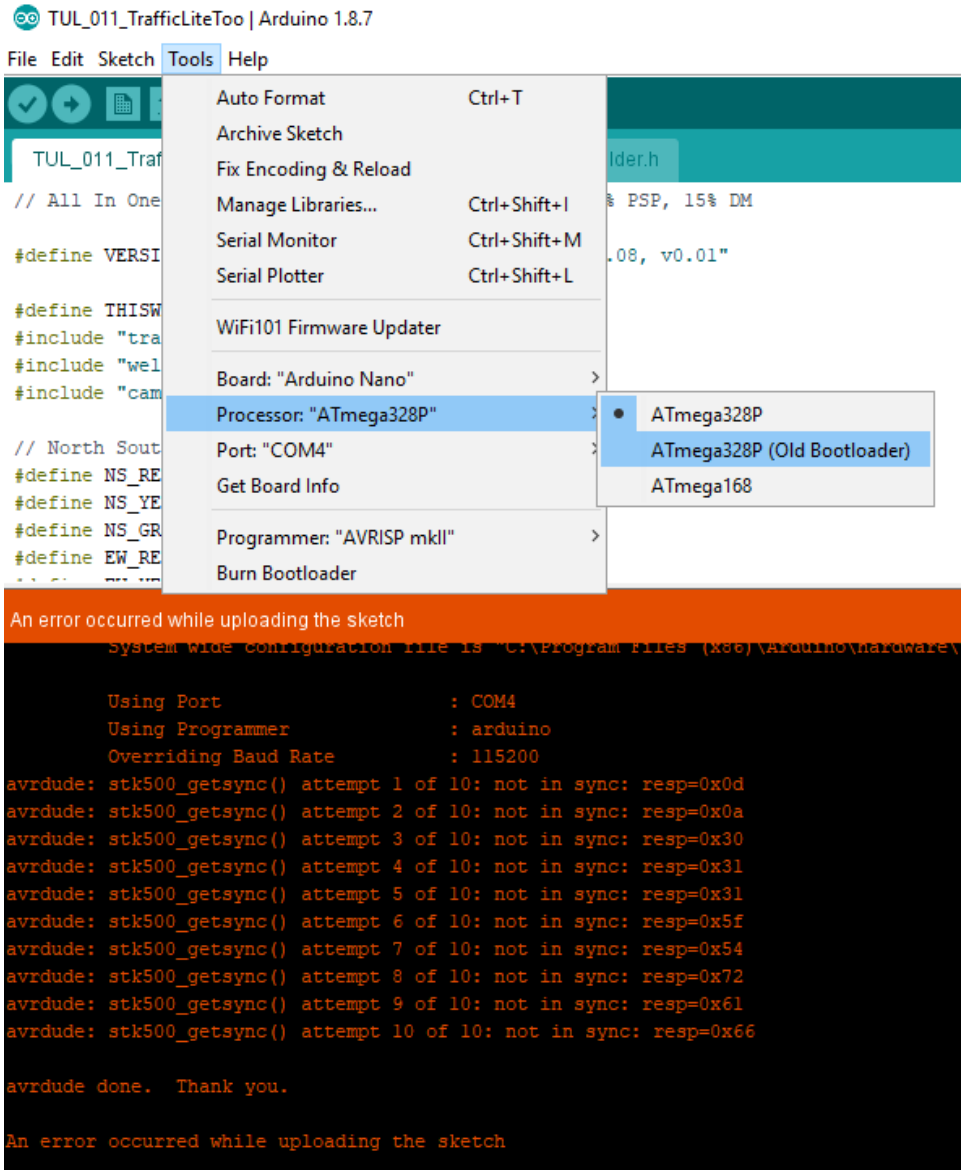
First Try…

- WIth the File → Examples → 01.Basics → **Blink** loaded in the IDE…
- Press Ctrl-R and note the stuff scrolling by in the bottom panel

```
Detecting Libraries Used…
Generating function prototypes…
Compiling sketch…
"C:\\Program Files (x86)\\Arduino\\hardware\\tools\\avr/bin/avr-g++" -c -g -Os
-w -std=gnu++11 -fpermissive -fno-exceptions -ffunction-sections
-fdata-sections -fno-threadsafe-statics -Wno-error=narrowing -MMD -flto
-mmcu=atmega328p -DF_CPU=16000000L -DARDUINO=10819 -DARDUINO_AVR_NANO
-DARDUINO_ARCH_AVR "-IC:\\Program Files
(x86)\\Arduino\\hardware\\arduino\\avr\\cores\\arduino" "-IC:\\Program Files
(x86)\\Arduino\\hardware\\arduino\\avr\\variants\\eightanaloginputs"
"C:\\Users\\Speed\\AppData\\Local\\Temp\\arduino_build_895887\\sketch\\Blink.in
o.cpp" -o
"C:\\Users\\Speed\\...\\arduino_build_895887\\sketch\\Blink.ino.cpp.o"
Compiling libraries…
…
…
… (and then)

Sketch uses 924 bytes (3%) of program storage space. Maximum is 30720 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for
local variables. Maximum is 2048 bytes.
```

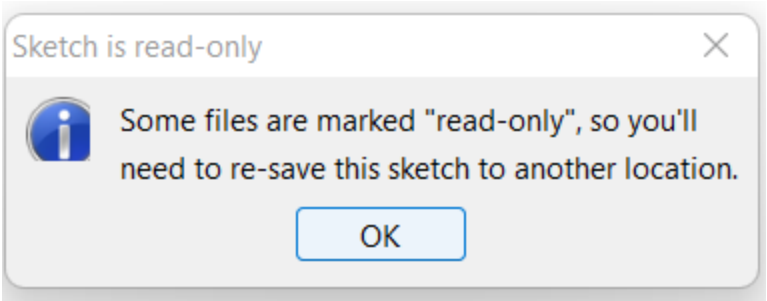
Upload Error?

- Ensure correct **Board**, **Port** and try Other Bootloader under Tools → **Processor**



Are you an Embedded Software (Firmware) Engineer?

- **Change** your **code** to match the red+yellow below
- **Verify/Compile** (Ctrl+R)
  - You might be forced to save the .ino file first (else Ctrl+S)
  - Note where the file goes, as well as the folder it goes into, when you have saved it. one.ino HAS to be in a folder one
  - C:\Users\<name>\Documents\Arduino\one\one.ino
- Now press Upload (Ctrl+U)
- Note the change in the LED on pin 13




```
void setup( ) {
  pinMode( 13, OUTPUT );
} // setup( )

void loop( ) {
  digitalWrite( 13, HIGH );
  delay( 750 );
  digitalWrite( 13, LOW );
  delay( 250 );
} // loop( )
```

Useful Arduino C/C++ commands for a beginner

pinMode(pin, IN/OUTPUT);
digitalWrite(pin, HIGH/LOW);
val_Digital_True_False = **digitalRead**(pin);
analogWrite(pwm_Pin, value);
val_Analog_0_To_1023 = **analogRead**(analog_Pin);
delay(millisecond);
val_Long = **millis**( );
if(x > 5) ;
while(j < 10) { j++; }
for(j = 0; j < 10; j++) { ; }
val = **map**(value, fromLo, fromHi, toLo, toHi);
random(min, max - 1);

Wiring? Processing? Just think it is C or C++ and move on...

https://www.arduino.cc/reference/en/

Language Reference

Arduino programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

Structure

- setup()
- loop()

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return

Variables

Constants

- HIGH | LOW
- INPUT | OUTPUT | INPUT_PULLUP
- LED_BUILTIN
- true | false
- integer constants
- floating point constants

Data Types

- void
- boolean
- char
- unsigned char

Functions

Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - *PWM*

Due & Zero only

- analogReadResolution()
- analogWriteResolution()

Variables: bool areWeReady = false; // = true

- The C and C++ languages define what are illegal names for variables
- There are many different types of variables and they all take up various amounts of memory
- A **char** is 8 bits (0-255), and **int** is usually 16 bits (depending on what you compile it for) (-32,768 to 32,767) and a **bool** is simply true or false (but HIGH or LOW and 1 or 0 is also used)
- To have some consistency across platforms, we do not like to use **char** or **int** or **unsigned int**, we prefer the predefined **uint8_t** and **int16_t** and **unint16_t** types
- To help remember what the variable type is, you will notice that I use a character or two in front: **u8**Value, **u16**Value or **i16**Value and **b**AreWeReady

This also helps with the CamelCase, a way to use English to name a variable with more descriptive words. The b, 8 or 16 is followed by a capital letter

Let's go!

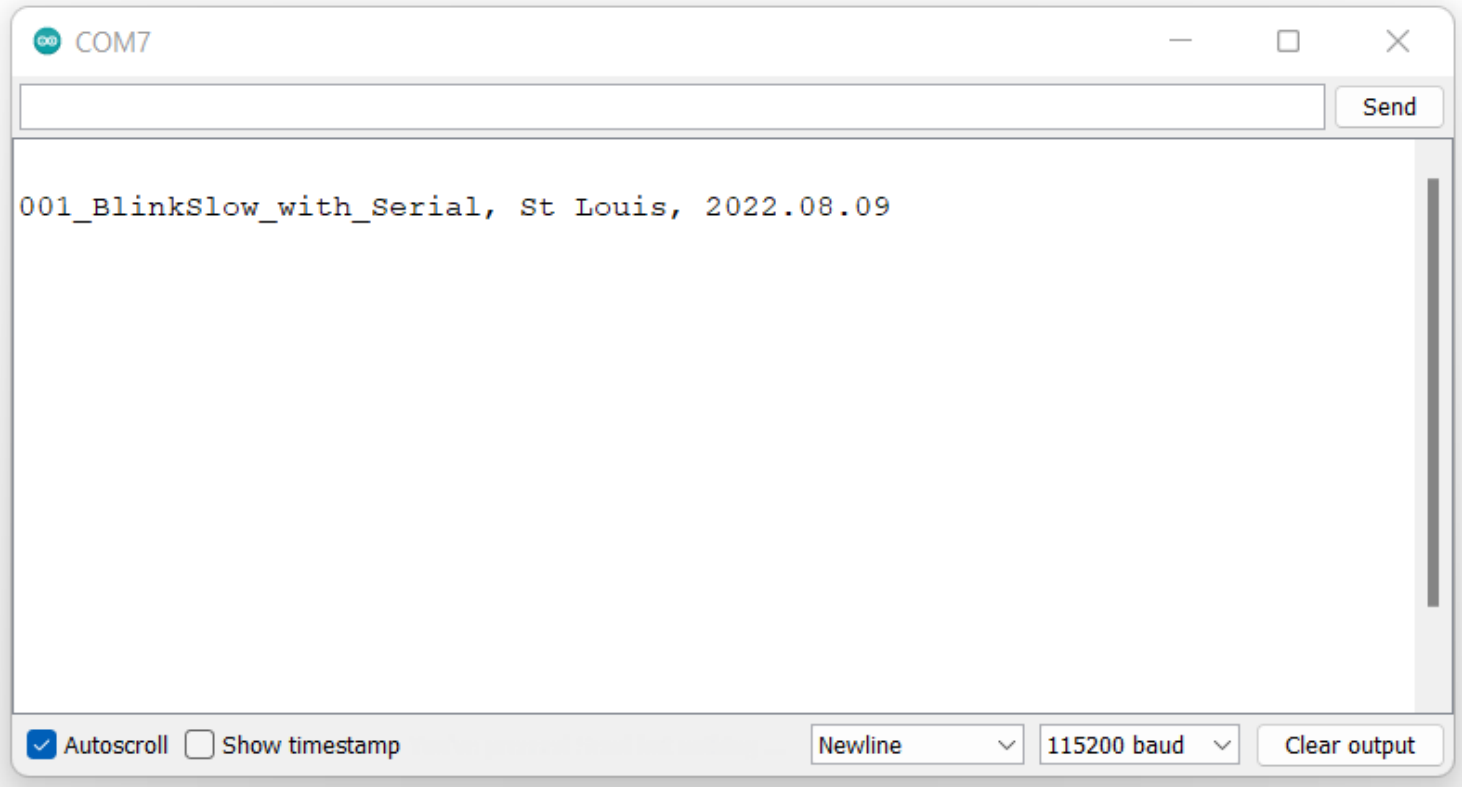
- Open **001_BlinkSlow_with_Serial.ino**

```
// Pin 13:  \__________/----------\__________/----------\__

#define LEDPIN      13

// Comments
void setup( ) {
  Serial.begin( 115200 );
  Serial.println( );
  Serial.println( "001_BlinkSlow_with_Serial, St Louis, 2022.08.09" );

  pinMode( LEDPIN, OUTPUT );
  digitalWrite( LEDPIN, HIGH );
} // setup( )

// Comments
void loop( ) {
  delay( 2000 );
  digitalWrite( LEDPIN, HIGH );

  delay( 2000 );
  digitalWrite( LEDPIN, LOW );
} // loop( )
```

Serial Port Monitor → Ctrl+Shift+M

- **Press Ctrl-Shift-M** (this opens the Serial Monitor from the IDE's Tools menu)
- **Set baud rate** in bottom right to 115200
- Loading Serial Monitor OR changing baud is like pressing the RESET button

- And this is how you find out 3 years later what is on the ATMEGA328 chip!



Tip: If your boss does not allow you to install TeraTerm, RealTerm, putty or Serial Port Monitor, install the **Arduino IDE**! Who can tell which Windows version lost Hyper Terminal?

IDE saving a Sketch

- A sketch is the .ino file in a folder with the same name
  - It is added to the build process when main.cpp is compiled

- No spaces in filename

- No ~~leading numbers or~~ funny characters

- main.cpp calls
  setup( ) once, and
  loop( ) from a repeating for(;;) { … }

- It also checks for incoming serial data, to switch to the bootloader (already programmed in) to upload the next program when you click Upload

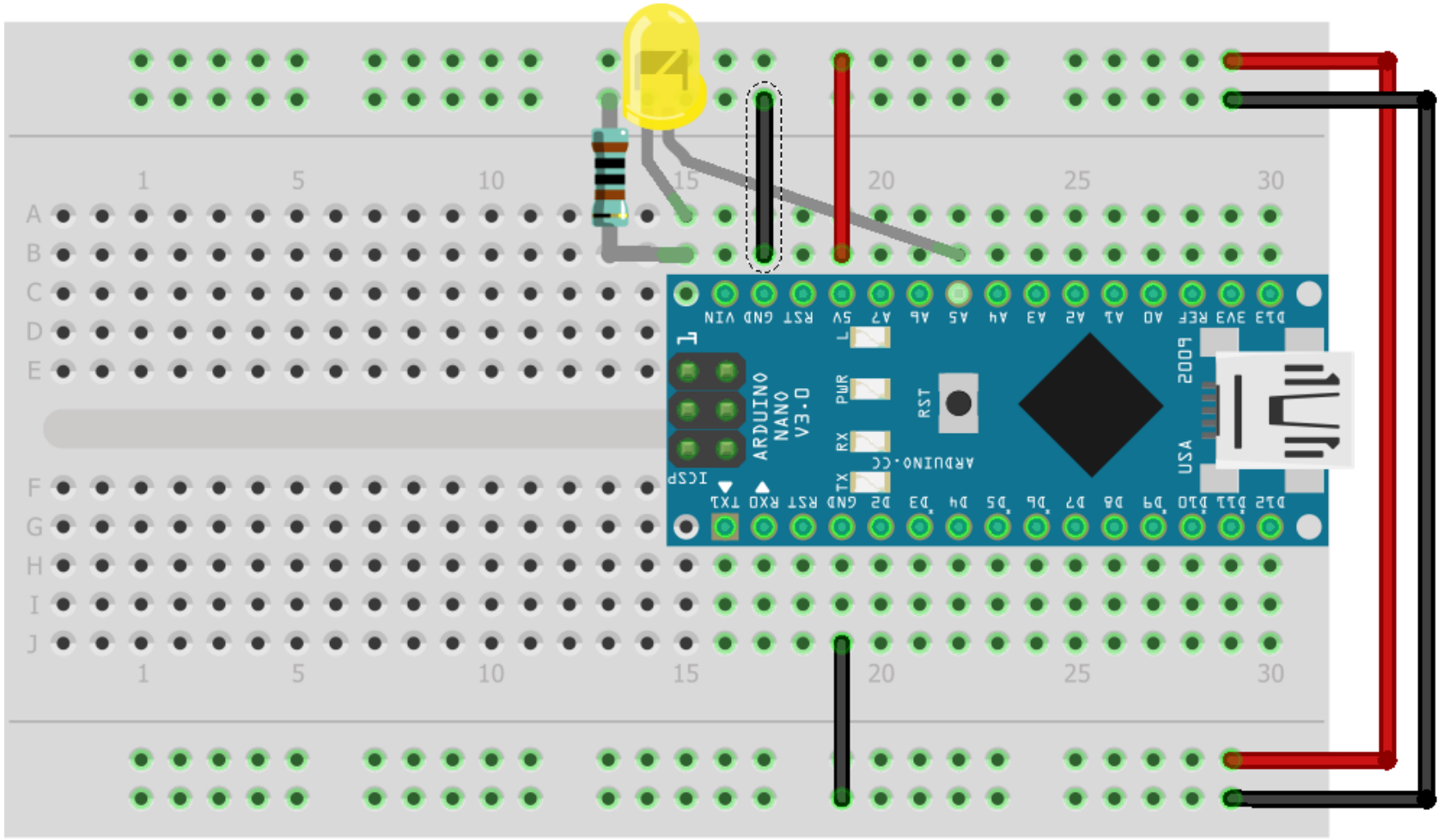
Next…

- Open **002_BlinkFaster.ino**

```
#define VERSION_STR    "002_BlinkFaster, St Louis, 2022.08.09"

/*
 * Blink by setting the ONTIME and the OFFTIME separately
 * \__________________/----  ,  \__________________/----  ,  \__
 * x                  x      ,  x                  x      ,  x
*/
…
#define LEDPIN           13
#define OFFTIME        1900
#define ONTIME          100
…
void loop( ) {
  digitalWrite( LEDPIN, LOW );
  Serial.println( "Off" );
  delay( OFFTIME );

  digitalWrite( LEDPIN, HIGH );
  Serial.println( "On" );
  delay( ONTIME );
}
```

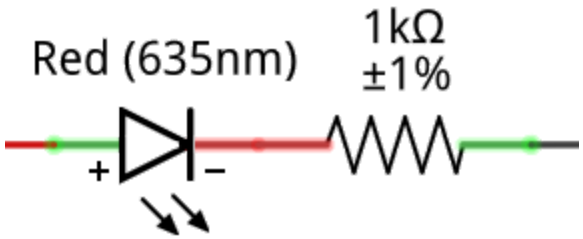
Build…

- Install the black and red wires!



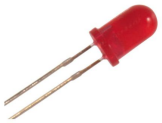
LEDs and **their** Resistors

An LED works like a one-way street. Just like cars accomplishing something by following everyone else in the same direction, an LED only allows current to flow in the direction of the triangle's arrow (from anode to cathode). It makes light once the forward voltage (Vf) across the diode is exceeded and electrical current starts flowing.



If you take a quick look at the datasheet for this Radio Shack 5 mm red LED (https://www.radioshack.com/products/5mm-red-led), you'll see that the Vf is somewhere between 2.1 and 2.8 Volts:

Forward voltage: 2.1 V (typical)/ 2.8 (max) @ 20 mA



So to make sure it will turn on, you need to do two things, provide more than 2.1 Volts **and put a resistor big enough in series** to avoid reaching the maximum rated current:

Forward Current (If): 25 mA (**Max**)

Assume we have a 5 V source to turn the LED on with and we believe the 2.0 Vf voltage, we need to use 5 - 2.1 V or 2.9 V across the current limiting resistor. From V = I x R, we can tell that if the current through the LED is the same as the current through the resistor, and we choose to run the LED with medium brightness at 3 mA:

R = V / I or R = 2.9 / 0.003 = **967 Ohms**. ← looks like 1k Ohm to me!

See datasheet or guess: I → 3 mA, V_{F_red} = ~1.9 V; $V_{F\_yel/ora}$ = ~2.0 V; V_{F_grn} = ~2.1 V; $V_{F\_blu/whi}$ = ~3.4 V;

- Run **003_TwoBeaconBlink.ino**

More than 1 beacon on the Arduino now!

Pay attention to the red ‘x’s shown here:

```
/*
 * Making two beacons work! Need external LED and resistor on pin A5
 */

// \_________________/----  ,  \_________________/----  ,  \____
// --\_________________/--  ,  --\_________________/--  ,  --\__
// x x               x x    ,  x x               x x    ,  x x

#define LEDPIN1          13
#define LEDPIN2          A5

#define TIME1           250
#define TIME2          1850
#define TIME3           250
#define TIME4            50
```

- Every delay( ) value needs to be calculated between every pin change

```
void loop( ) {
  digitalWrite( LEDPIN1, LOW );
  Serial.println( "Off 1" );
  delay( TIME1 );

  digitalWrite( LEDPIN2, LOW );
  Serial.println( "Off 2" );
  delay( TIME2 );

  digitalWrite( LEDPIN1, HIGH );
  Serial.println( "On 1" );
  delay( TIME3 );

  digitalWrite( LEDPIN2, HIGH );
  Serial.println( "On 2" );
  delay( TIME4 );
} // loop( )
```

Timing in software, 3 ways…

It works, kind of: **Blocking** -> delay( )

Just wait here, do nothing else.

Better: **Polling** -> check every so often, millis( )

Is it time yet?

Best: **Interrupts** -> Let ME interrupt YOU when it is ready. Egg timer, microwave, kettle whistling

But, the ATMEGA328P only has **3 timers** to generate timer interrupts!

So, **polling** with **millis( )** it is!

https://www.arxterra.com/10-atmega328p-interrupts/#Interrupt_Basics

Polling with millis( )

- A value that increments every millisecond from power up (and clears on a reset)
- Will overflow in about 49.7 days (but not a problem here)
- **Write down** the time (save it as “previous”), and **check every so often** if a certain amount of time has passed since “previous”
- If something needs to repeat, update the “previous” and keep checking if another period has expired.

- Run **004_Millis.ino**, GO!

```
#define VERSION_STR    "004_Millis, St Louis, 2022.08.09"

#define LEDPIN1         13
#define TIME1          250

// Read the pin and write the opposite
void toggle( uint8_t u8ThePin ) {
  digitalWrite( u8ThePin, !digitalRead( u8ThePin ) );
} // void toggle( uint8_t )

// Code in loop( ) will execute over and over, forever after setup( ) was called
// The "for (;;) { }" repeat forever loop in main.cpp calls loop over and over
void loop( ) {
  ulNow = millis( );

  if( ulNow - ulPrevious > TIME1 ) {
    // very important to update ulBefore for next time
    ulPrevious = ulNow;
    toggle( LEDPIN1 );
  } // if
} // loop( )
```

OOP: Object Oriented Programming

What is object-oriented programming?

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP

- Don’t fear, you were already using OOP with Serial.begin(115200) and Serial.println( )
- It allows us to have a Beacon keep track of its own variables, like pin number and the “previous”
- It allows us to call methods in Beacon that does not clutter all our code in the Sketch
- It gives us an easy way to create another Beacon with less duplicating code

Introducing: “Heartbeat”!

- If nothing works, how do you know your code is running?
- Let’s put a heartbeat in the system and then we know!
- I like it on pin 13, always! Why? Because the builtin LED is almost always there!
- **005_Blink_With_OOP.ino** it is!
- Sketch folders can also contain other files
- Notice the New Tab (Ctrl+Shift+N) down arrow at the top right
- It is common practice to #include .h file in C/C++, so we will do the same and stick our new OOP Class in there.
- Note, a **Class** tells what the code can do, but an **Object** is created (with a **Constructor)** to do the work.

Now for Beacon done by OOP!

- With millis( )

- Run **006_OOP_Beacon.ino**

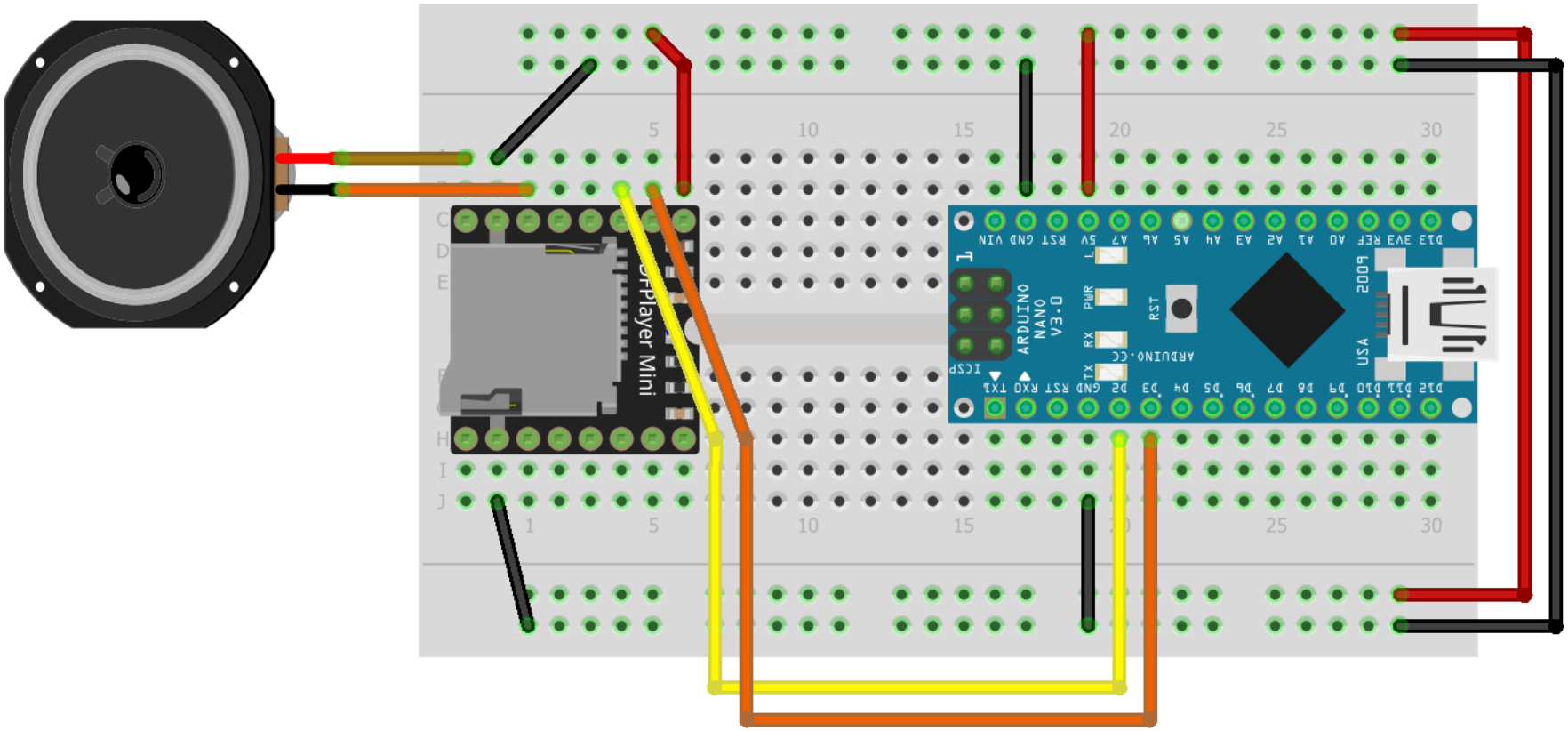
```
Heartbeat myHeart  = Heartbeat( LEDPIN1, TIMEON, TIMEOFF );
Heartbeat myBeacon = Heartbeat( BEACONPIN1, BCNTIMEON, BCNTIMEOFF );

void setup( ) {
  …
  myBeacon.begin( );
  myHeart.begin( );
} // setup( )

void loop( ) {
  myBeacon.update( );
  myHeart.update( );
} // loop( )
```

- How would you do a second beacon?

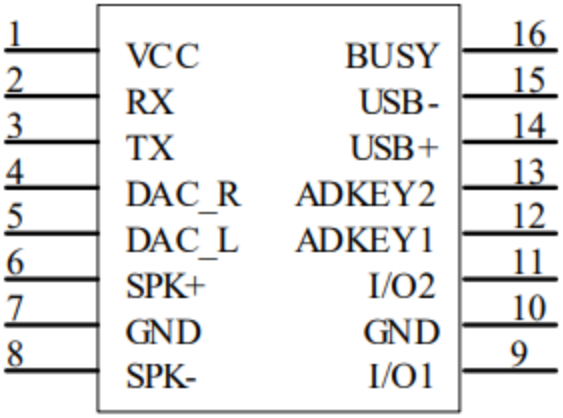
  Well, add another resistor and LED and then add myBeacon2 with a pin and on and off times. Use .begin( ) and .update( )!

Adding Sound…

- Install the black and red wires
- Install the Orange and Yellow wires
- Install the Speaker

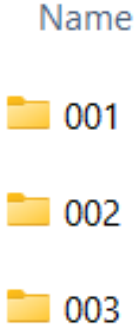


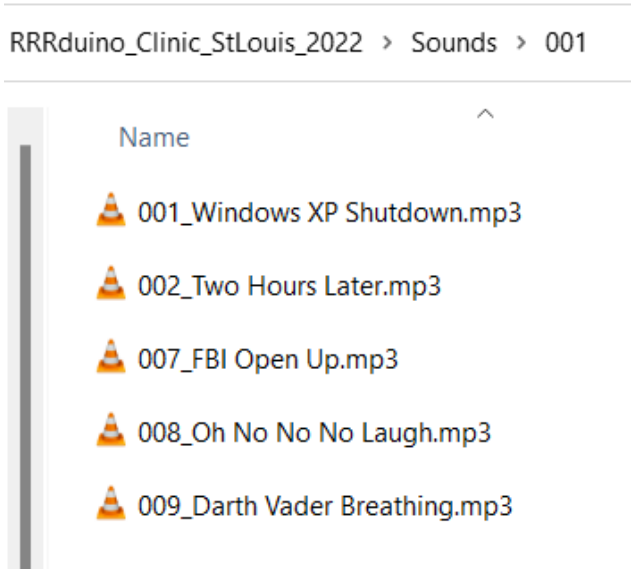
Pin Configuration and Summary



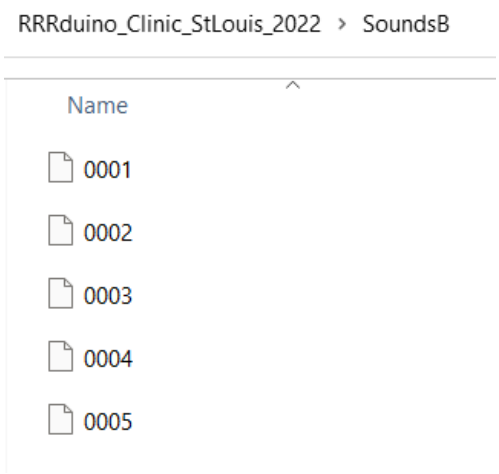
No	Pin	Description	Note
1	VCC	DC3.2~5.0V	
2	RX	UART serial input	3.3V TTL level
3	TX	UART serial output	3.3V TTL level
4	DAC_R	Audio output right channel	Drive an earphone or connect to an external amplifier
5	DAC_L	Audio output left channel	
6	SPK2	Speaker-	Drive speaker less than 3W
7	GND	Ground	Power GND
8	SPK1	Speaker+	Drive speaker less than 3W
9	IO1	Trigger port 1	Short press to play previous (long press to decrease volume)
10	GND	Ground	Power GND

uSD Card Contents…



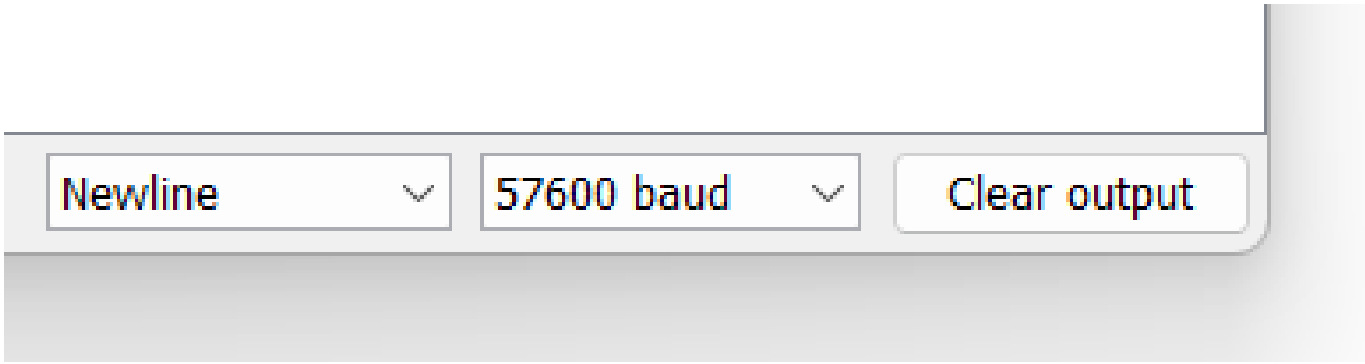


or



- Only thing that matters seems to be the order they were loaded onto the uSD card !!!
- Format + copy one by one

Switching Baud Rate to 57600 !!!



Play the first Sound…

- Run **007_Sound.ino**

```
#include "SoftwareSerial.h"
#include "DFRobotDFPlayerMini.h"

#define DFPLAYER_BAUDRATE       9600
#define SWRXPIN                    2
#define SWTXPIN                    3

DFRobotDFPlayerMini myDFPlayer;

mySoftwareSerial.begin( DFPLAYER_BAUDRATE );      // DFPlayer default baud rate

while( notReady ) {
  if( !myDFPlayer.begin( mySoftwareSerial ) ) {
    Serial.println( "DFPlayer begin( ) failed:" );
    Serial.println( "Check the wiring and/or insert a uSD card" );
    notReady = true;
    delay( 1000 );                                 // wait 1 second to try again
  } else {
    notReady = false;
  } // if ready
} // while
Serial.println( "DFPlayer ready!" );

setMP3Volume( volume );                           // set volume value, 0 to 30
printPlayerInfo( );
myDFPlayer.start( )
```

- Because myDFPlayer.setVolume(volume) does not work
- But, volumeUp and volumeDown do!

```
void setMP3Volume( uint8_t u8TheVolume ) {
  while( u8CurrentVolume != u8TheVolume ) {
    if( u8CurrentVolume > u8TheVolume ) {
      myDFPlayer.volumeDown( );                    // volume Down
    } else {
      if( u8CurrentVolume < u8TheVolume ) {
        myDFPlayer.volumeUp( );                    // volume Up
      } // if less
    } // if else
    delay( 1 );
    u8CurrentVolume = myDFPlayer.readVolume( );
  } // while
} // void setMP3Volume( uint8_t )
```

- Player and uSD card information

```
void printPlayerInfo( ) {
  Serial.print( F( "Files  : " ) ); Serial.println( myDFPlayer.readFileCounts( ) );
  Serial.print( F( "Folders: " ) ); Serial.println( myDFPlayer.readFolderCounts( )
);
  printCurrentFileNum( true );
  Serial.print( F( "Volume : " ) ); Serial.println( myDFPlayer.readVolume( ) );
  Serial.println( );
} // printPlayerInfo( )
```

- Echo what comes in from the Serial port

```
void loop( ) {
  // Send commands through serial port from PC (the USB cable provides a COMx,
ttyUSBx or ttyACMx interface)
  if( Serial.available( ) ) {
    Serial.write( Serial.read( ) );    // print anything we receive
  } // if serial data available

  myHeart.update( );
  myBeacon.update( );
} // loop( )
```

Sound with a Menu

```
Menu:
+     volume up
-     volume down
i     info
c     current
n     next
N     play next
p     previous
P     play previous
s     stop
S     start
R     reset
?     menu
```

```
void playTrackNum( uint8_t u8Track ) {
  myDFPlayer.stop( );
  if( u8Track > 0 ) {
    myDFPlayer.start( );
    if( u8Track > 1 ) {
      for( uint8_t u8i=1; u8i < u8Track; ++u8i ) {
        myDFPlayer.stop( );
        myDFPlayer.next( );
      } // for
    } // if > 1
  } // if > 0
} // void playTrackNum( uint8_t )
```

```
void loop( ) {
  // User sends commands through serial port from PC (the USB cable
  // provides the COMx, ttyUSBx or ttyACMx interface)
  if( Serial.available( ) ) {
  // processing of ALL incoming commands are listed in the include file:
#include "serialinterface.h"
  } // if serial available

  myHeart.update( );
} // loop( )
```

- Run **008_Sound_with_Menu.ino**

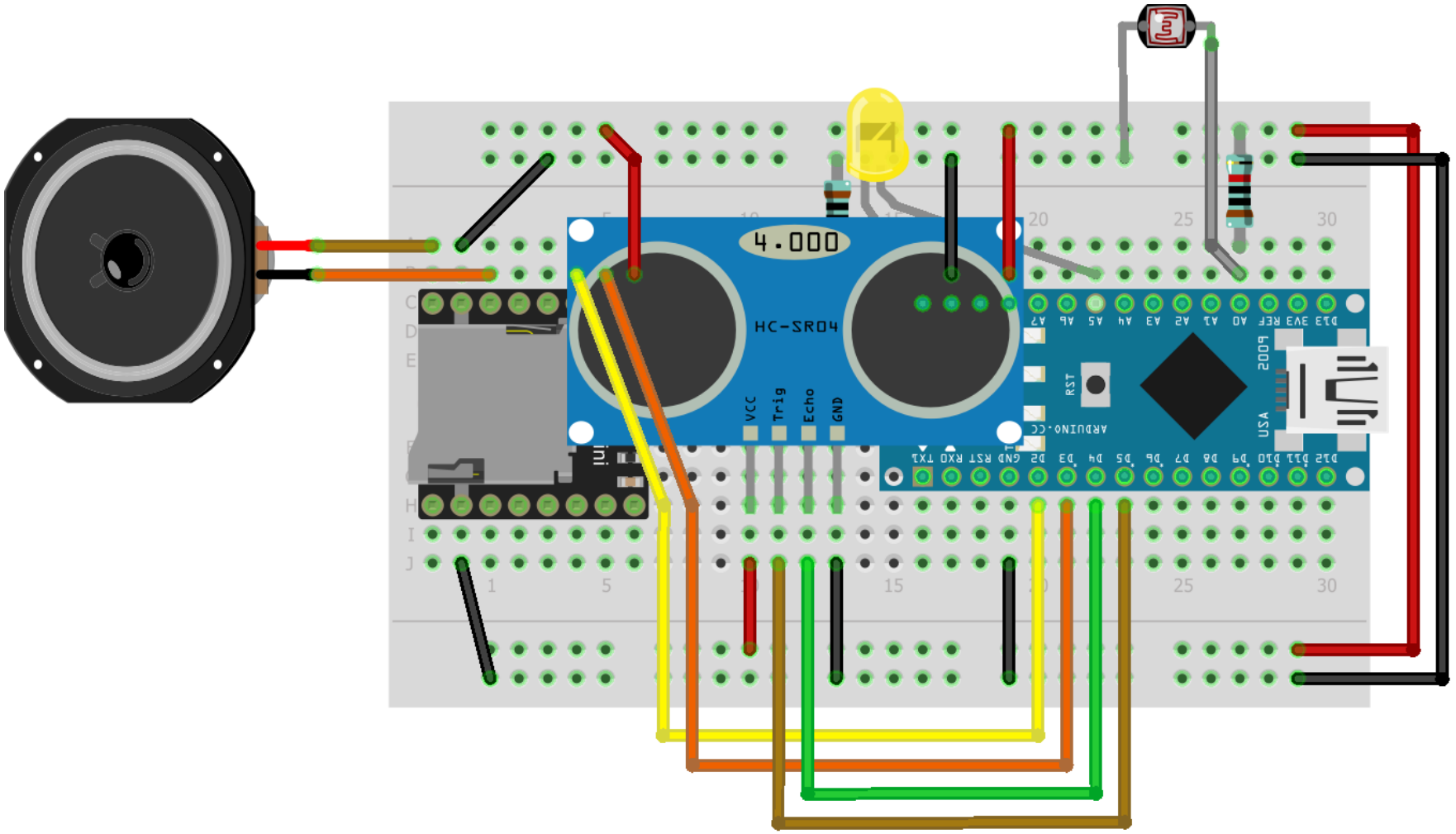
Ultrasonic Sensor

- "Trig" transmits 40 kHz ultrasonic sound pulses from left cylinder
- The other (right) is the receiver and listens for pulses
- When the receiver receives pulses, it creates an output pulse with a width proportional to the distance of the object
- Non-contact range detection: 2 cm to 400 cm (~13 feet)
- Distance = Speed x Time

https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/

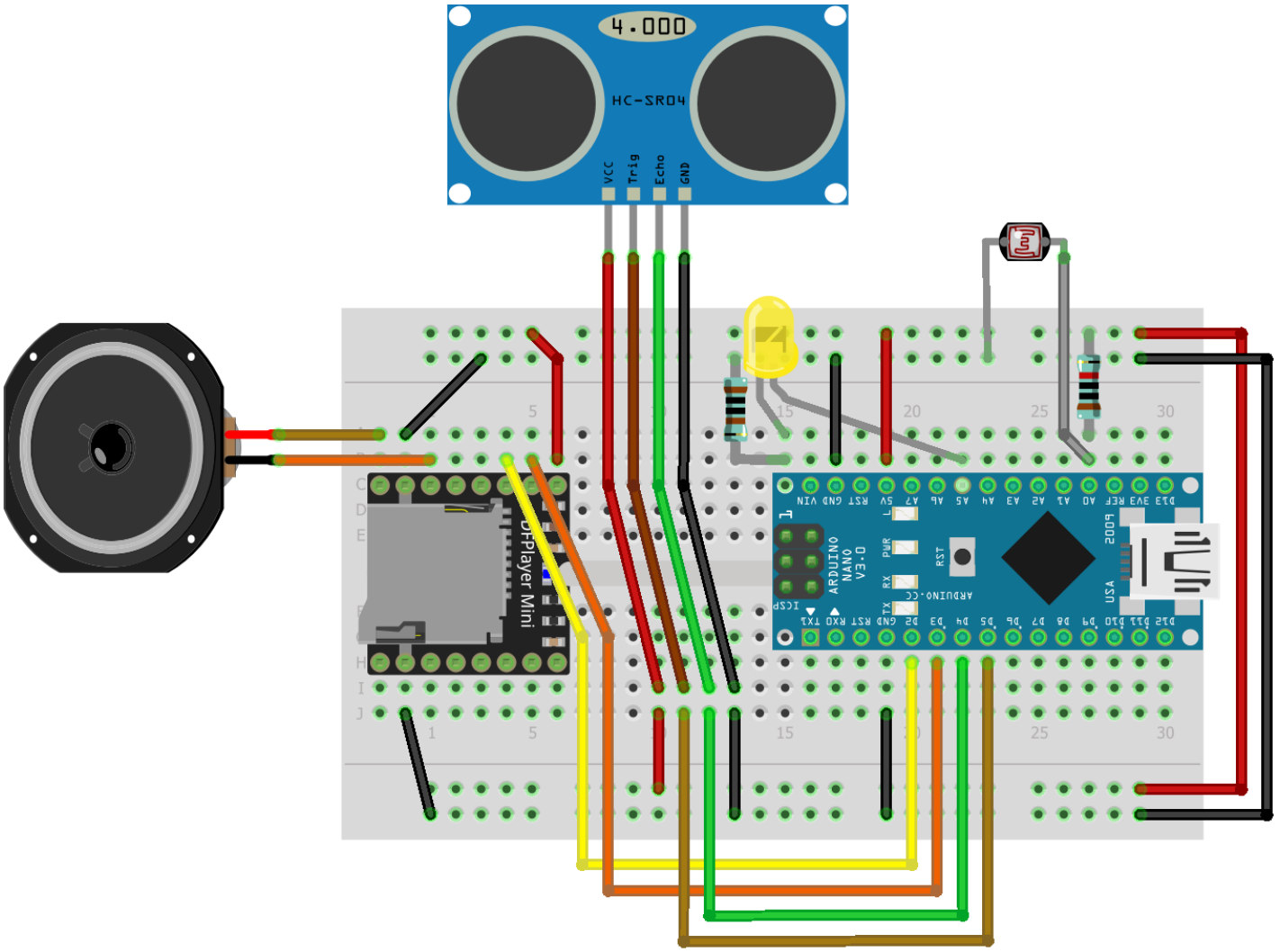
Ultrasonic Sensor

- Install HC-SR04 in the bottom half



Ultrasonic Sensor

Connect the bottom wires like the bottom shows:



Measuring the Time…

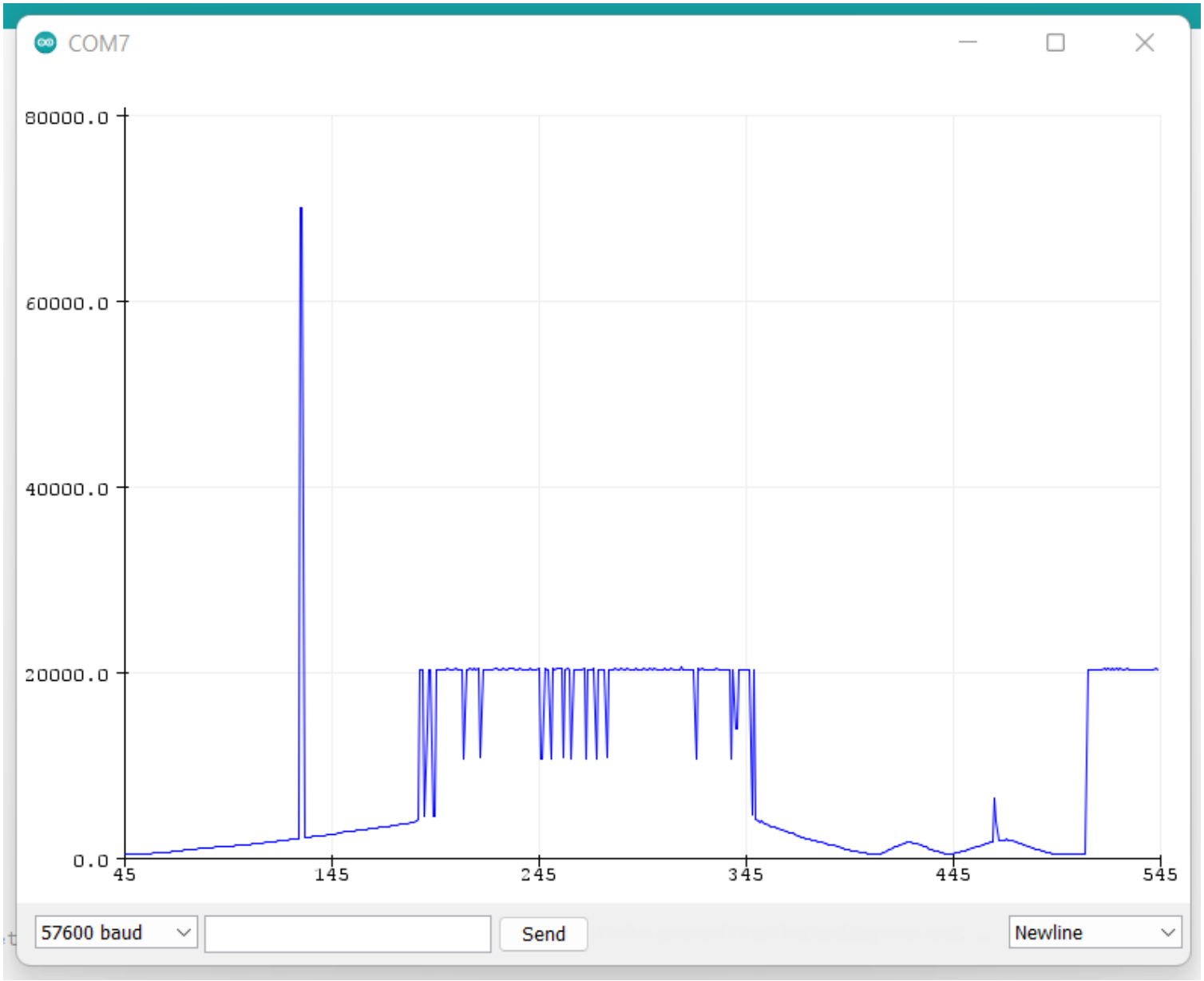
- Run **009_Ultrasonic_Sensor.ino**
- **Close** Serial Monitor
- Press Ctrl+Shift+L or open **Tools → Serial Plotter**

Blocking call, max 38 ms:

```
u32Duration = pulseIn( u8EchoPin, HIGH );
```

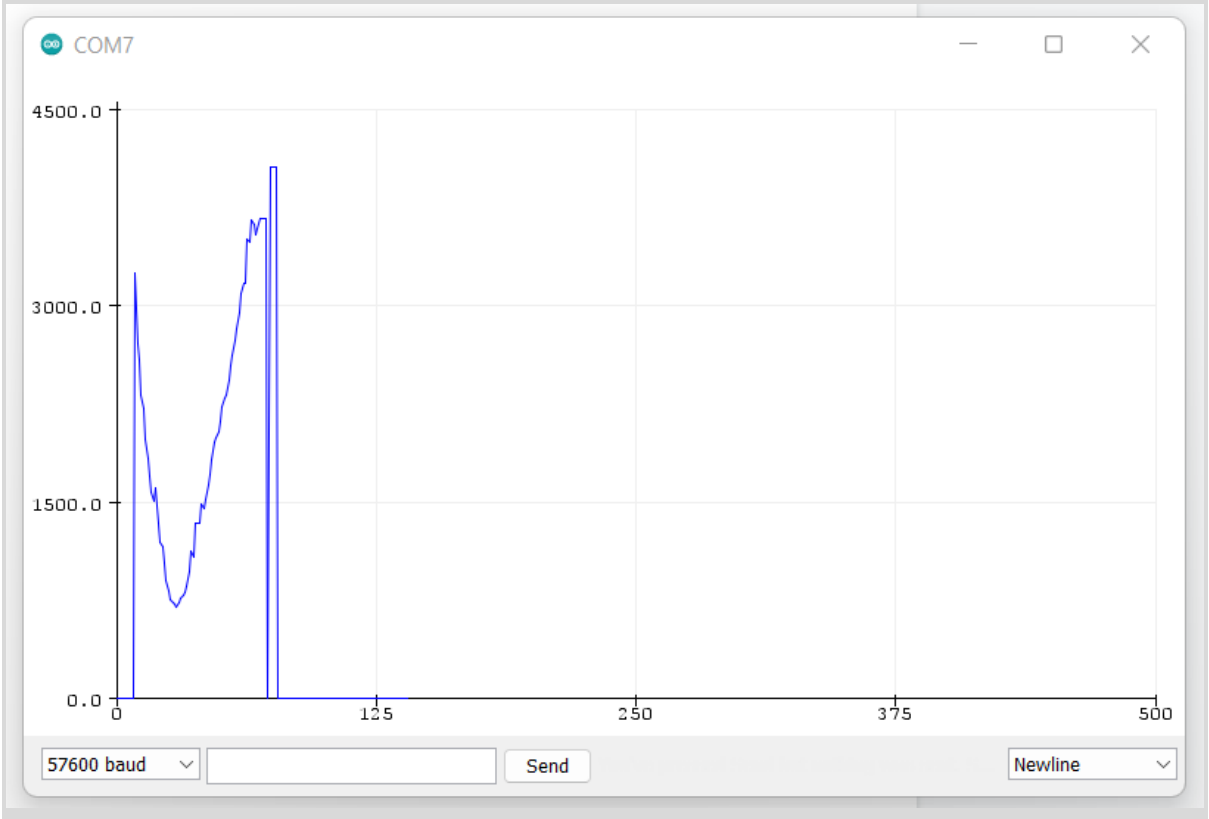
```
Pulse,
670         // 4 inches
620
611
614
607
632
10614       // open
10642
10641
10615
```

Serial Plot


COM7
80000.0
60000.0
40000.0
20000.0
0.0
45
145
245
345
445
545
57600 baud
Send
Newline

Remove Noise, Filter and Decide…

- Run **010_Ultrasonic_Sensor_Filtered.ino**



- Pick a useful number for max range (BIGVALUE), use **zero** for nothing
- No more overshoots and reset to zero if number goes stale
- Circular buffer to look at past

All together now…

- Run **011_All_Togegther.ino** // at least 5 files need, else change the code
- Based on the distance to the object, we play a different track
- Now you have enough tools to make sounds based on any sensor input you like!

```
if( myDFPlayer.readState() != 513 ) { // != DFPlayerBusy ) {
  uint32_t u32TriggerValue = myDistanceSensor.getValue( );

  if( u32TriggerValue > 0 ) {
    if( u32TriggerValue > BIGVALUE ) {
      // do nothing
    } else {
      if( u32TriggerValue > 5000 ) {
        playTrackNum( 2 );
      } else {
        if( u32TriggerValue > 2000 ) {
          playTrackNum( 3 );
        } else {
          if( u32TriggerValue > 1000 ) {
            playTrackNum( 4 );
          } else {
            if( u32TriggerValue > 500 ) {
              playTrackNum( 5 );
            } // if
```

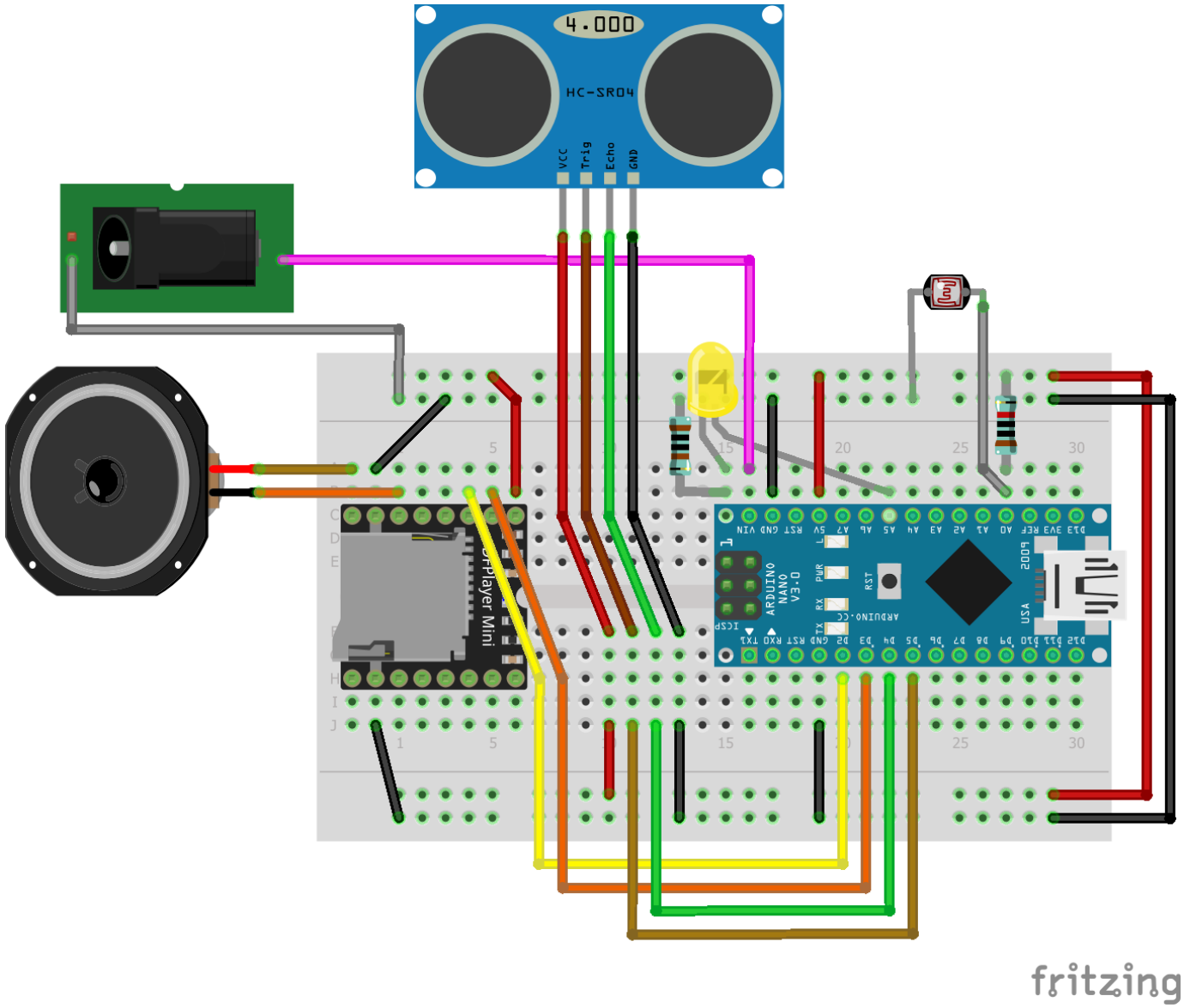
Light Sensor

- 10 bit Analog to Digital converter
- 0 - 1023
- Use 10 kOhm resistor with light sensor and measure in between

```
uint16_t u16AnalogValue = analogRead( A0 );

uint16_t u16MappedValue = map( u16AnalogValue, fromLow, fromHigh, toLow, toHigh );

if( u16MappedValue > SOMETHING ) { ; } else { ; }
```

Power

- Vin: 6 to 12 Vdc, with enough current to drive speaker or servo(s)



A Few Other Important Notes:

- Copying code from anywhere else, watch out for the quotes: “ and ” is not the same as "

- Keep the description and version in your VERSION_STR up to date, you need to be able to find the source, since the HEX file from the Arduino will not show you the C/C++ code again

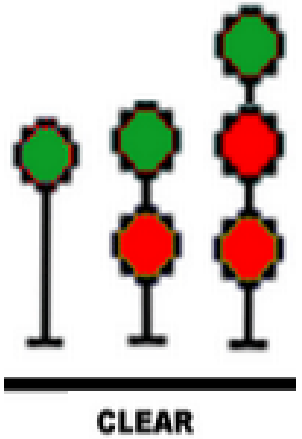
- Use dates and English words as much as you can. Easier to find and search later

- More interesting reading
https://roboticsbackend.com/the-arduino-language-in-10-points/

Any questions?

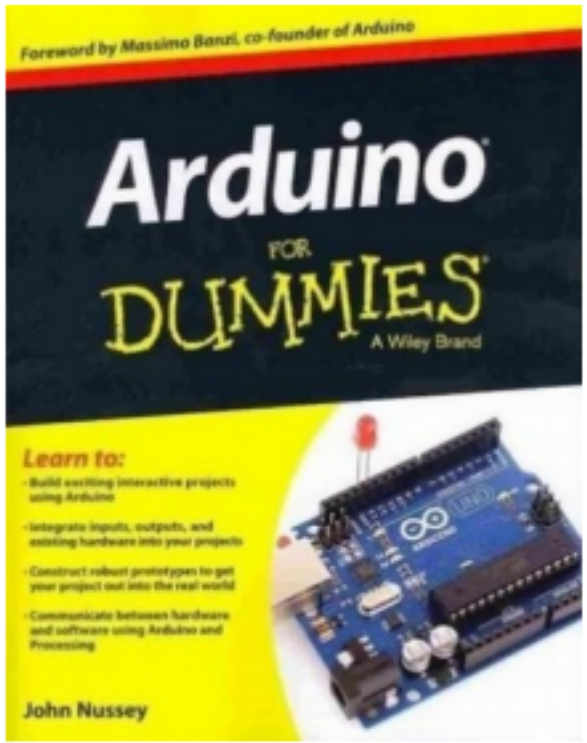
The only **dumb question** is the one **you did not ask**!

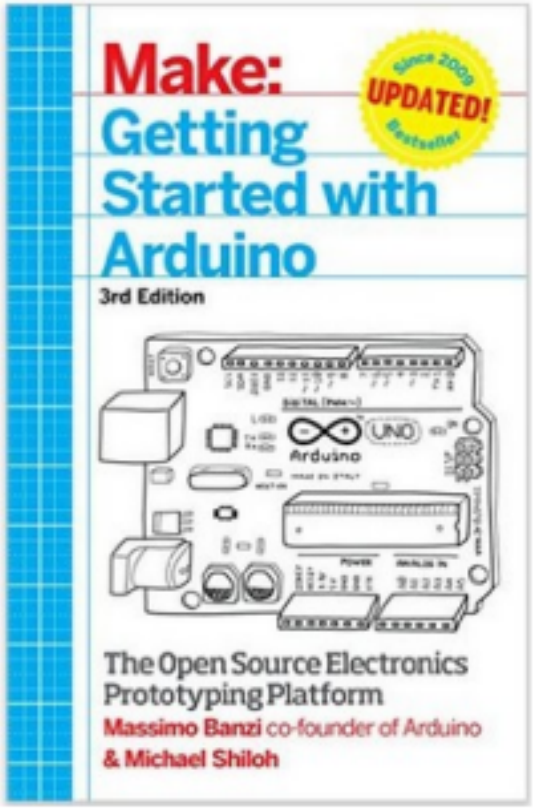
The End

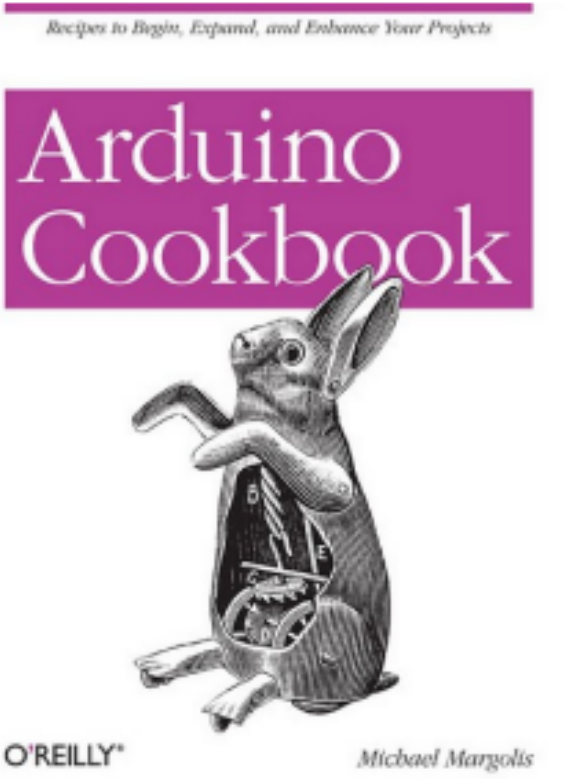


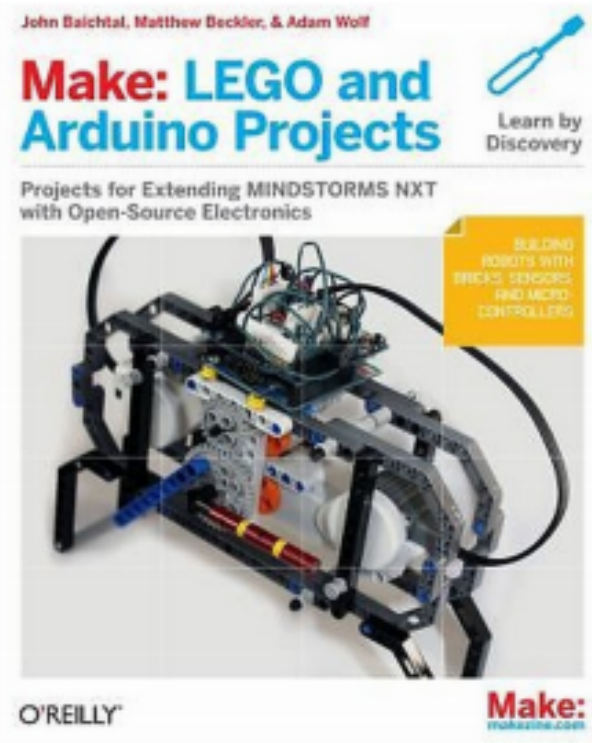
Thank you to Joel, David, Riley, *Bob, Donna, John, Stephanie, Alissa, Bianka, my Boss,*
and all the others that did not bother us while we were having fun doing this…

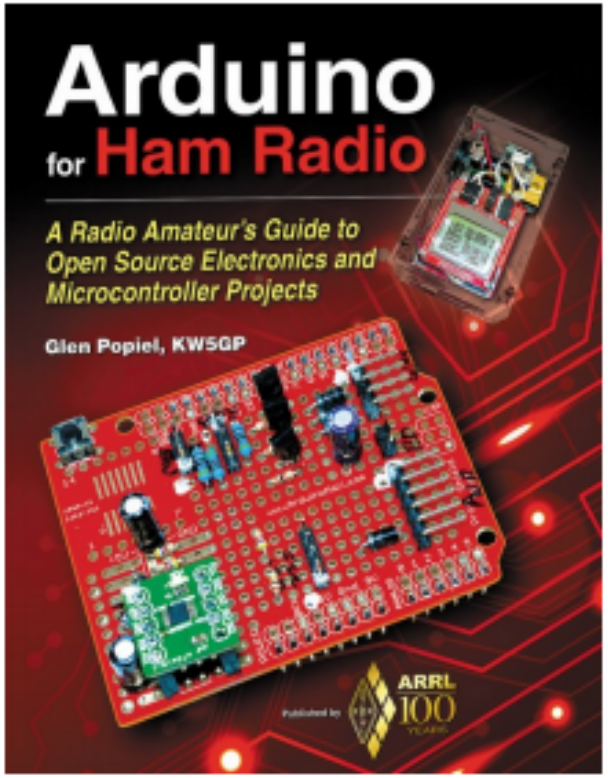
Get educated:






Recipes to Begin, Expand, and Enhance Your Projects
Arduino Cookbook
O'REILLY
Michael Margolis


John Baichtal, Matthew Beckler, & Adam Wolf
Make: LEGO and Arduino Projects
Learn by Discovery
Projects for Extending MINDSTORMS NXT with Open-Source Electronics
O'REILLY
Make:


Arduino for Ham Radio
A Radio Amateur's Guide to Open Source Electronics and Microcontroller Projects
Glen Popiel, KW5GP
ARRL
100

Changes:

v0.01, original

v0.02, numbers allowed in filename, OOP used with 115200 baud