...RRRduino...

Anyone said “ZoomTRAK smores”?

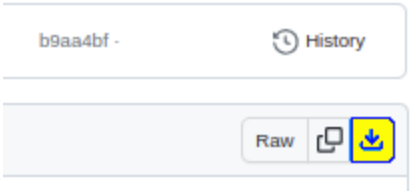
by Speed Muller *v0.02*

...RRRduino...

Anyone said “ZoomTRAK smores”?

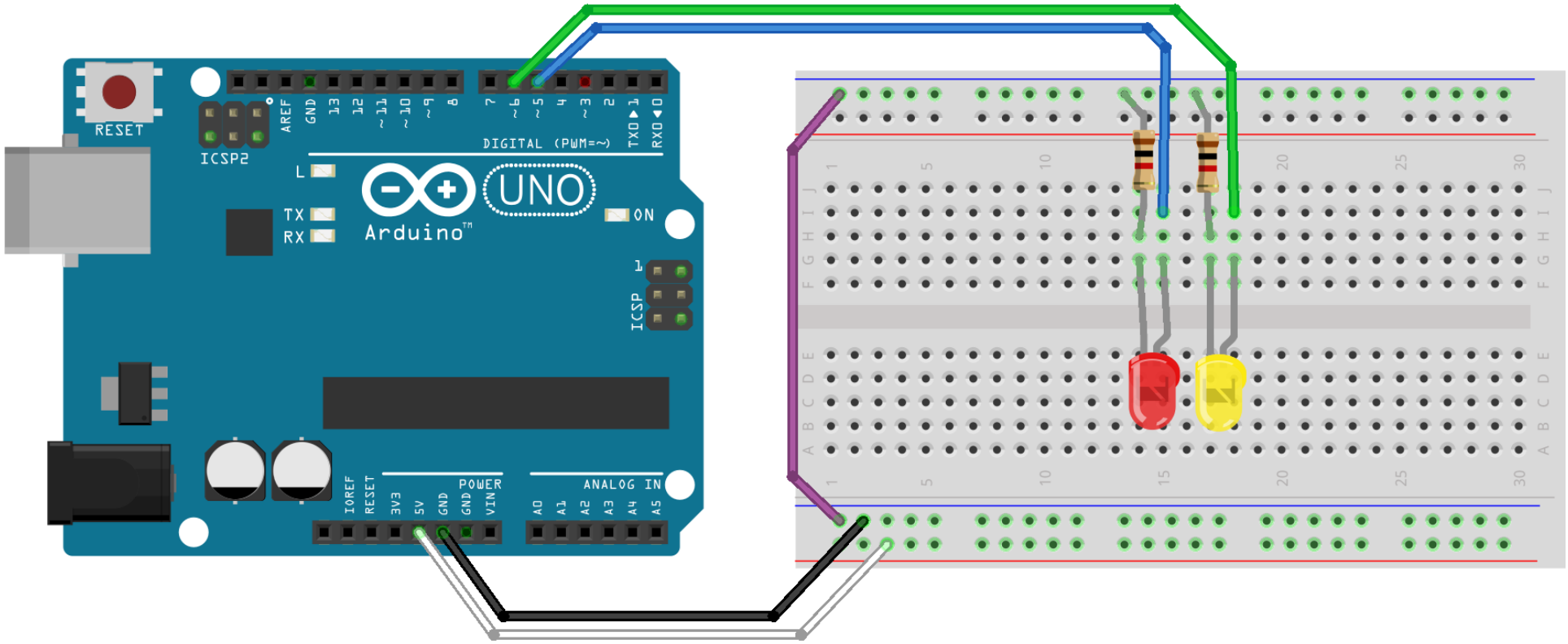
by Speed Muller *v0.02*

https://tinyurl.com/32n8fc8k



Todo, as we talk: Assemble / Build…

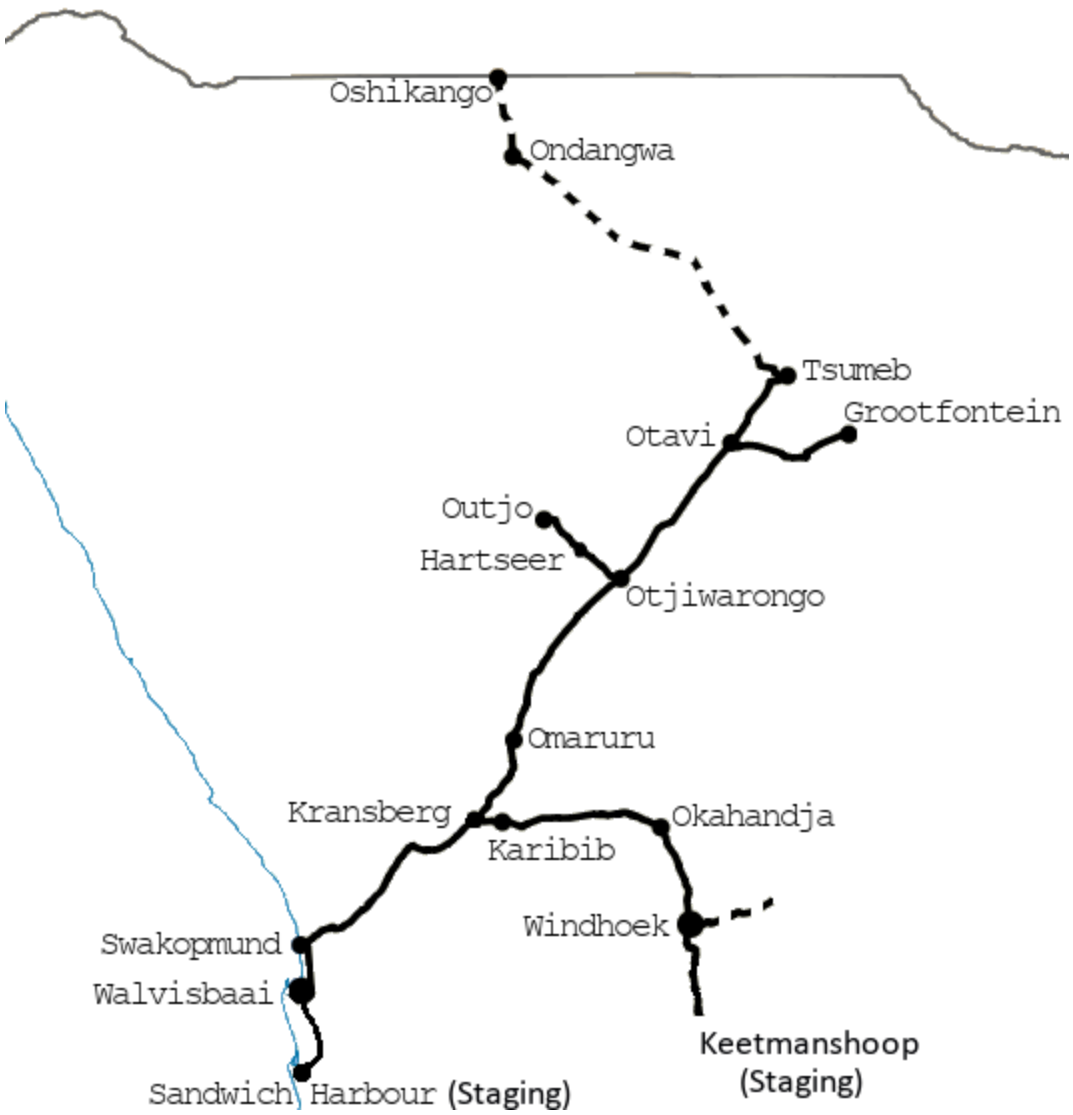
- Stick a **yellow** and **red LED** with the **flat side to the left** in the middle of the breadboard.
- Then a **blue wire from pin ~5** to the **red** LED and a **green wire from pin ~6** to the **yellow** LED's anodes (long pins for these specific devices, or the "round side")
- Then add **1 kOhm** resistors from the flat side to GND (blue breadboard rail)
- Install the **black** and **white** wires for power (5 Vdc to red rail with white wire and GND to blue rail with black wire)
- And connect a **purple** wire between the two blue breadboard rails



Namibia:


Lomé
Gulf of Guinea
Yaoundé
Equatorial Guinea
Uganda
Kampala
Kenya
Somalia
Nairobi
Rwanda
Burundi
DRC
Mombasa
Tanzania
Dar es Salaam
Luanda
Angola
Zambia
Malawi
Lusaka
Harare
Mozambique
Zimbabwe
Namibia
Botswana
Madagascar
Pretoria
Eswatini
Durban
South Africa
Cape Town
Gqeberha

Receive/Transmit → Rx/Tx: TransNamib → TxNamib (i)



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!

Key to the Clinic Slide Titles:

This is some information

Here is some more information

You have to do something…

And each step is likely in blue too!

Setup first!

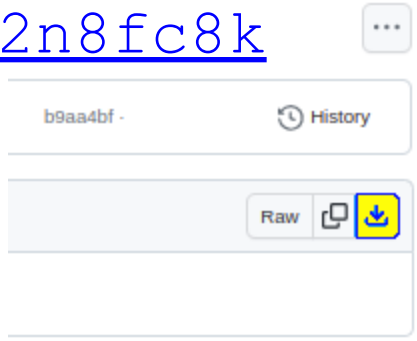
- Make sure everyone has the IDE installed on their computer (https://www.arduino.cc/en/software)

- Plug the Arduino Uno in and get a COM port to talk to.
- COM3, COM4 or other in Windows
- /dev/ttyACM0 or /dev/ttyUSB1 on “unix” derived systems, like Linux or /dev/cu.Serial on newer MacOS computers

- For the code in this series, go to:

https://tinyurl.com/32n8fc8k

- Download the zip file →



- Unzip the contents to your Arduino Sketch location (see File → Preferences)

IDE (*version 2.2.1 this time*):

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

Older: https://www.arduino.cc/en/software/OldSoftwareReleases

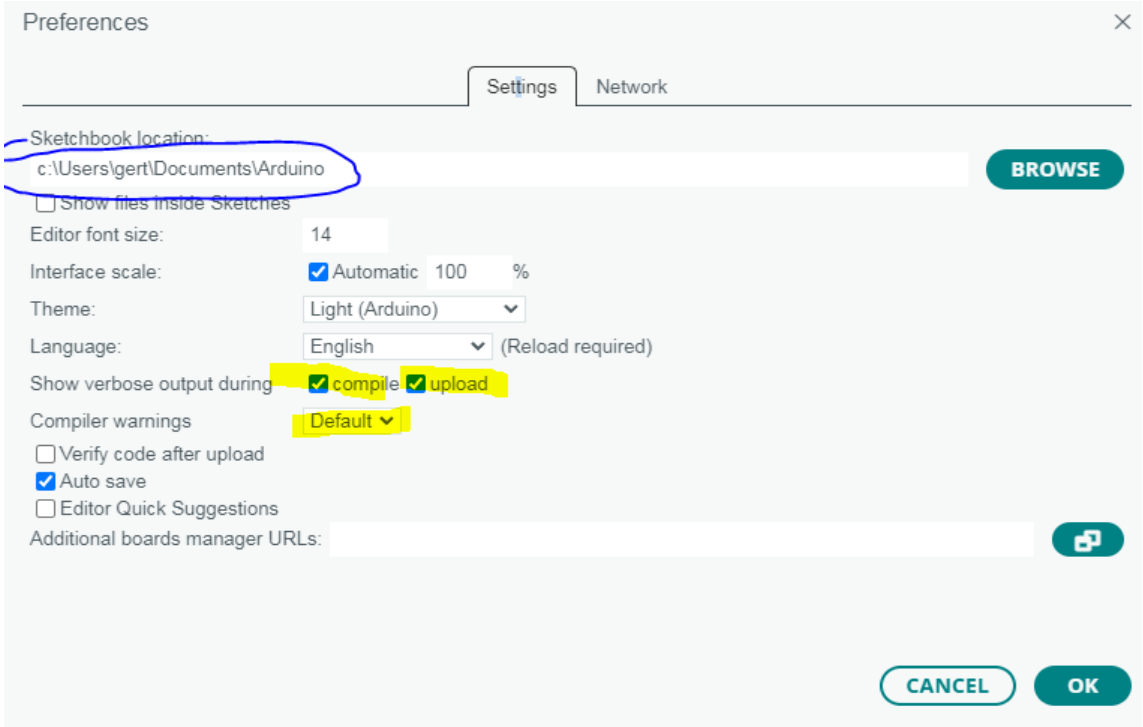
Just before we really start, set the IDE Preferences…

- Please open the File → Preferences (Ctrl+Comma) and set the following settings:

1. Show verbose output during: [x] **compilation** and [x] **upload**
2. **Change** Compiler warnings to: **Default**
3. And then make a mental note of **where** new Sketches will be saved on your computer. Usually something like

..\Documents\Arduino, which you need to know if you want to add a custom library, make a backup or share a Sketch.

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



Now for the Missing Introduction…

Because you can find this on the internet, we are going to skip over:

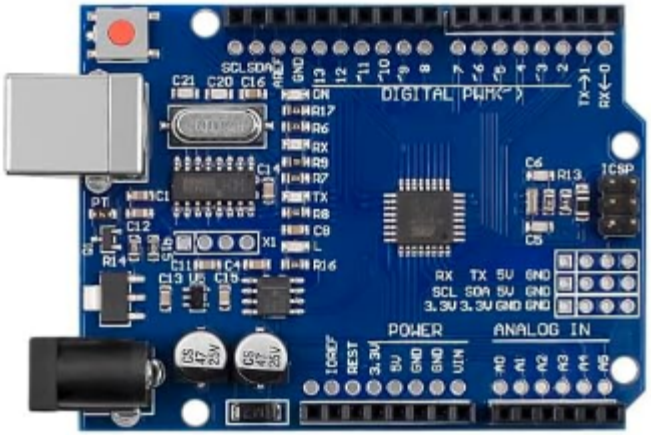
- "What is an Arduino?"
- "How did it come about?"
- "What is a microcontroller?"
- "How is a Raspberry Pi different?", or "Isn't the Pi Better?"
- "You will likely use Open Source software (Arduino IDE) as the tool"
- "You need one of the MANY choices to use as an Arduino platform"
- "You might like a Shield plugged into it to provide more functionality"
- "You need code: either write, copy, or edit to compile and upload onto the microcontroller"
- "And yes: It is just C/C++ code with 100s of libraries of code available"
  - *And if you disagree here: Please point us to a "library src folder" that does not contain a .cpp and/or .h file?*
  - *And explain why the compiler is avr-**gcc**/7.3.0-atmel3.6.1-arduino7/bin/avr-**gcc***

Also watch this: https://www.youtube.com/watch?v=BLrHTHUjPuw

So, where to start?

- Buy one ... and plug it in!

[PC → USB Cable → Uno]

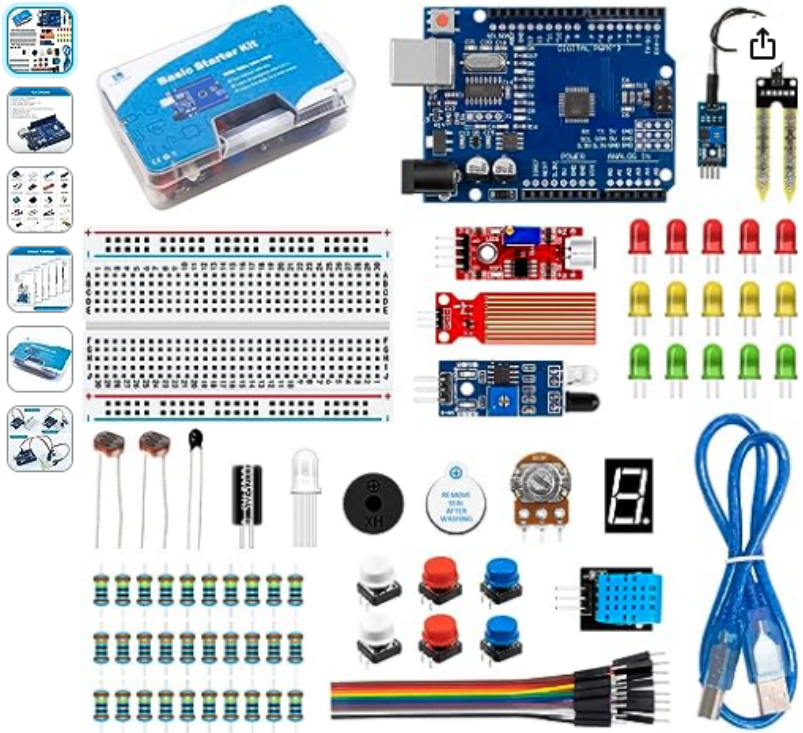


See a **solid** green* light and a **blinking** orange* light?

* colors may vary by vendor

Where did we get it?

https://a.co/d/8t39EeQ or https://a.co/d/23Lfms1



Roll over image to zoom in

LAFVIN Basic Starter Kit with R3 CH340,Breadboard + Retail Box Compatible with Arduino IDE

Visit the LAFVIN Store

4.0 ★★★★☆ 24 ratings

$15.99

Get **Fast, Free Shipping** with Amazon Prime

FREE Returns

Coupon: Apply $3 coupon Shop items › | Terms

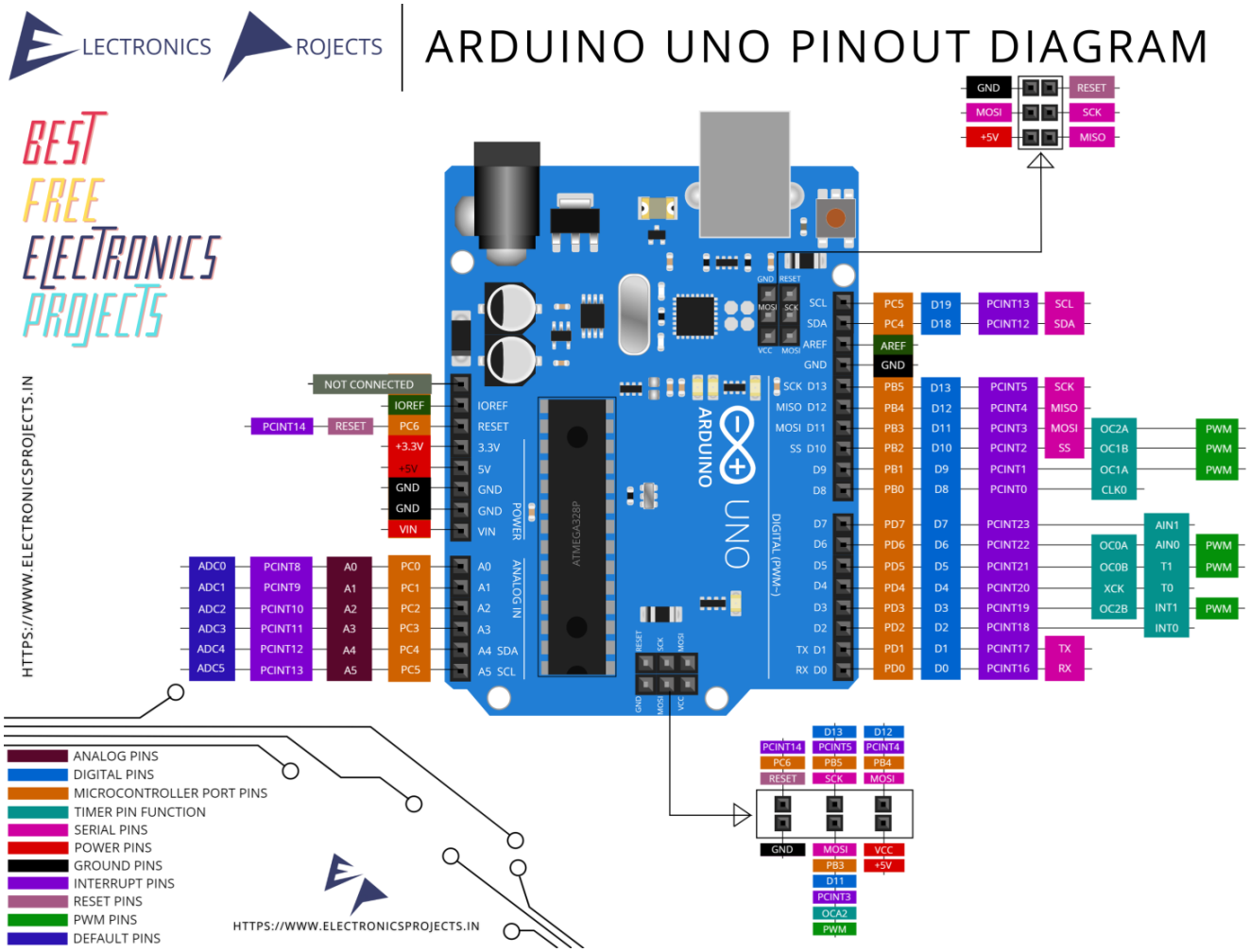
Get $50 off instantly: Pay $0.00 ~~$15.99~~ upon approval for Amazon Visa. No annual fee.

Brand	LAFVIN
Connectivity Technology	USB
Included Components	CD
Wireless Communication Standard	Bluetooth
Product Dimensions	10"L x 7.6"W x 3.5"H

About this item

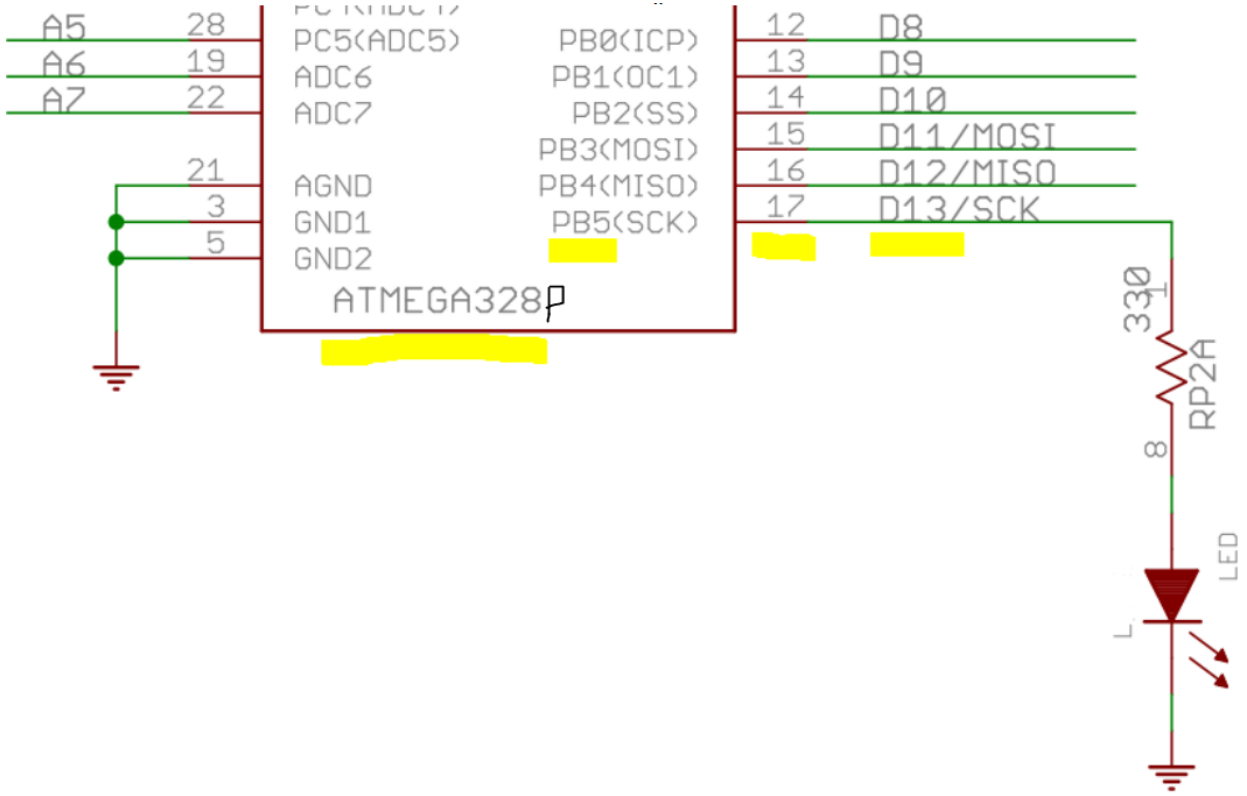
- The basic and cheap learning programming starter kit.
- All components compatible with IDE, can be used directly.
- 100% Compatible with program.
- Include High Quality R3 Controller Board,400 tie-points Breadboard,DHT11 Module,Soil Humidity Sensor ,LED, etc.
- A nice plastic packaging box is included for easy and neat storage.

And this is what we just plugged in:



- What we care about: the **brown** and **light blue** numbers in the 2nd columns on the left and right from the board, since those are the numbers our software code needs. Also written on your physical board.
- Brown numbers 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, on your board, have a wiggle, those can be **PWM** outputs...in simple terms, they could look like an Analog **output** using a resistor and capacitor as a filter.

ATMEGA328P Pin Ports vs Arduino Pin Numbers (when you get experienced)



- The Atmel ATMEGA328P-PU chip has pin 17 (labeled Port B bit 5 and Serial Clock) referred to as Arduino pin D13 … and hooked to an LED on the Arduino board. So if the code toggles Port B pin 5 (or 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 sink the current.

First example on the Uno

Uno still plugged in?

- Start the software 
- Tools → Board → Arduino AVR Boards → Arduino Uno // done only once
- Tools → Port → COMx (or /dev/ttyUSBx) // done only once
- File → Examples → 01.Basics → **Blink**

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

void loop( ) {
  digitalWrite( LED_BUILTIN, HIGH );
  delay( 1000 );  // 1,000 ms is 1 second
  digitalWrite( LED_BUILTIN, LOW );
  delay( 1000 );
} // loop( )
```

First Try…

- With File → Examples → 01.Basics → **Blink** loaded in the IDE…
- Press Ctrl+R (**Verify**) and note the stuff scrolling by in the bottom panel
- Now press **Upload** (Ctrl+U) // without knowing, you just compiled again!
- Notice **LED** on **D13** still blinking?

```
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\\\898EE7…708A9449F\\sketch\\Blink.ino.c
pp" -o "C:\\Users\\Speed\\...\\\898EE7…708A9449F\\sketch\\Blink.ino.cpp.o"
Compiling libraries…
…
… (and then)

Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for
local variables. Maximum is 2048 bytes.
```

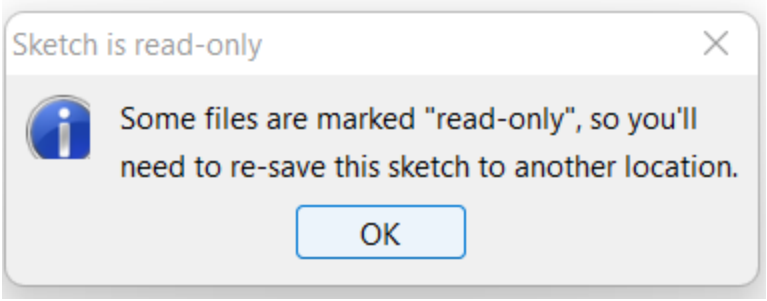
Embedded Software (Firmware) Engineer?

Change your **code** to match the red+yellow below

- Now press **Upload** (Ctrl+U)
  - 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 called **one**
  - C:\Users\<your name>\Documents\Arduino**one****one**.ino

- Notice the change in the LED on pin 13?

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

void loop( ) {
  digitalWrite( LED_BUILTIN, HIGH );
  delay( 250 );
  digitalWrite( 13, LOW );
  delay( 750 );
} // 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 bAreWeReady = 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 of the variable name: **u8**Value, **u16**Value or **i16**Value and **b**AreWeReady

This also helps with using CamelCase, a way to use English to name a variable with more descriptive words. The b, u8 or u16 is then followed by a capital letter.

Let's go!

- Open from the provided folders in Code/001_BlinkSlow_with_Serial/001_BlinkSlow_with_Serial.ino

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

#define LEDPIN      13    // LED_BUILTIN

// Comments
void setup( ) {
  Serial.begin( 115200 );
  Serial.println( );
  Serial.println( VERSION_STR );

  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 the **Serial Monitor** OR **changing baud** is like pressing the RESET button
- `And this is how you find out 3 years later what is on that 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 in which Windows version did we lose Hyper Terminal?

IDE saving a Sketch

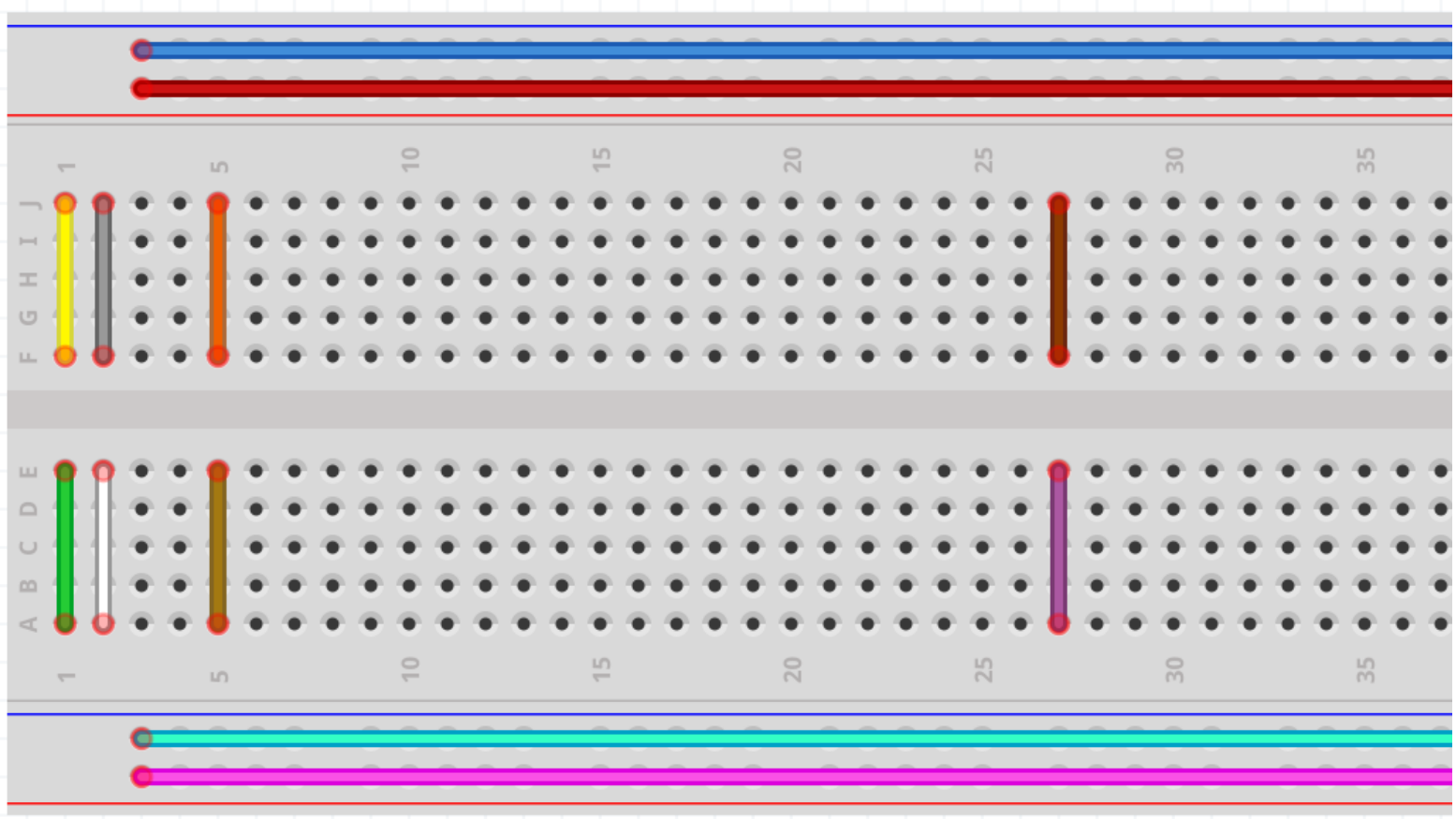
- A sketch is the .ino file in a folder with the same name
  - It is added to the build process when main.cpp is compiled
- Brand new sketch is named containing today's date
- No spaces in filename
- No ~~leading numbers or~~ funny characters
- main.cpp calls
  `setup( )` once, and
  `loop( )` over and over and over 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

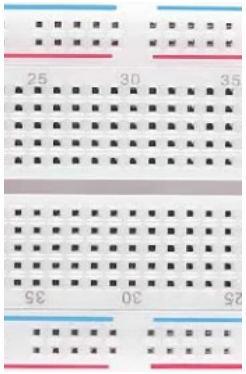
Next…

- Open **002_BlinkFaster.ino**

```
#define VERSION_STR    "002_BlinkFaster, ZoomTRAK, 2023.10.28"

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

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

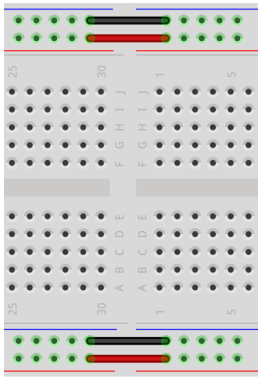
Breadboards, just a bunch of shorts!





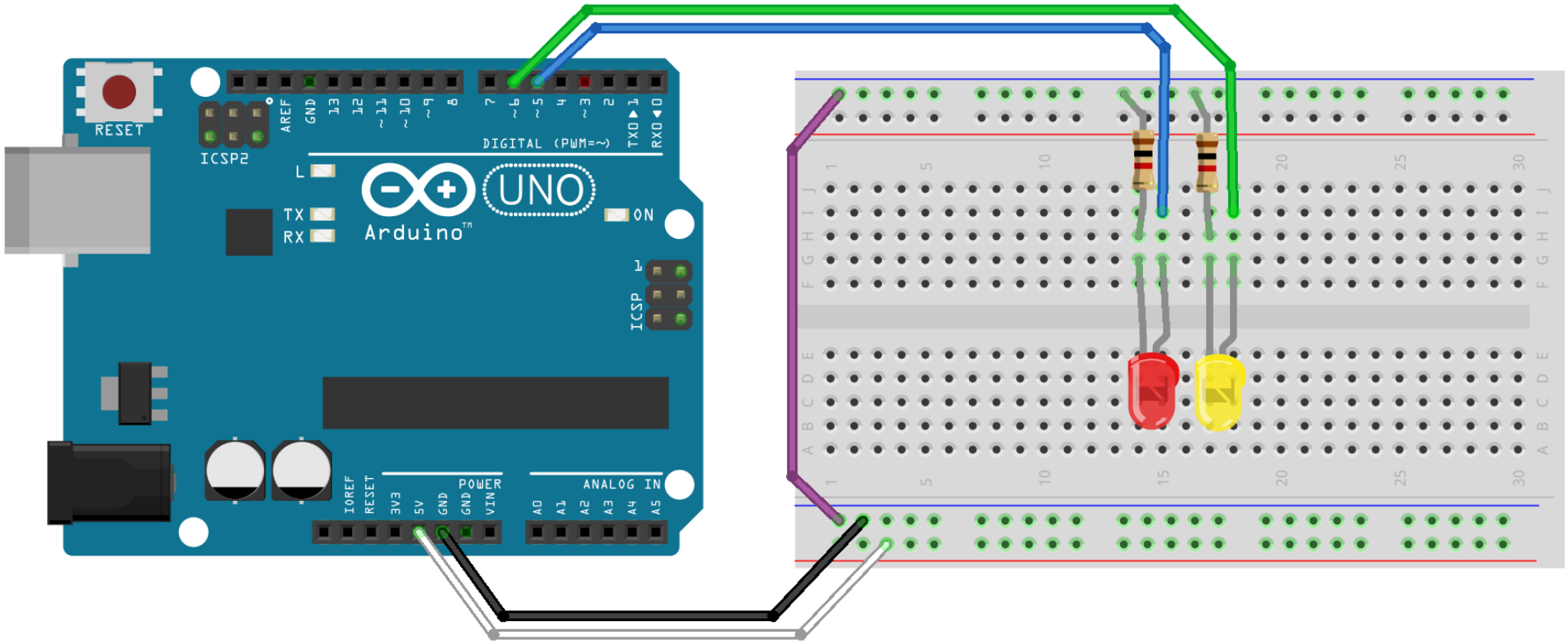
!!! Watch out for the ones with a gap in the blue and red bars:

They likely need to be “bridged” across the middle.



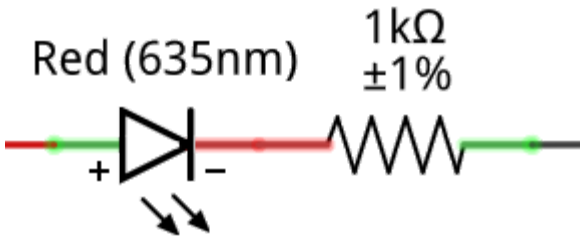
Assemble / Build…

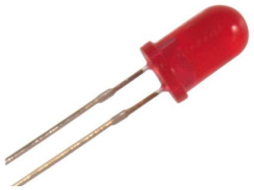
- Stick a **yellow** and **red LED** with the **flat side to the left** in the middle of the breadboard.
- Then a **blue wire from pin ~5** to the **red** LED and a **green wire from pin ~6** to the **yellow** LED's anodes (long pins for these specific devices, or the "round side")
- Then add **1 kOhm** resistors from the flat side to GND (blue breadboard rail)
- Install the **black** and **white** wires for power (5 Vdc to red rail with white wire and GND to blue rail with black wire)
- And connect a **purple** wire between the two blue breadboard rails



LEDs and **their** Resistors

An LED works like a **one-way street**. One-way streets allow cars 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 flows.





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) (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) and 2.8 V (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 trust the 2.1 Vf voltage, we want **5 - 2.1 V = 2.9 V** across the current limiting resistor. From V = I x R, and knowing the current through the LED is the same as the current through the resistor, we can choose to run the LED with some medium brightness at 3 mA:

R = V / I or R = 2.9 / 0.003 = **967 Ohms**. ← which smells 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;

More delays…

- Run **003_TwoBeaconBlink.ino**

More than 1 beacon on the Arduino now!

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

```
/*
 * Making two beacons work! Need external LED and resistor on pins 5 and 6
 */

// \__________________/----  ,  \__________________/----  ,  \____
// --\__________________/--  ,  --\__________________/--  ,  --\__
// x x               x x     ,  x x               x x     ,  x x

#define LEDPIN1          5
#define LEDPIN2          6

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

- Every delay( ) value needs to be calculated between each and 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, door bell.

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, layout is on for 5 hours only?)

- **We write down** the time (save it as “previous time”), and **check every so often** if a certain amount of time has passed since “previous time”

- If something needs to **repeat** on a periodic update the “previous time” with “now”, and keep checking if another **period** has expired.

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

```
#define VERSION_STR    "004_Millis, ZoomTRAK, 2023.10.28"

#define LEDPIN1           5
#define TIME1           500

// 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 )                          // FIND AND FIX THE ERROR !!!
  } // if
} // loop( )
```

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

```
#define VERSION_STR    "004_Millis, ZoomTRAK, 2023.10.28"

#define LEDPIN1           5
#define TIME1           500

// 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 );                              // BETTER!
  } // 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

It allows us to have a Heartbeat or Beacon keep track of its own variables, like **pin number**, the "**now**" 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
- And reuse the Beacon in any future project
- Don't fear, you were already using OOP with Serial.begin(115200) and Serial.println( )

Introducing: “Heartbeat”!

- Just another Beacon, we specify the **on** time and the **off** time
- 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, go!
- Sketch folders can also contain other files, the IDE will show source code like .h files
- Notice the New Tab (Ctrl+Shift+N) in the 3 dots menu at the top right
- We include our .h file with **#include** “file.h”, just like normal C/C++
- Note, a **Class** tells what the code can do, but an **Object** is created (with a **Constructor)** to do the work

Quick tour → What is in Heartbeat.h?

```
class Heartbeat {
private:
  uint8_t u8Pin;
  bool bPinHigh;
  unsigned long ulTimeoutOn;
  unsigned long ulTimeoutOff;
  unsigned long ulPrevious;

public:
  // Constructor
  Heartbeat( uint8_t u8ThePin, unsigned long ulTheTimeoutOn, unsigned
long ulTheTimeoutOff ) {
    u8Pin = u8ThePin;
    ulTimeoutOn = ulTheTimeoutOn;
    ulTimeoutOff = ulTheTimeoutOff;
    bPinHigh = false;
  } // constructor void Heartbeat( uint8_t, unsigned long, unsigned long )
```

```
// Set the pin mode and start low/off
void begin( ) {
  ulPrevious = millis( );
  pinMode( u8Pin, OUTPUT );  // output
  writeLo( );                // let's begin with off
} // begin ( )


// Write HI
void writeHi( ) {
  digitalWrite( u8Pin, HIGH );
  bPinHigh = true;
} // void writeHi( )


// Read and write the opposite
void writeLo( ) {
  digitalWrite( u8Pin, LOW );
  bPinHigh = false;
} // void writeLo( )
```

```
  // Check state, and then if it is time, and then toggle if so
  void update( ) {
    unsigned long ulNow = millis( );
    if( bPinHigh ) {
      if( ulNow - ulPrevious > ulTimeoutOn ) {
        ulPrevious = ulNow;
       writeLo( );
      } // if time
    } else {
      if( ulNow - ulPrevious > ulTimeoutOff ) {
        ulPrevious = ulNow;
        writeHi( );
      } // if time
    } // if HI
  } // void update( )
}; // class Heartbeat
```

Now for Beacon done by OOP!

- Using millis( ) as well inside the object's update( ) function

- Run **006_Beacon_and_Heart_OOP.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( )
```

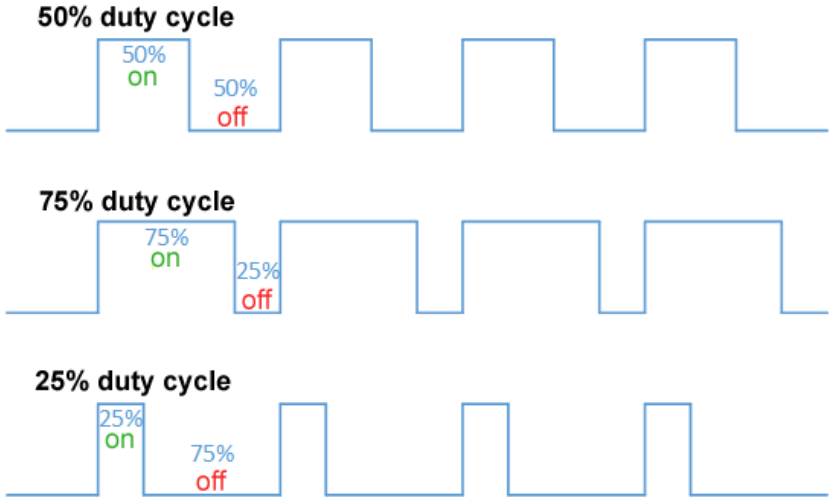
- How would you add a second beacon?

Well, pick an Arduino pin, add a resistor and LED and then add myBeacon2 with a pin and on and off times. myBeacon2**.begin**( ) and myBeacon2**.update**( )!

See 007_Two_Beacons_and_Heart_OOP.ino

What is PWM? (We need a dimmed LED)

- PWM pins can do Pulse Width Modulation



- Period (1 over frequency) and Duty cycle needed
- 100%: on, brightest on LED, 0%: off, dimmest possible; 50%: half on
- With a filter (in this LED case, your eyes) it appears as an analog output
- Much less expensive than connecting a D/A converter chip to the Uno
- Implemented on the Arduino with a simple analogWrite(pin, duty-cycle); command

(https://docs.arduino.cc/tutorials/generic/secrets-of-arduino-pwm)

Your eyes with flashing/changing lights

- Flicker Fusion Rate is the frequency of light that appears to be continuously on. It's usually given as ~25 Hz, but varies with intensity

- Your eye's rods: ~15 Hz and its cones: ~60 Hz

Arduino PWMs:

BOARD	PWM PINS	PWM FREQUENCY
Uno, Nano, Mini	3, 5, 6, 9, 10, 11	**490 Hz** (pins 5 and 6: 980 Hz)
Mega	2 - 13, 44 - 46	**490 Hz** (pins 4 and 13: 980 Hz)

see https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/ for more boards with Hz

analogWrite( )

Description

Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady rectangular wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite()) on the same pin.

You do not need to call pinMode() to set the pin as an output before calling analogWrite().

The analogWrite function has nothing to do with the analog pins or the analogRead function.

Syntax

```
analogWrite(pin, value)
```

Parameters

pin: the Arduino pin to write to. Allowed data types: int.

value: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: int.

Returns

Nothing

random( )

Description

The random function generates pseudo-random numbers.

Syntax

```
random( max )
    random( min, max )
```

Parameters

min: lower bound of the random value, inclusive (optional).
max: upper bound of the random value, exclusive.

Returns

A random number between min and max-1. Data type: long.

Campfire

- PWM for different brightness values, but also adding a random flicker
- Not fun with delay( ) when there is more than one thing happening!

```
// to do something periodic with random
while( 1 ) {
    x = random( 0, 100 );

    // do something
    delay( x );

    // do something else
    delay( 100 - x );
}
```

- Run **008_Campfire.ino**

Quick tour → What is in Campfire.h?

```
#define TRIGGERMAX 100
class Campfire {
  uint8_t u8Pin;
  uint8_t u8TriggerVal;
  uint8_t u8RandomMin;
  uint8_t u8RandomMax;
  unsigned long ulTimeout;
  unsigned long ulPrevious;

public:
  Campfire( uint8_t u8ThePin, unsigned long ulTheTimeout, uint8_t
u8TheTrigger = 127, uint8_t u8TheMin = 0, uint8_t u8TheMax = 255 ) {
    u8Pin = u8ThePin;
    u8 Trigger Value8 The u8TheTrigger;
    u8RandomMin = u8TheMin;
    u8RandomMax = u8TheMax;
    ulTimeout = ulTheTimeout;
  } // Campfire( uint8_t, unsigned long, [uint8_t], [uint8_t], [uint8_t] )
```

```
  void begin( ) {
    pinMode( u8Pin, OUTPUT );
    analogWrite( u8Pin, 0 ); // start with off
    ulPrevious = millis( );
  } // void begin( )

  void update( ) {
    unsigned long ulNow = millis( );
    if( ulNow - ulPrevious >= ulTimeout ) {
      ulPrevious = ulNow;
      if( random( 0, TRIGGERMAX ) > u8TriggerVal ) {
        analogWrite( u8Pin, random( u8RandomMin, u8RandomMax ) );
      } // if triggered
    } // if time
  } // void update( )
}; // class Campfire
```

- Every so **often**, when a random value is **bigger** than the **trigger**, write a new random value between **min** and **max** to the **PWM pin**.

Campfire by 2 lights

- Two of the same fires, but in different colors
- Run **009_Campfire_By_Two_Lights.ino**
- Now using pointers, so we could use them in an array if there where many

```
Campfire campfire1 = Campfire( CAMPFIRE1PIN, CAMPFIRE1TIMEOUT,
CAMPFIRE1TRIGGER, CAMPFIRE1MIN, CAMPFIRE1MAX ); // create the first
CampfireLite object using PWM pin 5

Campfire campfire2 = Campfire( CAMPFIRE2PIN, CAMPFIRE2TIMEOUT,
CAMPFIRE2TRIGGER,CAMPFIRE2MIN, CAMPFIRE2MAX ); // create the first
CampfireLite object using PWM pin 6
```

```
#define VERSION_STR "009_CampFire by Two lights, ZoomTRAK, 2023.10.28"


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

  campfire1.begin( );
  campfire2.begin( );
  myHeart.begin( );
} // setup

void loop( ) {
  campfire1.update( );
  campfire2.update( );
  myHeart.update( );
} // loop
```

analogRead()

Description

Reads the value from the specified analog pin. Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and the operating voltage(5V or 3.3V) into integer values between 0 and 1023. On an Arduino UNO, for example, this yields a resolution between readings of: 5 volts / 1024 units or, 0.0049 volts (4.9 mV) per unit.

On ATmega based boards (UNO, Nano, Mini, Mega), it takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

BOARD	OPERATING VOLTAGE	USABLE PINS	MAX RESOLUTION
Uno	5 Volts	A0 to A5	10 bits

Syntax

```
analogRead(pin)
```

Parameters

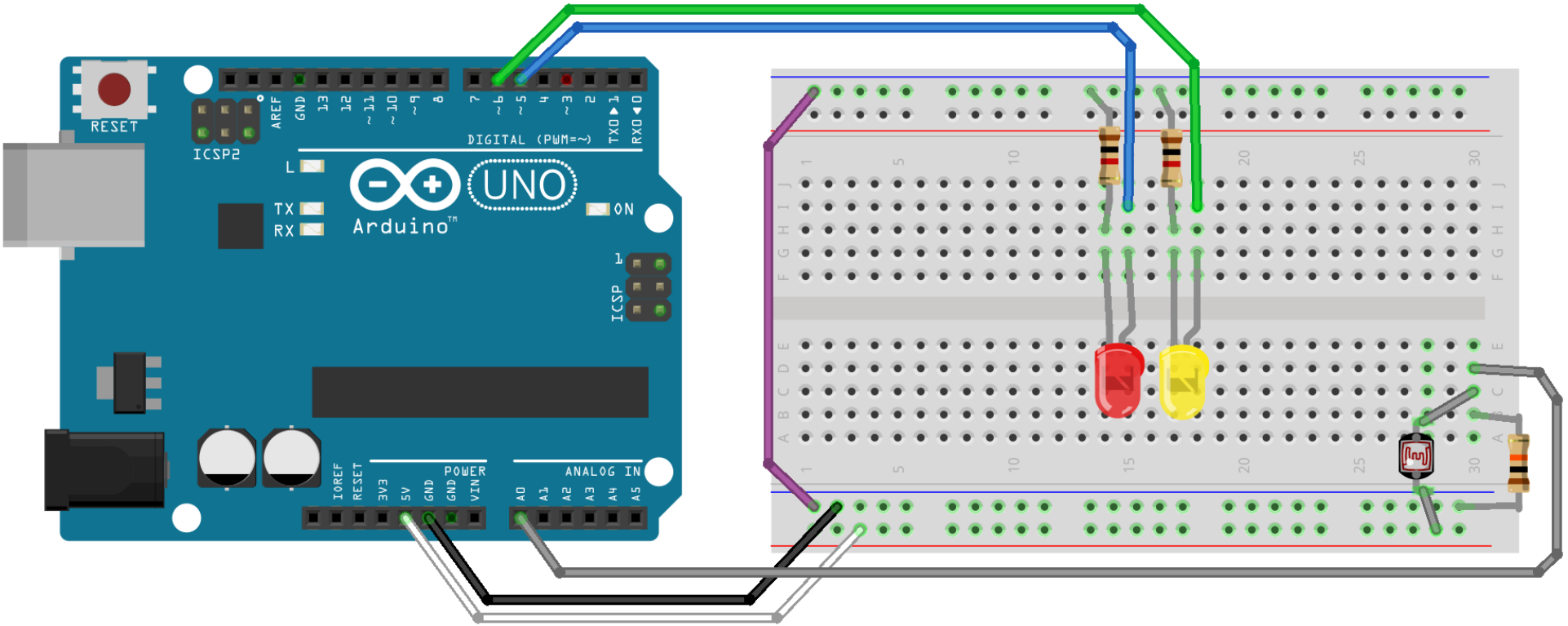
`pin`: the name of the analog input pin to read from (A0 to A5 on most boards, A0 to A6 on MKR boards, A0 to A7 on the Mini and Nano, A0 to A15 on the Mega).

Returns

The analog reading on the pin. Although it is limited to the resolution of the analog to digital converter (0-1023 for 10 bits). Data type: `int`.

Adding a Light Sensor…

- Install a **10** kOhm resistor to GND on the far right
- Install the Light Photoresistive Sensor between resistor and red bar, 5 Vdc
- Install a gray wire to pin A0 on the Arduino Uno from the Sensor



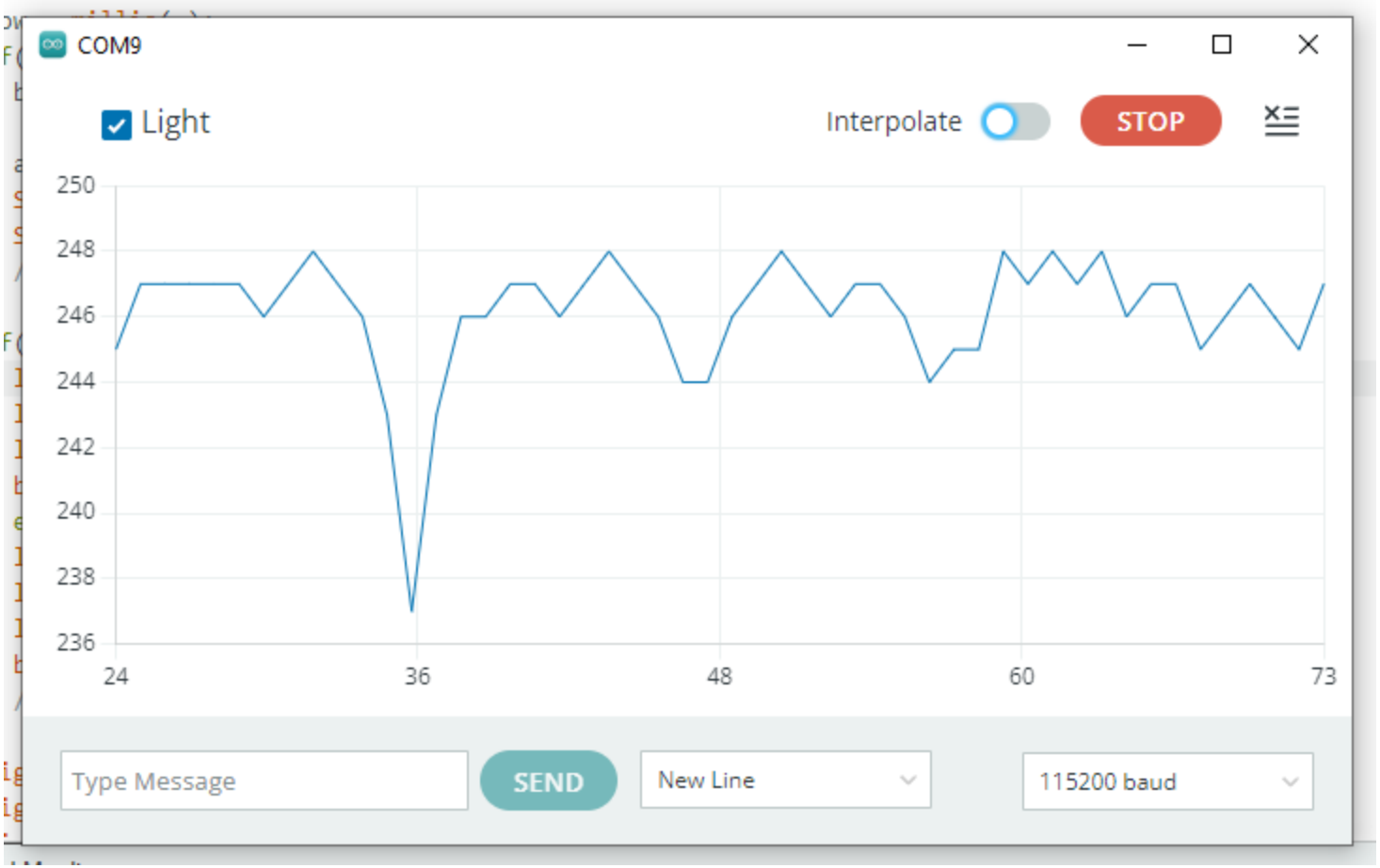
Light Sensor

- 10 bit Analog to Digital converter on ATMEGA328P → 0 - 1023

```
pinMode( ANAPIN, INPUT );

uint16_t u16AnalogValue = analogRead( ANAPIN );

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

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

- Run **010_Plot_Light**

Tools -> Serial Plotter



- Running **010_Plot_Light**
- Used to be Ctrl+Shift+L, not in 2.2.1; Now “scope” icon top right

Servos in general…

- A servo moves its “horn” to a position based on the width of a pulse on the control input.

- It will move straight (and fast) to the position demanded.

- It might use plenty of current (Amps) getting there.

- 1 ms to 2.5 ms wide pulses define the max range, typically in a 50 Hz (20 ms period) signal, for a 0 to 180 degree span.

- Continuous running Servos uses the same 1 to 2.5 ms range to run fastest one way, slow down and stop in the middle and run faster and faster in the opposite direction as the pulses increase in width.

- PWM capable pins on microcontrollers and inexpensive Arduinos allowed Model Railroads to use servos that used to be common in the more expensive Model Airplane and RC hobbies.

Install Servo Library

- ../Arduino/libraries, is where your libraries will go
- Sketch → Include Library → Add .Zip Library…
- Find and add servo-master.zip

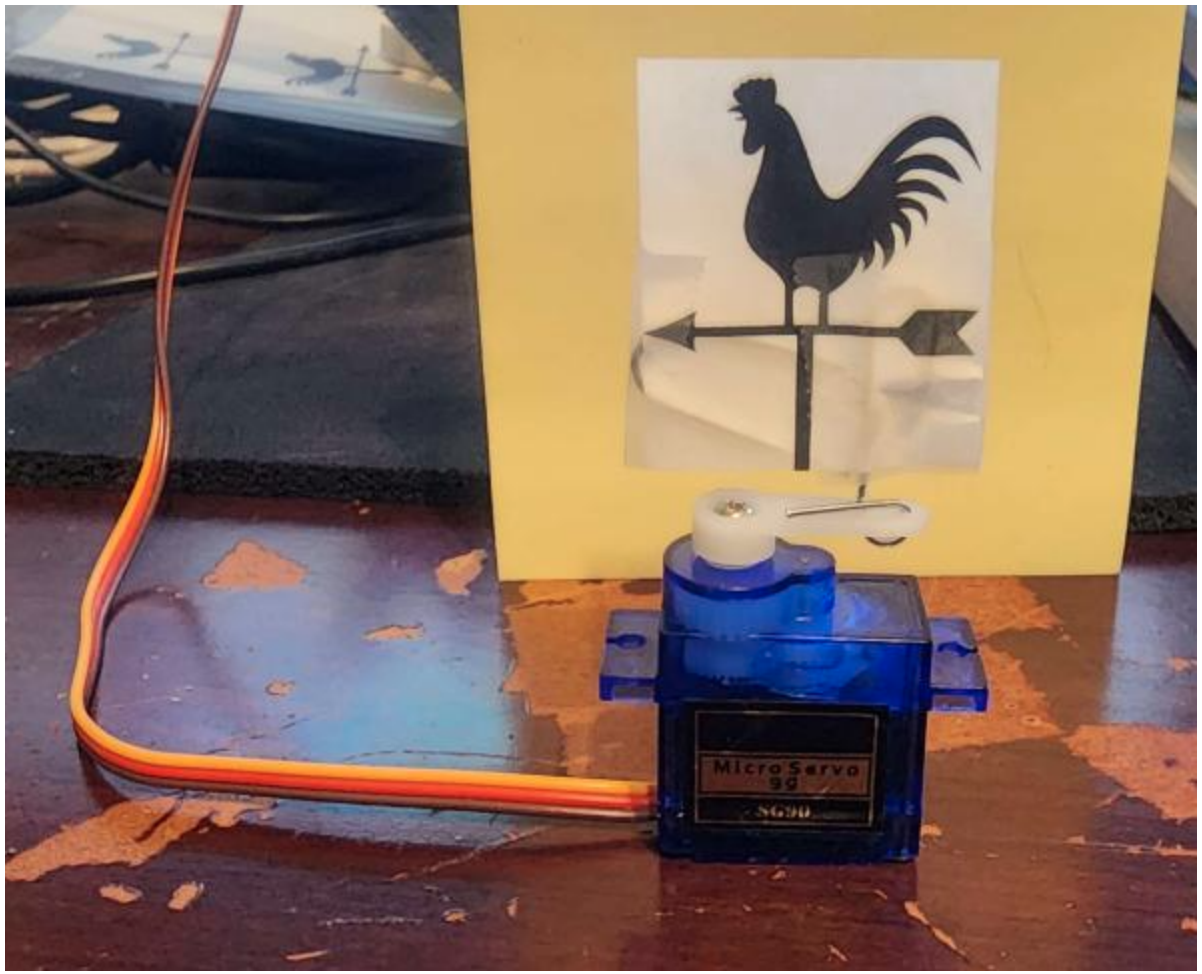
Servo.h

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

`#include <Servo.h>` gives us

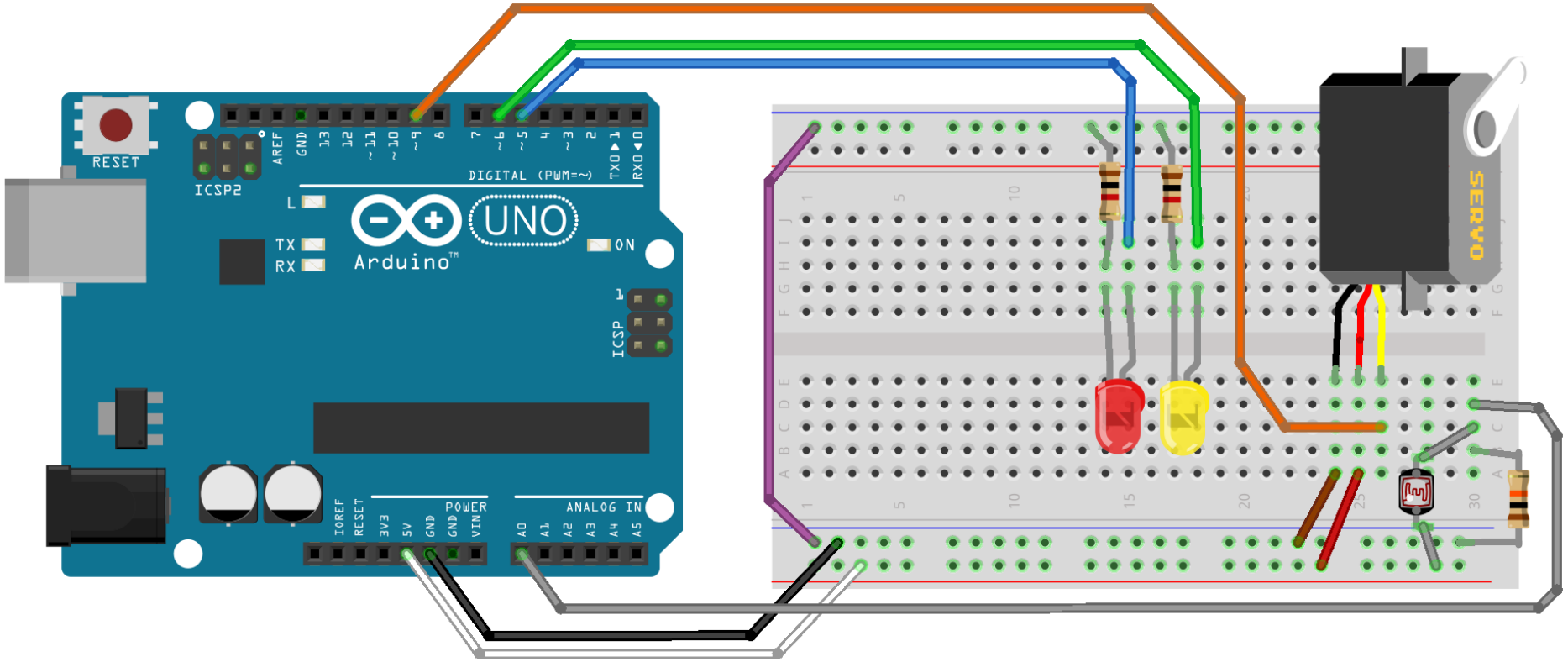
Methods

- attach( )
- write( )
- writeMicroseconds( )
- read( )
- attached( )
- detach( )



Connect our Servo…

- The servo is shown to be plugged into the breadboard, but **DON'T!**
- Plug the servo into the red, orange and brown wires, and then connect the orange wire to pin ~9, red to 5V and brown to GND.
- Run **011_Servo_by_Light.ino** and watch the Serial Plotter again



Fire only when dark

- Run **012_Campfire_By_Two_When_Dark.ino**

```
#define LIGHTLEVEL4FIRE 400

void loop( ) {
  …
  analogValue = analogRead( LIGHTSENSORPIN );
  if( analogValue <= LIGHTLEVEL4FIRE ) {
    startFire( );
  } else {
    stopFire( );
  } // if dark enough

  campfire1->update( );
  campfire2->update( );
…
```

And what you all really came for:

Smores when fire, only when dark!

- Run **013_Campfire_By_Two_When_Dark_With_Servo_Smores.ino**
- Servo does not move right away, delayed by TimeToUpdate

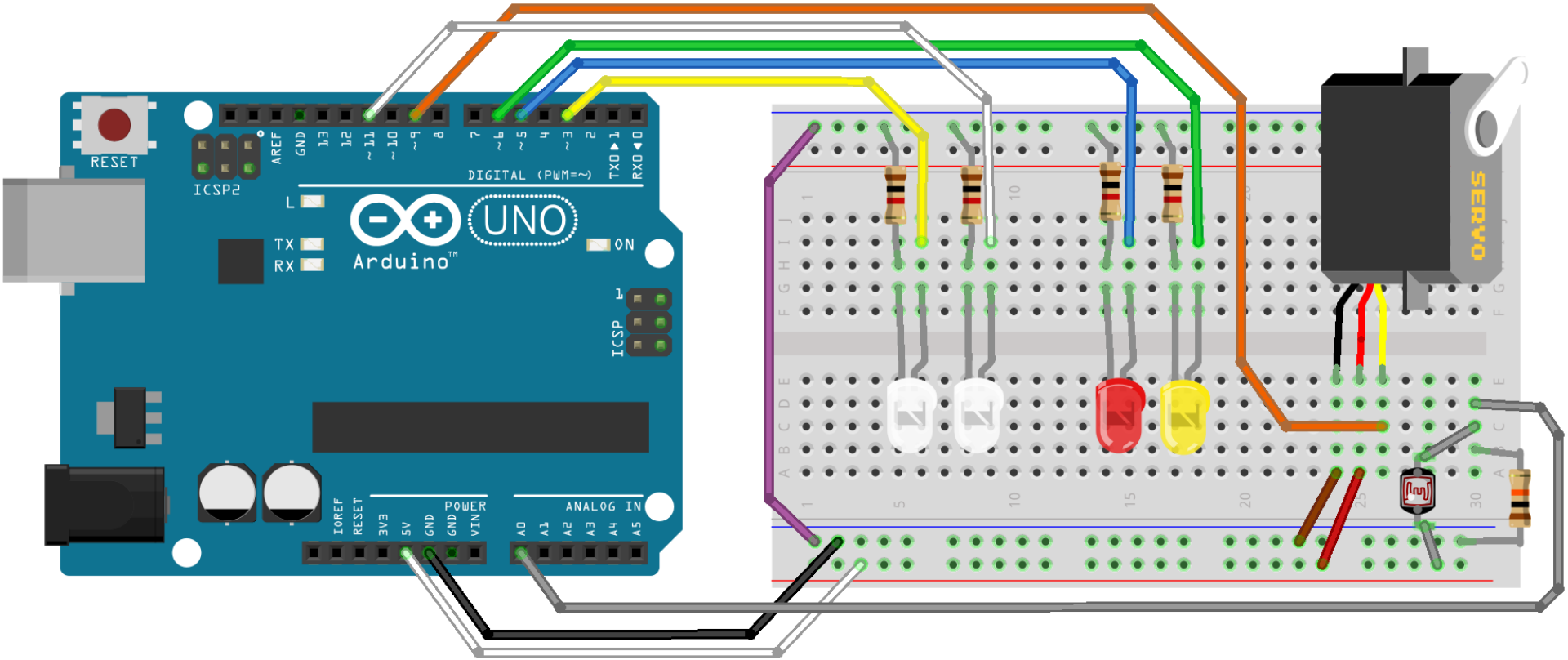
Quick Tour: what is in SmoreServo.h

```
  /* command( ): set the new position to go to */
  void command( uint8_t theCommandedPosition ) {
    u8CommandedPosition = theCommandedPosition;
  } // void command( uint8_t )

  /* isAtCommand( ): return true if at commanded position */
  bool isAtCommand( ) {
    return( u8Position == u8CommandedPosition );
  } // bool isAtCommand( )

/* output( ): move the servo if powered */
  void output( ) {
    if( bPower ) {
      myServo.write( u8Position );
    } // if bPower
  } // void output( )
```

```
  /* update( ): from the current position, move closer to the commanded position if different */
   void update( ) {
     ulNow = millis( );
     if( ulNow - ulBefore >= u16Timeout ) {
       ulBefore = ulNow;
       if( u8Position == u8CommandedPosition ) {
         // do nothing
       } else {
         if( u8Position < u8CommandedPosition ) {
           u8Position++;
           output( );
         } else {
           u8Position--;
           output( );
         } // decrease
       } // not there yet
     } // if time to update
   } // void update( )
}; // class SmoreServo( )
```

Fireflies

- Two white LEDs with 1 kOhm resistors added. Resistor and LED flat side towards GND
- Yellow and White wires to pins ~3 and ~11 at Uno



And the fire fades out at the end!

- Run **014_Camping_All.ino**
- Keep track of **u8Flickering** to reduce to zero and reduce

u8RandomMaxUsed

```
if( ( u8Flickering < 255 ) && ( u8Flickering > 0 ) ) {
  u8Flickering--;
  if( u8RandomMaxUsed > u8RandomMin ) {
    u8RandomMaxUsed--;
  } // if room to decrease
} // if Flickering
```

- Conditional compile:

```
#define TWOFIREFLIES

#ifdef TWOFIREFLIES
FireFly *myFly2 = new FireFly( FIREFLY2PIN, FLYOFFDELAYMIN, FLYOFFDELAYMAX, FLYMIN );
#endif
```

- Three states (disabled, ready(on) or fading)

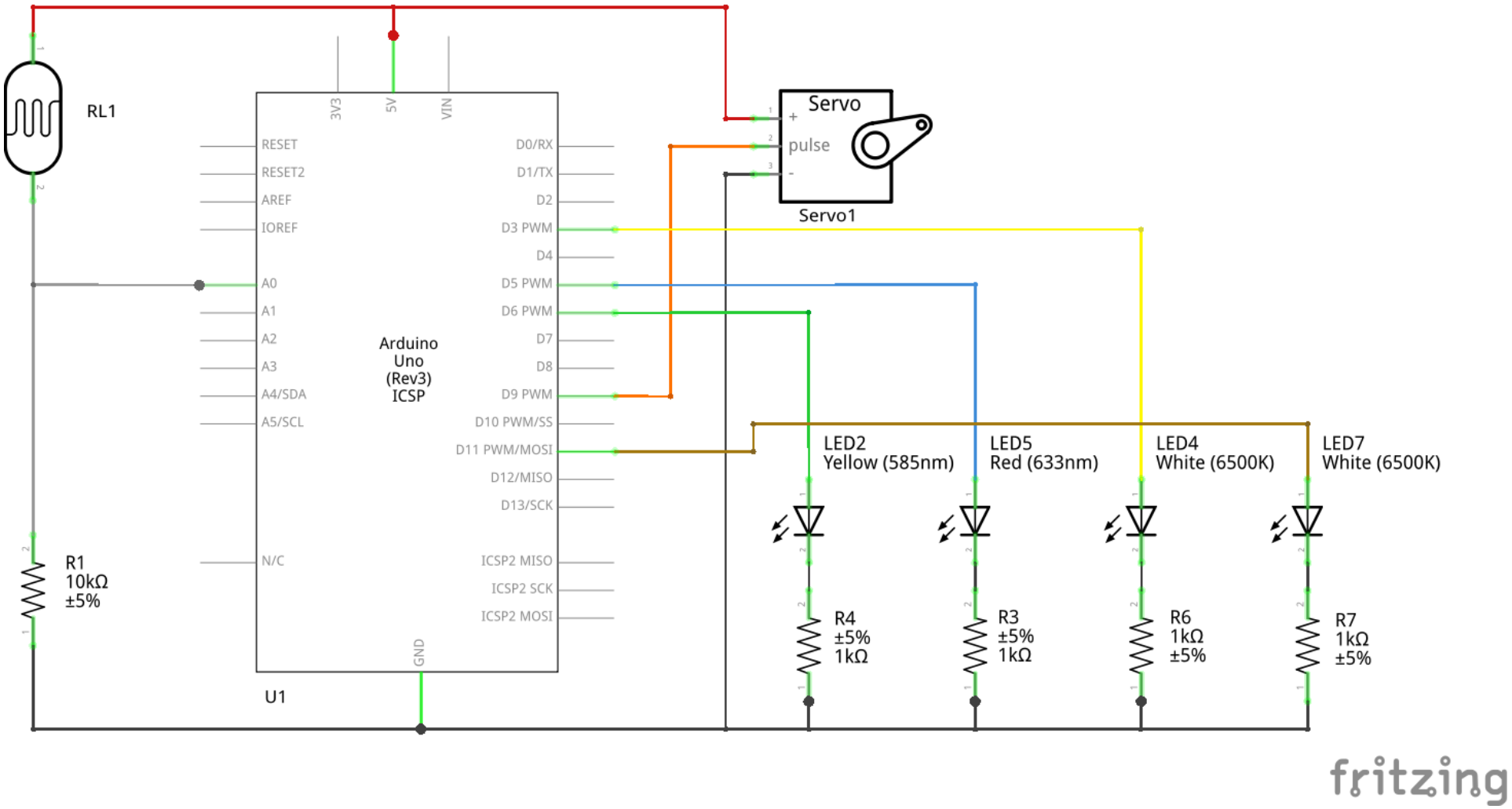
```
#define DISABLED      0
#define FADING      254
#define READY       255

#define MAXVAL      255
#define UPDATETIME    5
#define DECREASEBY    3

  // Set the pin and time up
  void begin( ) {
    calcNextRandom( );
    pinMode( u8PWMPin, OUTPUT );
    writeLo( ); // let's begin with off
    ulPrevious = millis( );
  } // begin ( )

  // output PWM value to ~pin
  void output( ) {
    analogWrite( u8PWMPin, u8OutputVal );
  } // void output( )
```

```
// Check fading, if zero: calculate next random time
// If output is zero calculate next random brightness
void update( ) {
  unsigned long ulNow = millis( );

  if( u8Mode == FADING ) {
    if( ulNow - ulPrevious > UPDATETIME ) {
      ulPrevious = ulNow;
      if( u8OutputVal > DECREASEBY ) {
        u8OutputVal -= DECREASEBY;
      } else {
        u8OutputVal = 0; // zero if close enough
      } // if able to decrease else off
      output( );
      if( u8OutputVal == 0 ) {
        calcNextRandom( );
        u8Mode = READY;
      } // if off
    } // if time to update
```

```
    } // if mode decreasing (FADING)
    if( u8Mode == READY ) {
      if( ulNow - ulPrevious > ulNextOnDelay ) {
        ulPrevious = ulNow;
        u8Mode = FADING;
        u8OutputVal = random( u8RandomMin, MAXVAL );
        output( );
      } // if time
    } // if mode READY
  } // void update( )
```

Sketch uses 6458 bytes (20%) of program storage space. Maximum is 32256 bytes.
Global variables use 414 bytes (20%) of dynamic memory, leaving 1634 bytes for local variables. Maximum is 2048 bytes.

Schematics


RL1
3V3
5V
VIN
RESET
RESET2
AREF
IOREF
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
ICSP2 MISO
ICSP2 SCK
ICSP2 MOSI
GND
Arduino
Uno
(Rev3)
ICSP
U1
Servo
+
pulse
-
Servo1
R1
10kΩ
±5%
LED2
Yellow (585nm)
LED5
Red (633nm)
LED4
White (6500K)
LED7
White (6500K)
R4
±5%
1kΩ
R3
±5%
1kΩ
R6
1kΩ
±5%
R7
1kΩ
±5%
fritzing

Rest of the kit

- Obstacle avoidance module? Occupancy detection?
- Soil Humidity Sensor? Run trains when it's raining outside?
- Sound sensor?
- Water level detection?
- 7 segment display? 7 more LEDs?
- DHT11, humidity and temperature?
- Buzzers?
- Potentiometer?
- Button switches?
- Tilt switch?
- Thermistor?
- RGB LED? Welder?

https://www.dropbox.com/sh/bies3jplqb5wm9s/AABZWa1Pwr_TH3t5yEkWoi6qa?dl=0

What would you like it to do?

Power

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

- Computer's USB is typically not enough when you turn 3 servos on at the same time!

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

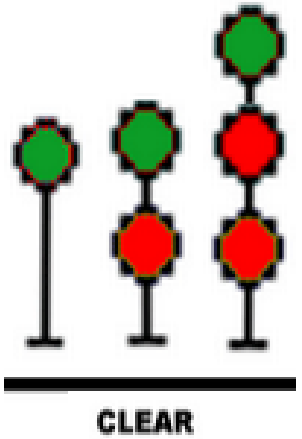
- Use dates and English words as much as you can. Easier to find and search later
- Speed likes YYYY.MM.DD HH:MM, 2023.08.21 14:15

- More 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

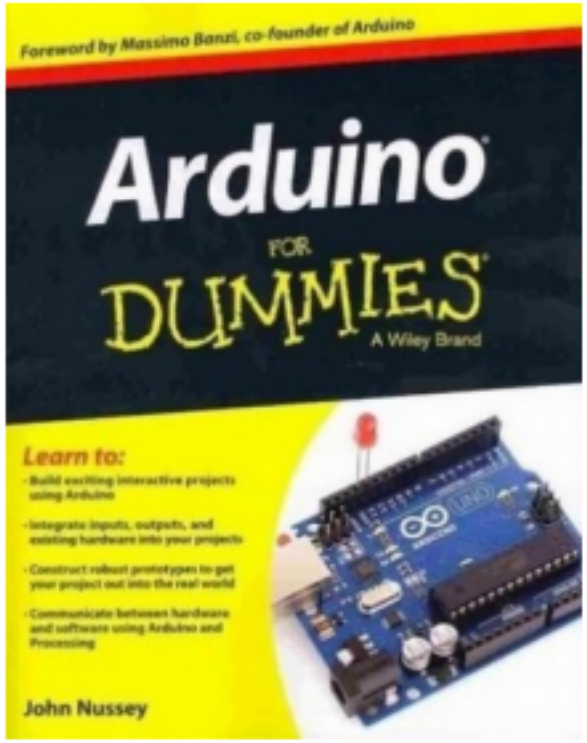


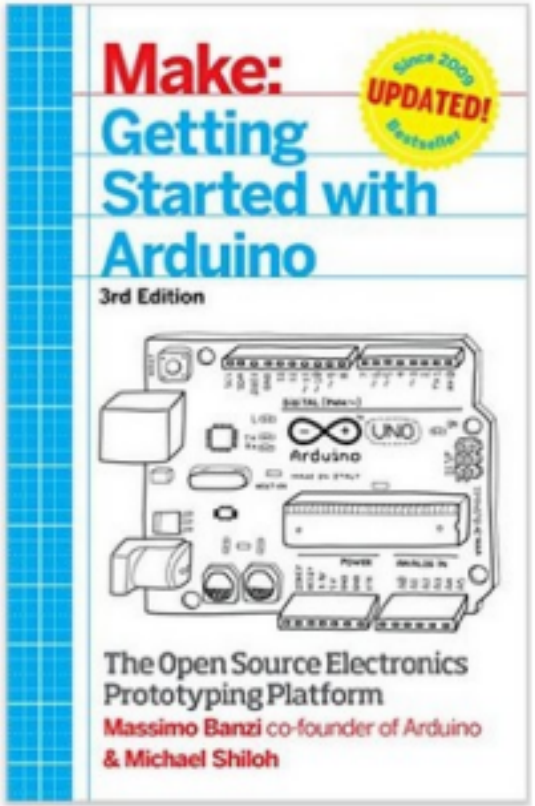
Thank you to Joel, David, Riley, *Bob, Donna, John, Stephanie, Alissa, Bianka, my Boss,*
and all the others that did not bother us while we were having fun doing this…

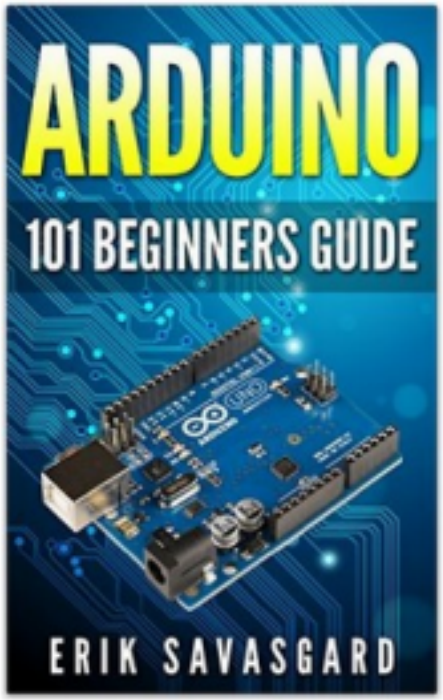
Get educated:

Watch this: https://www.youtube.com/watch?v=BLrHTHUjPuw

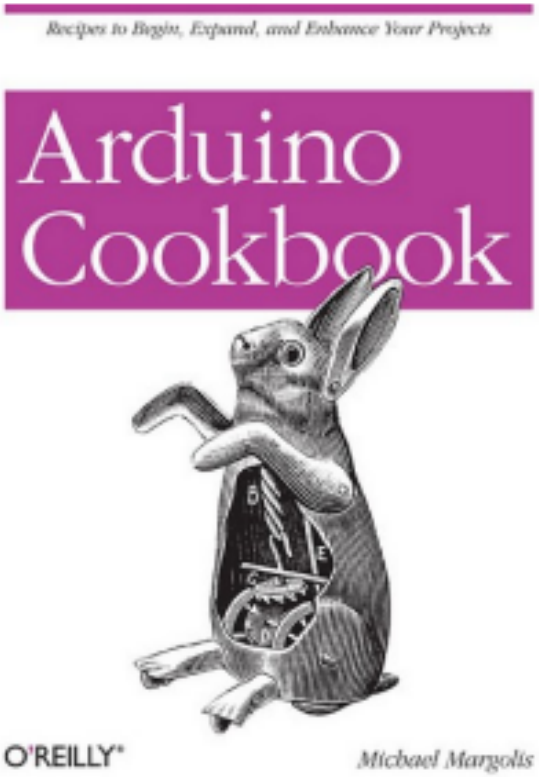
Read any of these:

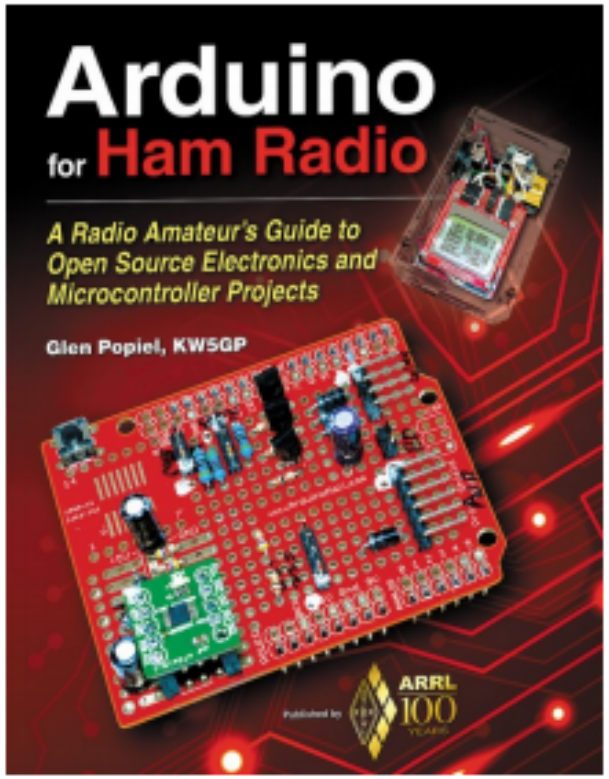


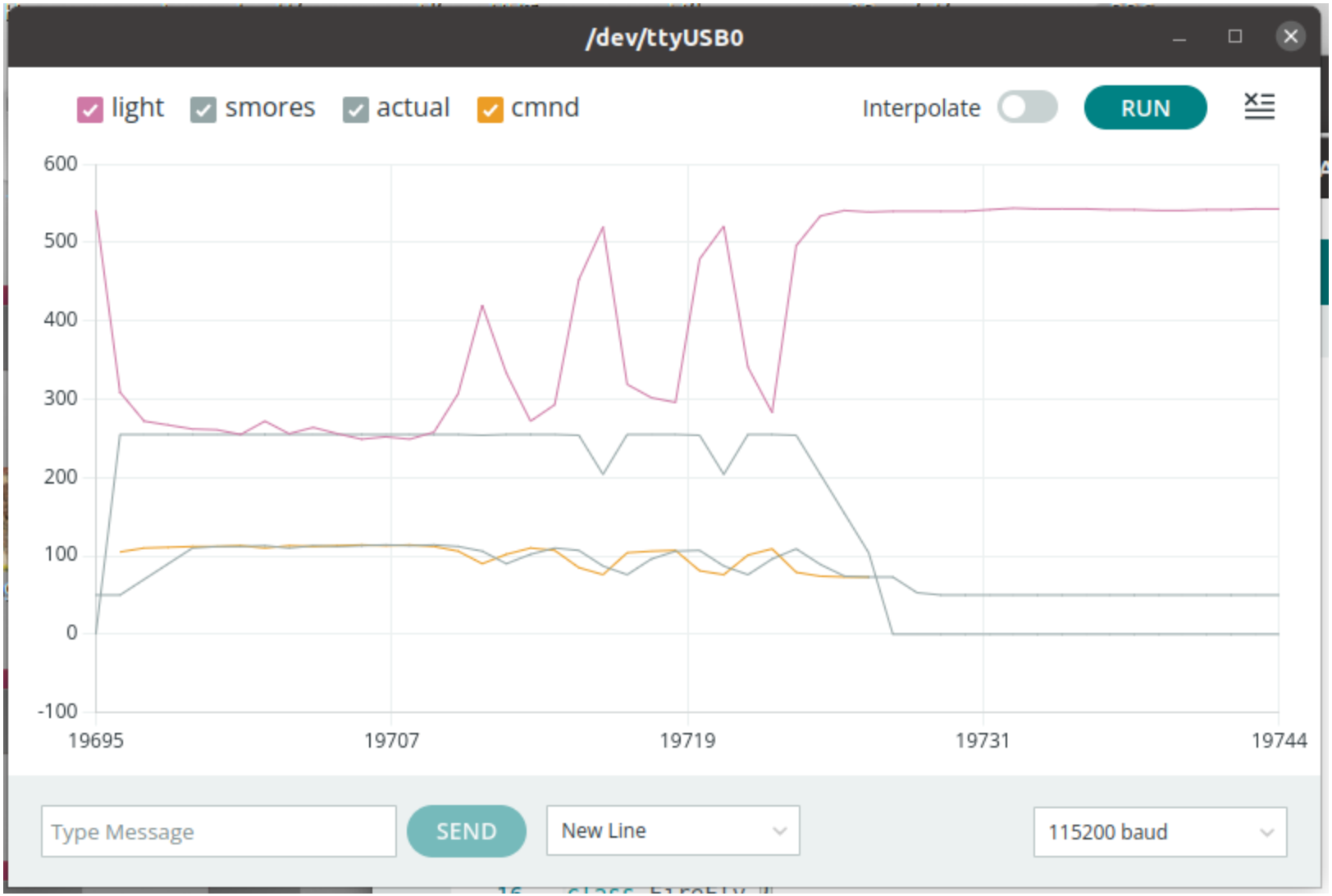




And get some confidence and jump in here:







Changes:
v0.01, original
v0.02, smaller breadboard, only wires needed, tinyURL to github.