...RRduino...

in Model Railroading

by Speed

(aka Gert 'GM' 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 then go ahead, take that nap!

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 kind of **lights**, **motors**, **sensors** and **other devices**; 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, that 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 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

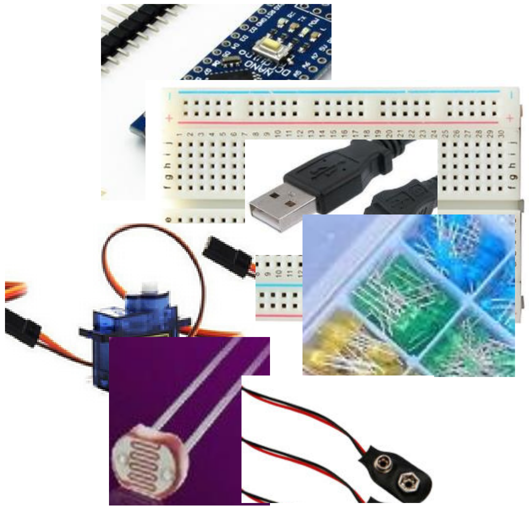
Pretty obviously, the open source model had always been used to fuel innovation for software and never **hardware**. If they had to make it work, they had to find a suitable licensing solution that could apply to the board. After a little investigation, Banzi and team looked at the whole thing from a 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!

Division 3 opted for a Make and Take Clinic in 2015!

We bring (or what your $$$ gets you):

- ☐ 1 x **Arduino Nano** (with a USB connector, read: "no separate programmer")
- ☐ 1 x USB to **Mini USB Cable**
- ☐ 1 x **9V** battery **cable**
- ☐ 1 x **9g Micro Servo** Motor (Grade Crossing, Semaphore?)
- ☐ 2 x 3mm **RED** LEDs (Crossing Gates?)
- ☐ 2 x 3mm **YELLOW** LEDs
- ☐ 2 x 3mm **GREEN** LEDs (Signals?)
- ☐ 2 x 3mm BLUE LEDs (Ambulance? Firetruck?)
- ☐ 2 x 3mm **WHITE** LEDs - yes, that was white, :)
- ☐ 10 x 1k 1% resistors (Why? Another clinic!)
- ☐ 1 x Push Button
- ☐ 1 x Light sensitive Photo Resistors
- ☐ 1 x 400 contact breadboard
- ☐ and some wires to connect some or all of these together!



You bring:

- ☐ a Laptop, or a friend with his/her laptop.
- ☐ a 9V battery (and maybe a solder iron too)
- ☐ an IDEA! Yes, you tell ahead of time what you want your Arduino to do going home, and we get the code 99% there.

– Usually a 2 PART CLINIC –

☐ 1) **What+Solder+Laptop:** The history, what it is, and what you can do with it.
Then we solder (a half clinic on soldering too) the pins and wires on
And install the software on your laptop or computer.
Your Arduino will have a blinking LED at the end of this

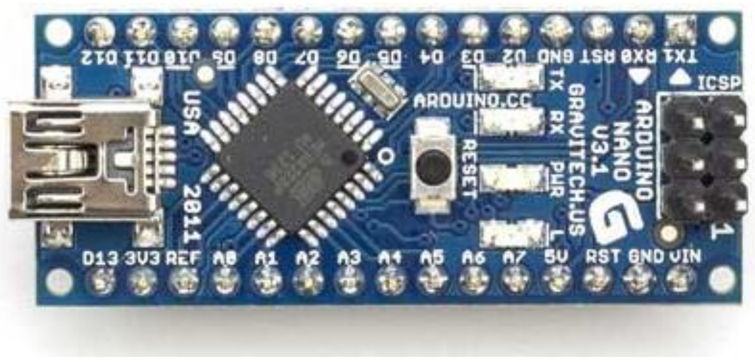
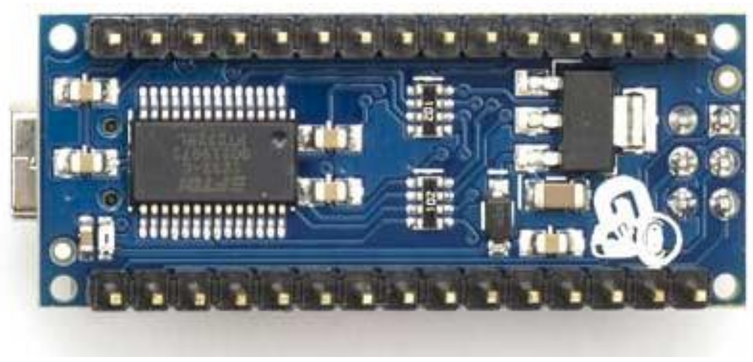
☐ 2) **Coding 101:** We get into the software and get your IDEA on your board!

*(yes, we are going to use the term "click" a few times and Larry will follow along on **my** laptop!)*

Where to Start...

Buy one ... and plug it in!

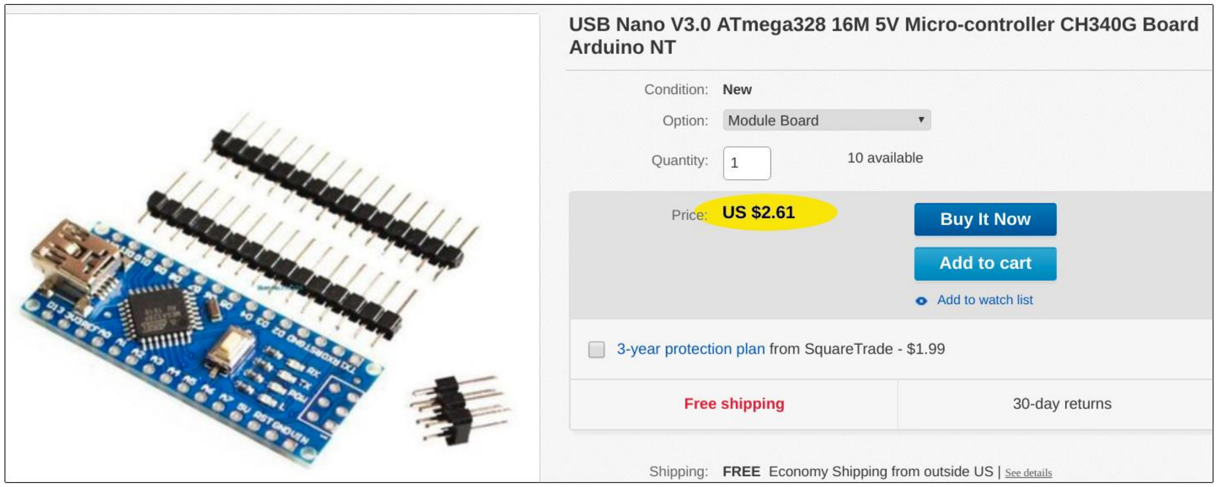
[PC → USB Cable → Nano]

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

Where did we get it?

- Search Ebay for “**Arduino Nano USB**” … I buy the Hong Kong ones, feels like they ship faster.
- Version 3.0 is new, 5V is good too.
- The CH340G USB/Serial chip requires 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)


USB Nano V3.0 ATmega328 16M 5V Micro-controller CH340G Board Arduino NT

Condition: New

Option: Module Board

Quantity: 1 10 available

Price: US $2.61

3-year protection plan from SquareTrade - $1.99

Free shipping 30-day returns

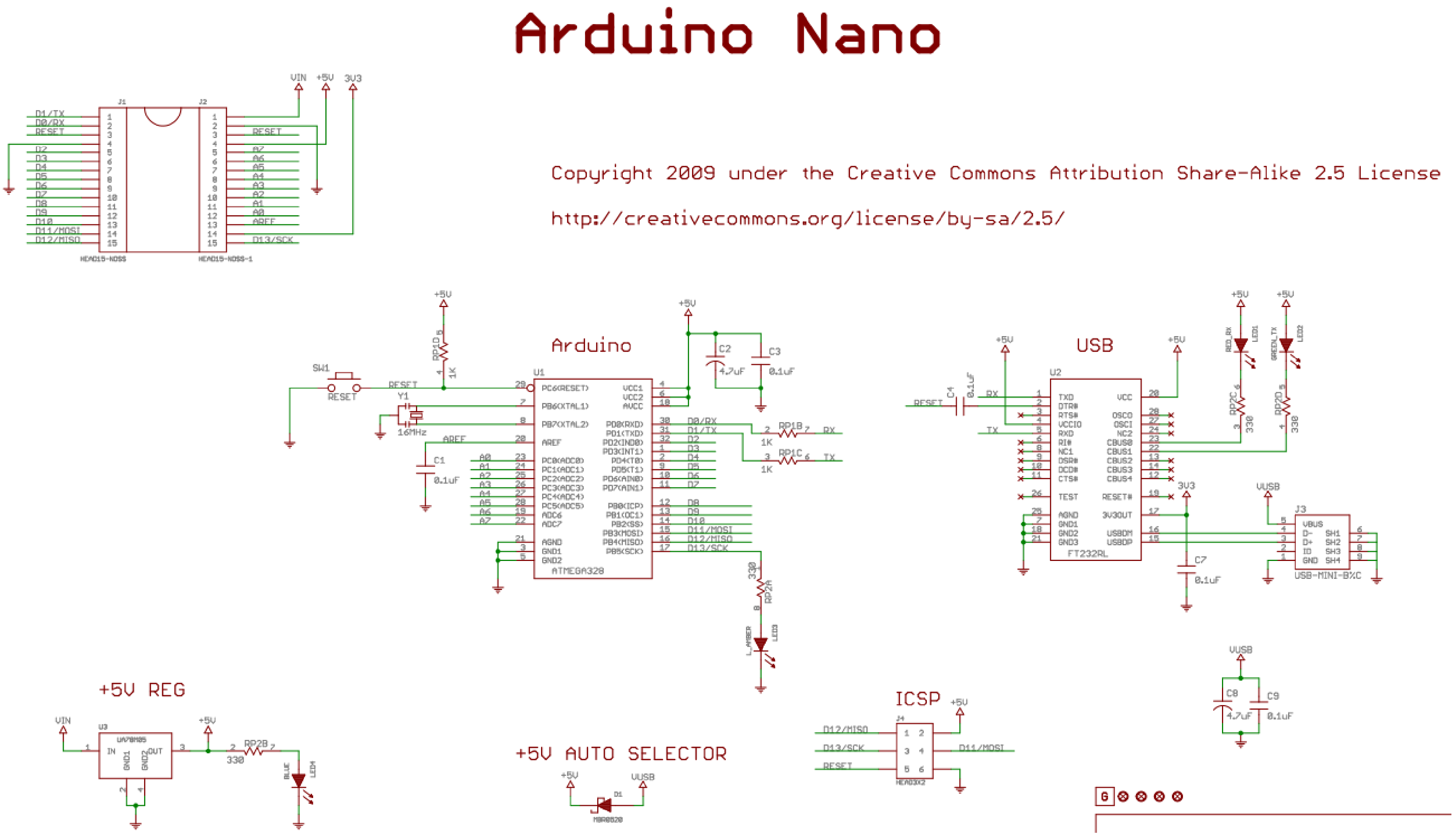
Shipping: FREE Economy Shipping from outside US | See details

- Of course, Adafruit, Sparkfun and Amazon would help you too.
- Even Micro Center has them in stock.

Here is what we just plugged in…

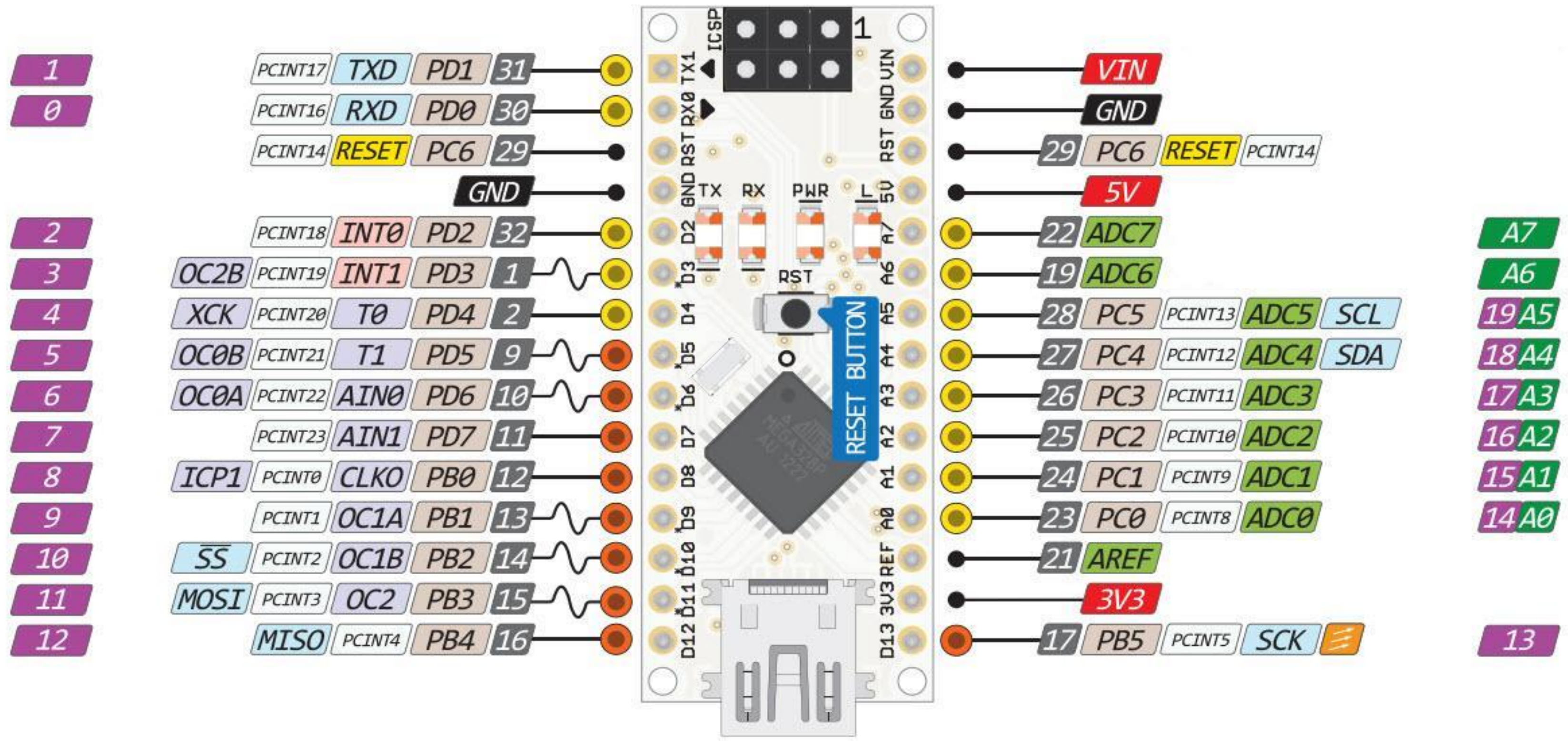
Schematic

(Remember, Open Source, Open Hardware)



And almost always shown like this instead…

“Arduino Nano”

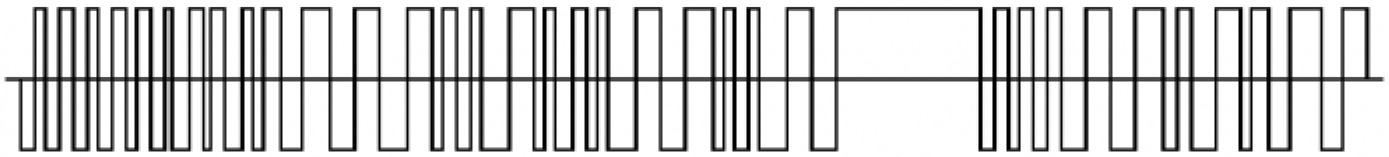


What you care about, is the purple and green numbers on the far left and right, since those are the numbers our software needs.
Green indicated pins needed for Analog **inputs**, but all the pins can do digital things. Except ADC6 and ADC7.
The other thing to note is the “tilde” lines on D3, D5, D6, D9, D10 and D11 that has a wiggle, those can be **PWM** outputs...in simple terms, they could look like an Analog **output** with a resistor and capacitor as filter.

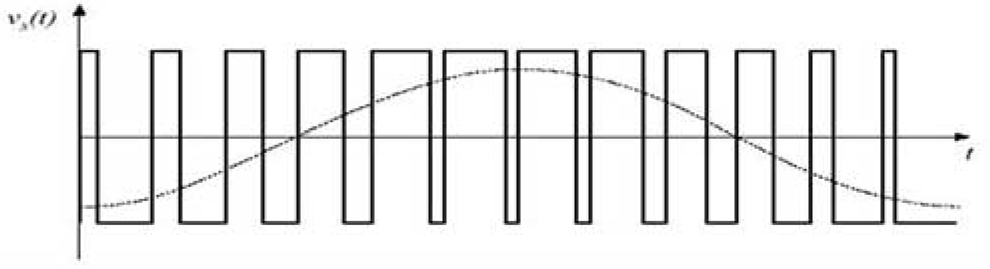
Now, do you understand the differences in the following?

- Digital vs Analog? On-Off (0 Vdc - 5 Vdc) vs 3.1415926535 Vdc
- Input vs Output? Listening vs Talking? Digital inputs decides if it is above or below mid-voltage. Analog?
- PWM vs Analog? Frequency with Duty Cycle (still On-Off) versus Sine wave
- PWM vs DCC? Fixed frequency vs Stretched digital pulses

DCC:

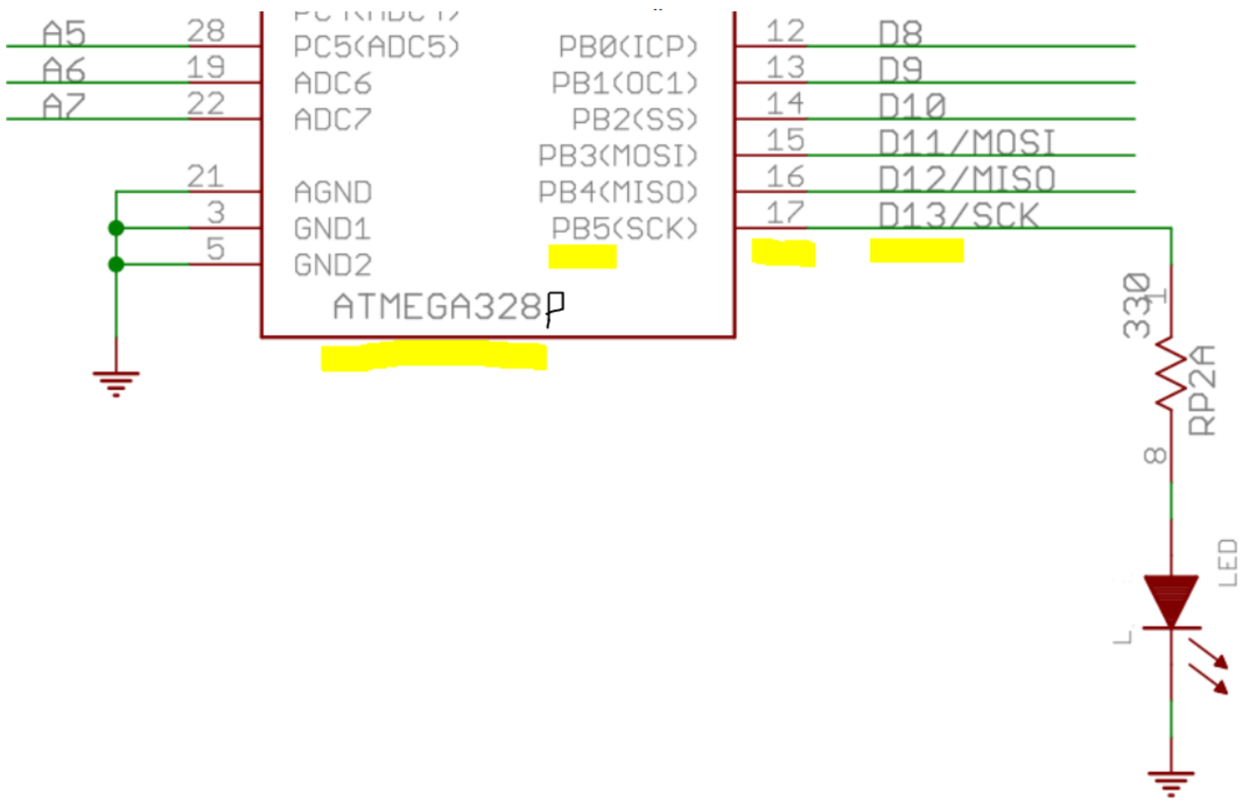


vs PWM:



(Note: DCC Bandwidth-> 180, 3 byte packets per second. One way, Can not read on the main line, only write.)

Pins Ports and Numbers



- The Atmel ATMEGA328**P**-Px 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.
- For Schematic and PCB designers, do not buy the cheaper ATMEGA328-PU chip, it takes plenty of skill to configure the avrdude.conf files and restarting the software in order to program the $0.50 less expensive silicon chip. The **P** after 328**P** is for the low power model, and that is the favorite.

OK, hold 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.

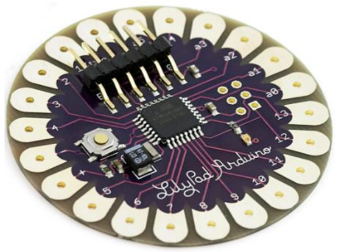
- It has a **following**, even today’s kids know about it

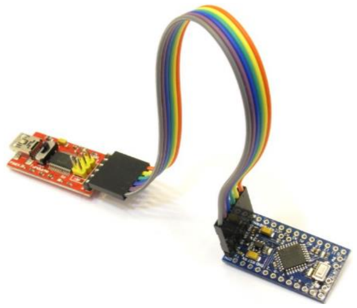
- It is **world wide**! It has websites, blogs, examples, even Facebook and Twitter, don’t you?

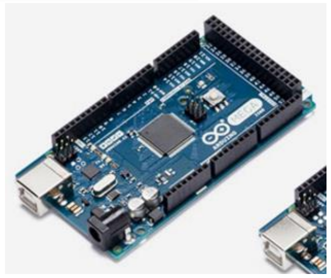
...It is a whole **ecosystem** with tools...

Ok, you said Hardware?

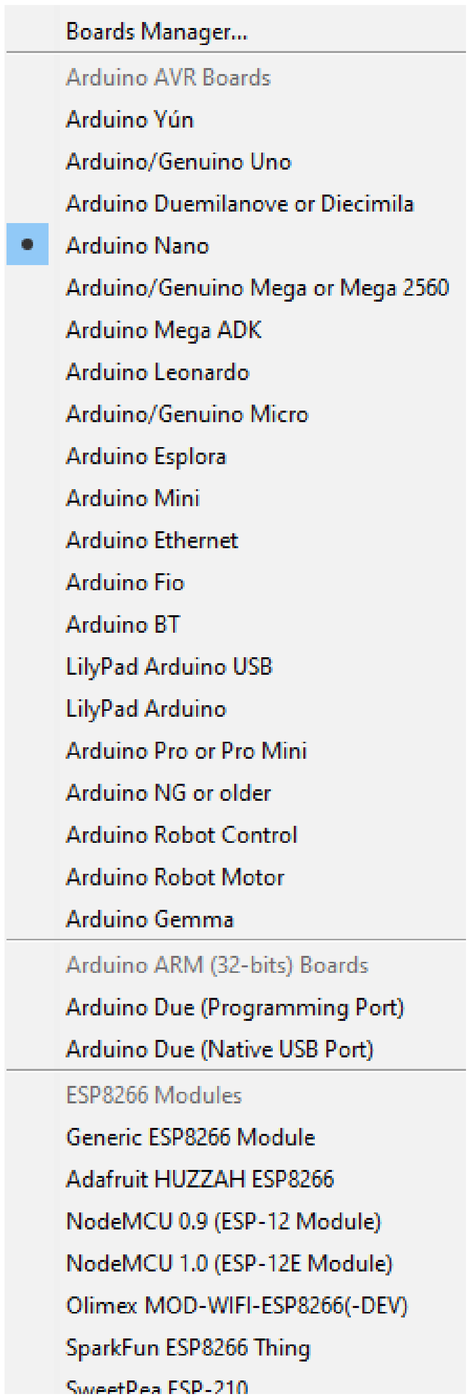
UNO came first: Atmel ATMEGA328P in a PU package, with an “NFL” size USB plug:







Then an surface mount UNO version was made.
wait, my Nano has that chip too!!!
...and a mini USB connector
...which the Micro does not have.


Boards Manager...

Arduino AVR Boards

Arduino Yún

Arduino/Genuino Uno

Arduino Duemilanove or Diecimila

● Arduino Nano

Arduino/Genuino Mega or Mega 2560

Arduino Mega ADK

Arduino Leonardo

Arduino/Genuino Micro

Arduino Esplora

Arduino Mini

Arduino Ethernet

Arduino Fio

Arduino BT

LilyPad Arduino USB

LilyPad Arduino

Arduino Pro or Pro Mini

Arduino NG or older

Arduino Robot Control

Arduino Robot Motor

Arduino Gemma

Arduino ARM (32-bits) Boards

Arduino Due (Programming Port)

Arduino Due (Native USB Port)

ESP8266 Modules

Generic ESP8266 Module

Adafruit HUZZAH ESP8266

NodeMCU 0.9 (ESP-12 Module)

NodeMCU 1.0 (ESP-12E Module)

Olimex MOD-WIFI-ESP8266(-DEV)

SparkFun ESP8266 Thing

SweetPea ESP-210

Uno: ATmega328p First, USB, DIP or SMD

Yun: ATmega32u4 and the Atheros AR9331. Linux distribution OpenWrt-Yun. Built-in Ethernet and WiFi.

Duemila...2009: ATmega168 First, USB B.

Nano: ATmega328p Small, USB mini, don’t pick favorites.

Mega 2560: ATmega2560. USB, It has 54 I/Os

Leonardo: ATmega328p First, USB B.

Micro: ATmega32U4, USB built in, little harder to work with, since timing is important

Esplora: Nintendo?

Mini: Tiny, was ATmega168 now ATmega328p , need programmer

Ethernet: Your guess is as good as mine!

FIO: 3.3 V ATmega328p

BT: Bluetooth?

Pro or Pro Mini: SparkFun, ATmega328p need programmer. 3.3V and 5V versions

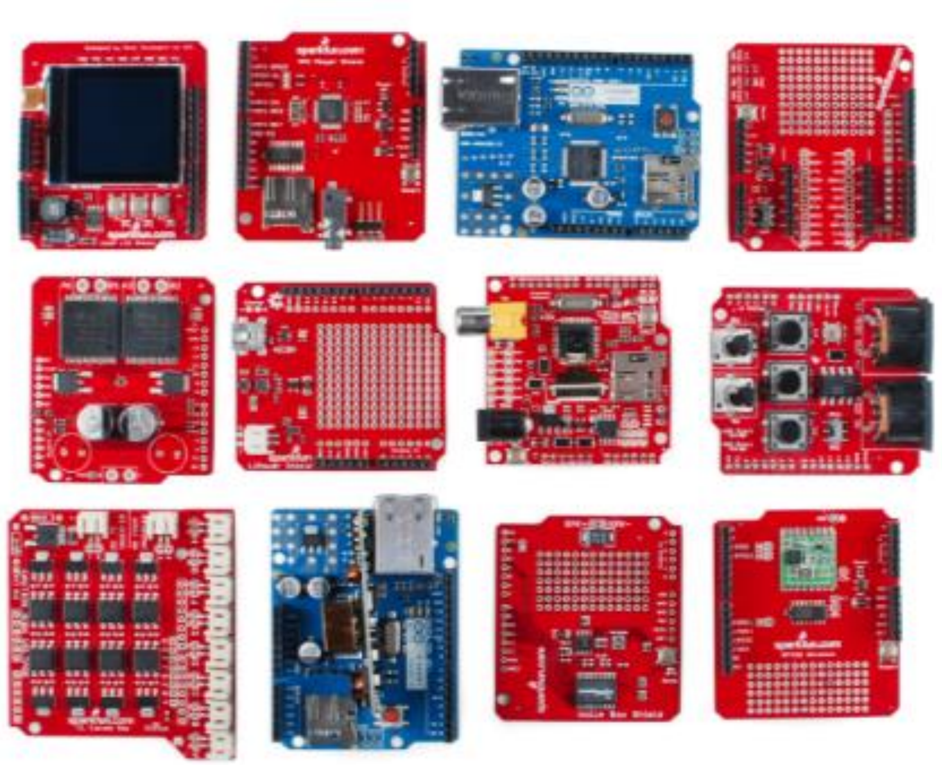
Zero: SAMD21 MCU, 32-bit ARM Cortex® M0+ core.

Due: Atmel SAM3X8E ARM Cortex-M3 CPU. First 32-bit ARM core microcontroller. 54 digital I/Os

Arduino vs Genuino: Lawyers...companies sued, read the story!

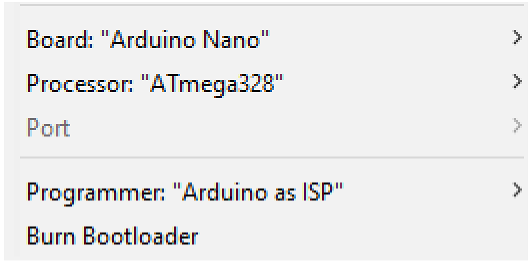
A short word on Shields

- That is how Arduinos connect to the world, if it 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 and Eagle is free



Programmers

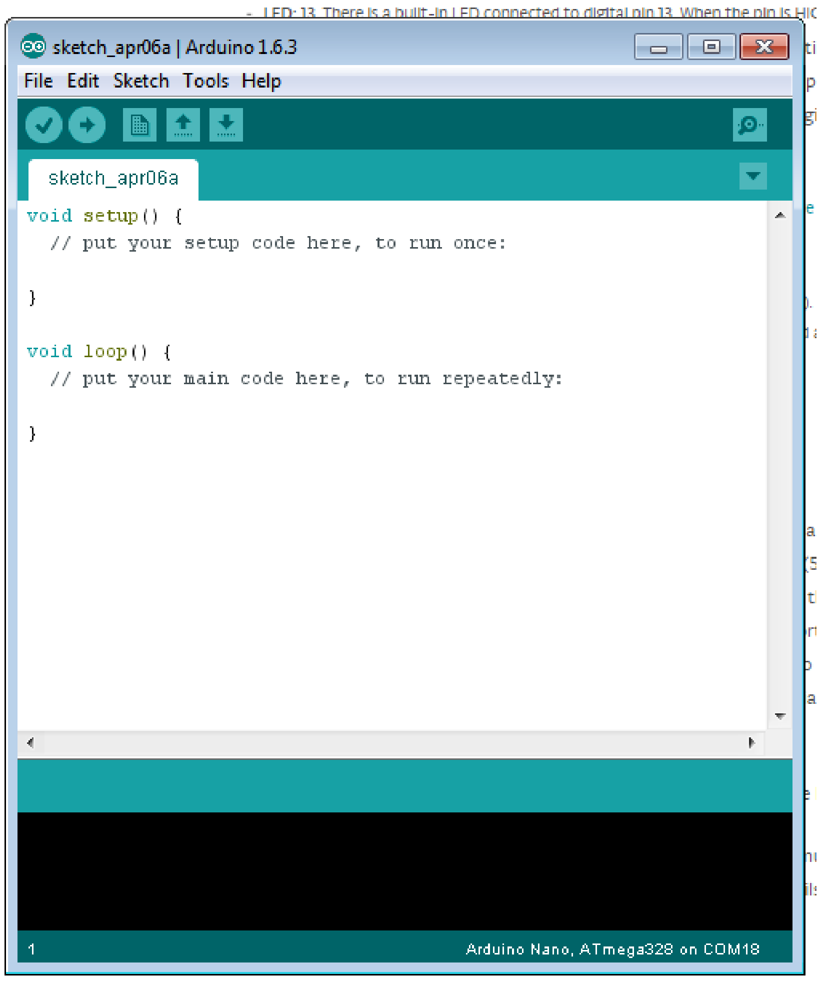
- Bootloader
- FTDI most common, CH340G needs device driver. Read up about FTDI-gate.
- An Arduino can become a programmer to program another
- ATMEGA328P-xx vs no’P’-xx
  - Lower power version vs lower cost version
  - Difference in Social Security #
- Watch out, there is now an “older” bootloader under Processor too.



Board: "Arduino Nano"
Processor: "ATmega328"
Port
Programmer: "Arduino as ISP"
Burn Bootloader

And then you said Software!

- **IDE** (integrated development environment), free download
    - Windows, Mac and Linux (last one sudo apt-get arduino, Raspberry Pi too)
    - Windows, go at least bigger than version 1.8.2
    - 1.8.9 is newest during this Tulsa Clinic in December 2018
- **Fancy editor** colors and calling things in the background!
- **Select** your board, processor, programmer and serial port from the Tools menu
- Type **Code**, **verify** (compile) and and then **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 now also a **Serial Plotter** (Ctrl+Shift+L)


sketch_apr06a | Arduino 1.6.3
File Edit Sketch Tools Help
sketch_apr06a
1
Arduino Nano, ATmega328 on COM18

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

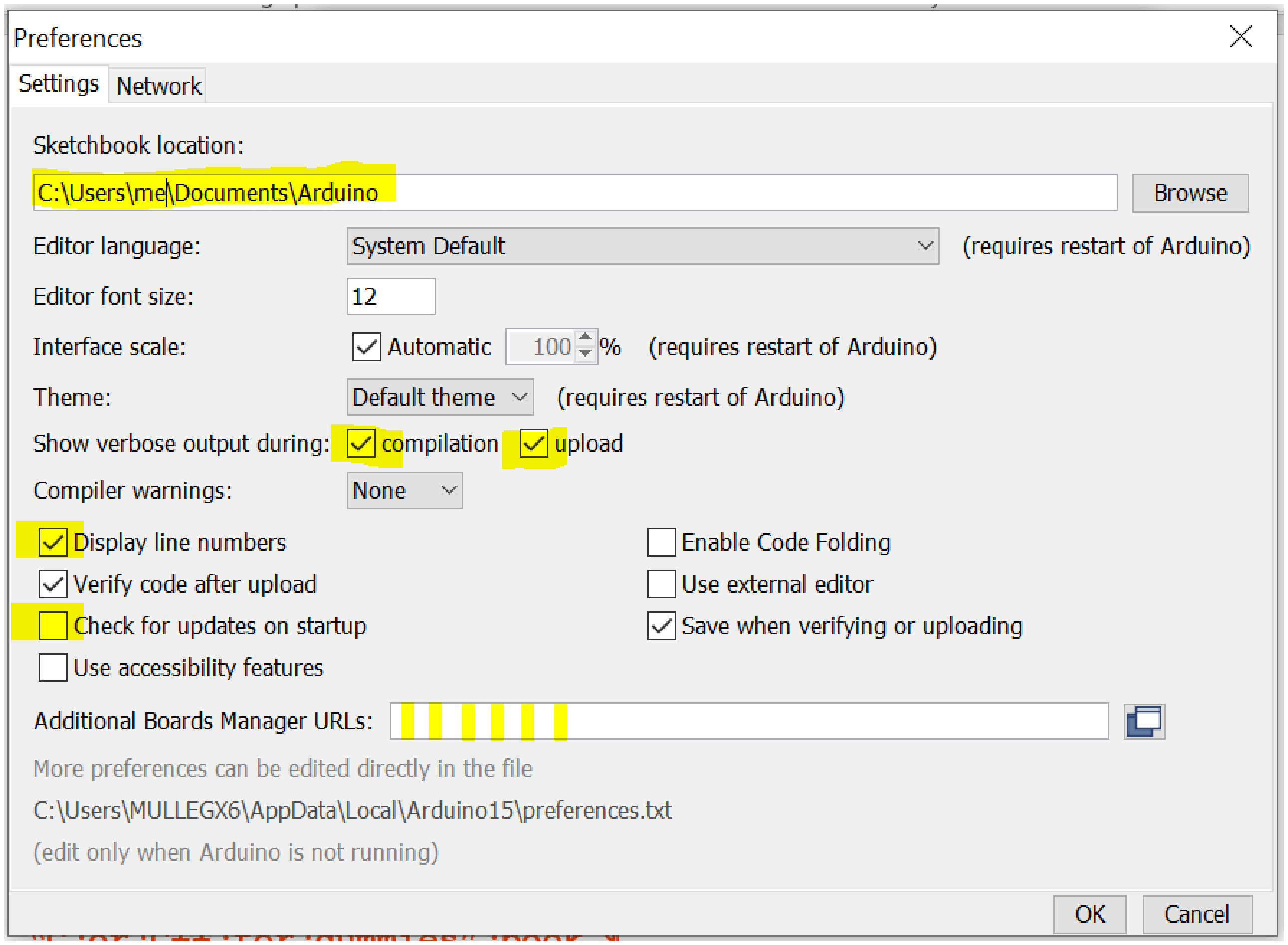
Programming

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

Getting the first example on the Arduino

- ☐ Uno or Nano plugged in?
- ☐ Start the software
- ☐ Set the File->Preferences (Ctrl+comma) // only once
- ☐ Tools → Board → Arduino Uno // only once
- ☐ Tools → Processor → ATmega328 // only once
- ☐ Tools → Port → COMx (or /dev/ttyUSB0) // only once
  (or /dev/ttyACM0)
- ☐ File → Examples → 01.Basics → **Blink**

Preferences...



Preferences

Settings | Network

Sketchbook location: C:\Users\me\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: [✓] Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: [✓] compilation [✓] upload

Compiler warnings: None

- [✓] Display line numbers
- [✓] Verify code after upload
- [ ] Check for updates on startup
- [ ] Use accessibility features
- [ ] Enable Code Folding
- [ ] Use external editor
- [✓] Save when verifying or uploading

Additional Boards Manager URLs:

More preferences can be edited directly in the file
C:\Users\MULLEGX6\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

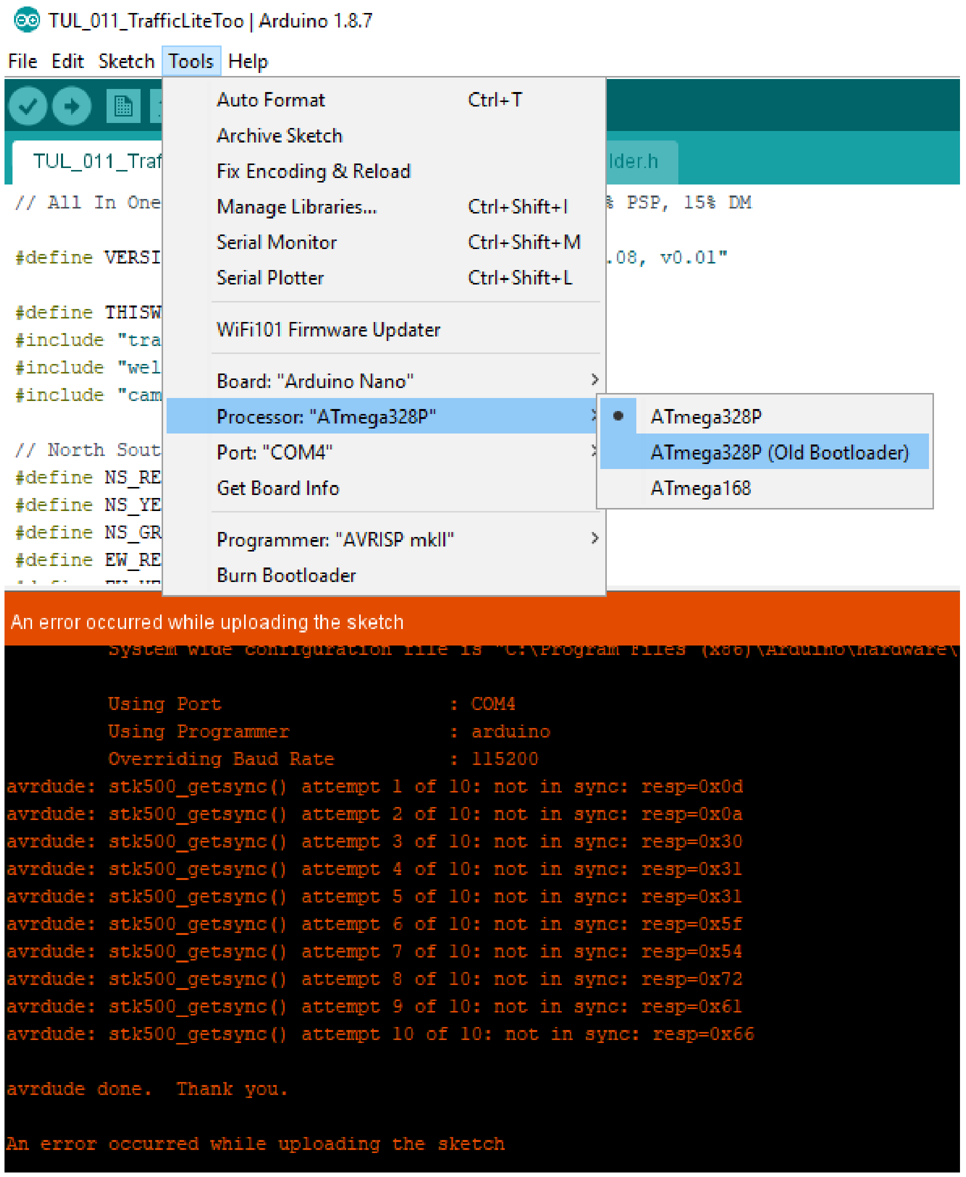
- Remember where your files are!
- 2021.10.08_2130_MyFirstArduinoProgram (Folder must match Filename) (Chronological and Alphabetical order)
- Backup often!

```
// Tulsa 2021.10.08 Arduino for Beginners Make and Take Clinic

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

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

- Save... (Ctrl+s)
- Compile... (or verify, Ctrl+r) ← this will Save first
- Upload... (Ctrl+u) ← but this will Save, Compile AND Upload (**so only use it!**)

Upload Error using the Nano?

Correct **Board**, **Port** and Other Bootloader in **Processor**:


TUL_011_TrafficLiteToo | Arduino 1.8.7
File Edit Sketch Tools Help
TUL_011_Traf
lder.h
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino Nano"
Processor: "ATmega328P"
Port: "COM4"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
● ATmega328P
ATmega328P (Old Bootloader)
ATmega168

```
// All In One                                    % PSP, 15% DM

#define VERSI                                    .08, v0.01"

#define THISW
#include "tra
#include "wel
#include "cam

// North Sout
#define NS_RE
#define NS_YE
#define NS_GR
#define EW_RE
```

An error occurred while uploading the sketch

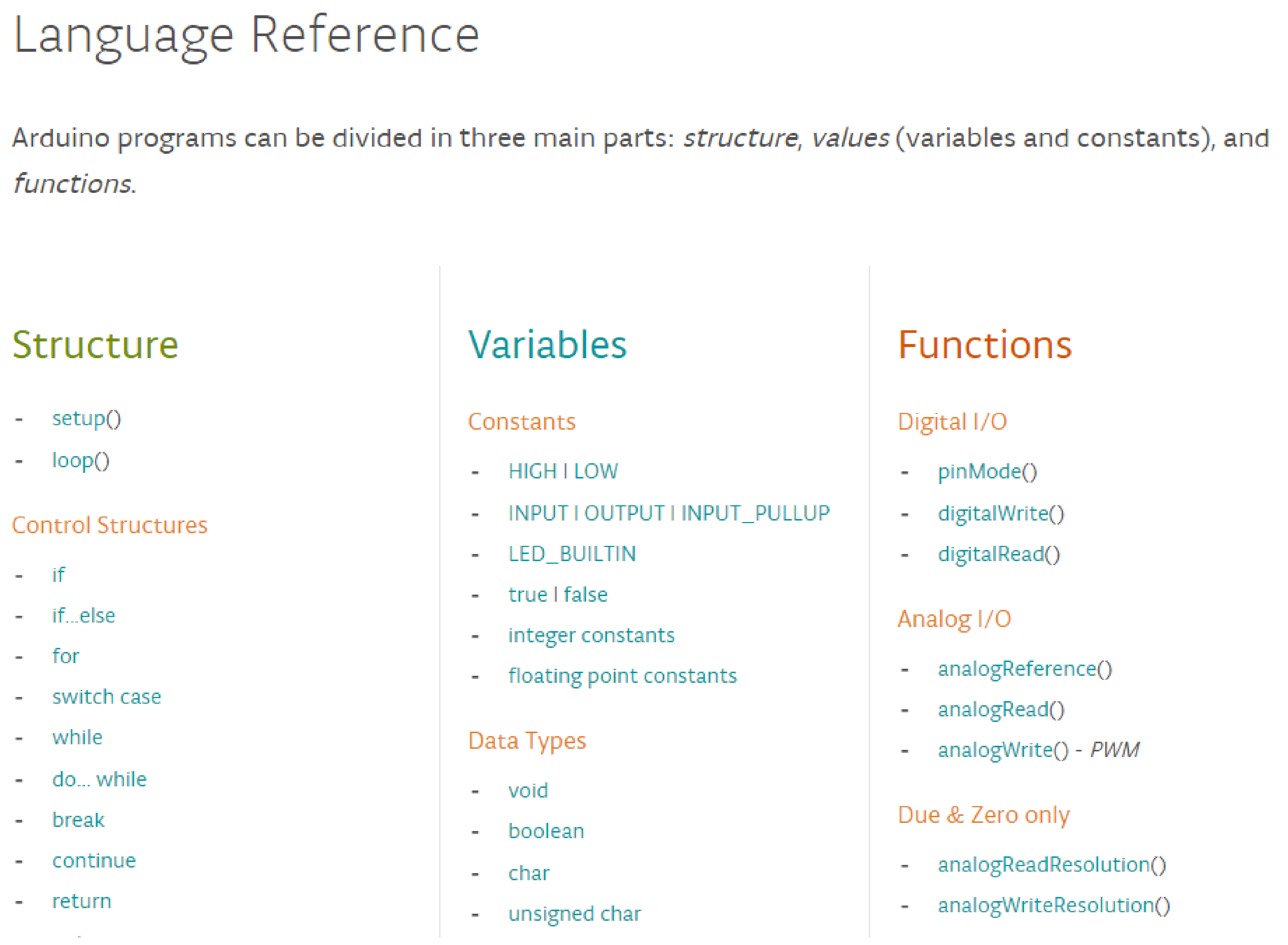
```
         System wide configuration file is "C:\Program Files (x86)\Arduino\hardware\

         Using Port                    : COM4
         Using Programmer              : arduino
         Overriding Baud Rate          : 115200
avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0x0d
avrdude: stk500_getsync() attempt 2 of 10: not in sync: resp=0x0a
avrdude: stk500_getsync() attempt 3 of 10: not in sync: resp=0x30
avrdude: stk500_getsync() attempt 4 of 10: not in sync: resp=0x31
avrdude: stk500_getsync() attempt 5 of 10: not in sync: resp=0x31
avrdude: stk500_getsync() attempt 6 of 10: not in sync: resp=0x5f
avrdude: stk500_getsync() attempt 7 of 10: not in sync: resp=0x54
avrdude: stk500_getsync() attempt 8 of 10: not in sync: resp=0x72
avrdude: stk500_getsync() attempt 9 of 10: not in sync: resp=0x61
avrdude: stk500_getsync() attempt 10 of 10: not in sync: resp=0x66

avrdude done.  Thank you.

An error occurred while uploading the sketch
```

A few useful 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); (but think ahead!)
val_Long = **millis**( );
if(x > 5) { ; } else { ; }
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 buy the "C or C++ for dummies" book.

https://www.arduino.cc/en/Reference/HomePage



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

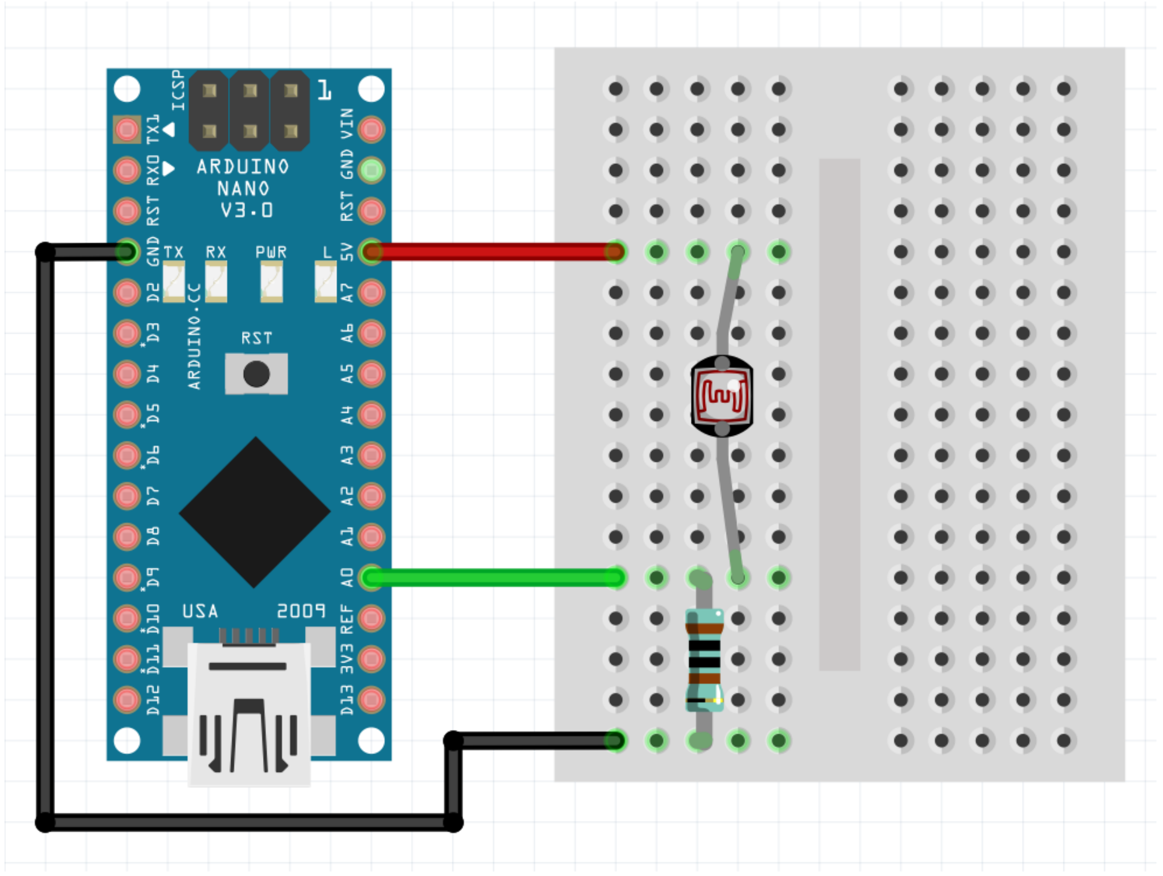
- analogReference()
- analogRead()
- analogWrite() - *PWM*

Due & Zero only

- analogReadResolution()
- analogWriteResolution()

2nd Example

Reading an analog value and send data over Serial



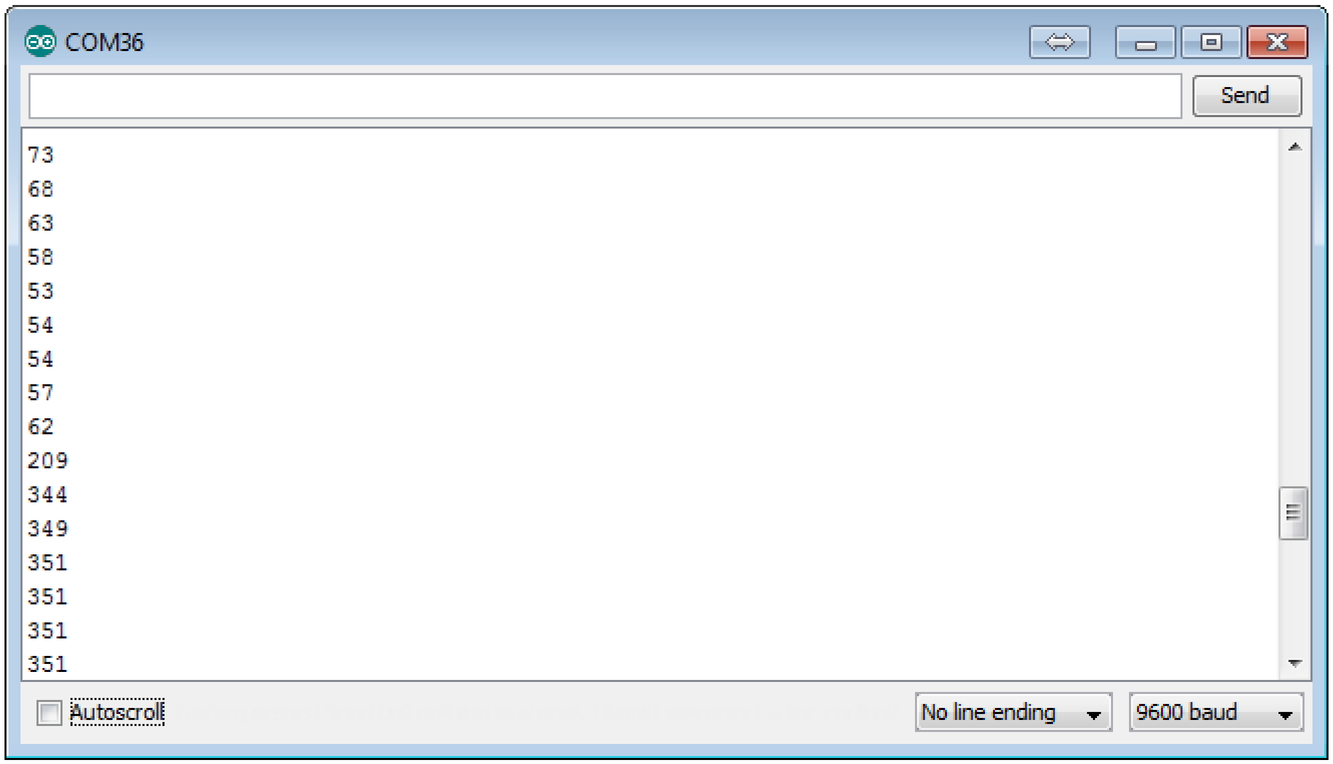
Tip: For making these drawings, check out fritzing.org, you can draw on the breadboard or schematic and even create a PCB layout to make a pc board.

- Open readLight.ino

```
// Reading an analog value and send data over Serial
// 10-bit analog to digital converter.
// It will map input voltages between 0 and 5 volts into
//               integer values between 0 and 1023
#define VERSION_STR  "readLight, 2021.10.08, 07:58, ver 0.01"
#define ANALOG_PIN   A0
#define LED_PIN      13
int lightValue;
void setup( ) {
  pinMode( LED_PIN, OUTPUT );
  pinMode( ANALOG_PIN, INPUT );
  Serial.begin( 115200 );  // baud-rate, press Ctrl+Shift+M after upload
  Serial.println( );
  Serial.println( VERSION_STR );
} // setup( )
void loop( ) {
  lightValue = analogRead( ANALOG_PIN );
  Serial.println( lightValue );
  if ( lightValue > 100 )
    digitalWrite( LED_PIN, HIGH );
  else
    digitalWrite( LED_PIN, LOW );
  delay( 250 );
} // loop( )
```

Wait for it...

- ☐ Press Ctrl-Shift-M (this opens the Serial Monitor from the IDE)
- ☐ Notice dark to light...put your hand over the circuit
- ☐ Watch the LED at about 100



```
73
68
63
58
53
54
54
57
62
209
344
349
351
351
351
351
```

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?

3rd and last one, take home prize!

- Disconnect all breadboard wires, we only need the built in LED (pin 13)
- Open welder.ino

```
#define LED_PIN 13
void setup( ) {
  pinMode( LED_PIN, OUTPUT );      // set digital pin 13 as an output: LED
} // setup( )

void loop( ) {
  int j = random( 0, 100 );        // welding for a random number of cycles
  while( j > 0 ) {
    digitalWrite( LED_PIN, HIGH ); // LED is on...
    delay( random( 0, 100 ) );     // ...for a random number of
                                   // milliseconds between -1 and 100
    digitalWrite( LED_PIN, LOW );  // LED is then turned off...
    delay( random( 0, 100 ) );     // ...for another randomly chosen
                                   //   interval between -1 and 100
                                   // milliseconds
    j--;                           // reduce j by one, same as j = j - 1;
  } // while j
  j = random( 0, 8000 );     // now pick a random number of ms
  delay( j );                // and stay off,
                             // you need to move to another spot to weld
} // loop( )
```

RrrDuino for real Model Railroads

So, what can an **Arduino** do for my railroad? (See the whole list at TxNamib.com)

Since we can turn things on and off:

Lighting:

- Turn LEDs on and off at specified or random times
- Welding, cutting, torching
- Cycle lights in a building, house, or church
- Streetlights
- **Traffic lights**
- **Grade crossing lights**
- Signals for the railroad
- **Flickering lamps, like an Arc Welder and Bathroom light on another pin.**
- Candles, campfires, fire in a building, or just a stove...
- Police car lights, Fire Trucks, Ambulances...

Servos and steppers:

- **A grade crossing gates**
- **A wigwag**
- An animated scene, windmill, gate, logs, manufacturing plant
- **Change a turnout position**
- Moving an uncoupling magnet in and out of position
- A belt driven sawmill

A helicopter, or just a windsock
An overhead crane, like the kind in an intermodal yard picking containers off cars
A forklift lifting
A water tank lowering its spout
A car or engine washing station
Building doors opening or sliding
Controlling your turntable or transfer table
Moving blocks of ice at an icehouse
The guy popping out of a tower!

Sound:
A grade crossing with ding, ding, ding...
After the waterspout is down...water!
A coal loading facility
clickity clack wheel sounds, anywhere
"All Aboard?"
"64 wheels counted"

Data:
How 'bout an LCD or LEDs to show train departure time, or expected time of arrival

Since we can sense things:
Train occupancy
Wigwag
Grade crossing
Streetlights at dusk
Counting Axles: "86 Axles, zero defects!"

Speed:
Train speed, how fast were you going?

Touch:
Yes, **turn it on or off now**

Temperature:
Time to turn the fan on
Boiler is hot, let's go!

Humidity, water, pressure, you name it, if it has an optical, digital or analog output, we can measure it!

Acceleration:
If we speed up, the tires might scream
When we brake, we might want screeching tires

Tilt:
Oops, here comes that water
Or the crane is down, active that electromagnet to pick the metal up

And of course, anything you can imagine!!!

...and for the somewhat advanced

DCC:

An accessory decoder, so your DCC throttle (or JMRI for that matter) could tell accessory #1042 to move a turnout, or turn something on, all over the DCC bus. (**And another site**)

...and for the even little more advance

LCC or OpenLCB:

Moving a few Servos

Since we can talk to things (Infrared, BLE (Bluetooth), WiFi, Morse code):

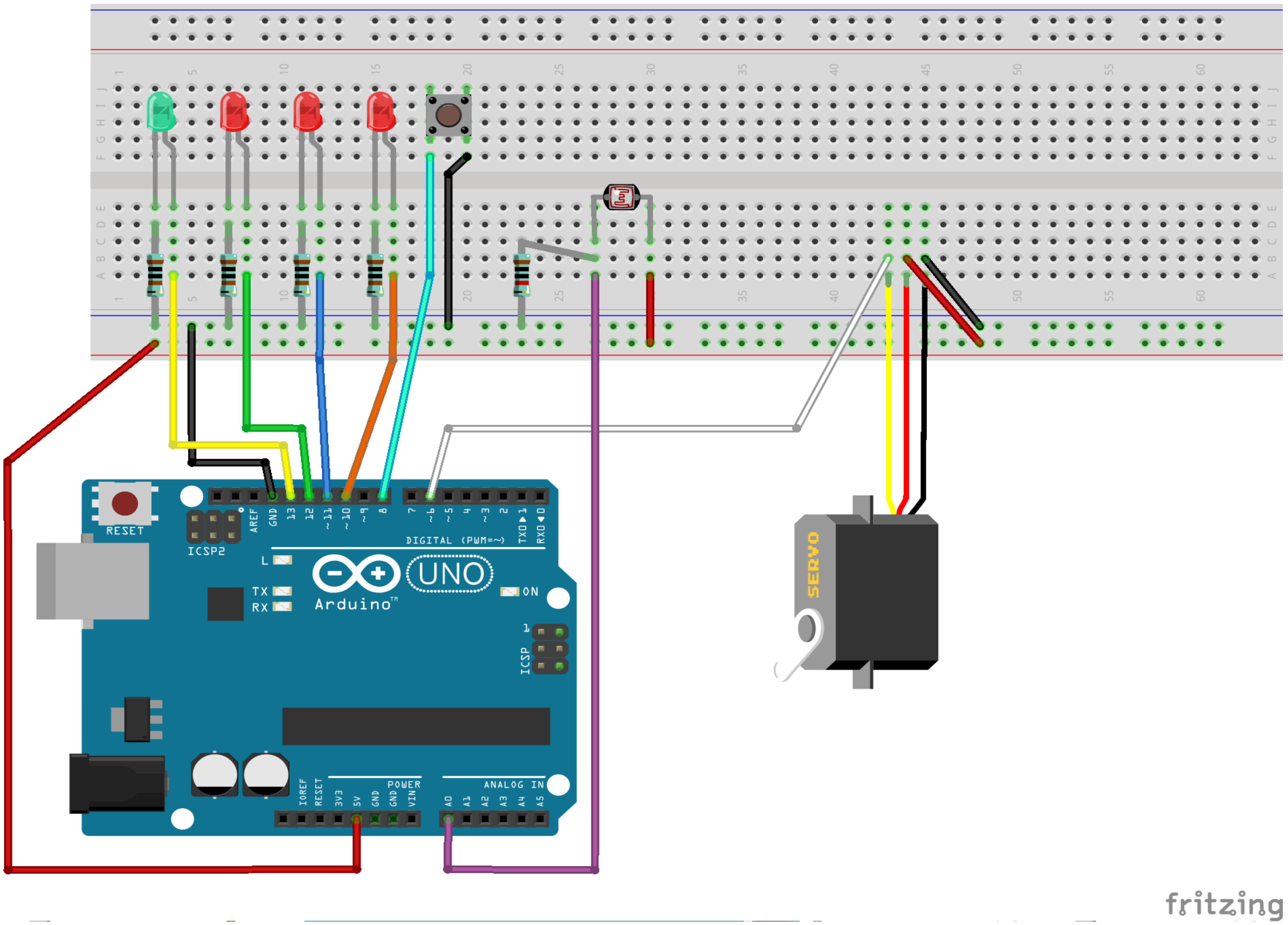
We could send data, this train left
Next train should be arriving at...
Block occupancy, tell the next or opposing signal of such
Controlling the lights and speed of your **Faller self driving car?**

We might talk to our home automation system to turn the A/C on, or heat
We could email a friend
We could update a website, or post on Facebook

Here we go…

Tulsa 2021: RRRduino Beginner's Clinic

Wire this together first!



```
/* Tulsa 2021.10.08 Arduino for Beginners Make and Take Clinic
 * This is my second program: 002_better.ino
 * To flash the LED at pin 13
 * Author: Speed Muller, Date: 2021.10.08, 08:00  */

#define VERSION_STR "002_better, 2021.10.08, 08:00, ver 0.01"

#define LED       13
#define ONTIME   750
#define OFFTIME  250

void setup( ) {
  Serial.begin( 115200 );

  pinMode( LED, OUTPUT );

  Serial.println( );
  Serial.println( VERSION_STR );
} // setup( )

void loop( ) {
  digitalWrite( LED, HIGH );
  delay( ONTIME );

  digitalWrite( LED, LOW );
  delay( OFFTIME );
  //Serial.print( "." );
} // loop( )

// Notes: Resistors are sticky where they go into the band, cut it off
//        Every LED needs its OWN current limiting resistor, Speed rule #114
//        Open the Serial Port Monitor with Ctrl+Shift+M
//        Ctrl-u will compile and upload
```

Why C/C++? main.cpp is hidden to you:

```
int main( ) {
  // setup all the things Arduino needs to set up, like the a Serial port

  // then call setup( )
  setup( );

  while( 1 ) {
    // do some interrupt things to see if new code is coming
    // yada yada

    // then call loop( )
    loop( );
  } // while( 1 )

    return -1;
} // int main( )
```

Note the compiler…

```
../arduino-1.8.13/hardware/tools/avr/bin/avr-g++
```

```
// Speed Rule #006: Keep an eye on the compiler's last breath:
```

```
Sketch uses 2644 bytes (8%) of program storage space. Maximum is 32256 bytes.
Global variables use 298 bytes (14%) of dynamic memory, leaving 1750 bytes for local
variables. Maximum is 2048 bytes.
```

```
/* To flash two LEDs, at pin 13 and pin 12 alternating
 * Author: Speed Muller, Date: 2021.10.08, 08:03  */

#define VERSION_STR "003_more, 2021.10.08, 08:03, ver 0.01"
#define LED1       13
#define LED2       12
#define ONTIME   500
#define OFFTIME  500

void setup( ) {
  Serial.begin( 115200 );
  pinMode( LED1, OUTPUT );
  pinMode( LED2, OUTPUT );

  Serial.println( );
  Serial.println( VERSION_STR );
} // setup( )

void loop( ) {
  digitalWrite( LED1, HIGH );
  digitalWrite( LED2, LOW );
  delay( ONTIME );
  digitalWrite( LED1, LOW );
  digitalWrite( LED2, HIGH );
  delay( OFFTIME );
  //Serial.print( "." );
} // loop( )

// You could also place the 2nd LED between Vcc and pin 13 to get the same effect.
// In other words, when pin 13 is LOW, LED2 is on and then HIGH, LED1 is on instead.
```

```
/* To flash three LEDs, at pin 13, pin 12 and pin 11 in sequence
 * Author: Speed Muller,  Date: 2021.10.08, 08:04  */

#define VERSION_STR "004_sequence, 2021.10.08, 08:04, ver 0.01"

#define LED1       13
#define LED2       12
#define LED3       11

#define ONTIME1   250
#define ONTIME2   250
#define ONTIME3   250

void setup( ) {
  Serial.begin( 115200 );
  pinMode( LED1, OUTPUT );
  pinMode( LED2, OUTPUT );
//  pinMode( LED3, OUTPUT );
  Serial.println( );
  Serial.println( VERSION_STR );
} // setup( )

void loop( ) {
  digitalWrite( LED1, HIGH );
  delay( ONTIME1 );
  digitalWrite( LED1, LOW );
  digitalWrite( LED2, HIGH );
  delay( ONTIME2 );
  digitalWrite( LED2, LOW );
  digitalWrite( LED3, HIGH );
  delay( ONTIME3 );
  digitalWrite( LED3, LOW );
  //Serial.print( "." );
} // loop( )

// Pins can also sink current, so the LED can be connected to VCC (5V)
// and the other end to the pin, and when the pin goes low, the LED turns on!
```

```
/* To flash three LEDs, at pin 13, pin 12 and pin 11 in sequence
 * Author: Speed Muller, Date: 2021.10.08, 08:05  */

#define VERSION_STR "005_sequence_OOP, 2021.10.08, 08:05, ver 0.01"

/* Light with time on and time off specified  */
class Light {
private:
  uint8_t pin;
  uint16_t timeOn, timeOff;
  bool state = false;
  unsigned long now, before;

public:
  /* Constructor */
   Light( uint8_t thePin, uint16_t theTimeOn, uint16_t theTimeOff ) {
    pin = thePin;
    timeOn = theTimeOn;
    timeOff = theTimeOff;
    pinMode( pin, OUTPUT );
  } // constructor Light( uint8_t, uint16_t, uint16_t )

 /* begin( ): set the start time */
  void begin( ) {
    now = before = millis( );
  } // void begin( )

  /*  on( ): turn pin HIGH and set state to true  */
  void on( ) {
    digitalWrite( pin, HIGH );
    state = true;
  } // void on( )
  /* off( ): turn pin LOW and set state to false  */
  void off( ) {
    digitalWrite( pin, LOW );
    state = false;
  } // void off( )

  /* update( ): from the state it is in, determine if the time has expired and flip the state   */
  void update( ) {
    now = millis( );
    if( state == true ) {
      if( now - before >= timeOn ) {
        before = now;
        off( );
      } // if time on is over
    } else {
     if( now - before >= timeOff ) {
        before = now;
        on( );
      } // if time off is over
    } // if on or off
  } // void update( )
}; // class Light( )


#define BAUD 115200

#define LED1       13
#define LED2       12
#define LED3       11

#define ONTIME1   250
#define ONTIME2   250
#define ONTIME3   250

#define OFFTIME1  500
#define OFFTIME2  500
```

```
#define OFFTIME3  500

#define DELAY     250

Light light1( LED1, ONTIME1, OFFTIME1 );
Light light2( LED2, ONTIME2, OFFTIME2 );
Light light3( LED3, ONTIME3, OFFTIME3 );

void setup( ) {
  Serial.begin( BAUD );

  light1.begin( );
  delay( DELAY );
  light2.begin( );
  delay( DELAY );
  light3.begin( );

  Serial.println( );
  Serial.println( VERSION_STR );
} // setup( )

void loop( ) {
  light1.update( );
  light2.update( );
  light3.update( );
} // loop( )

// And now Light can move to light.h and we can use all kinds of lights: random,
fading, flickering, etc!
```

```
/* To flash three LEDs, at pin 13, pin 10 and pin 11 in sequence, but roll them!
 * Author: Speed Muller, Date: 2021.10.08, 08:06 */

#define VERSION_STR "006_sequence_OOP_test_roll, 2021.10.08, 08:06, ver 0.01"
#define BAUD 115200

#define LED1       13
#define LED2       12
#define LED3       11

#define ONTIME1   250
#define ONTIME2   250
#define ONTIME3   250

#define OFFTIME1  500
#define OFFTIME2  500
#define OFFTIME3  500

#define DELAY     125



// Try drastic changes in the ONTIME and OFFTIME for any of the LEDs,
// forget that they might be in sequence. Try DELAY 0
```

```
/* When the boom is down, turn the tip of the boom's light on,
 *   and flash the two other lights on the boom (as well as the ones under
 *   the hoods).
 * Author: Speed Muller, Date: 2021.10.08, 08:07 */

#define VERSION_STR "007_Boom_LEDs, 2021.10.08, 08:07, ver 0.01"
#define BAUD 115200

// heartbeat
#define LED1       13

// Boom LEDs
#define LED2       12
#define LED3       11
#define LED4       10

#define ONTIME1   100
#define ONTIME2   500
#define ONTIME3   500
#define ONTIME4   500

#define OFFTIME1  900
#define OFFTIME2    0
#define OFFTIME3  500
#define OFFTIME4  500

#define DELAY       0

Light light4( LED4, ONTIME4, OFFTIME4 );
```

```
/* When the boom is going down, simulated with a pin going low on pin 8, turn the tip of
 *   the boom's light on, and flash the two other lights on the boom (as well as the ones
 *   under the hoods).
 * Author: Speed Muller, Date: 2021.10.08, 08:08 */

#include "light.h"

#define VERSION_STR "008_Enable_Boom_LEDs, 2021.10.08, 08:08, ver 0.01"
#define BAUD 115200

// heartbeat
#define LED1       13

// Boom LEDs
#define LED2       12
#define LED3       11
#define LED4       10

// Enable pin
#define INPUTPIN    8

Light light1( LED1, ONTIME1, OFFTIME1 );
Light light2( LED2, ONTIME2, OFFTIME2 );
Light light3( LED3, ONTIME3, OFFTIME3 );
Light light4( LED4, ONTIME4, OFFTIME4 );

void setup( ) {
  Serial.begin( BAUD );

  pinMode( INPUTPIN, INPUT_PULLUP );
...
} // setup( )
```

```
void loop( ) {
  if( digitalRead( INPUTPIN ) == LOW ) {
    light2.poweron( );
    light3.poweron( );
    light4.poweron( );
  } else {
    light2.poweroff( );
    light3.poweroff( );
    light4.poweroff( );
  } // if pin is active (which is LOW )

  light1.update( );
  light2.update( );
  light3.update( );
  light4.update( );
} // loop( )

// Notice that your heartbeat is still going, so the code must be running.
// Speed rule 202: Always have a heartbeat!
```

light.h:

```
/* Light with time on and time off specified */
class Light {
private:
  uint8_t pin;
  uint16_t timeOn, timeOff;
  bool state = false;
  bool power = false;
  unsigned long now, before;
public:
  /* Constructor  */
   Light( uint8_t thePin, uint16_t theTimeOn, uint16_t theTimeOff ) {
    pin = thePin;
    timeOn = theTimeOn;
    timeOff = theTimeOff;
    pinMode( pin, OUTPUT );
  } // constructor Light( uint8_t, uint16_t, uint16_t )

  /* Destructor  */
  ~Light( ) { } // destructor ~Light( )

  /* begin( ): set the start time */
  void begin( ) {
    now = before = millis( );
    power = true;
  } // void begin( )

  /* on( ): turn pin HIGH and set state to true, if there is power */
  void on( ) {
    if( power ) {
      digitalWrite( pin, HIGH );
    }
    state = true;
  } // void on( )

  /* off( ): turn pin LOW and set state to false   */
  void off( ) {
    digitalWrite( pin, LOW );
    state = false;
  } // void off( )

  /* poweron( ): turn power back on, state is still last state */
  void poweron( ) {
    power = true;
  } // void poweron( )

  /* poweroff( ): turn pin LOW, but keep state */
  void poweroff( ) {
    digitalWrite( pin, LOW );
    power = false;
  } // void poweroff( )

  /* update( ): from the state it is in, determine if the time has expired and flip the state */
  void update( ) {
    now = millis( );
    if( state == true ) {
      if( now - before >= timeOn ) {
        before = now;
        off( );
      } // if time on is over
    } else {
     if( now - before >= timeOff ) {
        before = now;
        on( );
      } // if time off is over
    } // if on or off
  } // void update( )
}; // class Light( )

// end of file //

// Library files can go into the Arduino/libraries folder, so they can be maintained
// in one place alone.
```

```
/* When the boom is going down, simulated with a pin going low on pin 8, or the
 *   light on a photoresistor indicating occupancy, turn the tip of the boom's
 *   light on, and flash the two other lights on the boom (as well as the ones under
 *   the hoods).
 *
 * But, in 009 we will just read the light every second and see what we get!
 * Author: Speed Muller, Date: 2021.10.08, 08:09 */

#include "light.h"

#define VERSION_STR "009_Enable_Boom_LEDs_with_Light, 2021.10.08, 08:09, ver 0.01"
#define BAUD 115200
...
// Light sensor pin
#define ANAPIN     A0

#define TIMEOUT  1000

unsigned long now, before;

void setup( ) {
  Serial.begin( BAUD );

  pinMode( INPUTPIN, INPUT_PULLUP );
  pinMode( ANAPIN, INPUT );

...
  now = before = millis( );
} // setup( )
```

```
void loop( ) {
  now = millis( );
  if( now -before > TIMEOUT ) {
    before = now;
    Serial.println( analogRead( ANAPIN ) );
  } // if time to read


} // loop( )



// Now is a good time to look at the Serial Plotter in the Arduino IDE!
// Press Ctrl+Shift+L after closing the Serial Monitor
```

```
/* Plotting some numbers...Ctrl+Shift+L
 * Author: Speed Muller, Date: 2021.10.08, 08:09b  */
#include "light.h"
#define VERSION_STR "009b_Plotting, 2021.10.08, 08:09b, ver 0.01"
#define BAUD      115200

// 10-bit A/D, 0-1023
#define MAXPOINT    1023
#define SETPOINT     100

unsigned long now, before;
uint16_t analogValue;
uint8_t servoValue;

void setup( ) {
  Serial.println( "Setpoint,Light,Map" );    // Key / legend
  Serial.print( "0," );                      // just to scale the Ctrl+Shift+L graph
  Serial.print( MAXPOINT ); Serial.print( "," );
  Serial.println( 255 );
} // setup( )

void loop( ) {
  now = millis( );
  if( now -before > TIMEOUT ) {
    before = now;
    Serial.print( SETPOINT );   Serial.print( "," );

    analogValue = analogRead( ANAPIN );     Serial.print( analogValue );
    servoValue = map( analogValue, 0, 1023, 0, 255 );
    Serial.print( "," );      Serial.println( servoValue );
  } // if time to read
```

```
/* Servo into a "tilde" pin, like D6
 * Author: Speed Muller, Date: 2021.10.08, 08:10
 */
#define VERSION_STR "010_Servo_by_Light, 2021.10.08, 08:10, ver 0.01"

// include the Servo library
#include <Servo.h>

// Servo pin
#define SERVOPIN       6

Servo myServo;  // create a servo object

void setup( ) {
...
  myServo.attach( SERVOPIN );    // Attach the pin to the servo
...
} // setup( )

void loop( ) {
  now = millis( );
  if( now -before > TIMEOUT ) {
    before = now;

    analogValue = analogRead( ANAPIN );     Serial.print( analogValue );
    servoValue = map( analogValue, 0, 1023, 0, 255 );
    Serial.print( "," );      Serial.println( servoValue );

    // Set the servo
    myServo.write( servoValue );
  } // if time to read
} // loop( )
```

```
/* When the boom is going down, simulated with a pin going low on pin 8, or the
 *   light on a photoresistor indicating occupancy,
 *   turn the tip of the boom's light on, and flash the two other lights on the boom
 *  (as well as the ones under the hoods).
 * Author: Speed Muller, Date: 2021.10.08, 08:11  */

#include "light.h"

#define VERSION_STR "011_Enable_by_Light, 2021.10.08, 08:11, ver 0.01"

void loop( ) {
  if( ( digitalRead( INPUTPIN ) == LOW ) || ( analogValue < SETPOINT ) ) {
    light2.poweron( );
    light3.poweron( );
    light4.poweron( );
  } else {
    light2.poweroff( );
    light3.poweroff( );
    light4.poweroff( );
  } // if pin is active (which is LOW )

} // loop( )
```

```
/*
 * Tulsa 2021.10.08 Arduino for Beginners Make and Take Clinic
 *
 * When the boom is going down, simulated with a pin going low on pin 8, or the light
 *   on a photoresistor indicating occupancy, turn the tip of the boom's light on,
 *   and flash the two other lights on the boom (as well as the ones under the
 *   hoods).
 *
 *   Making an object with the servo, so it can slowly move between two positions.
 *
 * Author: Speed Muller, Date: 2021.10.08, 08:10
 */

// include the Servo library
#include <Servo.h>

#include "light.h"
#include "tulsaservo.h"

#define VERSION_STR "012_Servo_by_OOP, 2021.10.08, 08:12, ver 0.01"

// Servo pin
#define SERVOPIN           6

#define SERVOUPDATETIME  150

// TulsaServo( uint8_t theServoPin, uint16_t theTime, uint8_t theUpPosition, uint8_t
theDownPosition )
TulsaServo boomServo( SERVOPIN, SERVOUPDATETIME, 80, 200 );

void setup( ) {
  boomServo.begin( );
  boomServo.raise( );
} // setup( )

void loop( ) {
  now = millis( );
  if( now -before > TIMEOUT ) {
    before = now;

    analogValue = analogRead( ANAPIN );
    Serial.println( analogValue );
  } // if time to read

  if( ( digitalRead( INPUTPIN ) == LOW ) || ( analogValue < SETPOINT ) ) {
    light2.poweron( );
    light3.poweron( );
    light4.poweron( );
    boomServo.lower( );
  } else {
    light2.poweroff( );
    light3.poweroff( );
    light4.poweroff( );
    boomServo.raise( );
  } // if pin is active (which is LOW )

  light1.update( );
  light2.update( );
  light3.update( );
  light4.update( );
  boomServo.update( );
} // loop( )
```

tulsaservo.h:

```
/*
 * Servo with two positions, up and down, to move to on command with a delay inbetween steps
 */
class TulsaServo {
private:
  uint8_t servoPin;
  uint16_t timeout;
  uint8_t myPosition;
  uint8_t commandedPosition;
  uint8_t upPosition;
  uint8_t downPosition;

  bool power = false;
  unsigned long now, before;
  Servo myServo;               // the real servo

public:
  /* Constructor */
   TulsaServo( uint8_t theServoPin, uint16_t theTime, uint8_t theUpPosition, uint8_t theDownPosition ) {
    servoPin = theServoPin;
    timeout = theTime;
    upPosition = theUpPosition;
    downPosition = theDownPosition;

    // send servo to the middle of the two points at start-up
    commandedPosition = myPosition = (int8_t)( ( (int16_t)theUpPosition + (int16_t)theDownPosition ) / 2 );
    myPosition = commandedPosition;
  } // constructor TulsaServo( uint8_t, uint16_t, uint8_t, uint8_t )

  /* Destructor */
  ~TulsaServo( ) {
    mySero.detach( );
   } // destructor ~TulsaServo( )


  /* begin( ): set the start time */
  void begin( ) {
    now = before = millis( );
    power = true;
    myServo.attach( servoPin ); // Attach the servo to the pin
    output( );
  } // void begin( )


  /* lower( ): send the servo to the down position */
  void lower( ) {
    commandedPosition = downPosition;
  } // void lower( )
  /* raise( ): send the servo to the raised position */
  void raise( ) {
    commandedPosition = upPosition;
  } // void raise( )


  /* atTop( ): return true if at top */
  bool atTop( ) {
    return( myPosition == upPosition );
  } // bool atTop( )


  /* output( ): move the servo */
  void output( ) {
//    Serial.print( "Servo:" );
//    Serial.println( myPosition );
    myServo.write( myPosition );
  } // void output( )


  /* update( ): from the state it is in, determine if the time has expired and flip the state */
  void update( ) {
    now = millis( );
    if( now - before >= timeout ) {
      before = now;
      if( myPosition == commandedPosition ) {
        // do nothing
      } else {
        if( myPosition < commandedPosition ) {
          myPosition++;
          output( );
        } else {
          myPosition--;
          output( );
        } // decrease
      } // not there yet
    } // if time on is over
  } // void update( )
}; // class TulsaServo( )

/* end of file */
```

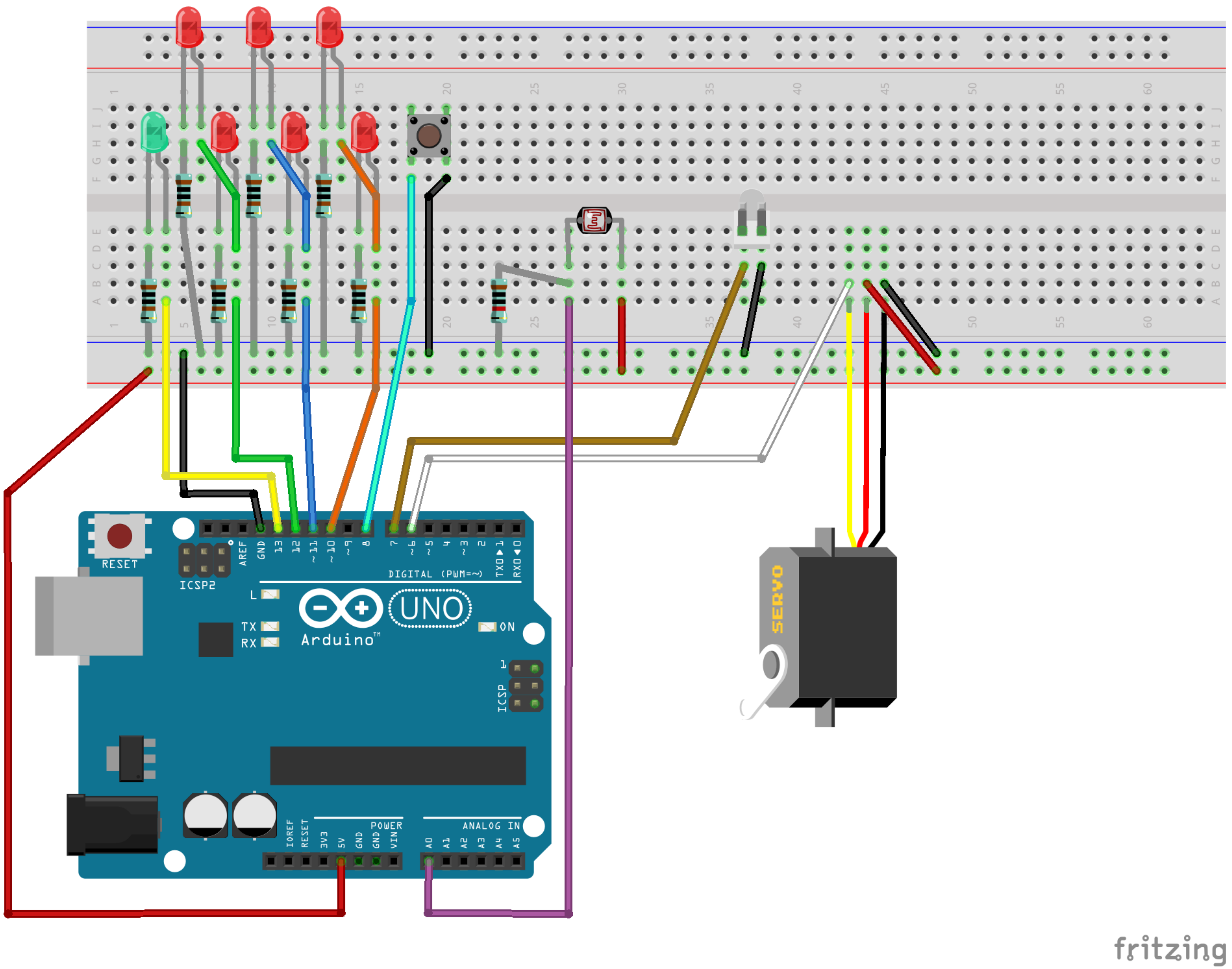
And now the last one:

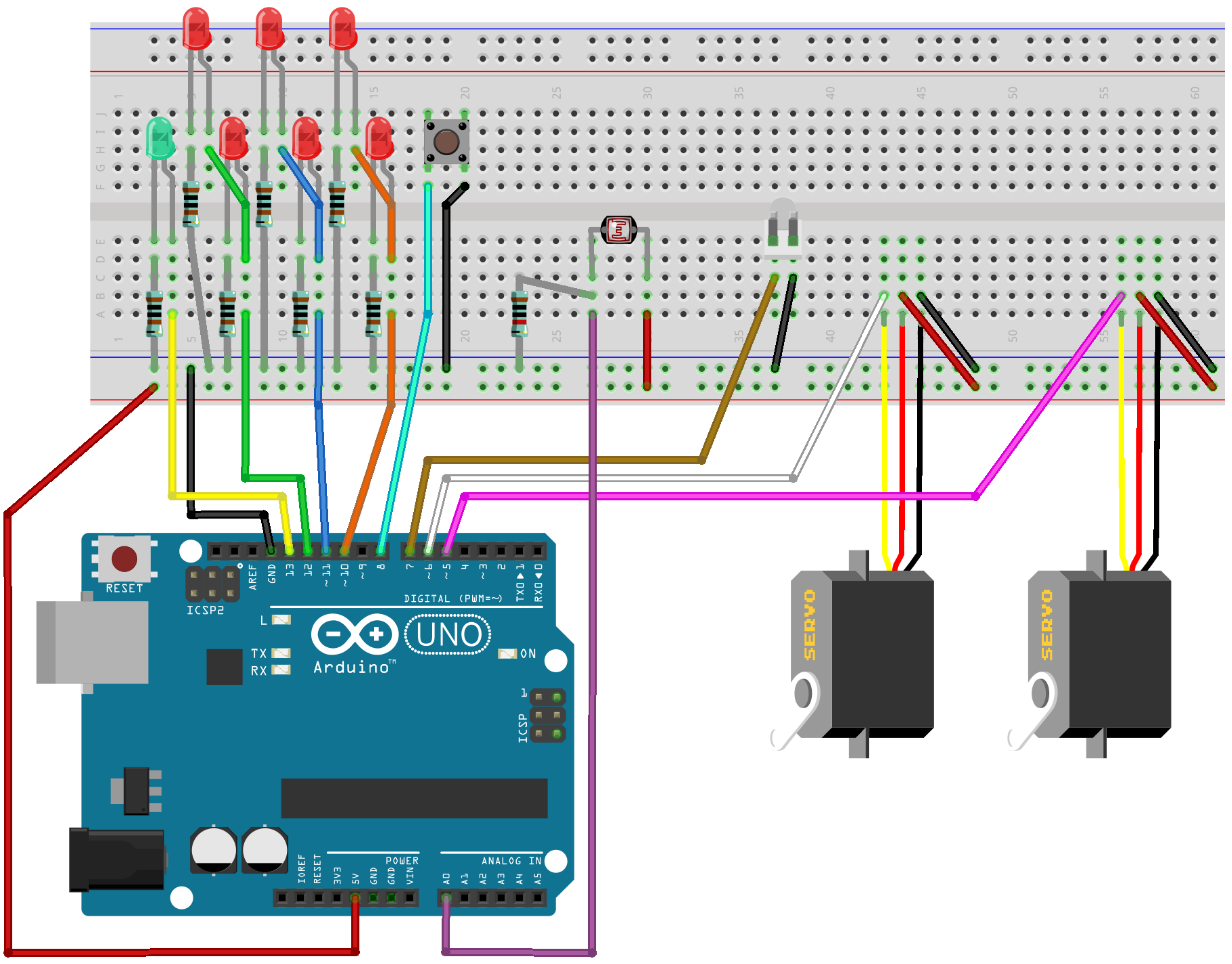
```
/*
 * Tulsa 2021.10.08 Arduino for Beginners Make and Take Clinic
 *
 * When the boom is going down, simulated with a pin going low on pin 8, or the light
 * on a photoresistor indicating occupancy, turn the tip of the boom's light on, and
 * flash the two other lights on the boom (as well as the ones under the hoods).
 *
 *   Making an object with the servo, so it can slowly move between two positions.
 *
 *   Keep the lights on until the servo is at the top
 *
 *   Enable sound pin.
 *
 * Servo into a "tilde" pin, like D6
 *
 * Author: Speed Muller, Date: 2021.10.08, 08:10
 */

// include the Servo library
#include <Servo.h>

#include "light.h"
#include "tulsaservo.h"

#define VERSION_STR "013_Gate_Crossing_Controller, 2021.10.08, 08:13, ver 0.01"
#define BAUD          115200

// heartbeat
#define LED1                 13
```

```
// Boom LEDs
#define LED2               10
#define LED3               11
#define LED4               12

// Enable sound pin
#define SOUNDPIN            7

// Enable pin
#define INPUTPIN            8

// Light sensor pin
#define ANAPIN             A0

// Servo pin
#define SERVOPIN            6

#define ONTIME1           100
#define ONTIME2           500
#define ONTIME5           500

#define OFFTIME1          900
#define OFFTIME2            0
#define OFFTIME5          500

#define SERVO_UPDATETIME 100

#define TIMEOUT          1000

// 10-bit A/D, 0-1023
#define MAXPOINT         1023
#define SETPOINT          250

Light light1( LED1, ONTIME1, OFFTIME1 );
Light light2( LED2, ONTIME2, OFFTIME2 );
Light light3( LED3, ONTIME5, OFFTIME5 );
Light light4( LED4, ONTIME5, OFFTIME5 );

// TulsaServo( uint8_t theServoPin, uint16_t theTime, uint8_t theUpPosition, uint8_t
theDownPosition )
TulsaServo boomServo( SERVOPIN, SERVO_UPDATETIME, 80, 200 );

unsigned long now, before;
uint16_t analogValue;
uint8_t servoValue;


void setup( ) {
  Serial.begin( BAUD );

  pinMode( INPUTPIN, INPUT_PULLUP );
  pinMode( ANAPIN, INPUT );
  pinMode( SOUNDPIN, OUTPUT );

  boomServo.begin( );

  light1.begin( );
  light2.begin( );
  light3.begin( );
  delay( ONTIME5 );
  light4.begin( );

  Serial.println( );
  Serial.println( VERSION_STR );

  now = before = millis( );
  boomServo.raise( );
} // setup( )
```

```
void loop( ) {
  now = millis( );
  if( now - before > TIMEOUT ) {
    before = now;
    analogValue = analogRead( ANAPIN );
    Serial.println( analogValue );
  } // if time to read

  if( ( digitalRead( INPUTPIN ) == LOW ) || ( analogValue < SETPOINT ) ) {
    light2.poweron( );
    light3.poweron( );
    light4.poweron( );
    boomServo.lower( );
    digitalWrite( SOUNDPIN, HIGH );
  } else {
    boomServo.raise( );
    if( boomServo.atTop( ) == true ) {
      light2.poweroff( );
      light3.poweroff( );
      light4.poweroff( );
      digitalWrite( SOUNDPIN, LOW );
    } // if done
  } // if pin is active (which is LOW )

  light1.update( );
  light2.update( );
  light3.update( );
  light4.update( );
  boomServo.update( );
} // loop( )
```

```
// Can do better than ALWAYS telling it to raise or lower it.
// Create a state and set or reset it.

// Can also put all the lights in one GradeCrossingLights class, to reduce all the
numbers needed here to create the lights.

//   Sketch uses 4300 bytes (13%) of program storage space. Maximum is 32256 bytes.
//   Global variables use 369 bytes (18%) of dynamic memory, leaving 1679 bytes
//   for local variables. Maximum is 2048 bytes.
```


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX▶1
RX◀0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
1
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
SERVO
fritzing


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX▶1
RX◀0
DIGITAL (PWM=~)
L
TX
RX
Arduino™
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
SERVO
SERVO
fritzing

A few Important notes:

- Copying code from **anywhere else**, watch out for the **quotes**: “ and ” is not the same as " (you need the latter one)
- 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 code again
- Use dates and English words as much as you can. Easier to find and search later

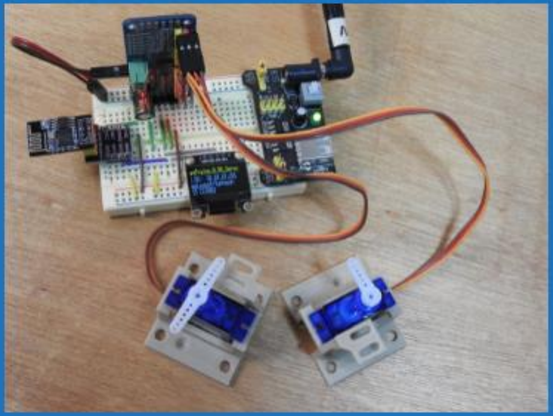
Special Plug for mqTrains!

Created with the Arduino IDE, but the ESP8266 is Espressif’s WiFi connected microcontroller.

mqTrains MQTT in the model train world downloads

mqTrains – Home

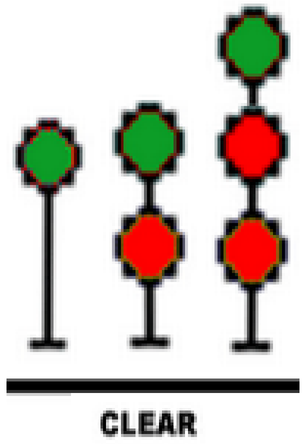
If you want a layout with less wires, why not use WiFi?



Any questions?

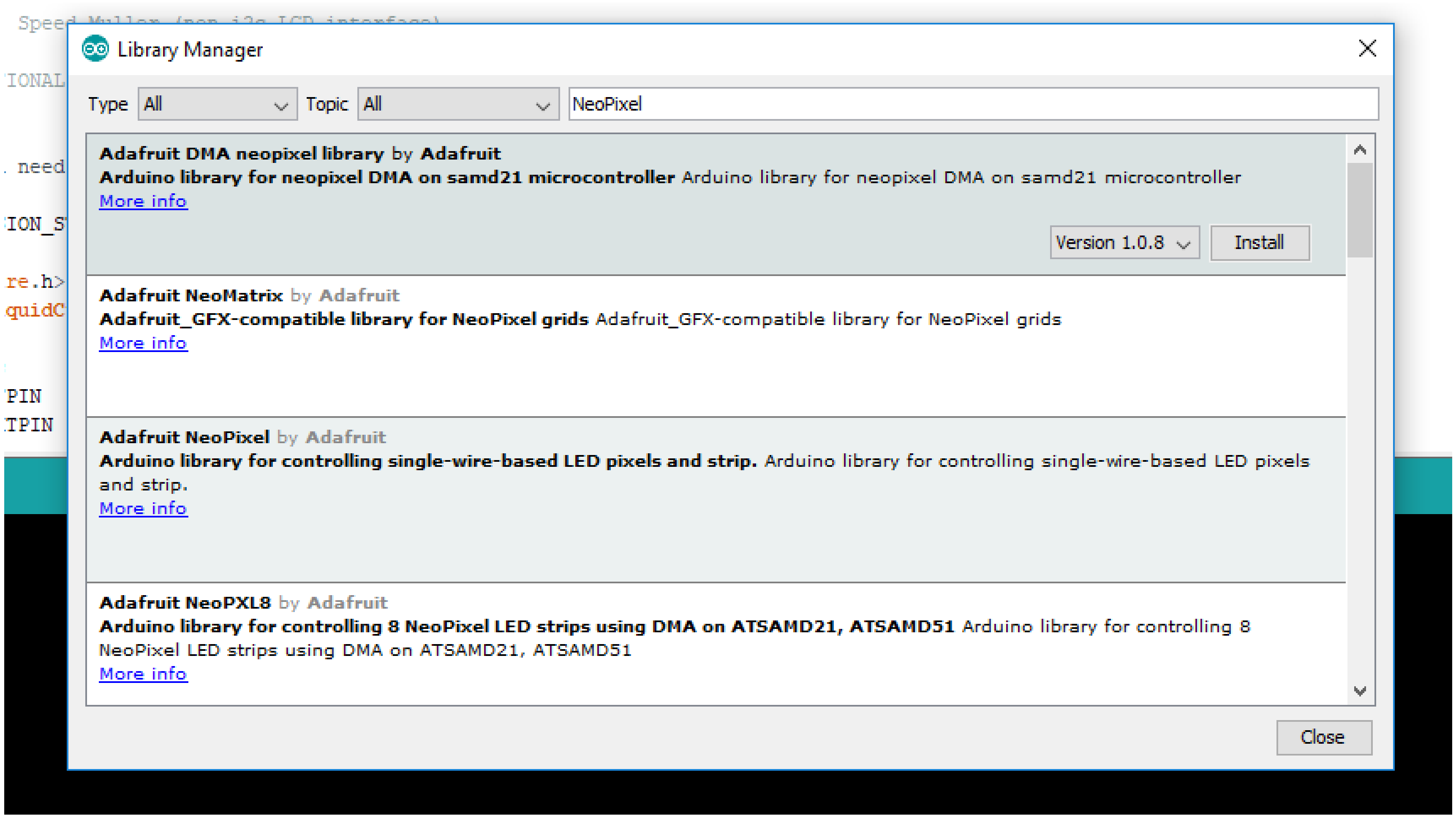
The only **dumb question** is the one **you did not ask**!

The End



Thank you to Ray, Dave, Allan, *Bob, Sandy, Harry, Stephanie, Alissa, my Boss, and all the others that did not bother us while we were having fun doing this…*

One wire, WS2811, WS2812, NeoPixels
(*Need to install Arduino Library, Ctrl+Shift+i*)



Adafruit DMA neopixel library by **Adafruit**
Arduino library for neopixel DMA on samd21 microcontroller Arduino library for neopixel DMA on samd21 microcontroller
More info
Version 1.0.8 Install

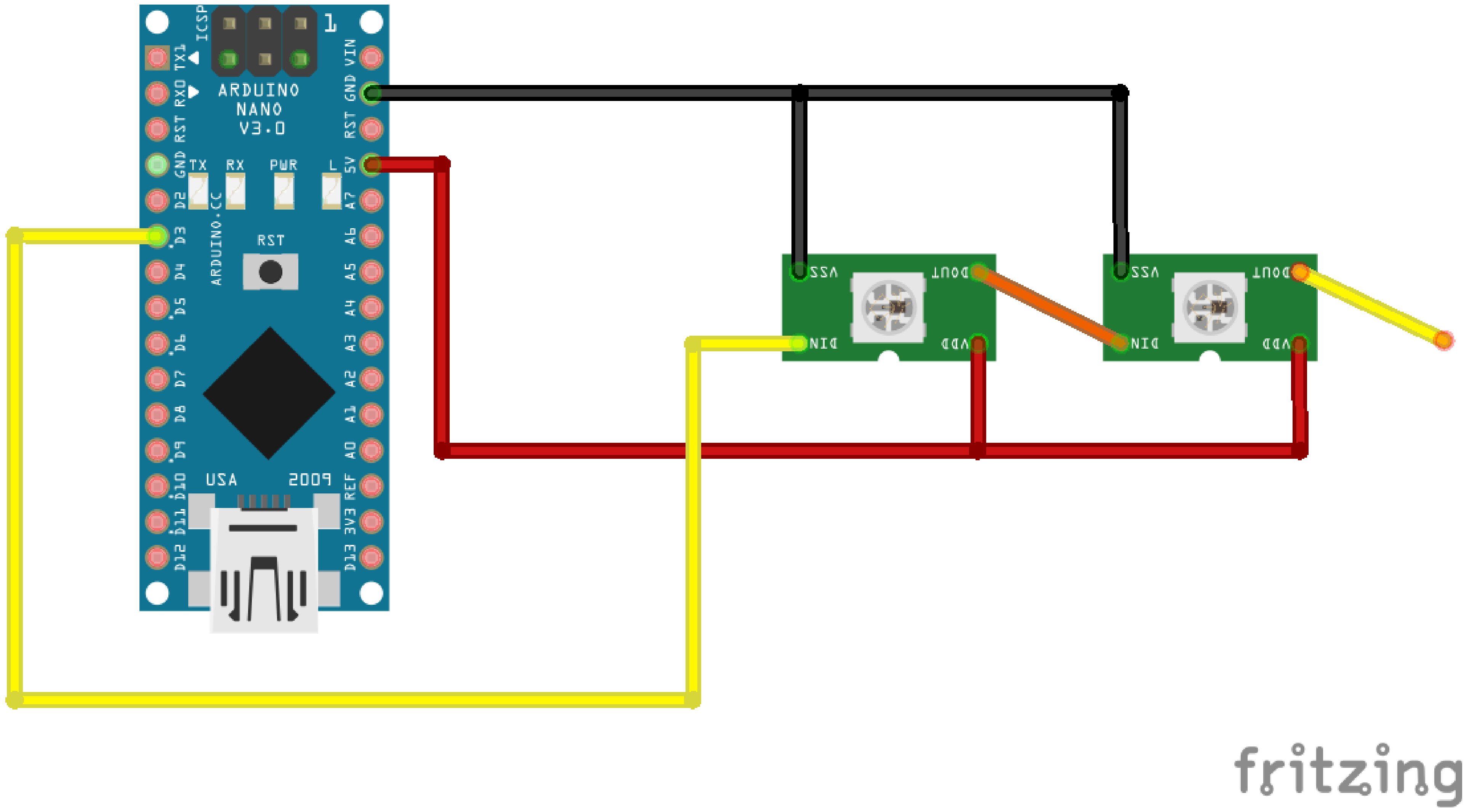
Adafruit NeoMatrix by **Adafruit**
Adafruit_GFX-compatible library for NeoPixel grids Adafruit_GFX-compatible library for NeoPixel grids
More info

Adafruit NeoPixel by **Adafruit**
Arduino library for controlling single-wire-based LED pixels and strip. Arduino library for controlling single-wire-based LED pixels and strip.
More info

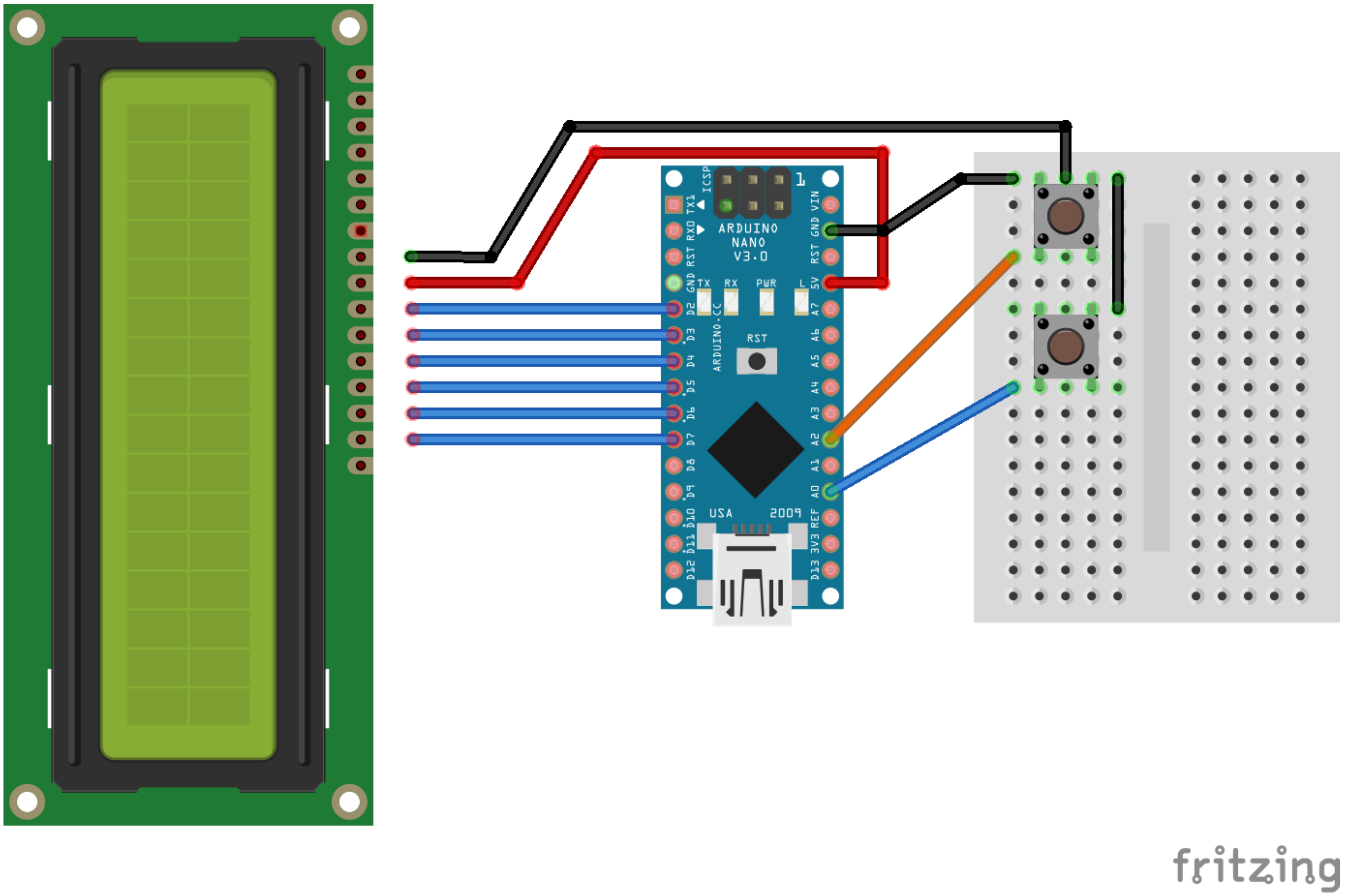
Adafruit NeoPXL8 by **Adafruit**
Arduino library for controlling 8 NeoPixel LED strips using DMA on ATSAMD21, ATSAMD51 Arduino library for controlling 8 NeoPixel LED strips using DMA on ATSAMD21, ATSAMD51
More info

How many? More than 50!

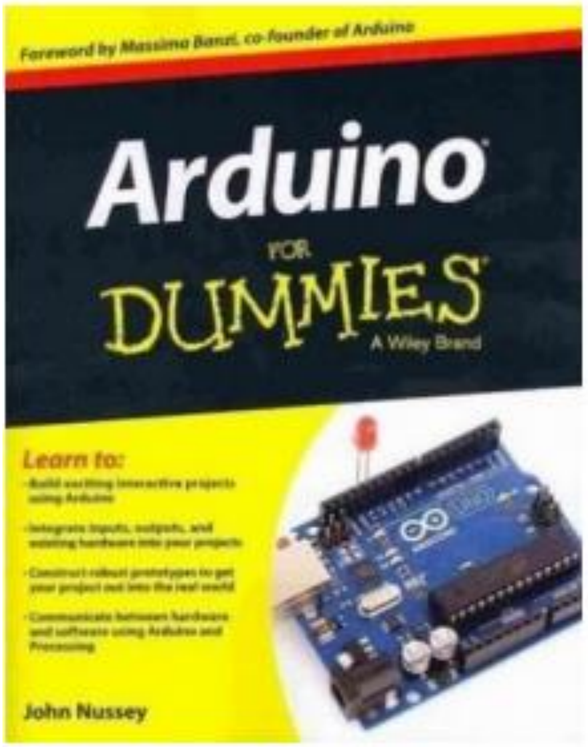
https://www.youtube.com/watch?v=-gpFhmG4abw

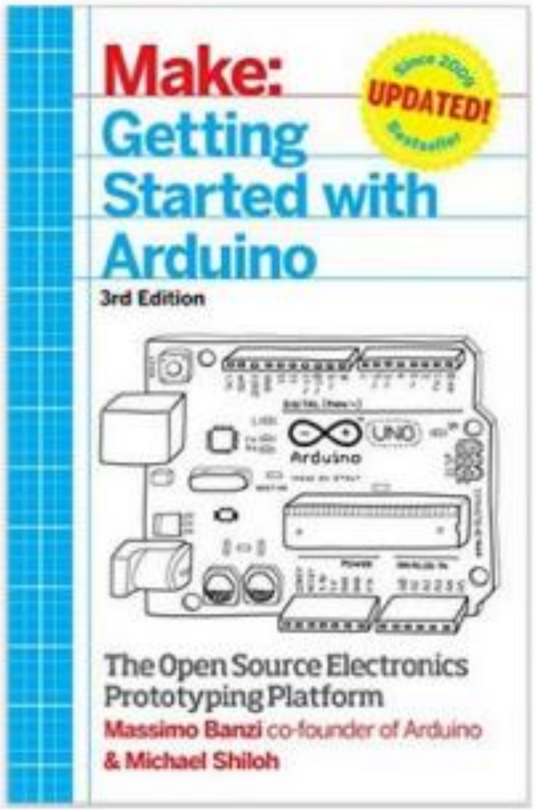


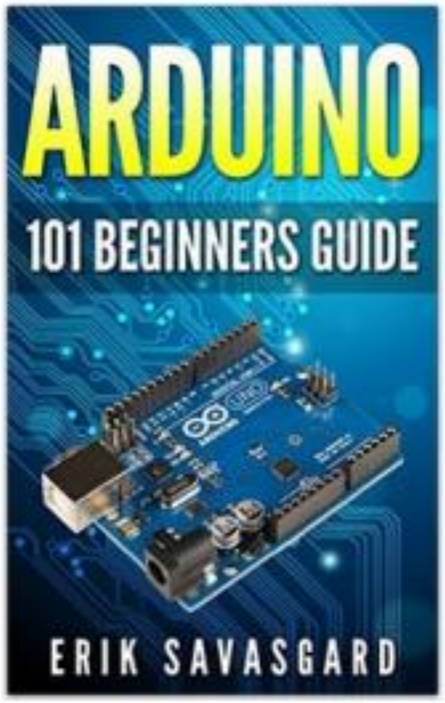
Speedometer with LCD

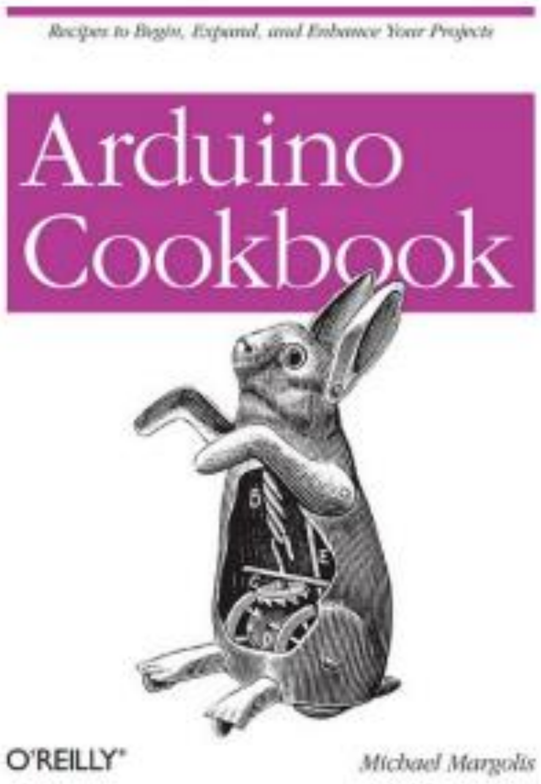


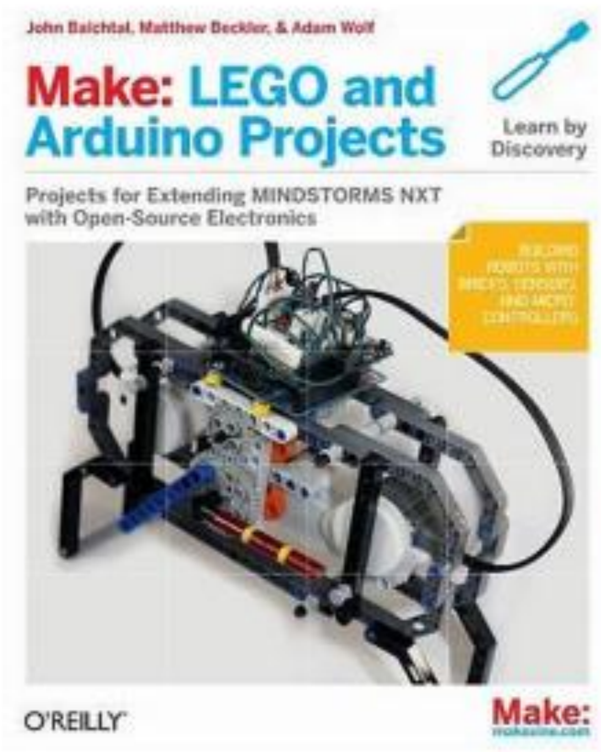
Get educated:

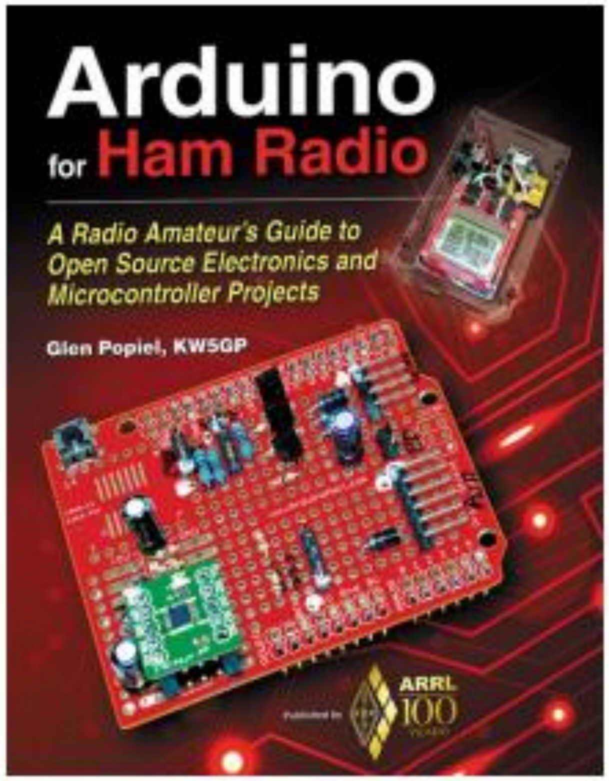



Make:
Getting Started with Arduino
3rd Edition
Since 2008
UPDATED!
Bestseller
The Open Source Electronics Prototyping Platform
Massimo Banzi co-founder of Arduino
& Michael Shiloh




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:
makezine.com


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

BOARDS (Compare Specs)

Arduino Uno

Arduino Leonardo

Arduino Due

Arduino Yún

Arduino Tre

Arduino Zero

Arduino Micro

Arduino Esplora

SHIELDS

Arduino GSM Shield

Arduino Ethernet Shield

Arduino WiFi Shield

Arduino Wireless SD

KITS

The Arduino Starter Kit

Arduino Materia 101

ACCESSORIES

TFT LCD screen

Arduino Mega ADK

Arduino Ethernet

Arduino USB Host Shield

USB/Serial Light Adapter

Arduino Mega 2560

Arduino Robot

Arduino Motor Shield

Arduino ISP

Arduino Mini

Arduino Nano

Arduino Wireless Proto Shield

Mini USB/Serial Adapter

LilyPad Arduino Simple

LilyPad Arduino SimpleSnap

Arduino Proto Shield

Advanced

Can't stand the difference between Verify and Upload and the fact that Upload verifies every time you need to program 25 boards?

Meet **arduino-builder**

C:\...\Arduino\TxHoMast_Test_Builder>**arduino-builder** -hardware="C:\Program Files (x86)\Arduino\hardware"
-tools="C:\Program Files (x86)\Arduino\tools-builder"
-tools="C:\Program Files (x86)\Arduino\hardware\tools"
-build-path="C:\Users\gert\Documents\Arduino\TxHoMast_Test_Builder\build"
-fqbn=a**rduino:avr:nano TxHoMast_Test.ino**

Now you also know where the .hex file is for upload next time: In the build-path folder!