Microcontrollers

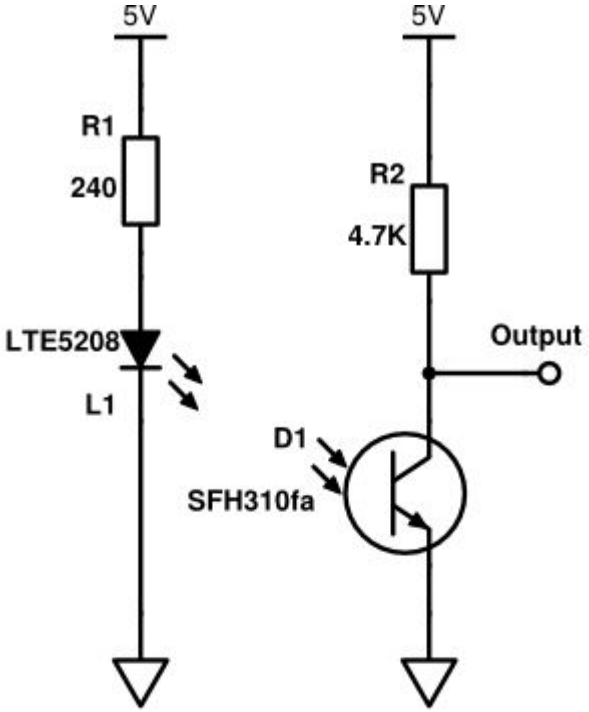
An introduction to microcontrollers through the arduino nano

Picking the right tool for the job

- Embedded system: An electronic system consisting of inputs and outputs that performs a specific role in a larger device
- Useful in measurement devices, tools and other stand alone devices
- Different types of electronic devices used
  - Circuitry
  - FPGA
  - Microcontroller
  - Single Board Computer
- Often combined in a single system if different functionalities needed

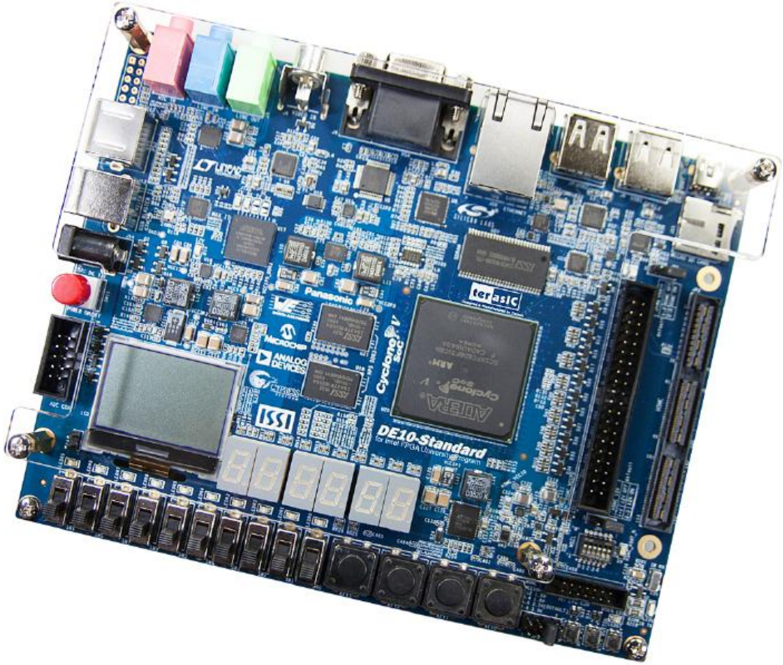
Circuitry

- Mostly Used for analog signals
- Nanosecond response times
- Very difficult to change
- Useful when interfacing with other devices



Field Programmable Gate Array (FPGA)

- Interface analog and digital
- Nanosecond response times
- Very high barrier to entry (cost)
- Easier to change

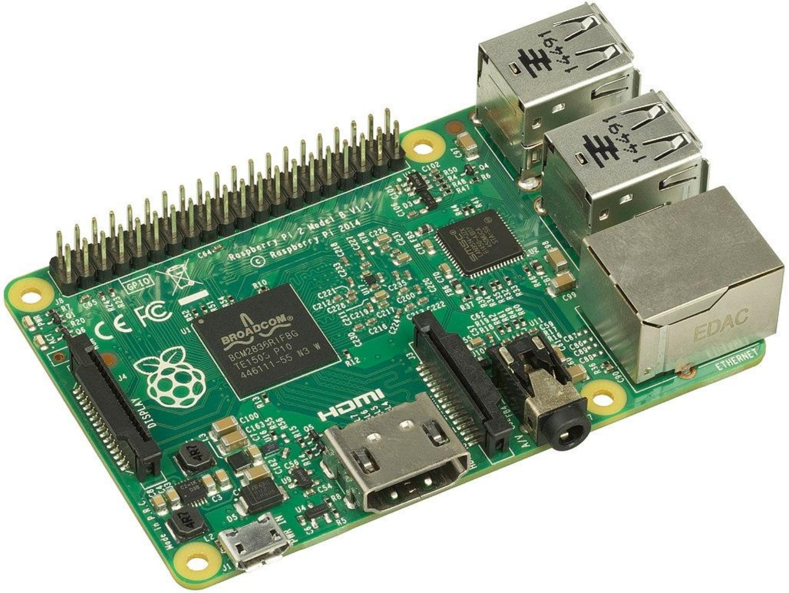


Microcontroller

- Interface analog and digital
- Microsecond response times
- Low barrier to entry
- Easily changeable (programmable)

Single Board Computer

- Most often only digital
- Millisecond response times
- Uses operating system
- Often used for higher level systems (servers, monitor displays, cameras)



Spectrum of Microcontrollers

Attiny85

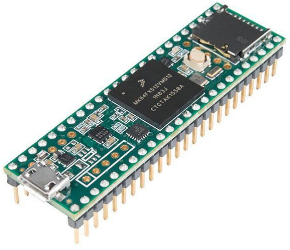
- 5 I/O
- 3.3-5V
- 16Mhz
- 8k Flash
- 0 Serial
- $1.50

Arduino Nano

- 22 I/O
- 5V
- 16Mhz
- 32k Flash
- 1 Serial
- $5.49

Arduino Mega

- 54 I/O
- 5V
- 16Mhz
- 256k Flash
- 4 Serial
- $21.99

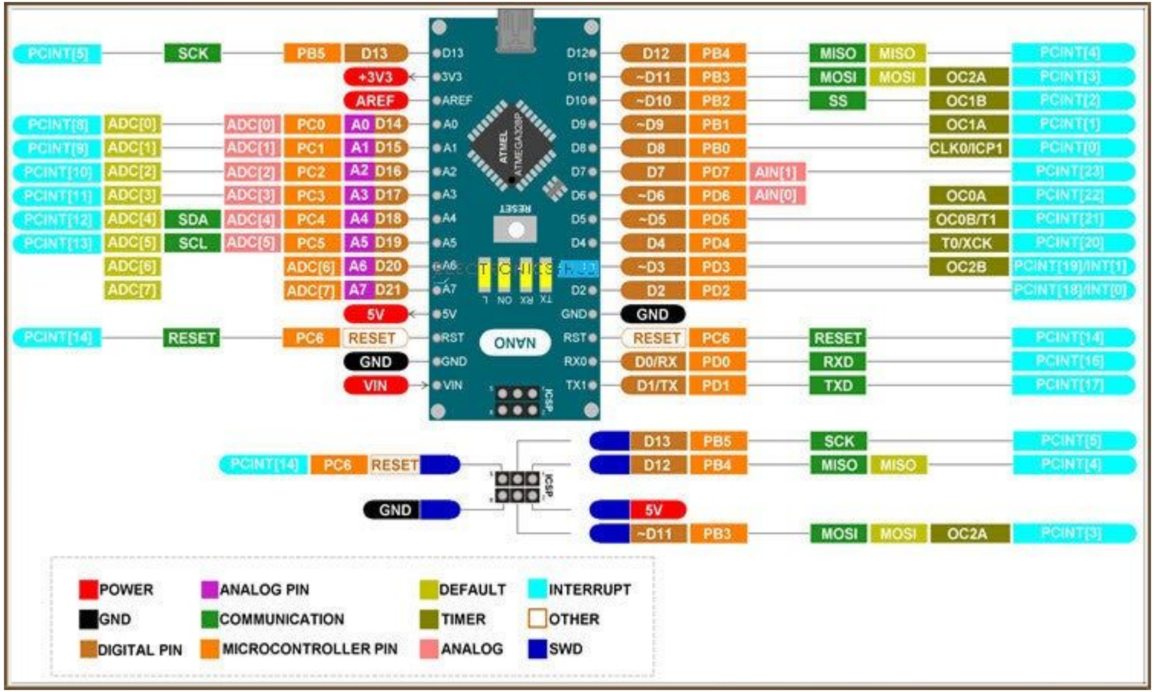


Teensyduino 4.1

- 55 I/O
- 3.3V
- 600Mhz
- 8M Flash
- 8 Serial
- $31.50

Anatomy of the Arduino Nano

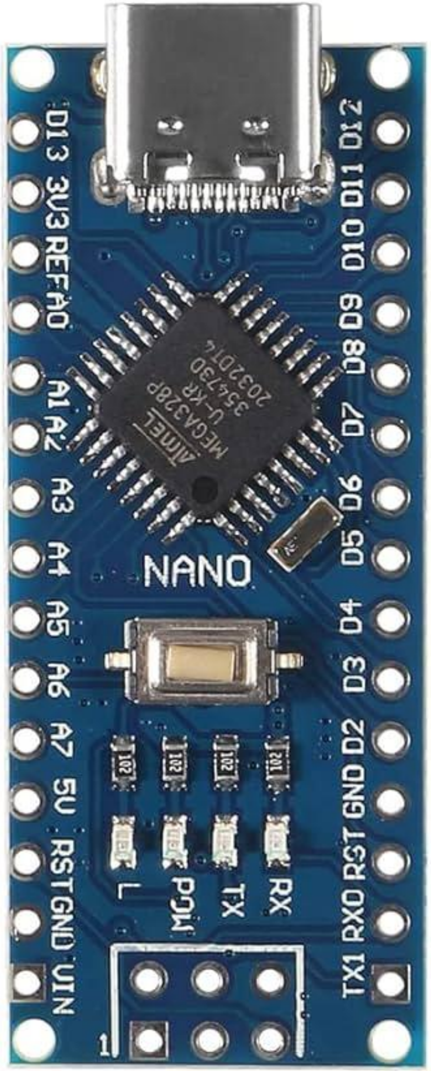
- Pinout tells what each pin does and how to access them
- Pins can have multiple functions based on how they are set up
- Peripherals interface outside data with the processor inside
- Information on Programming



PCINT[5] SCK PB5 D13
+3V3
AREF
PCINT[8] ADC[0] ADC[0] PC0 A0 D14
PCINT[9] ADC[1] ADC[1] PC1 A1 D15
PCINT[10] ADC[2] ADC[2] PC2 A2 D16
PCINT[11] ADC[3] ADC[3] PC3 A3 D17
PCINT[12] ADC[4] SDA ADC[4] PC4 A4 D18
PCINT[13] ADC[5] SCL ADC[5] PC5 A5 D19
ADC[6] ADC[6] A6 D20
ADC[7] ADC[7] A7 D21
5V
PCINT[14] RESET PC6 RESET
GND
VIN
D13 3V3 AREF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN
D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
ATMEL ATMEGA328P
RESET
L ON RX TX
NANO
ICSP
D12 PB4 MISO MISO PCINT[4]
~D11 PB3 MOSI MOSI OC2A PCINT[3]
~D10 PB2 SS OC1B PCINT[2]
~D9 PB1 OC1A PCINT[1]
D8 PB0 CLK0/ICP1 PCINT[0]
D7 PD7 AIN[1] PCINT[23]
~D6 PD6 AIN[0] OC0A PCINT[22]
~D5 PD5 OC0B/T1 PCINT[21]
D4 PD4 T0/XCK PCINT[20]
~D3 PD3 OC2B PCINT[19]/INT[1]
D2 PD2 PCINT[18]/INT[0]
GND
RESET PC6 RESET PCINT[14]
D0/RX PD0 RXD PCINT[16]
D1/TX PD1 TXD PCINT[17]
D13 PB5 SCK PCINT[5]
PCINT[14] PC6 RESET
D12 PB4 MISO MISO PCINT[4]
ICSP
GND
5V
~D11 PB3 MOSI MOSI OC2A PCINT[3]
POWER
GND
DIGITAL PIN
ANALOG PIN
COMMUNICATION
MICROCONTROLLER PIN
DEFAULT
TIMER
ANALOG
INTERRUPT
OTHER
SWD

Components on a Nano

- USB Port
- Atmega328p
- 16MHz Crystal
- Reset Button
- 4 LEDs
  - Pin 13 LED
  - Power
  - Serial Transmit
  - Serial Receive

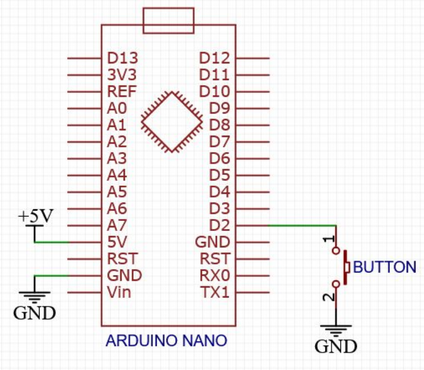


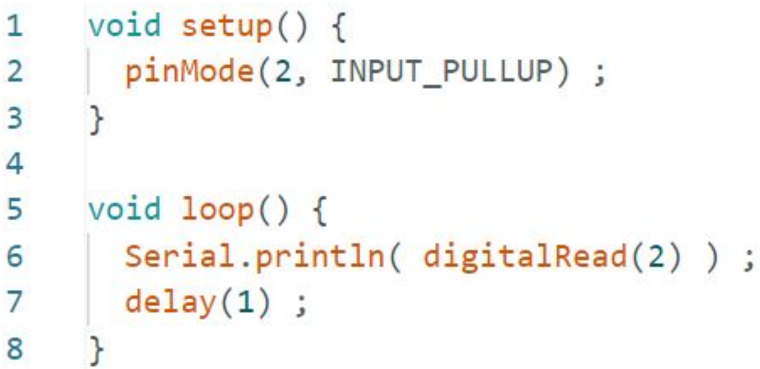
Digital I/O Pins

- All Pins labeled [D#] can be used as digital I/O
- Converts between a digital voltage on the pins (0V or 5V) and a value in the program
  - HIGH=true=1 ⇔ 5V
  - LOW=false=0 ⇔ 0V
  - Essentially rounds up or down if the voltage in between
- There is an onboard LED connected to [D13] that will light up if it is set to HIGH

Digital I/O Pins

- Must be setup before use using pinmode function
- Then read from or written to based on how it was set up
  - Set pinmode to OUTPUT to write 0V or 5V to a pin
  - Set to INPUT to read voltage as boolean
  - INPUT_PULLUP is a special mode that attaches an internal pullup resistor, helpful for reading buttons and such

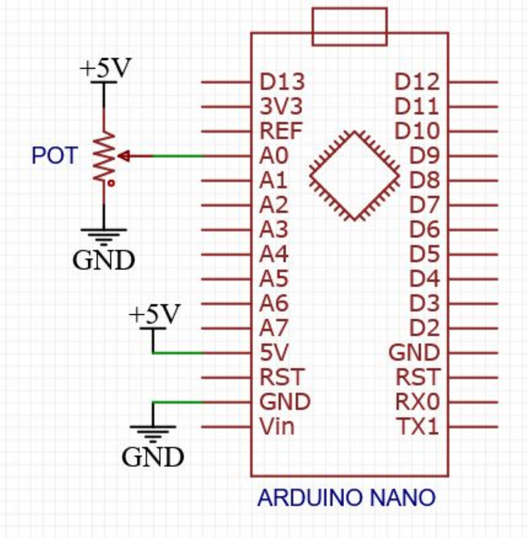


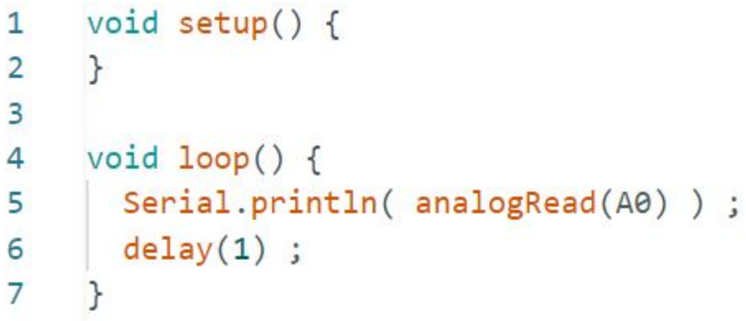


```
void setup() {
  pinMode(2, INPUT_PULLUP) ;
}

void loop() {
  Serial.println( digitalRead(2) ) ;
  delay(1) ;
}
```

Analog Pins

- Analog Pins labeled as [A#]
- Analog pins read in the voltage on a pin as an integer from 0-1023
  - 0V ⇔ 0
  - 2.5V ⇔ 511
  - 5V ⇔ 1023
- Helpful for using potentiometers as knobs, as it gives a measure of how turned it is

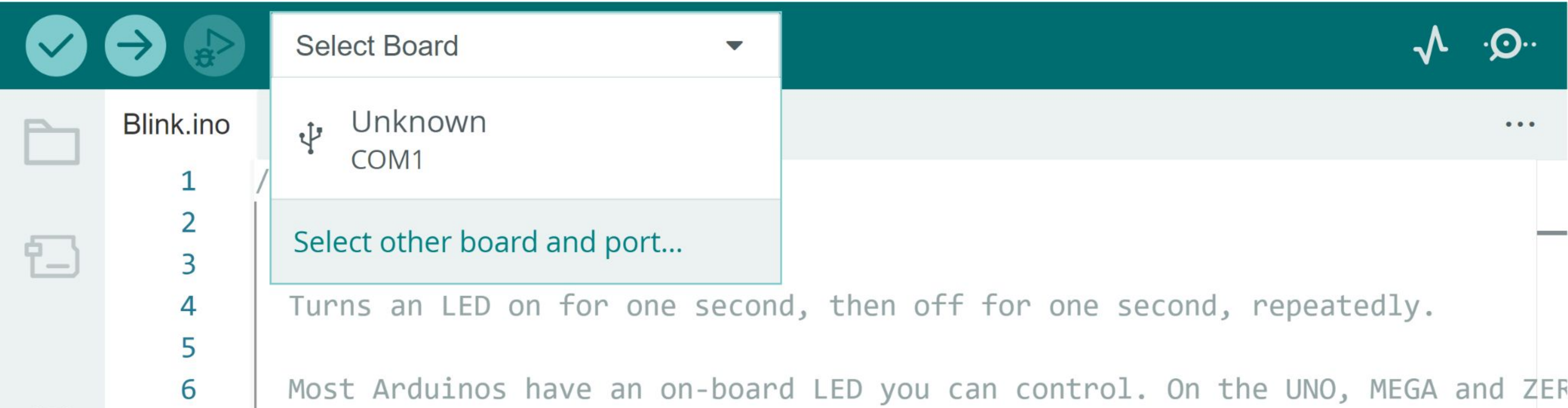




```
void setup() {
}

void loop() {
  Serial.println( analogRead(A0) ) ;
  delay(1) ;
}
```

How to Upload Programs

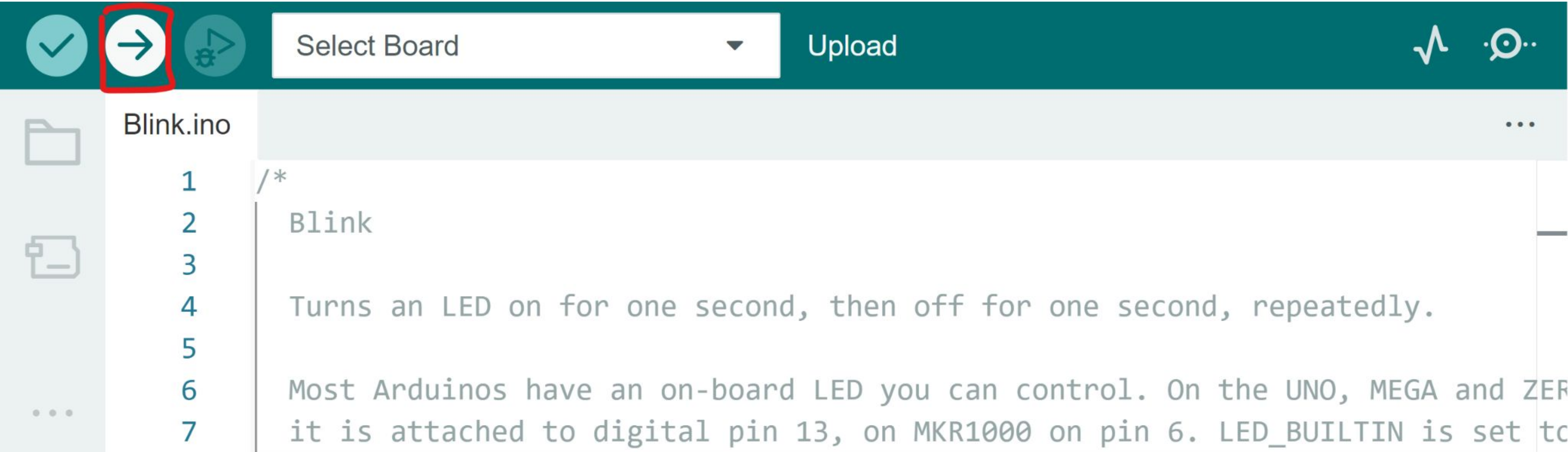
- To start you must tell Arduino IDE what type of microcontroller you are using so it can import the known data and functions that apply
- Click on select another board and port and then choose “Arduino Nano” and whatever COMM port that comes up



```
/
Turns an LED on for one second, then off for one second, repeatedly.

Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZER
```

How to Upload Programs

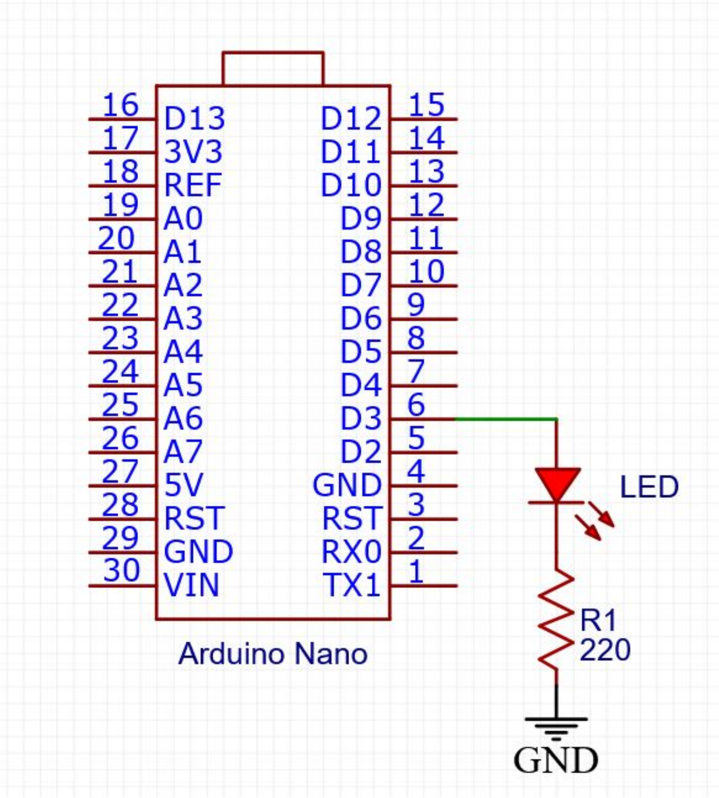
- Once the board is selected and you are ready to upload your program, click on the arrow in the top left to upload to the board



```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZER
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
```

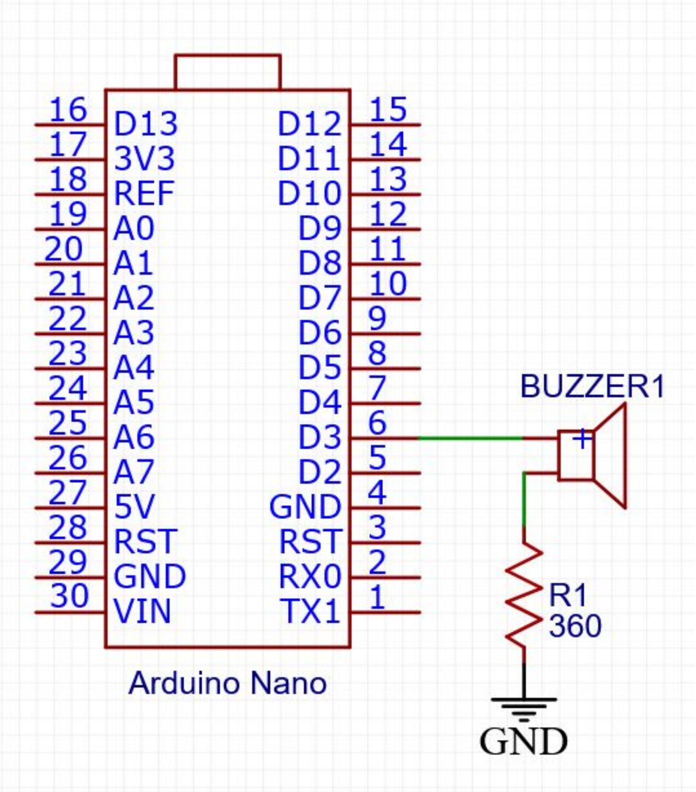
PWM Pins

- PWM (Pulse Width Modulation) labeled as [~D#]
- PWM is a method of approximating an analog signal by generating a rectangular wave with a variable “duty cycle” (percent of the wave that is HIGH vs LOW)
- Useful for dimming an LED
    - Duty cycle roughly represents how bright the LED is



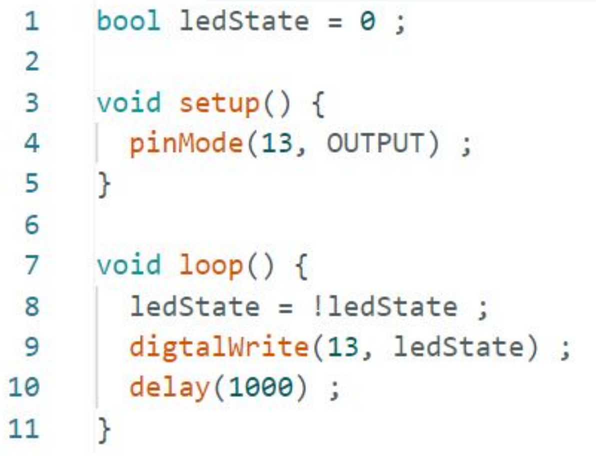
PWM Pins

- This can approximate writing an analog value since it will average to an analog voltage
  - This is written to using the same scale as reading an analog voltage
- Use with tone function to play notes from a buzzer

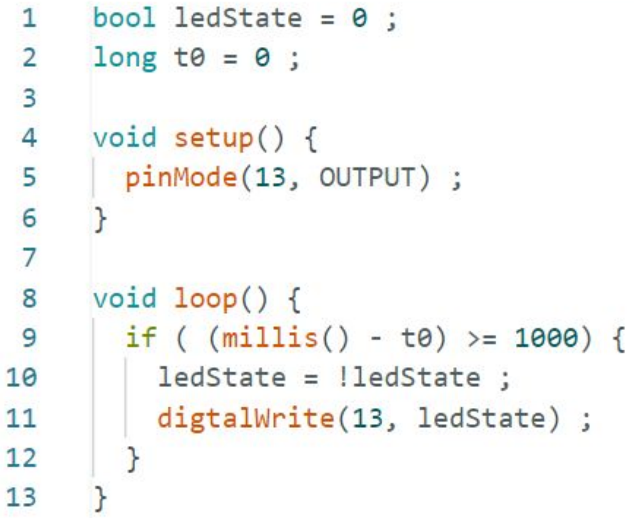


Timing

- delay() is used to stop the program and wait for an amount of milliseconds
  - Delay is a blocking function so no code will execute until it has waited the specified time
- millis() and micros() reference the timer and return how long the program has been running
  - Can use similarly to delay but without blocking



```
bool ledState = 0 ;

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

void loop() {
  ledState = !ledState ;
  digtalWrite(13, ledState) ;
  delay(1000) ;
}
```



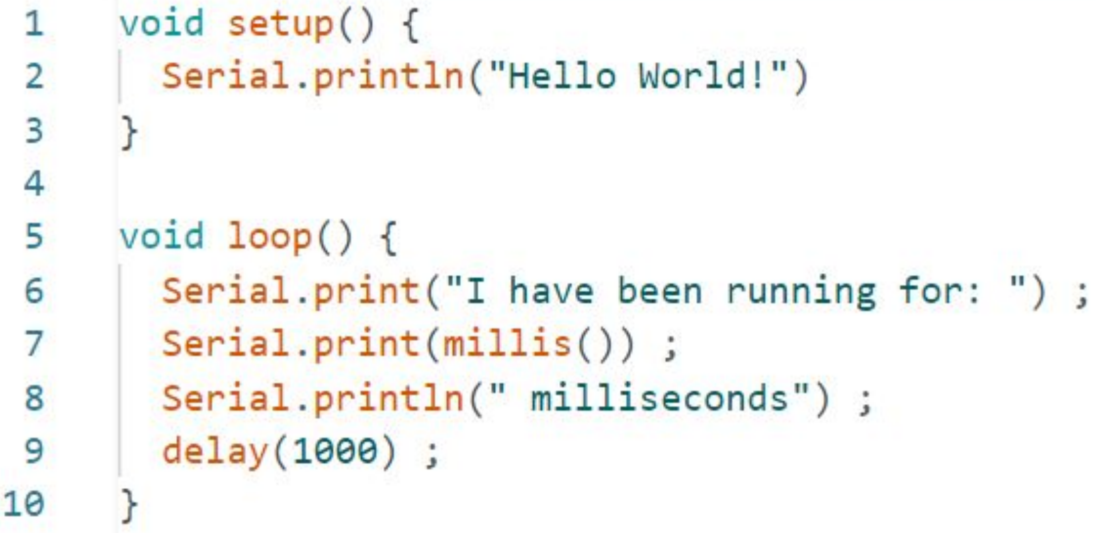
```
bool ledState = 0 ;
long t0 = 0 ;

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

void loop() {
  if ( (millis() - t0) >= 1000) {
    ledState = !ledState ;
    digtalWrite(13, ledState) ;
  }
}
```

Interrupts

- Interrupts allow you to run a function on the rising or falling edge when a pin changes state
  - Only pins D0 and D1 have this on arduino nano
- These are used for the best response times
- Can be very tricky so these are beyond the scope of this

Serial Communication

- Most often used to communicate to the computer over the USB port
- Often used for debugging
  - Print the text to the Serial monitor in arduino IDE for easy debugging
- This also writes to pin D0 and D1 for communication to other devices easier



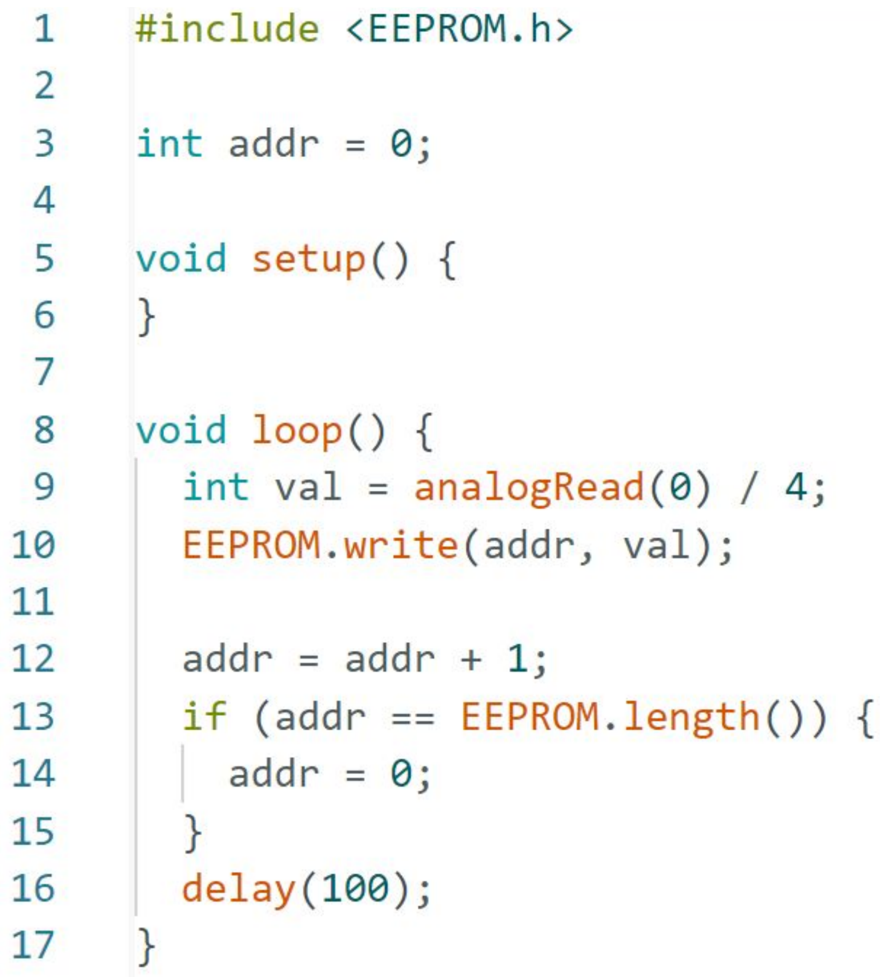
```
void setup() {
  Serial.println("Hello World!")
}

void loop() {
  Serial.print("I have been running for: ") ;
  Serial.print(millis()) ;
  Serial.println(" milliseconds") ;
  delay(1000) ;
}
```

I2C and SPI

- The nano includes two other communication protocols I2C and SPI
- Both are often used to connect extra peripherals like sensors, displays, or other various output devices
- Most often used bundled into another library
- I2C (integrated integrated circuit)
  - Uses pins A4 and A5 for communication
  - Uses <Wire.h> library
- SPI (Serial Peripheral Interface)
  - Uses Pins D11, D12, D13 for communication
  - Uses the <SPI.h> library

EEPROM

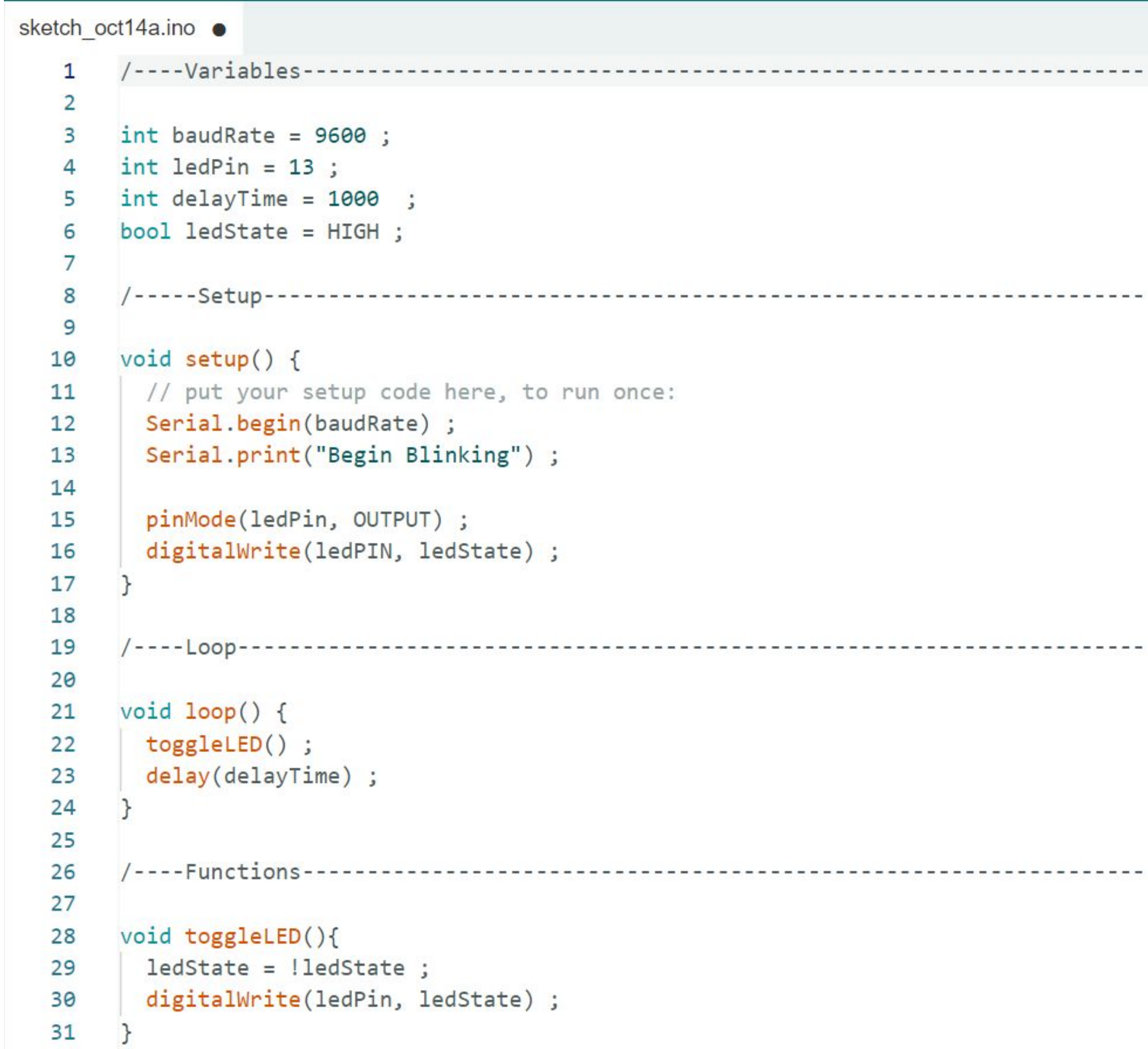
- The EEPROM can store data even when powered off
- Slow to write and with only 1024 Bytes of storage
- Most often used for user settings and such
- Easiest to use a library to access



```
#include <EEPROM.h>

int addr = 0;

void setup() {
}

void loop() {
  int val = analogRead(0) / 4;
  EEPROM.write(addr, val);

  addr = addr + 1;
  if (addr == EEPROM.length()) {
    addr = 0;
  }
  delay(100);
}
```

Anatomy of an Arduino Program

How the code works



```
/----Variables-----------------------------------------------------------------

int baudRate = 9600 ;
int ledPin = 13 ;
int delayTime = 1000  ;
bool ledState = HIGH ;

/-----Setup--------------------------------------------------------------------

void setup() {
  // put your setup code here, to run once:
  Serial.begin(baudRate) ;
  Serial.print("Begin Blinking") ;

  pinMode(ledPin, OUTPUT) ;
  digitalWrite(ledPIN, ledState) ;
}

/----Loop----------------------------------------------------------------------

void loop() {
  toggleLED() ;
  delay(delayTime) ;
}

/----Functions-----------------------------------------------------------------

void toggleLED(){
  ledState = !ledState ;
  digitalWrite(ledPin, ledState) ;
}
```

Arduino Programming Cheat Sheet

Primary source: Arduino Language Reference
https://www.arduino.cc/reference/en/

Structure & Flow

Basic Program Structure

```
void setup() {
  // Runs once when sketch starts
}
void loop() {
  // Runs repeatedly
}
```

Control Structures

```
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
for (int i = 0; i < 10; i++) { ... }
break;     // Exit a loop immediately
continue;  // Go to next iteration
switch (var) {
  case 1:
    ...
    break;
  case 2:
    ...
    break;
  default:
    ...
}
return x;  // x must match return type
return;    // For void return type
```

Function Definitions

```
<ret. type> <name>(<params>) { ... }
e.g. int double(int x) {return x*2;}
```

Operators

General Operators

=	assignment		
+	add	-	subtract
*	multiply	/	divide
%	modulo		
==	equal to	!=	not equal to
<	less than	>	greater than
<=	less than or equal to		
>=	greater than or equal to		
&&	and	||	or
!	not		

Compound Operators

++	increment
--	decrement
+=	compound addition
-=	compound subtraction
*=	compound multiplication
/=	compound division
&=	compound bitwise and
|=	compound bitwise or

Bitwise Operators

&	bitwise and	|	bitwise or
^	bitwise xor	~	bitwise not
<<	shift left	>>	shift right

Pointer Access

&	reference: get a pointer
*	dereference: follow a pointer

Built-in Functions

Pin Input/Output

Digital I/O - pins 0-13 A0-A5

```
  pinMode(pin,
    {INPUT|OUTPUT|INPUT_PULLUP})
  int digitalRead(pin)
  digitalWrite(pin, {HIGH|LOW})
```

Analog In - pins A0-A5

```
  int analogRead(pin)
  analogReference(
    {DEFAULT|INTERNAL|EXTERNAL})
```

PWM Out - pins 3 5 6 9 10 11

```
  analogWrite(pin, value) // 0-255
```

Advanced I/O

```
tone(pin, freq_Hz, [duration_msec])
noTone(pin)
shiftOut(dataPin, clockPin,
  {MSBFIRST|LSBFIRST}, value)
shiftIn(dataPin, clockPin,
  {MSBFIRST|LSBFIRST})
unsigned long pulseIn(pin,
  {HIGH|LOW}, [timeout_usec])
```

Time

```
unsigned long millis()
  // Overflows at 50 days
unsigned long micros()
  // Overflows at 70 minutes
delay(msec)
delayMicroseconds(usec)
```

Math

```
min(x, y)    max(x, y)    abs(x)
sin(rad)     cos(rad)     tan(rad)
sqrt(x)      pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
```

Random Numbers

```
randomSeed(seed) // long or int
long random(max) // 0 to max-1
long random(min, max)
```

Bits and Bytes

```
lowByte(x)    highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn)  // bitn: 0=LSB 7=MSB
```

Type Conversions

```
char(val)        byte(val)
int(val)         word(val)
long(val)        float(val)
```

External Interrupts

```
attachInterrupt(interrupt, func,
 {LOW|CHANGE|RISING|FALLING})
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```

Libraries

Serial - *comm. with PC or via RX/TX*

```
begin(long speed)  // Up to 115200
end()
int available()  // #bytes available
int read()   // -1 if none available
int peek()   // Read w/o removing
flush()
print(data)    println(data)
write(byte)    write(char * string)
write(byte * data, size)
SerialEvent()  // Called if data rdy
```

SoftwareSerial.h - *comm. on any pin*

```
SoftwareSerial(rxPin, txPin)
begin(long speed)  // Up to 115200
listen()         // Only 1 can listen
isListening()  // at a time.
read, peek, print, println, write
  // Equivalent to Serial library
```

EEPROM.h - *access non-volatile memory*

```
byte read(addr)
write(addr, byte)
EEPROM[index]  // Access as array
```

Servo.h - *control servo motors*

```
attach(pin, [min_usec, max_usec])
write(angle)  // 0 to 180
writeMicroseconds(uS)
   // 1000-2000; 1500 is midpoint
int read()    // 0 to 180
bool attached()
detach()
```

Wire.h - *I²C communication*

```
begin()      // Join a master
begin(addr) // Join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(byte)              // Step 2
send(char * string)
send(byte * data, size)
endTransmission()       // Step 3
int available() // #bytes available
byte receive()  // Get next byte
onReceive(handler)
onRequest(handler)
```

Variables, Arrays, and Data

Data Types

bool	true | false
char	-128 - 127, 'a' '$' etc.
unsigned char	0 - 255
byte	0 - 255
int	-32768 - 32767
unsigned int	0 - 65535
word	0 - 65535
long	-2147483648 - 2147483647
unsigned long	0 - 4294967295
float	-3.4028e+38 - 3.4028e+38
double	currently same as float
void	return type: no return value

Strings

```
char str1[8] =
  {'A','r','d','u','i','n','o','\0'};
  // Includes \0 null termination
char str2[8] =
  {'A','r','d','u','i','n','o'};
  // Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

Numeric Constants

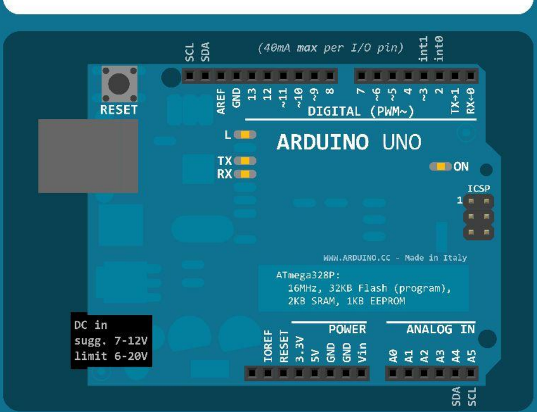
123	decimal
0b01111011	binary
0173	octal - base 8
0x7B	hexadecimal - base 16
123U	force unsigned
123L	force long
123UL	force unsigned long
123.0	force floating point
1.23e6	1.23*10^6 = 1230000

Qualifiers

static	persists between calls
volatile	in RAM (nice for ISR)
const	read-only
PROGMEM	in flash

Arrays

```
byte myPins[] = {2, 4, 8, 3, 6};
int myInts[6];    // Array of 6 ints
myInts[0] = 42;   // Assigning first
                  // index of myInts
myInts[6] = 12;   // ERROR! Indexes
                  // are 0 though 5
```



by Mark Liffiton
version: 2024-02-14

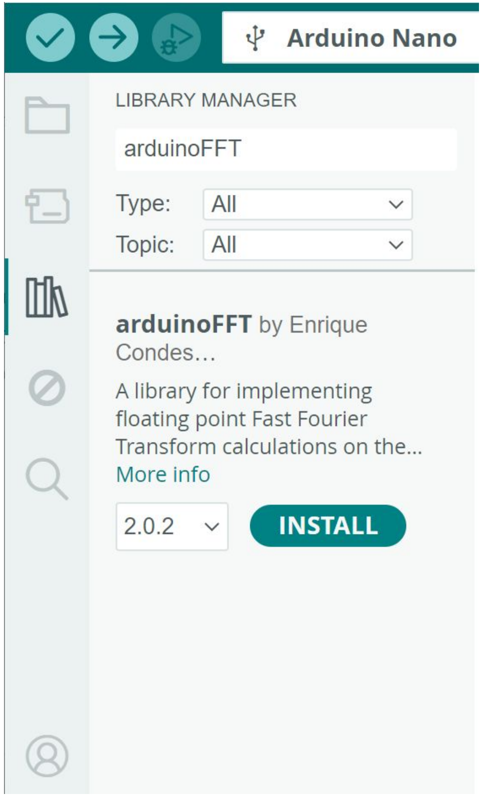
source: https://github.com/liffiton/Arduino-Cheat-Sheet/

Adapted from:
- Original: Gavin Smith
- SVG version: Frederic Dufourg
- Arduino board drawing: Fritzing.org

Libraries

- Very first part of the code imports libraries
- These allow you to use code other people have made
- Must have the libraries installed through the library manager first

```
#include "arduinoFFT.h"
```



LIBRARY MANAGER

arduinoFFT

Type: All

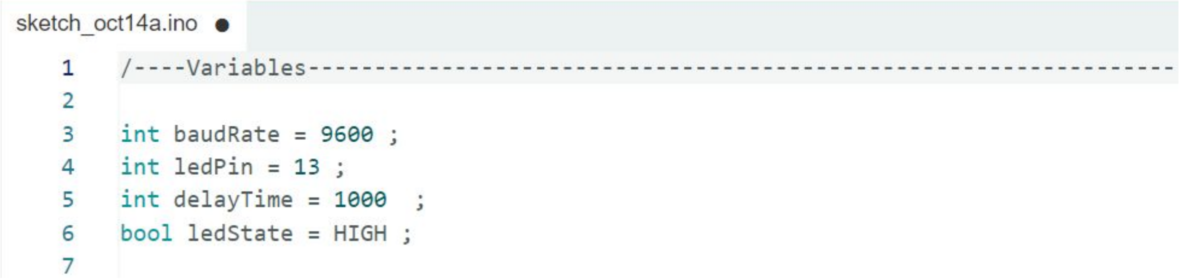
Topic: All

arduinoFFT by Enrique Condes…

A library for implementing floating point Fast Fourier Transform calculations on the...

Variables

- Written in C so variables are statically typed
  - Int, bool, float, array, and String most often used types
- Global variables often placed before the setup portion
  - Variables only active in the scope they are defined
- Usually comes after any library imports but before the setup function

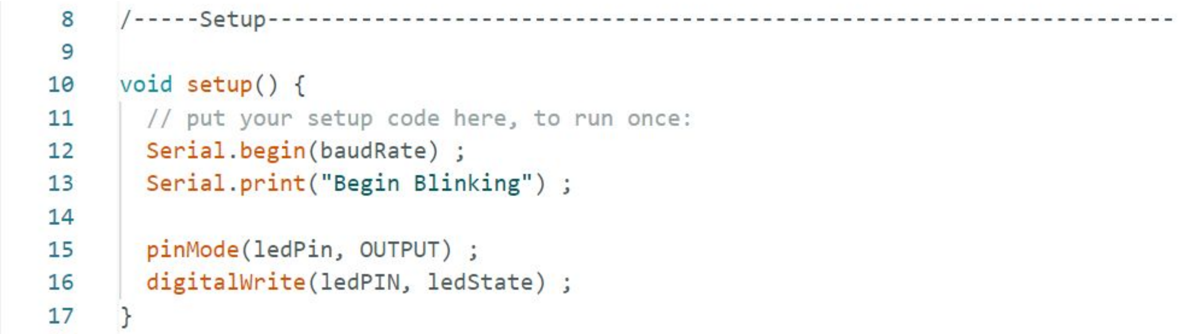


sketch_oct14a.ino ●

```
/----Variables------------------------------------------------------------------

int baudRate = 9600 ;
int ledPin = 13 ;
int delayTime = 1000  ;
bool ledState = HIGH ;

```

Setup

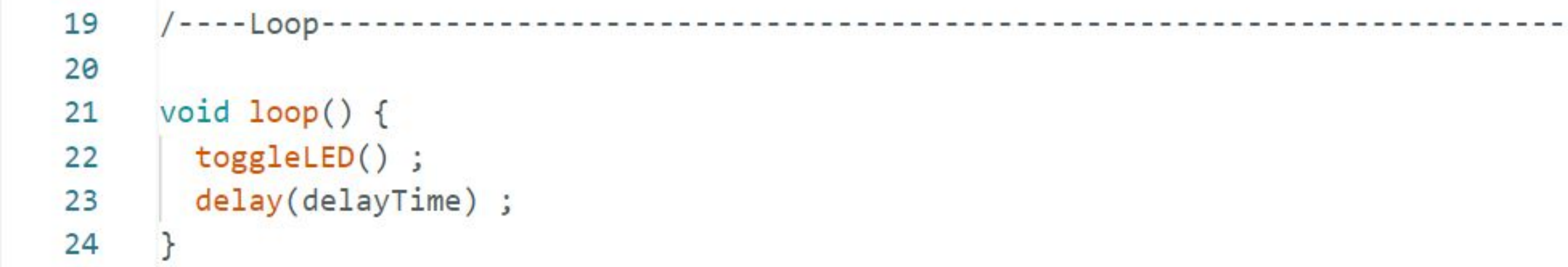
- This code is run once upon starting the program
- This is where you usually initialize anything used in the program including:
  - Pin Modes and their starting condition
  - The Serial port for communication
  - Any objects used
- All code inside the setup function



```
/-----Setup----------------------------------------------------------------------

void setup() {
  // put your setup code here, to run once:
  Serial.begin(baudRate) ;
  Serial.print("Begin Blinking") ;

  pinMode(ledPin, OUTPUT) ;
  digitalWrite(ledPIN, ledState) ;
}
```

Loop

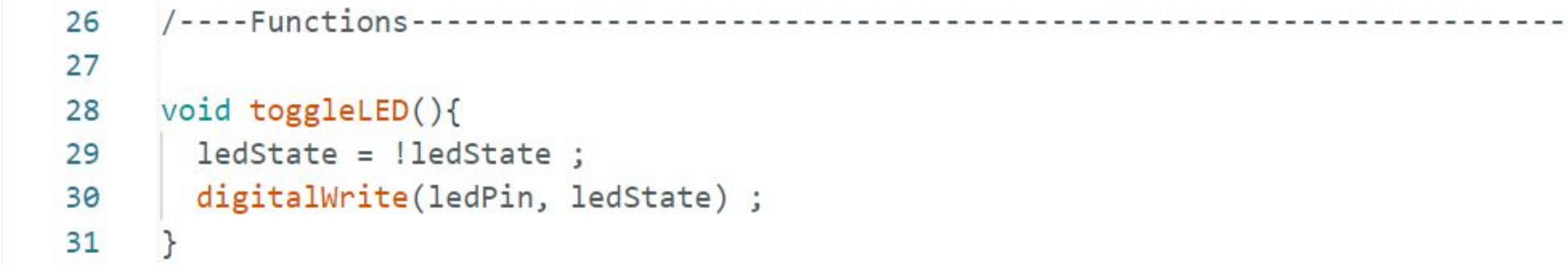
- The main part of the program that is looped constantly once started
- This is the meat of the program where you interact with things and perform logic
- All code must be inside the loop function



```
/----Loop----------------------------------------------------------------------

void loop() {
  toggleLED() ;
  delay(delayTime) ;
}
```

Functions

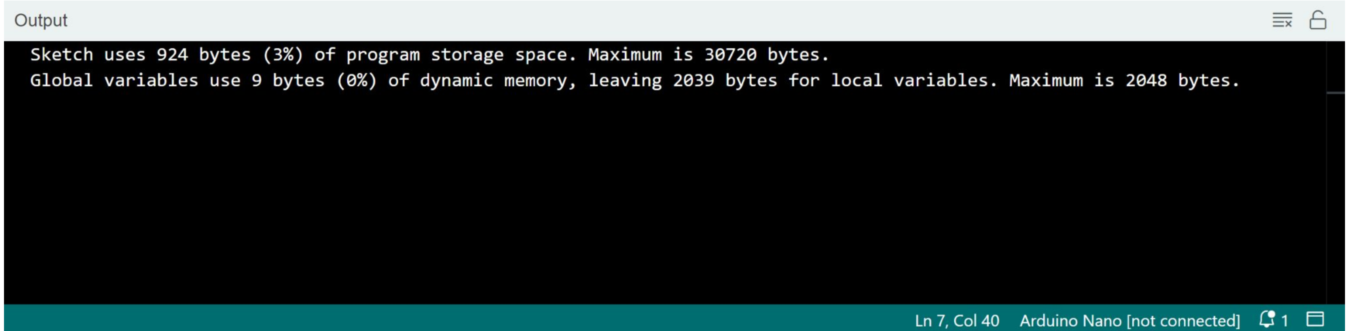
- After the Loop is where functions are usually defined
- The type at the beginning defines the output of the function
    - Void gives no function output
    - “return [value]” to get an output



```
/----Functions------------------------------------------------------------------

void toggleLED(){
  ledState = !ledState ;
  digitalWrite(ledPin, ledState) ;
}
```

How Memory Works on a Nano

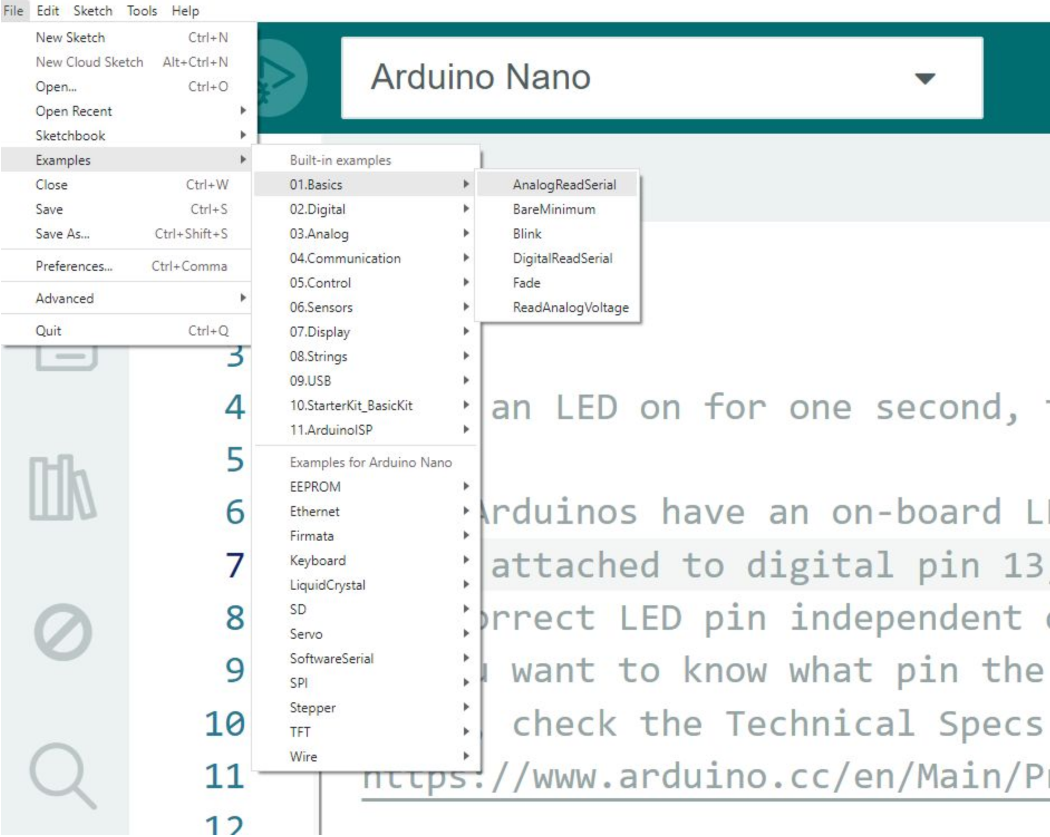
- Microcontrollers have limited memory for both programs and variables on board
  - Flash memory hold the instructions that define the program and is limited to 30720 bytes on a nano
  - RAM holds the data for any variables used in the program and is limited to 2048 bytes on a nano
  - EEPROM is not part of the running program so it is a peripheral part of the system



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

Example Programs!

- Under Files tab with example programs
- Very useful if you get stuck, and very well documented usually



```
 an LED on for one second,

Arduinos have an on-board L
 attached to digital pin 13
orrect LED pin independent
 want to know what pin the
  check the Technical Specs
https://www.arduino.cc/en/Main/Pr
```

Now Build Your Own!

- Chasing Lights with controllable frequency
- Stacker style hit the LED in the middle
- Potentiometer controlled dimmable LED
- Whack a mole style game
- Simon says/Pattern repetition
- Buzzer Piano