Arduino for (Neuro)Biologists

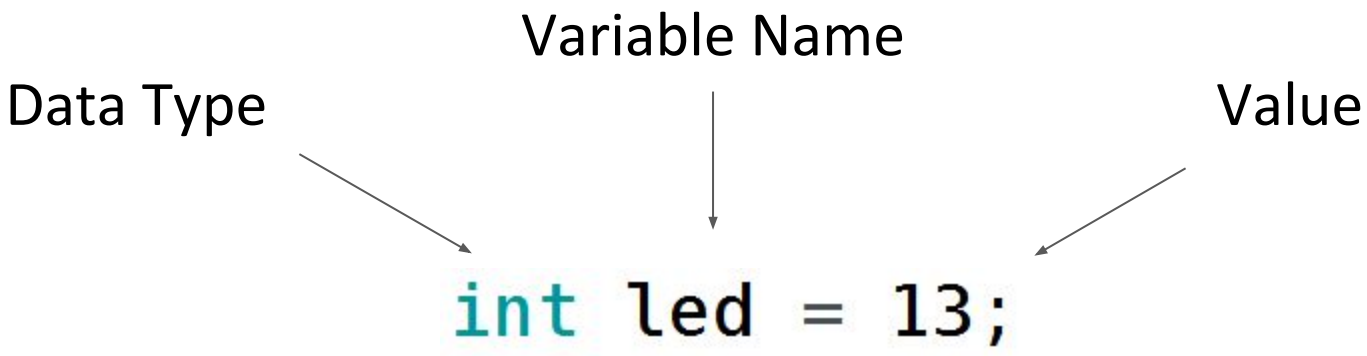
Day 3: Software

Arduino programming concepts, advice, and caveats

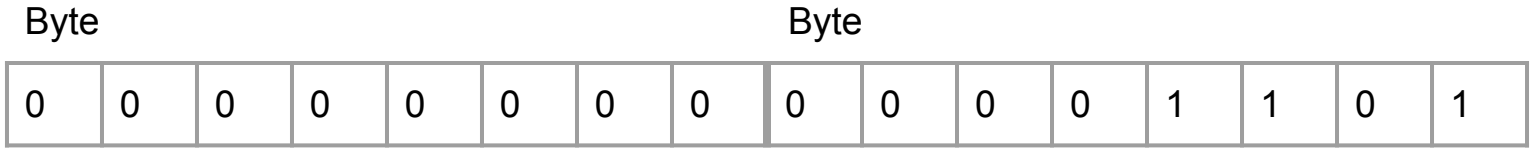
- Datatypes: limitations and advice
- Memory types and usage
- Functions & libraries
- Timing & interrupts
- Arduino - PC communication


sketch | Arduino 1.6.7
File Edit Sketch Tools Help
sketch
int led = 13;
void setup() {
pinMode(led, OUTPUT);
}
void loop() {
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
}
Done Saving.
Invalid library found in /home/brett/sketchbook/libraries/btserial: /home/brett/sketchbook/librar
12
Teensy 3.2 / 3.1, Serial, 72 MHz optimized, US English on /dev/ttyACM0


sketch | Arduino 1.6.7
File Edit Sketch Tools Help
sketch
int led = 13;
void setup() {
pinMode(led, OUTPUT);
}
void loop() {
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
}
Done Saving.
Invalid library found in /home/brett/sketchbook/libraries/btserial: /home/brett/sketchbook/librar
12
Teensy 3.2 / 3.1, Serial, 72 MHz optimized, US English on /dev/ttyACM0

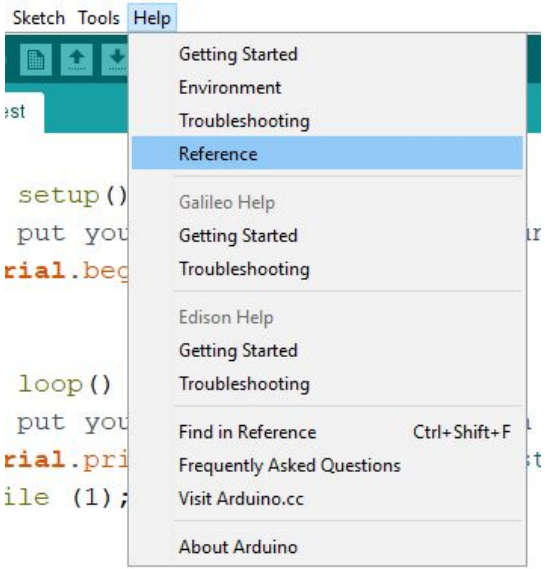

Variable Name
Data Type
Value
int led = 13;

```
int led = 13;
```



Data Types

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



- setup()
- loop()

Control Structures

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

Further Syntax

- ; (semicolon)
- {} (curly braces)
- // (single line comment)
- /* */ (multi-line comment)
- #define
- #include

Arithmetic Operators

- = (assignment operator)
- + (addition)

Constants

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

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- short
- float
- double
- string - char array
- String - object
- array

Digital I/O

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

Analog I/O

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

Due & Zero only

- analogReadResolution()
- analogWriteResolution()

Advanced I/O

- tone()
- noTone()
- shiftOut()
- shiftIn()
- pulseIn()

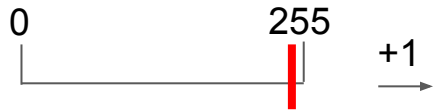
Time

- millis()
- micros()
- delay()
- delayMicroseconds()

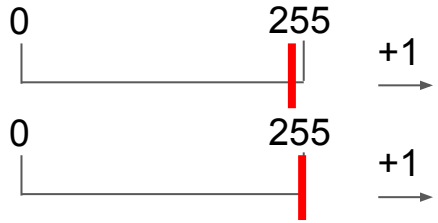
	Number of bytes	Minimum value	Maximum value	N to fill RAM
int	2	-32,768	32,767	1024
unsigned int	2	0	65535	1024
char	1	-128	127	2048
byte	1	0	255	2048
long	4	-2,147,483,648	2,147,483,647	512
unsigned long	4	0	4,294,967,295	512

```
byte a = 254;
```

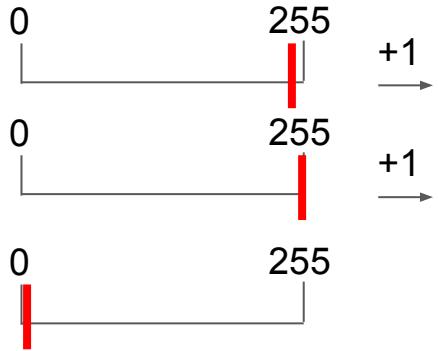
```
byte a = 254;
a = a + 1;  // a == 255
```



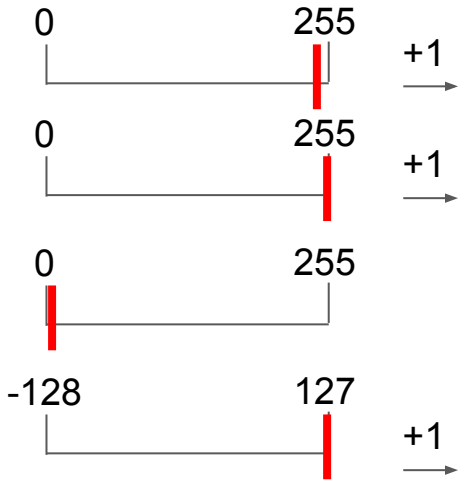
```
byte a = 254;
a = a + 1;  // a == 255
a = a + 1;
```



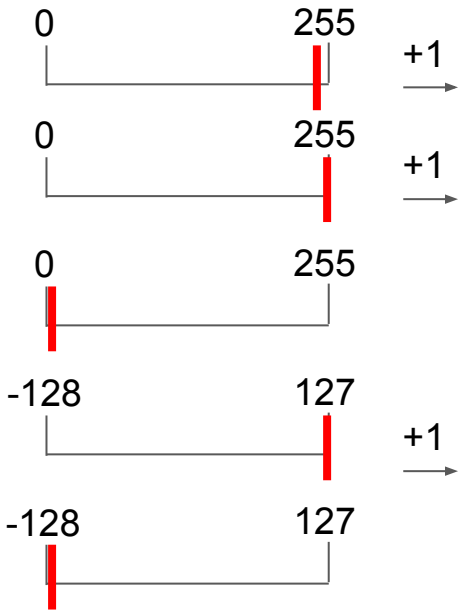
```
byte a = 254;
a = a + 1;   // a == 255
a = a + 1;   // a == 0
```



```
byte a = 254;
a = a + 1;   // a == 255
a = a + 1;  // a == 0

char b = 127;
b = b + 1;
```



```
byte a = 254;
a = a + 1;   // a == 255
a = a + 1;  // a == 0

char b = 127;
b = b + 1; // a == -128
```



	Number of bytes	Minimum value	Maximum value
unsigned int	2	0	65535
int	2	-32,768	32,767
byte	1	0	255
char	1	-128	127
unsigned long	4	0	4,294,967,295
long	4	-2,147,483,648	2,147,483,647

	Number of bytes	Minimum value	Maximum value
unsigned int	2	0	65535
int	2	-32,768	32,767
byte	1	0	255
char	1	-128	127
unsigned long	4	0	4,294,967,295
long	4	-2,147,483,648	2,147,483,647
float	4	-3.4028235E+38	3.4028235E+38
double	4	-3.4028235E+38	3.4028235E+38

```
void int_speed() {
  unsigned long us = 0;
  us = micros();
  for (int i=0; i<100; i=i+1) {
    int_value = int_value + int_value;
  }
  us = micros() - us;
  Serial.print("100 adds of ints took ");
  Serial.print(us, DEC);
  Serial.println(" microseconds");
  delay(1000);
}
```

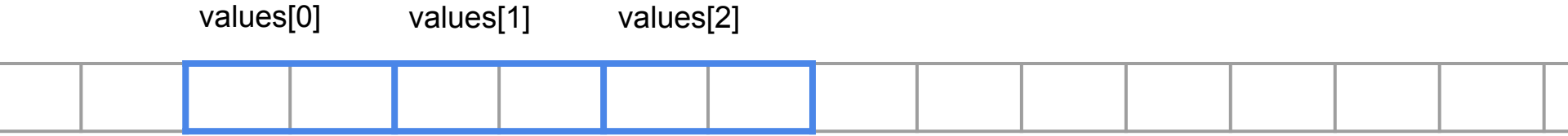
```
100 adds of ints took 40 microseconds
100 adds of ints took 44 microseconds
100 adds of ints took 40 microseconds
100 adds of ints took 40 microseconds
100 adds of ints took 44 microseconds
100 adds of ints took 48 microseconds
100 adds of ints took 44 microseconds
```

```
void float_speed() {
  unsigned long us = 0;
  us = micros();
  for (int i=0; i<100; i=i+1) {
    float_value = float_value + float_value;
  }
  us = micros() - us;
  Serial.print("100 adds of floats took ");
  Serial.print(us, DEC);
  Serial.println(" microseconds");
  delay(1000);
}
```

```
100 adds of floats took 752 microseconds
100 adds of floats took 752 microseconds
100 adds of floats took 748 microseconds
100 adds of floats took 756 microseconds
100 adds of floats took 752 microseconds
100 adds of floats took 752 microseconds
```

Arrays

```
int values[3];
```

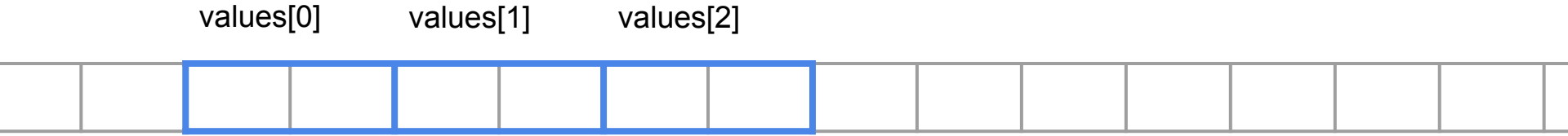


```
int values[3];
values[0] = 20000;
values[1] = 25000;
values[2] = 20000;
```

```
int values[] = {20000, 25000, 20000};
```

Arrays

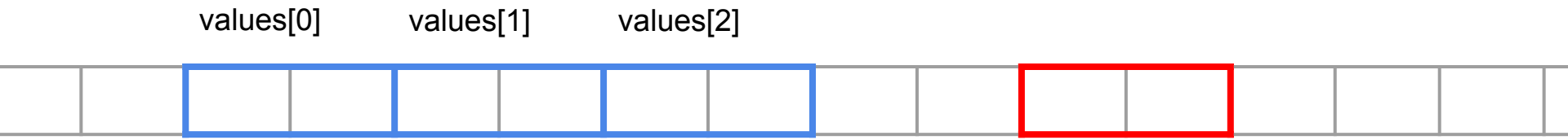
```
int values[3];
```




```
values[4] = 25000;
```

Arrays

```
int values[3];
```



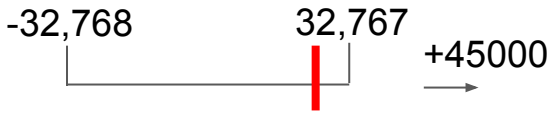

```
values[4] = 25000;
```

Don’t do this!

```
int values[3];
values[0] = 20000;
values[1] = 25000;
values[2] = 20000;

int sum = 0;
for (int i=0; i<3; i=i+1) {
  sum = sum + values[i];
}
```

```
int values[3];
values[0] = 20000;
values[1] = 25000;
values[2] = 20000;

int sum = 0;
for (int i=0; i<3; i=i+1) {
  sum = sum + values[i];
}
```

sum == -536 !!



Strings

```
char greeting[] = "Hi ";
char question[] = "how are you?";

char message[] = greeting + question;
```

Strings

```
char greeting[] = "Hi ";
char question[] = "how are you?";

char message[] = greeting + question;

String greeting = "Hi ";
String question = "how are you?";

String message = greeting + question;
```

Memory

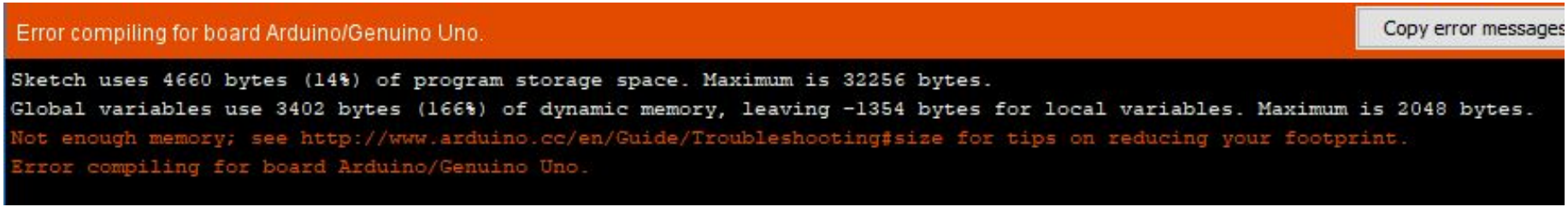
- RAM: working memory
- Flash: where your program is stored
- EEPROM: user writeable (calibration values, unique identifiers, etc…)

	Size	**Volatile?**	**Write Cycles**	**Speed**
RAM	2 kBytes	Yes	-	125 ns
Flash	32 kBytes	No	10,000	187.5 ns
EEPROM	1 kByte	No	100,000	3.3 ***ms***

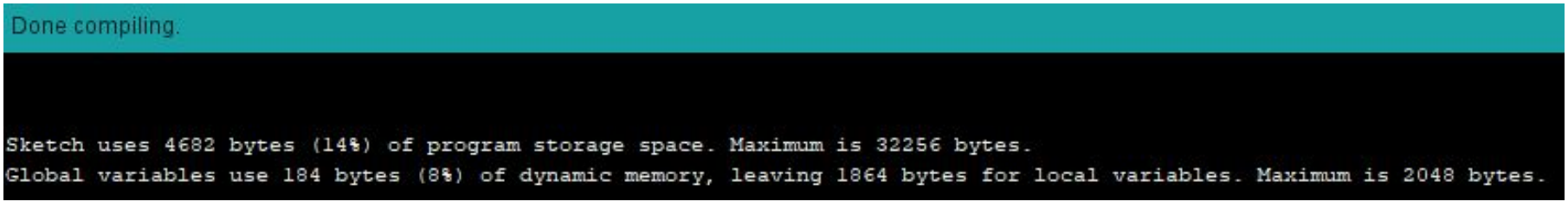
Running out of RAM

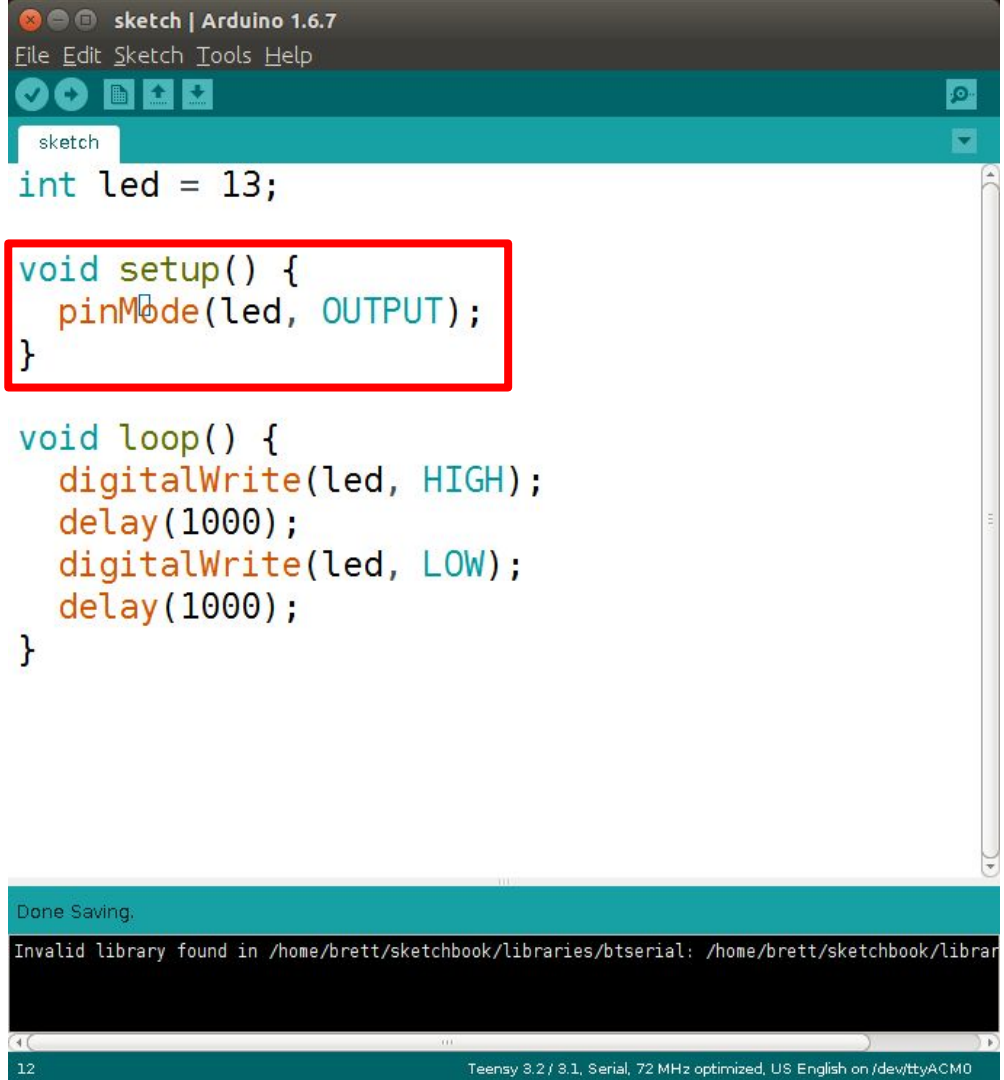
- The program will warn you (mostly correct)
- Use smaller data types when possible
  - long -> int -> char
  - unsigned long -> unsigned int -> byte
- Use a different ‘Arduino’ with more RAM (Mega has 8Kb, Teensy 3.2 has 64Kb!)
- Store long strings that won’t be modified in Flash...

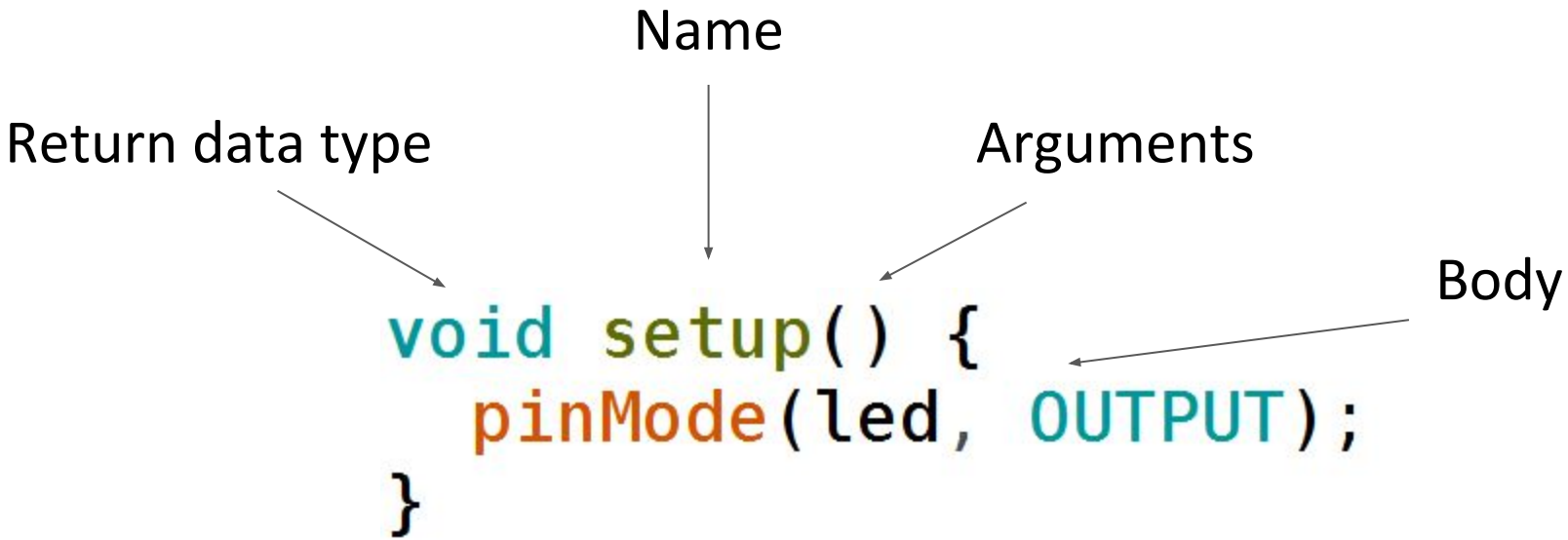
Store strings in flash



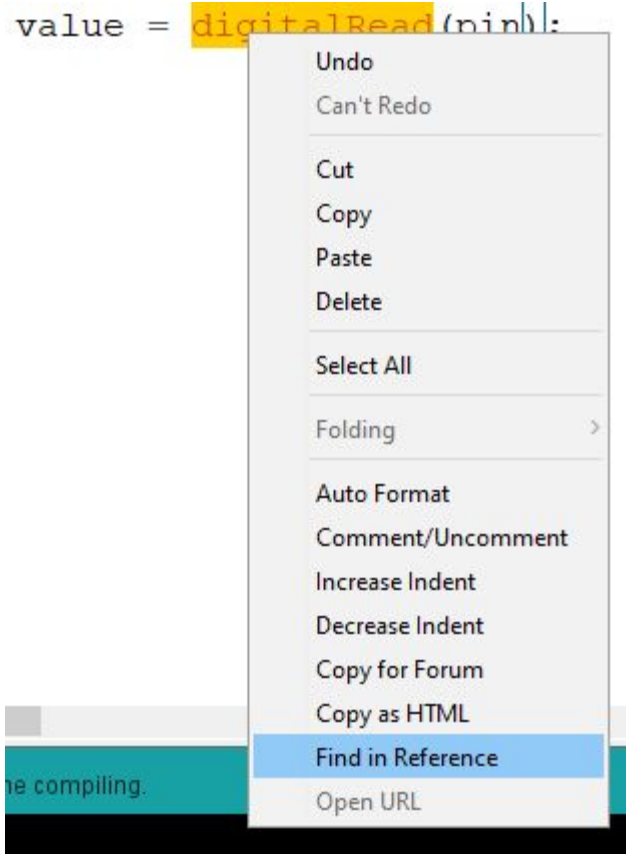

```
Serial.print(F("This string is in flash"));
```




sketch | Arduino 1.6.7
File Edit Sketch Tools Help
sketch
int led = 13;
void setup() {
pinMode(led, OUTPUT);
}
void loop() {
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
}
Done Saving.
Invalid library found in /home/brett/sketchbook/libraries/btserial: /home/brett/sketchbook/librar
12
Teensy 3.2 / 3.1, Serial, 72 MHz optimized, US English on /dev/ttyACM0


Name
Return data type
Arguments
Body
void setup() {
pinMode(led, OUTPUT);
}

```
value = digitalRead(pin);
```



digitalRead()

Description

Reads the value from a specified digital pin, either HIGH or LOW.

Syntax

digitalRead(pin)

Parameters

pin: the number of the digital pin you want to read (*int*)

Returns

HIGH or LOW

Example

Sets pin 13 to the same value as pin 7, declared as an input.

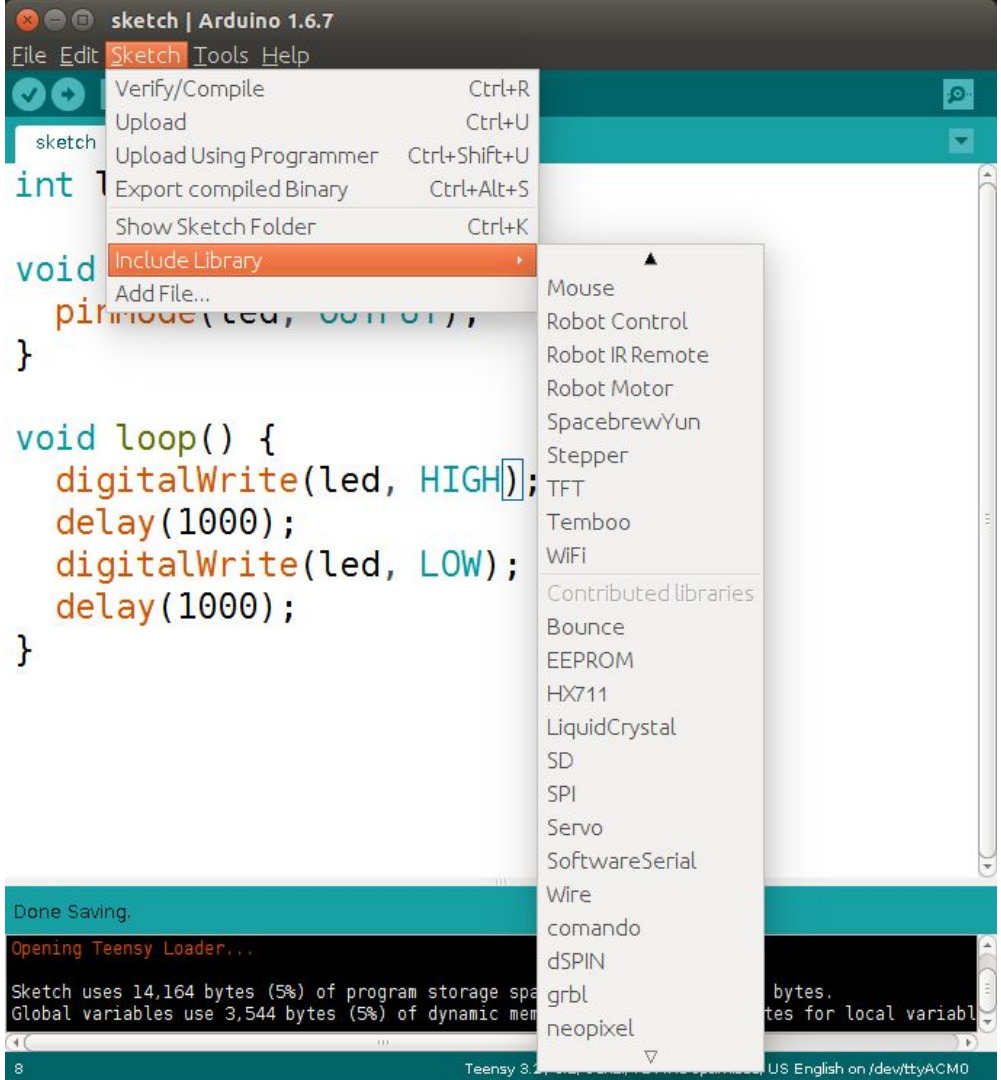
```
int ledPin = 13; // LED connected to digital pin 13
int inPin = 7;   // pushbutton connected to digital pin 7
int val = 0;     // variable to store the read value

void setup()
{
```

Libraries

Collections of functions

Usable when added to sketch

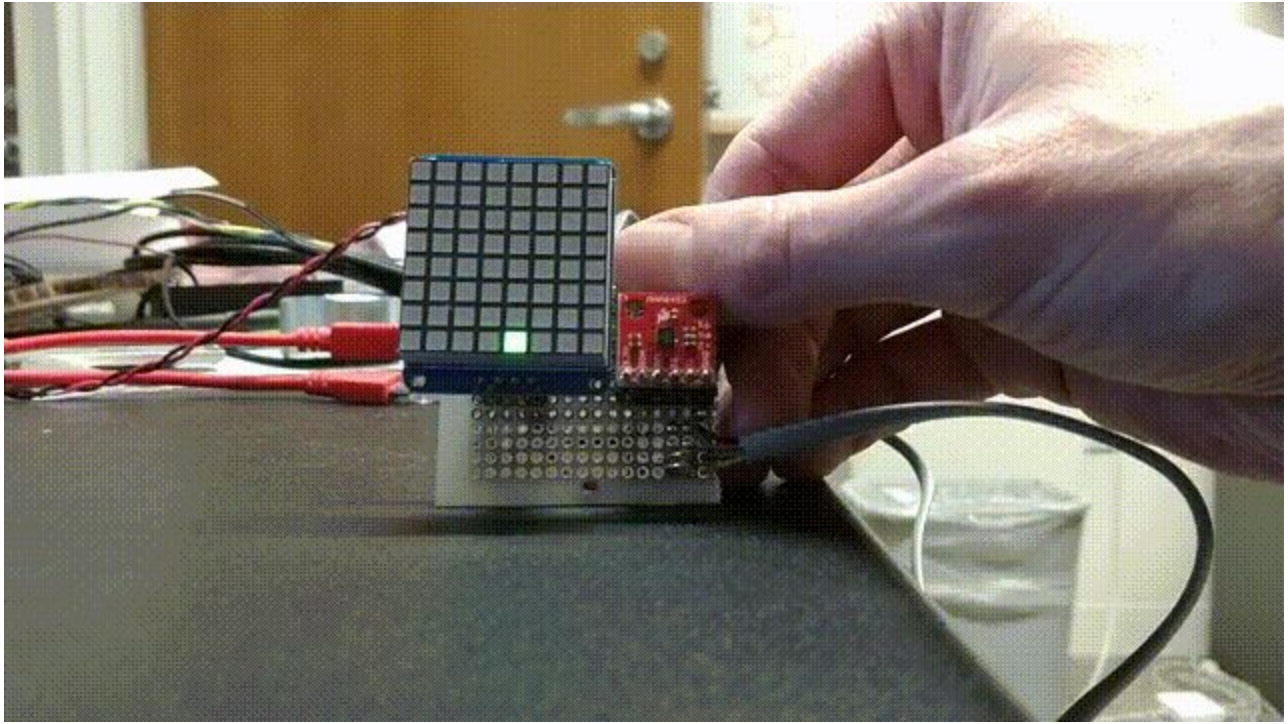
- Include Library
- `#include <EEPROM.h>`



Communication Libraries

- Serial (UART, USB)
  - 2 Wires (plus power and ground): RX TX (pins 0 and 1)
  - Connects to a single other device
  - `Serial.print(F("This string is in flash"));`
- SPI: Serial Peripheral Interface (Library: SPI)
  - 3 Wires (plus power and ground): MOSI MISO SCK CS (pins 11, 12, 13, 10)
  - Can communicate with multiple devices (requires a separate CS wire per device)
- I2C/TWI: Inter-Integrated Circuit/Two-Wire Interface (Library: Wire)
  - 2 wires (plus power and ground): SDA SCL (pins A4, A5)
  - Can communicate with multiple devices (each device must have a unique address)

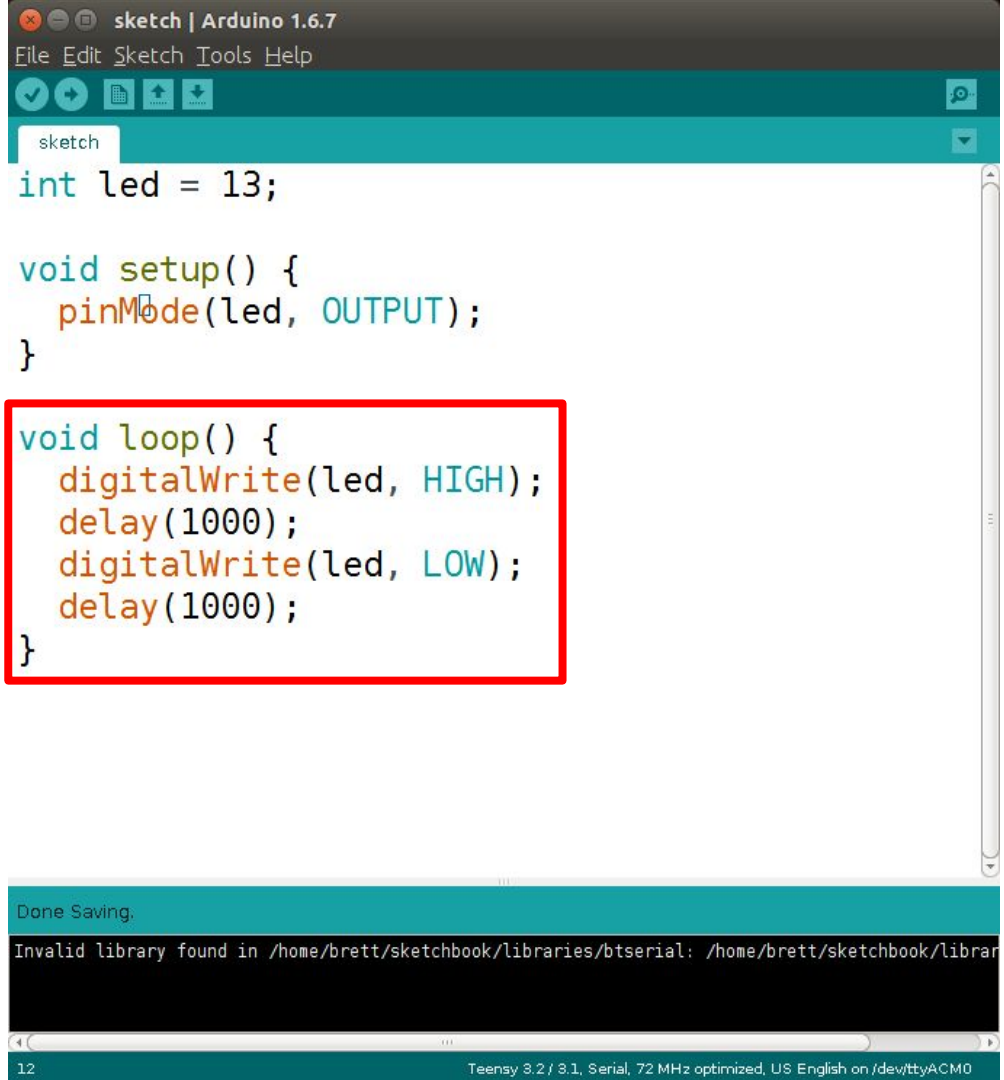
```
#include <Wire.h>

#define MMA8452_ID 29  // the ID of the accelerometer

void setup() {
  Wire.begin();  // join I2C bus
}

void loop() 
  byte message = 13;
  Wire.beginTransmission(MMA8452_ID); // start transmitting
  Wire.write(message);
  Wire.endTransmission(); //Stop transmitting

  Wire.requestFrom(MMA8452_ID, 1); //Ask for 1 byte
  while(!Wire.available()) ; //Wait for the data to come back
  byte value = Wire.read(); //Return this one byte
}
```



```
#include <Wire.h>

#define MMA8452_ID 29  // the ID of the accelerometer

void setup() {
  Wire.begin();  // join I2C bus
}

void loop() 
  byte message = 13;
  Wire.beginTransmission(MMA8452_ID); // start transmitting
  Wire.write(message);
  Wire.endTransmission(); //Stop transmitting

  Wire.requestFrom(MMA8452_ID, 1); //Ask for 1 byte
  while(!Wire.available()) ; //Wait for the data to come back
  byte value = Wire.read(); //Return this one byte
}
```

Preprocessor

```
#include <Wire.h>

#define MMA8452_ID 29

Wire.beginTransmission(MMA8452_ID);
Wire.beginTransmission(29);
```


sketch | Arduino 1.6.7
File Edit Sketch Tools Help
sketch
int led = 13;
void setup() {
pinMode(led, OUTPUT);
}
void loop() {
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
}
Done Saving.
Invalid library found in /home/brett/sketchbook/libraries/btserial: /home/brett/sketchbook/librar
12
Teensy 3.2 / 3.1, Serial, 72 MHz optimized, US English on /dev/ttyACM0

Timing

```
delay(ms);  // up to ~50 days
delayMicroseconds(us);  // 3 to 16383 us

unsigned long ms = millis();
unsigned long us = micros();
```

Timing

```
delay(ms);  // up to ~50 days
delayMicroseconds(us);  // 3 to 16383 us

unsigned long ms = millis();
unsigned long us = micros();
```

Watch out for **overflows!**

- millis : 2^{32} milliseconds ~50 days
- micros: 2^{32} microseconds ~70 minutes

Blink with button input

Blink LED on for 1 second, off for 1 second

If a button is pressed, blink for on for 3 (instead of 1) seconds

```
void led_on() {
  if (led_state) return;
  digitalWrite(LED, HIGH);
  led_on_time = millis();
  led_state = 1;
  Serial.print("                                    LED on time: ");
  Serial.println(led_on_time, DEC);
  Serial.print("                                    LED was off for ");
  Serial.print(led_on_time - led_off_time, DEC);
  Serial.println(" milliseconds");
}
```

Simple blink

```
void simple_blink() {
  led_off();
  delay(off_duration);
  led_on();
  delay(default_on_duration);
}
```

Simple blink: timing

```
                                    LED on time: 2102069
                                    LED was off for 1002 milliseconds
LED off time: 2103129
LED was on for 1060 milliseconds
                                    LED on time: 2104130
                                    LED was off for 1001 milliseconds
LED off time: 2105190
LED was on for 1060 milliseconds
                                    LED on time: 2106191
                                    LED was off for 1001 milliseconds
LED off time: 2107251
LED was on for 1060 milliseconds
```

Simple blink (attempt to) add button

```
void simple_blink_with_button() {
  led_off();
  delay(off_duration);
  led_on();
  if (digitalRead(BUTTON) == LOW) {
    delay(duration_when_pressed);
  } else {
    delay(default_on_duration);
  }
}
```

Better blink

```
void better_blink() {
  led_off();
  start_time = millis();
  while ((millis() - start_time) < off_duration) {
    // do other things
  };
  start_time = millis();
  led_on();
  while ((millis() - start_time) < default_on_duration) {
    // do other things
  };
}
```

Better blink: timing

```
                                    LED on time: 2338958
                                    LED was off for 1000 milliseconds
LED off time: 2339958
LED was on for 1000 milliseconds
                                    LED on time: 2340958
                                    LED was off for 1000 milliseconds
LED off time: 2341958
LED was on for 1000 milliseconds
                                    LED on time: 2342958
                                    LED was off for 1000 milliseconds
LED off time: 2343958
LED was on for 1000 milliseconds
```

Better blink with button

```
void blink_with_button() {
  on_duration = default_on_duration;
  led_off();
  start_time = millis();
  while ((millis() - start_time) < off_duration) {
    if (digitalRead(BUTTON) == LOW) {
      on_duration = duration_when_pressed;
    }
  };
  start_time = millis();
  led_on();
  while ((millis() - start_time) < on_duration) {
    if (digitalRead(BUTTON) == LOW) {
      on_duration = duration_when_pressed;
    }
  };
}
```

Better blink with button: timing

```
                                        LED on time: 2398539
                                        LED was off for 1000 milliseconds
LED off time: 2399539
LED was on for 1000 milliseconds
                                        LED on time: 2400539
                                        LED was off for 1000 milliseconds
Button pressed
LED off time: 2403539
LED was on for 3000 milliseconds
                                        LED on time: 2404539
                                        LED was off for 1000 milliseconds
LED off time: 2407539
LED was on for 3000 milliseconds
```

Timing tips

Code takes time to execute: Serial.print(lots of text…)

Using delay prevents you from doing other things

Poll the clocks (millis, micros), check for the elapsed time

Look for (and test) timing libraries (elapsedMillis)

Test (prints & o-scope)

Dealing with timer overflows

millis : ~50 days

micros: ~70 minutes

Is it going to be a problem?

Beware of comparisons against times

```
if (millis() > some_time) {
```

Instead, compare against durations and keep variables as unsigned long

```
unsigned long past_time;
unsigned long duration;

if ((millis() - past_time) >= duration)
```

Dealing with timer overflows

Other means of tracking time? (Computer, RTC)

ellapsedMillis: http://playground.arduino.cc/Code/ElapsedMillis

Test...

```
#include <util/atomic.h>

void set_millis(unsigned long ms) {
  extern unsigned long timer0_millis;
  ATOMIC_BLOCK (ATOMIC_RESTORESTATE) {
    timer0_millis = ms;
    //timer0_micros
  }
}
```

How does the Arduino keep time?

Onboard crystal oscillates at 16 MHz

- 20-30 ppm drift
- Temperature dependant
- Don’t use this for wall time. Use a RTC (2-3 ppm, temperature compensated)

After ~1 ms, 1 is added to the “millis” counter

An **interrupt** is used to perform the addition

Interrupts

Event that pauses, interrupts your code, for (hopefully) a short period of time to run other code

Can be triggered by timers/clocks or digital inputs

Used by many common functions (delay, millis, analogRead, Serial, I2C, SPI)

You can write your own interrupts but be careful!

- In interrupts, many common functions won't work (delay, millis, etc…)
- To use a variable in an interrupt, it must be volatile `volatile byte count = 0;`
- Can cause unstable code especially if the interrupt is not very short

Communicating with your computer

Arduino is connected to computer via USB

Connection is Serial (UART) over USB

To connect you will need to know:

- the port
  - Windows: COM1, COM2, etc...
  - Mac/Linux: /dev/ttyUSB0, /dev/ttyACM0, etc...
- the baud rate (speed): 300, 9600, 115200, etc…

When you open the port, it will reset the Arduino

	Arduino	Matlab	Python
Open port	Serial.begin(9600);	p = serial('COM7', 'BaudRate', 9600); fopen(p);	p = serial.Serial('COM7', 9600)
Write message	Serial.print(...);	fwrite(p, bytes, 'uint8', 'sync');	p.write(bytes)
Check for message	Serial.available();	n = p.BytesAvailable;	n = p.inWaiting()
Read message	Serial.read();	bytes = fread(p, n);	bytes = p.read(n)

Parsing serial messages

```
if (Serial.available()) {
  char id = Serial.read();
  if (id == '\r') {
  } else if (id == '\n') {
  } else if (id == 'a') {
    my_int = Serial.parseInt();
  } else if (id == 'b') {
    my_float = Serial.parseFloat();
  } else {
    Serial.print("Invalid identifier ");
    Serial.println(id);
  }
}
```

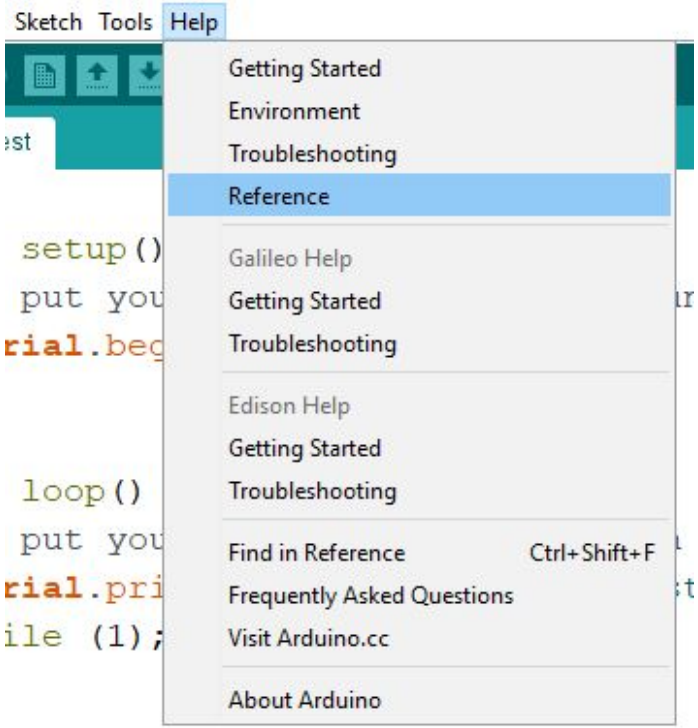
Advice

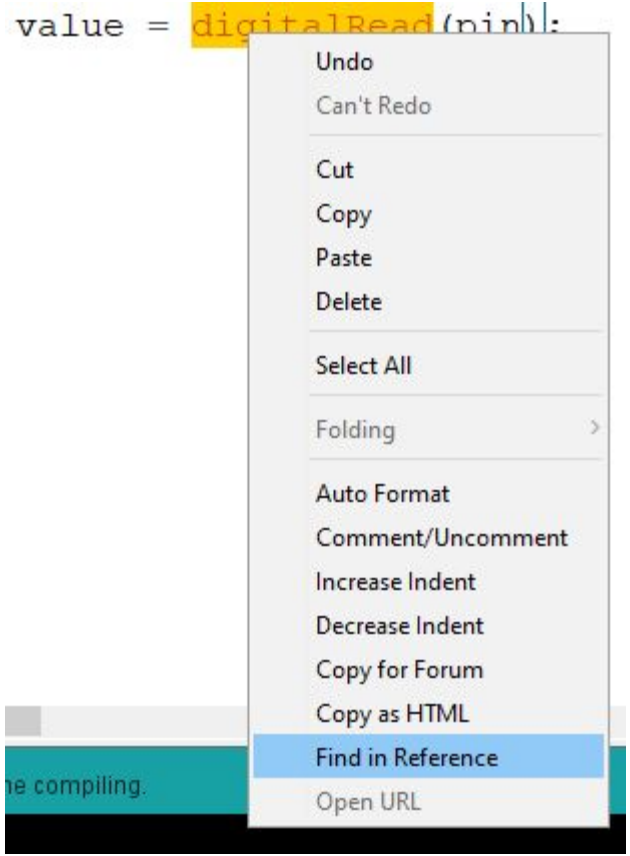
It will fail in interesting new ways

It is normal to spend more time debugging than building/programming

Try to break big problems into small steps

When debugging, be systematic (if I delete this code, does the problem go away?)


Sketch Tools Help
Getting Started
Environment
Troubleshooting
Reference
Galileo Help
Getting Started
Troubleshooting
Edison Help
Getting Started
Troubleshooting
Find in Reference Ctrl+Shift+F
Frequently Asked Questions
Visit Arduino.cc
About Arduino
setup()
put you
rial.beg
loop()
put you
rial.pri
ile (1);


value = digitalRead(pin);
Undo
Can't Redo
Cut
Copy
Paste
Delete
Select All
Folding
Auto Format
Comment/Uncomment
Increase Indent
Decrease Indent
Copy for Forum
Copy as HTML
Find in Reference
Open URL
compiling.

Other Useful References

https://learn.adafruit.com/

https://learn.sparkfun.com/

http://arduino.stackexchange.com/

https://www.google.com/

Us!

Day 3 project ideas

Continue working on previous projects

Try other sensor/actuator

Use parseInt/parseFloat to set training parameters from the serial monitor

Output trial information to serial in csv format

Send data to and read data from the Arduino using MATLAB/Python/etc

Why use EEPROM?

- 3.3 ms write!
- Easily available non-volatile write access
  - calibration values
  - saved state or other limited data
  - unique id (for many Arduinos running same code)

```
#include <EEPROM.h>

int address = 0;   // 0 to 1024
byte value = 12;   // must be a byte

EEPROM.write(address, value);
value = EEPROM.read(address);
```

```
#include <EEPROM.h>

int address = 0;  // 0 to 1024
byte value = 12;  // must be a byte

EEPROM.write(address, value);
value = EEPROM.read(address);
```

```
#include <EEPROM.h>

int address = 0;   // 0 to 1024
byte value = 12;   // must be a byte

EEPROM.write(address, value);
value = EEPROM.read(address);
```

```
#include <EEPROM.h>

int address = 0;  // 0 to 1024
byte value = 12;  // must be a byte

EEPROM.write(address, value);
value = EEPROM.read(address);
```

```
#define INTERRUPT_PIN 2
volatile byte count = 0;

void setup() {
  pinMode(INTERRUPT_PIN, INPUT);
  attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), tick, CHANGE);
}

void loop() {
  // do other stuff
}

void tick() {
  count = count + 1;
}
```

Disabling interrupts

Why? For very time-critical code

Remember, many functions require interrupts (delay, Serial, Wire, EEPROM, etc…)

They will not work when interrupts are disabled

```
noInterrupts();
// time critical code
interrupts();
```