

Physical Computing

Designing Physical Interactions for a Digital World

DESN 265

Spring 2012

Thursday 2PM – 5:30PM

Online and QC Makerspace

Professor Danne Woo

pcomp.dannewoo.com

danne.woo@qc.cuny.edu

Week 1-9

Week 1: What is Physical Computing?

Week 2: Introduction to Electronics

Week 3: Arduino, Hello World

Week 4: Analog Input and Output

Week 5: Digital and Analog Review

Week 6: Enclosures

Week 7: Serial Communication, Processing and p5.js

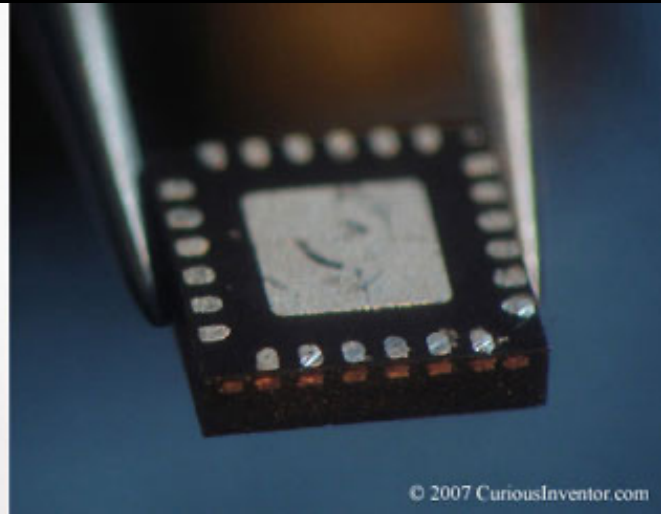
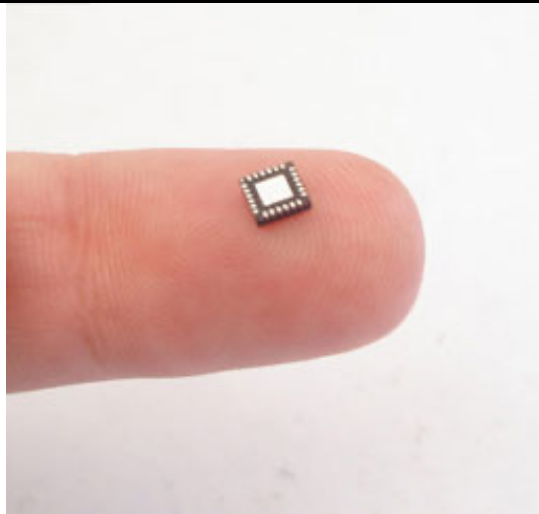
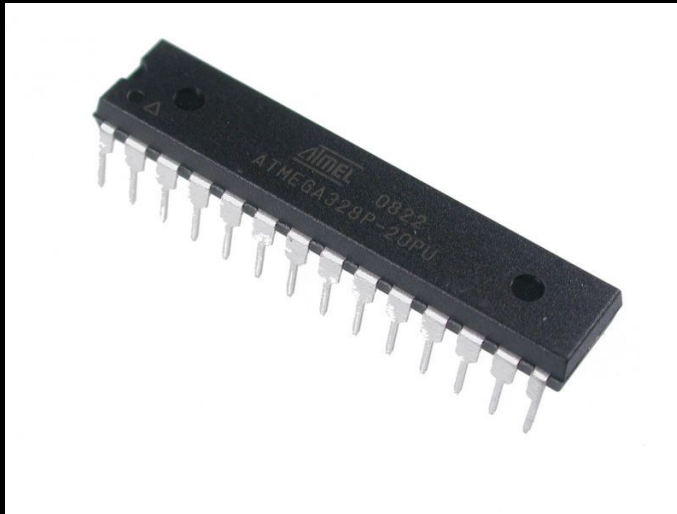
Week 8: Soldering Workshop

Week 9: Midterm Presentation

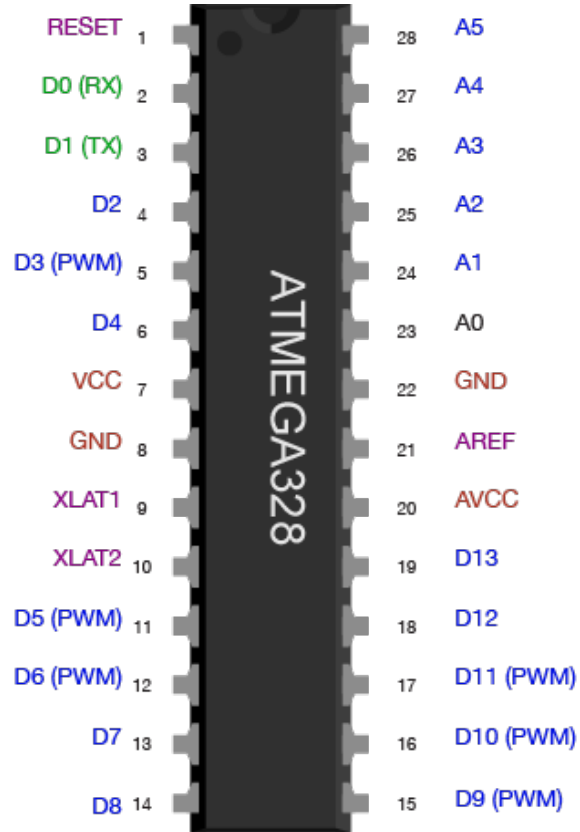
Microcontroller

Very simple computer processor
that is designed for a basic task
like listening to sensors.

Microcontroller



Arduino / Atmega328

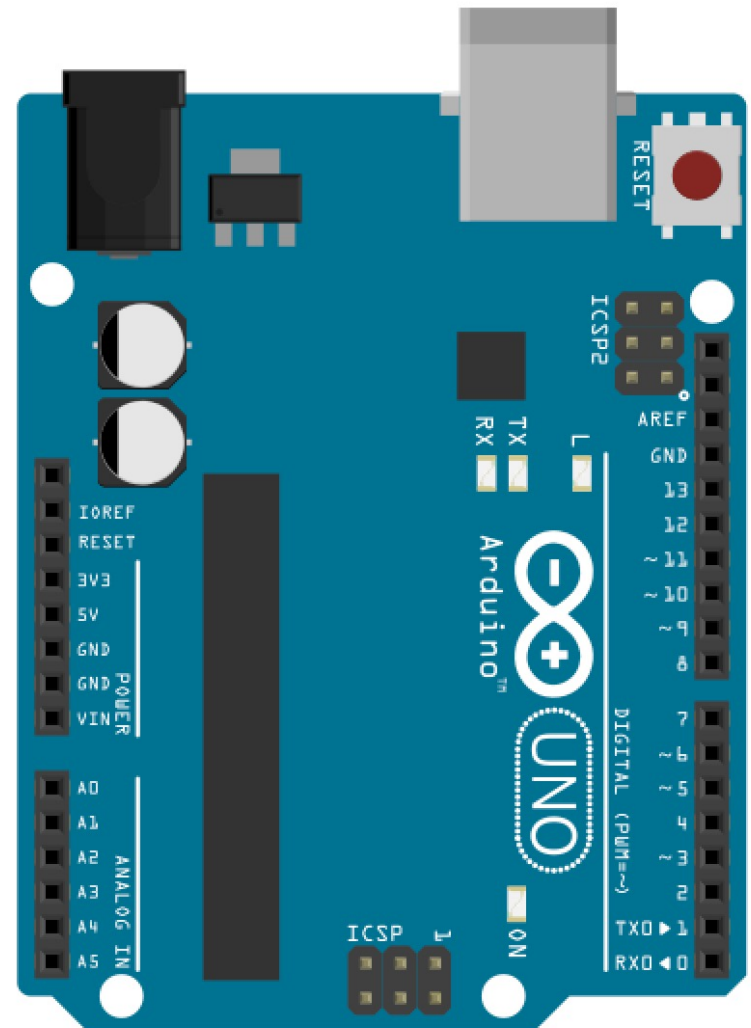


Power connectors

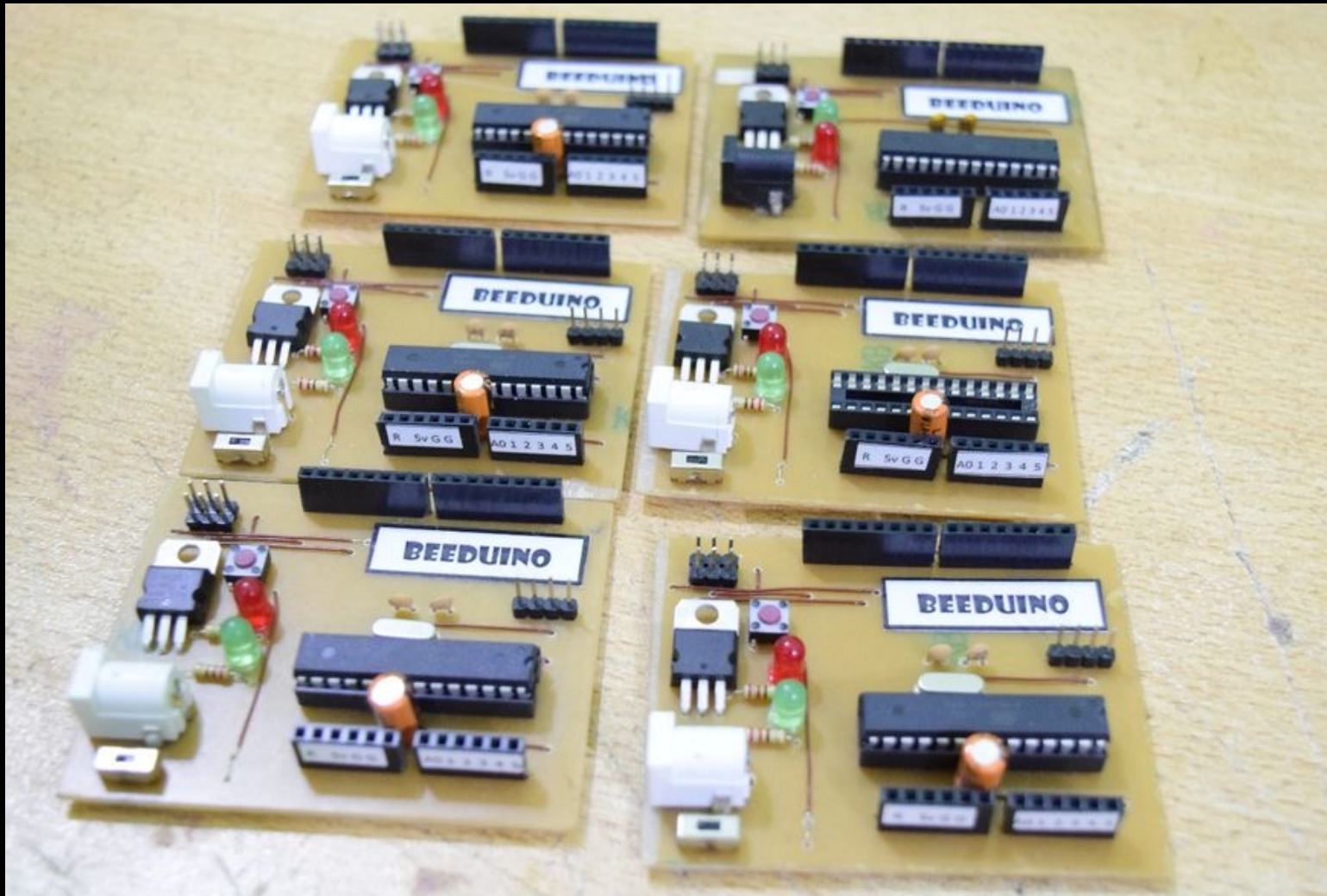
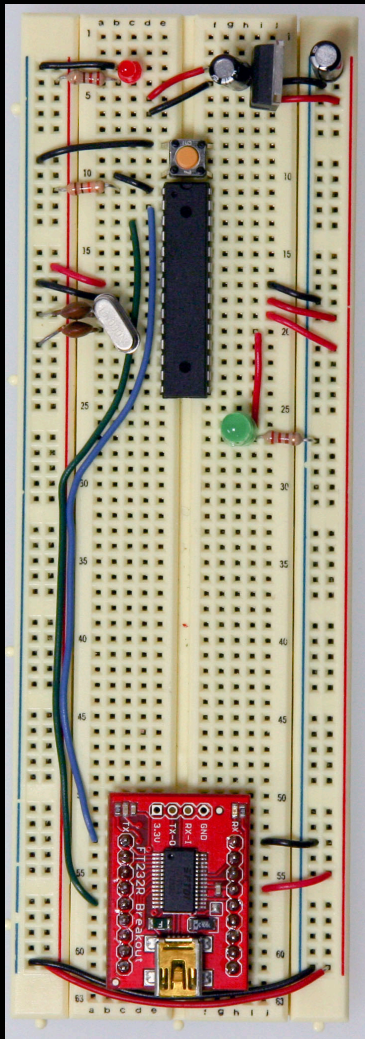
GPIO connectors

Communication connectors
(also GPIO)

Special function connectors



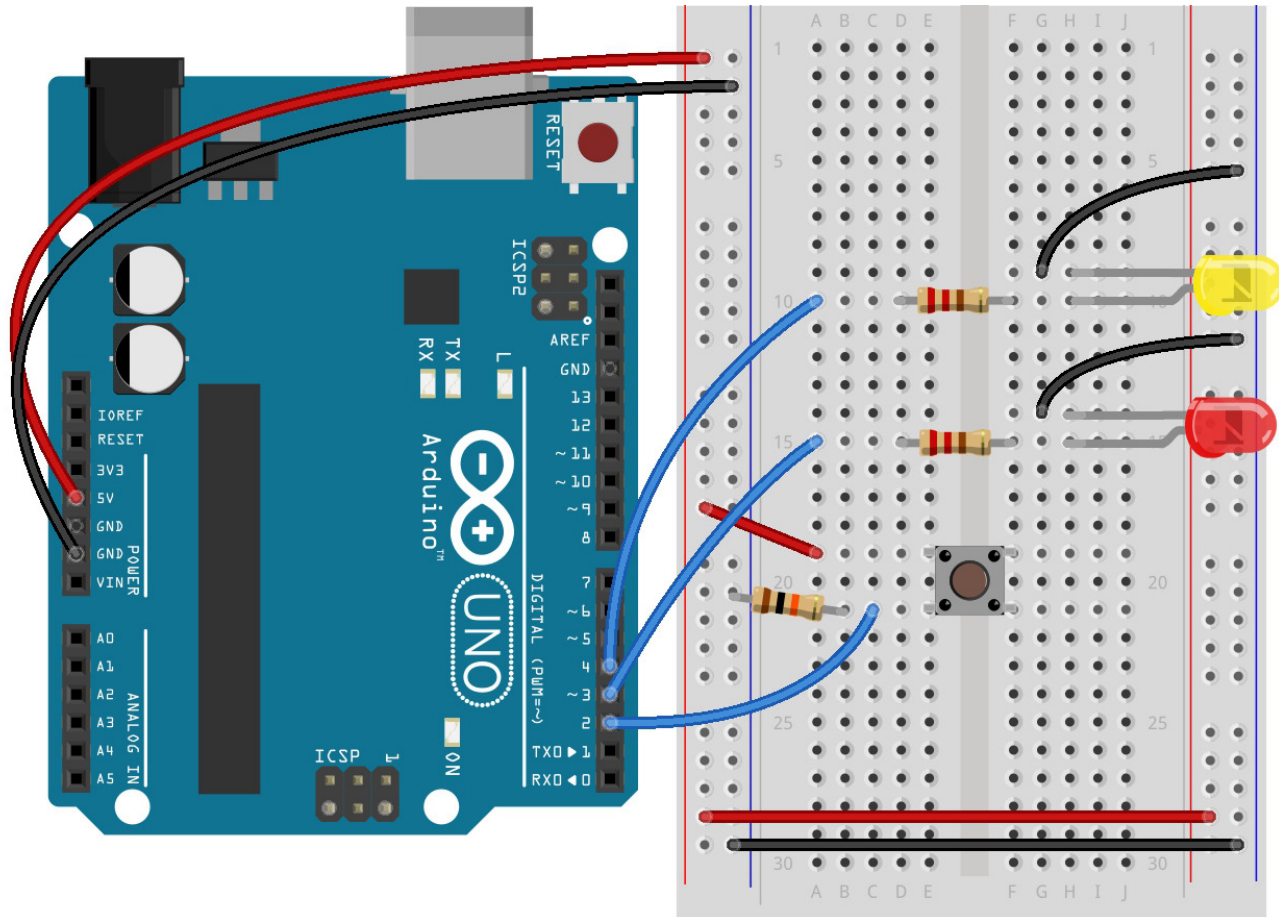
Arduino Open Source



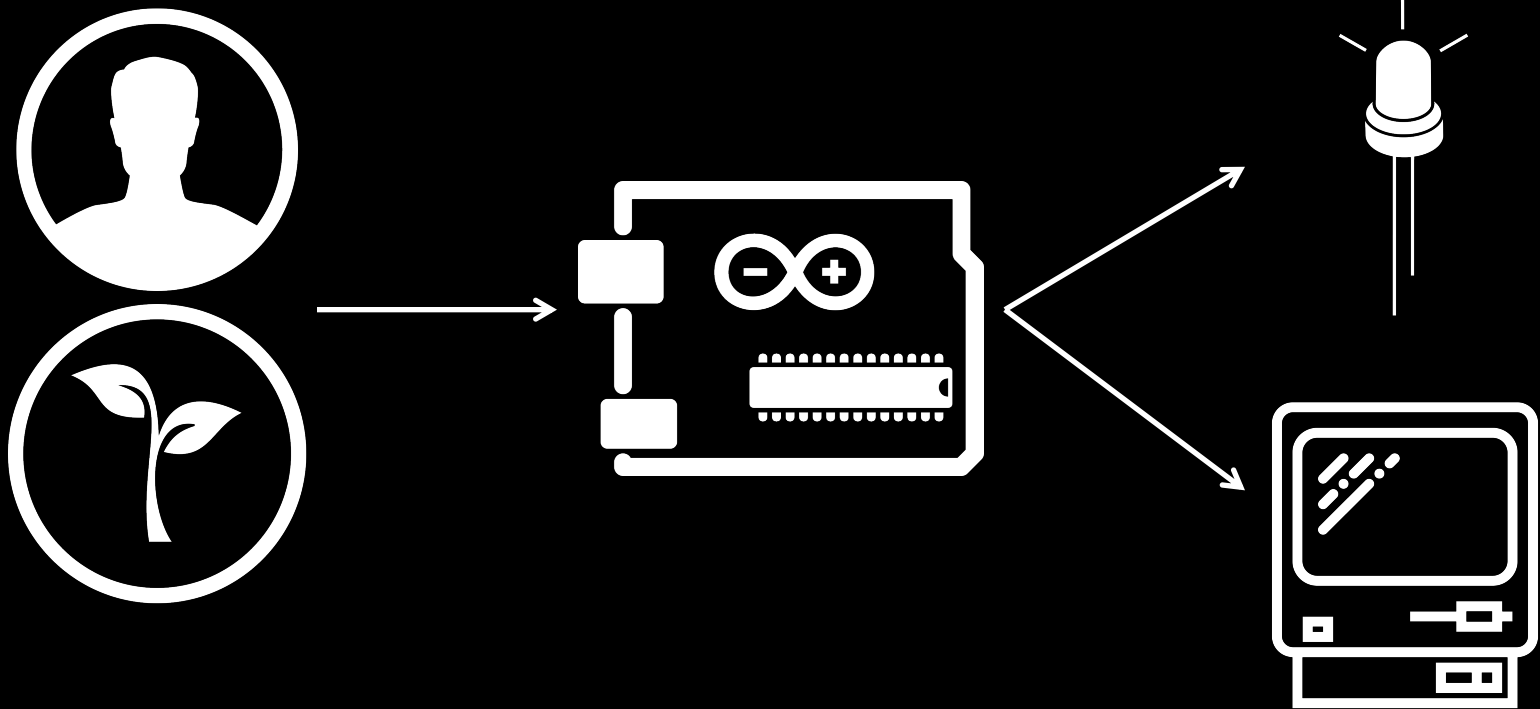
<https://www.arduino.cc/en/Main/Standalone>

<http://www.instructables.com/id/Beedduino-Homemade-Arduino-on-a-Budget/>

Arduino



Arduino

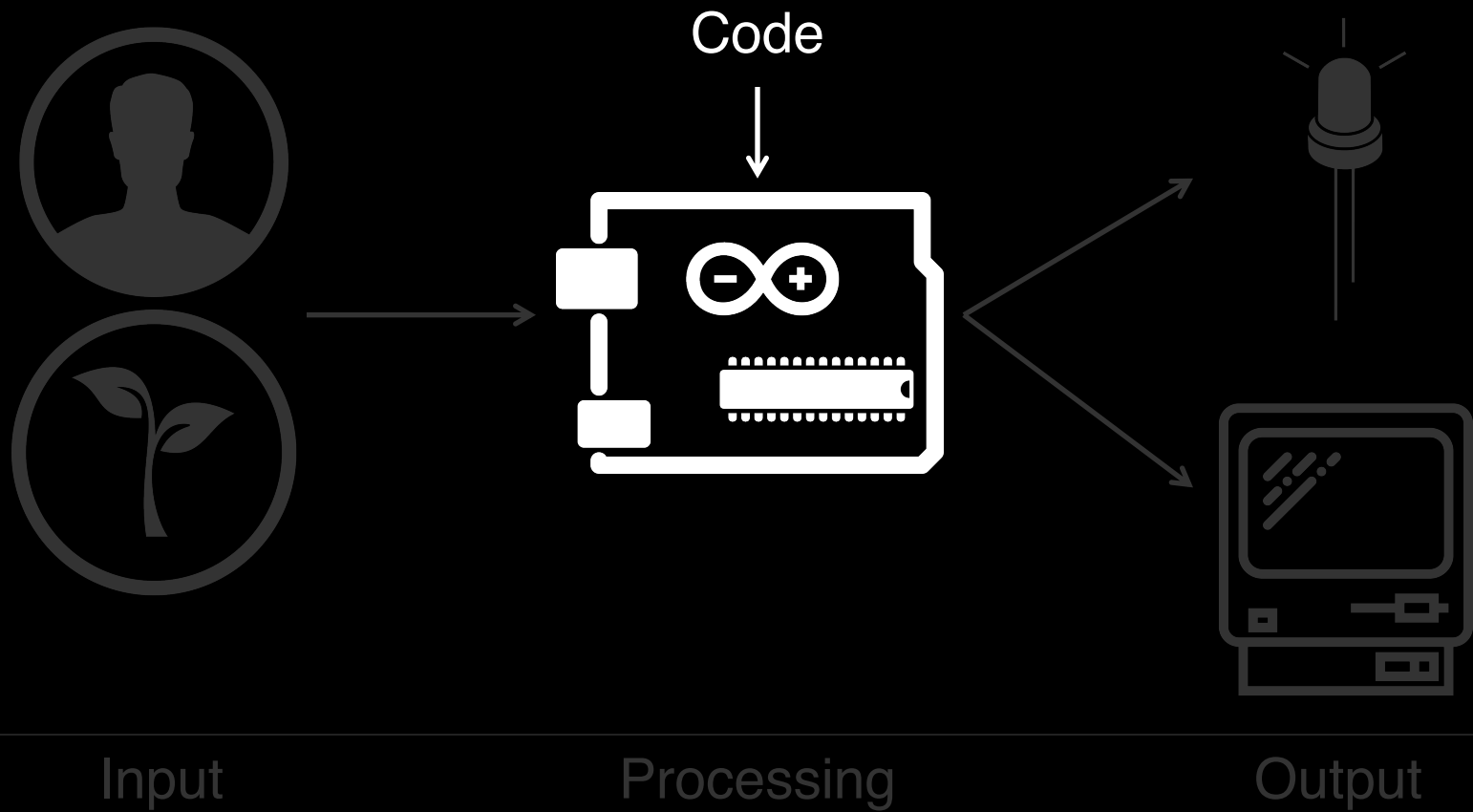


Input

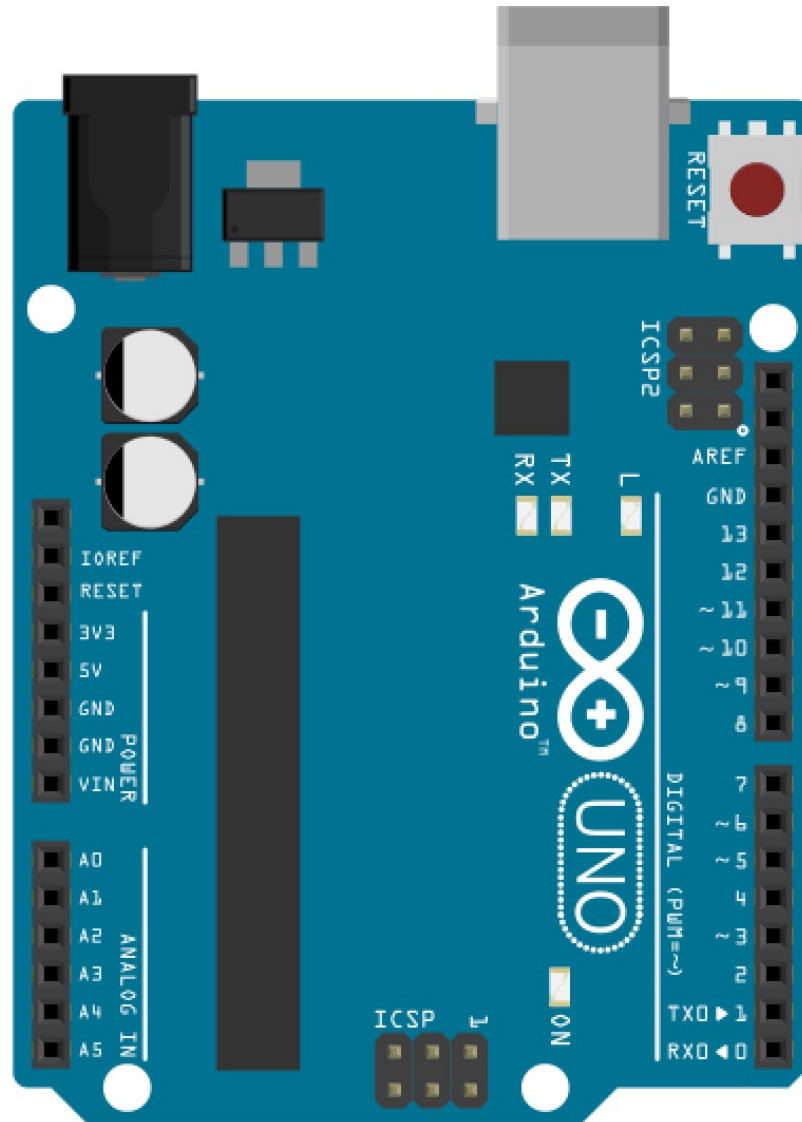
Processing

Output

Arduino

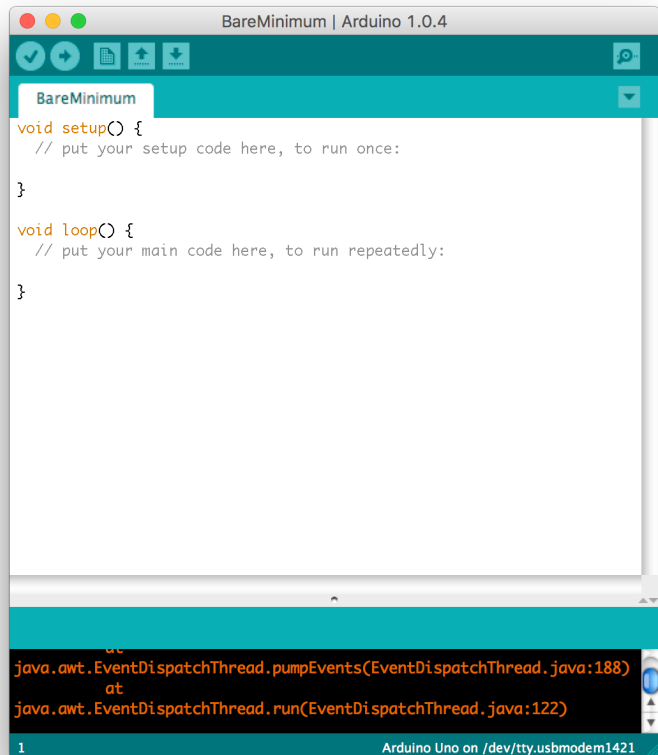


Arduino



Arduino IDE

Local IDE



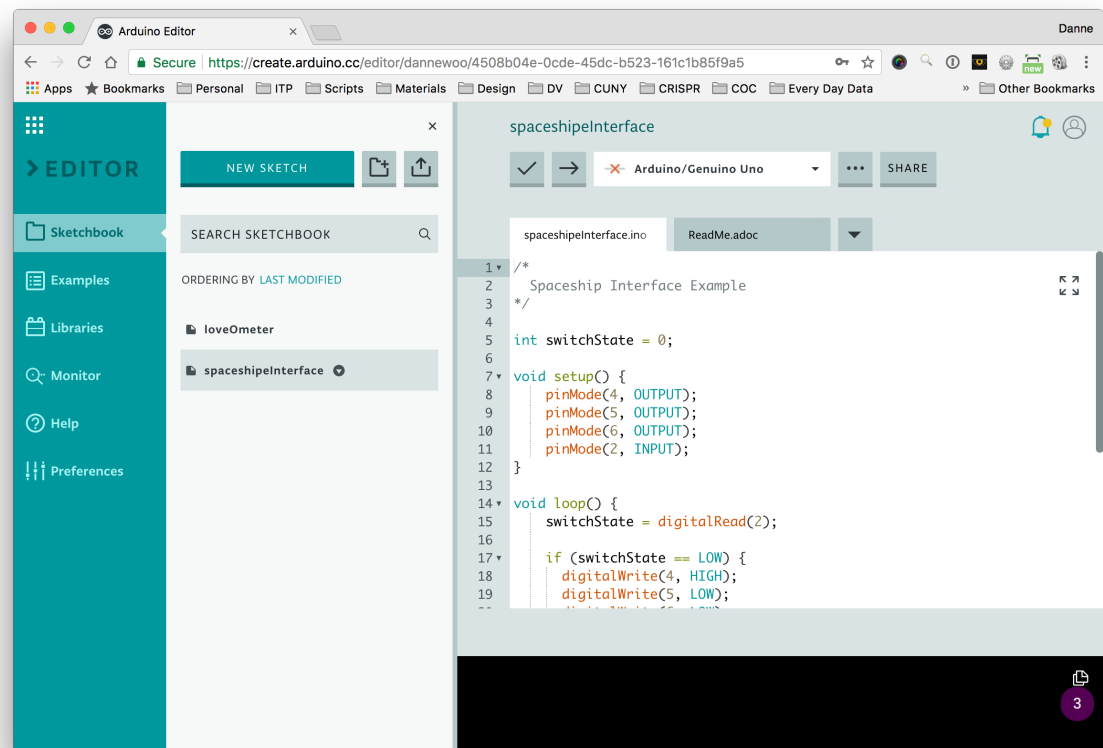
The screenshot shows the local Arduino IDE window titled "BareMinimum | Arduino 1.0.4". The interface includes a toolbar with icons for running, uploading, and saving. The main text area contains the following code:

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

At the bottom, a status bar indicates "Arduino Uno on /dev/tty.usbmodem1421". A terminal window at the bottom shows the following output:

```
java.awt.EventQueue.pumpEvents(EventDispatchThread.java:188)  
at  
java.awt.EventQueue.run(EventDispatchThread.java:122)
```

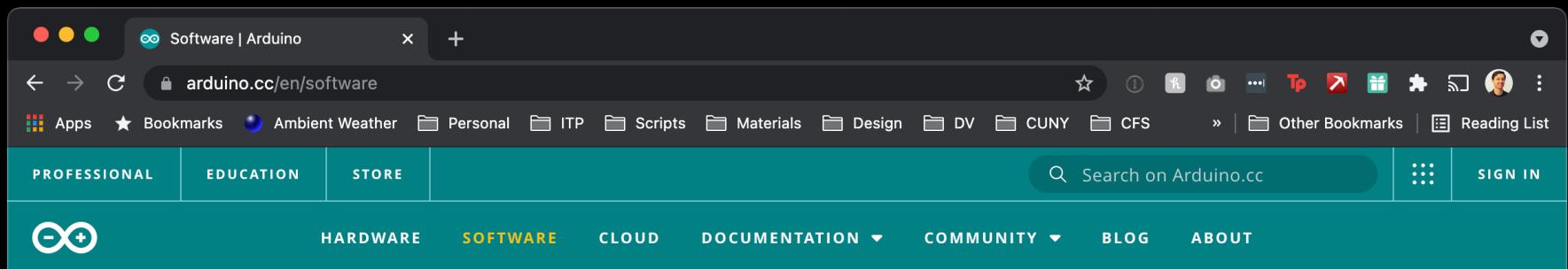
Web IDE



The screenshot shows the Arduino Web IDE interface in a browser. The URL is <https://create.arduino.cc/editor/dannewoo/4508b04e-0cde-45dc-b523-161c1b85f9a5>. The interface includes a sidebar with options like "NEW SKETCH", "Sketchbook", "Examples", "Libraries", "Monitor", "Help", and "Preferences". The main area displays a sketch titled "spaceshipInterface" for the "Arduino/Genuino Uno" board. The code is as follows:

```
spaceshipInterface.ino  
ReadMe.adoc  
1 /*  
2  * Spaceship Interface Example  
3  */  
4  
5 int switchState = 0;  
6  
7 void setup() {  
8   pinMode(4, OUTPUT);  
9   pinMode(5, OUTPUT);  
10  pinMode(6, OUTPUT);  
11  pinMode(2, INPUT);  
12 }  
13  
14 void loop() {  
15   switchState = digitalRead(2);  
16  
17   if (switchState == LOW) {  
18     digitalWrite(4, HIGH);  
19     digitalWrite(5, LOW);  
20   }
```

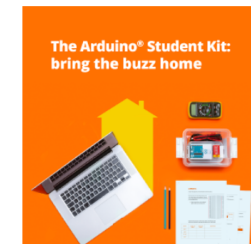
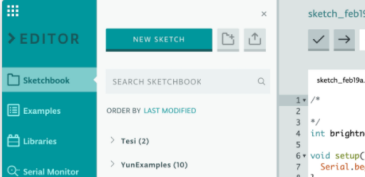
Arduino IDE



Arduino Web Editor

Start coding online and save your sketches in the cloud. The most up-to-date version of the IDE includes all libraries and also supports new Arduino boards.

[CODE ONLINE](#) [GETTING STARTED](#)



Downloads



Arduino IDE 1.8.16

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the [Getting Started](#) page for Installation instructions.

SOURCE CODE

Active development of the Arduino software is [hosted by GitHub](#). See the instructions for [building the code](#). Latest release source code archives are available [here](#). The archives are PGP-signed so

DOWNLOAD OPTIONS

Windows Win 7 and newer

Windows ZIP file

Windows app Win 8.1 or 10



Linux 32 bits

Linux 64 bits

Linux ARM 32 bits

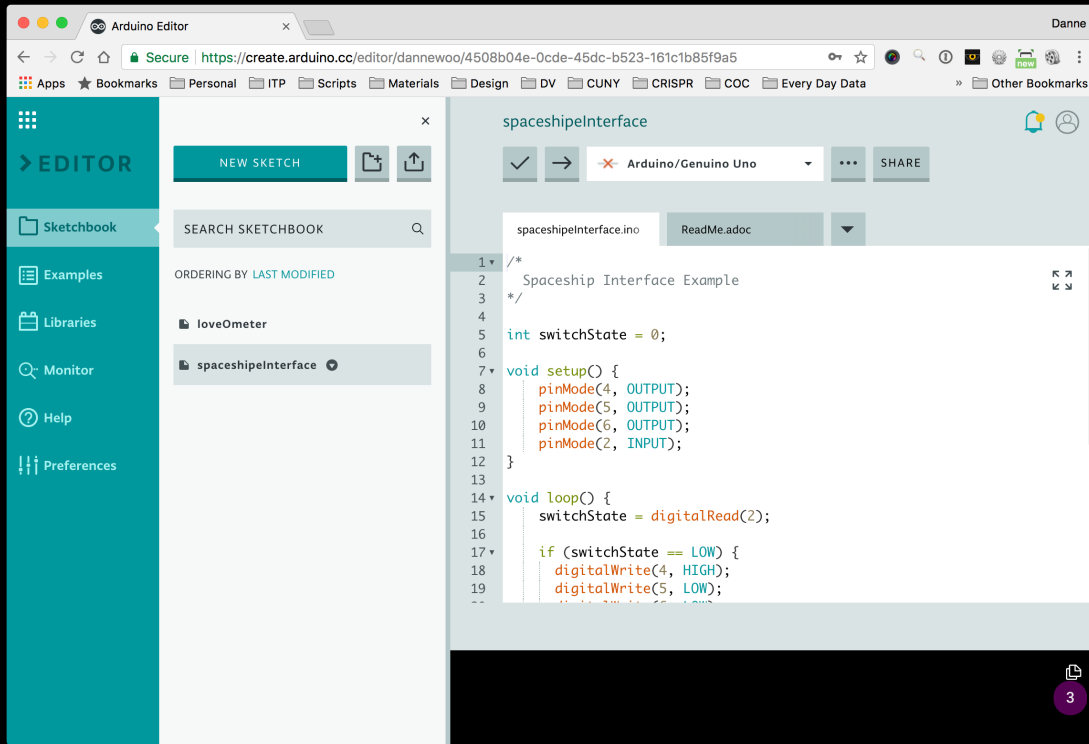
Linux ARM 64 bits

Mac OS X 10.10 or newer

[Release Notes](#) [Checksums \(sha512\)](#)



Arduino IDE



Toolbar / Arduino Connection

Tabs

Text Editor

Message Pane / Errors

Menu

Sub Menu

Processing

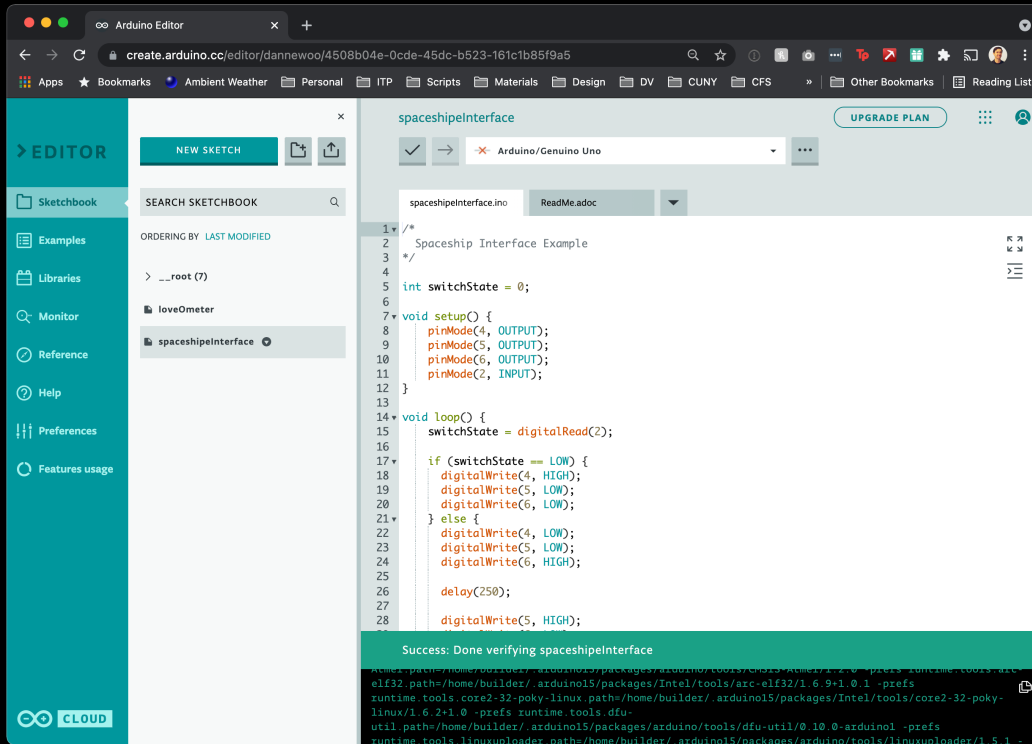
The screenshot displays the p5.js Web Editor interface. At the top, a browser window shows the URL `editor.p5js.org`. Below the browser, the p5.js logo is on the left, and navigation links for File, Edit, Sketch, and Help are in the center. On the right, there are links for English, Log in, and Sign up. Below the navigation bar, there are icons for running (play), stopping (square), and auto-refresh, along with the text "Cypress parade" and a settings gear icon. The main area is split into two panes: a code editor on the left and a preview area on the right. The code editor shows a file named `sketch.js` with the following code:

```
1 function setup() {  
2   createCanvas(400, 400);  
3 }  
4  
5 function draw() {  
6   background(220);  
7 }
```

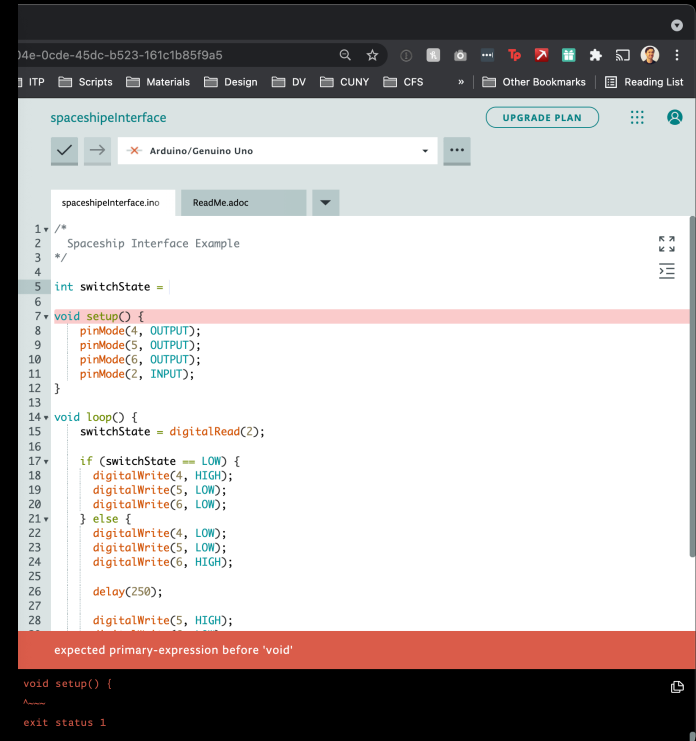
The preview area is currently blank. At the bottom, there is a console panel with the label "Console" and a "Clear" button with a dropdown arrow.

Message Pane

Success



Error



Sketchbook

The screenshot displays the Arduino IDE Sketchbook interface. The left sidebar contains navigation options: EDITOR, Sketchbook, Examples, Libraries, Monitor, Reference, Help, Preferences, and Features usage. The Sketchbook panel shows a search bar and a list of projects under 'Physical_Computing (2)', including 'Week_3 (2)' and 'Week_4 (2)'. The 'Button' project is selected. The main editor area shows the code for the 'Button' project, which is a simple LED control program. The code is as follows:

```
1 /*  
2  Queens College, Physical Computing Week 3  
3  Button  
4  Danne Woo  
5  */  
6  
7  int button = 2;  
8  int redLED = 3;  
9  int greenLED = 4;  
10  
11 void setup() {  
12   pinMode(button, INPUT);  
13   pinMode(redLED, OUTPUT);  
14   pinMode(greenLED, OUTPUT);  
15 }  
16  
17 void loop() {  
18   if (digitalRead(button) == HIGH) {  
19     digitalWrite(redLED, LOW);  
20     digitalWrite(greenLED, HIGH);  
21   } else {  
22     digitalWrite(redLED, HIGH);  
23     digitalWrite(greenLED, LOW);  
24   }  
25 }
```

At the bottom, a success message states: "Success: Done uploading Button". Below this, the avrdude output is visible:

```
avrdude: verifying ...  
avrdude: 924 bytes of flash verified  
avrdude done. Thank you.
```

Examples

The screenshot displays the Arduino Editor web interface. On the left, a teal sidebar contains the 'EDITOR' section with a list of options: Sketchbook, Examples (selected), Libraries, Monitor, Reference, Help, Preferences, and Features usage. Below this is the 'CLOUD' logo. The main content area is divided into two panes. The left pane, titled 'SEARCH EXAMPLES', shows a list of example categories under the heading 'SHOWING EXAMPLES FOR UNO'. The categories are: 01.BASICS (6), 02.DIGITAL (9), 03.ANALOG (6), 04.COMMUNICATION (12), 05.CONTROL (6), 06.SENSORS (4), 07.DISPLAY (2), 08.STRINGS (14), 09.USB (7), 10.STARTERKIT_BASICKIT (14), and 11.ARDUINOISP (1). The right pane shows the code editor for a sketch named 'Button'. The code is as follows:

```
1 /*
2  Queens College, Physical Computing Week 3
3  Button
4  Danne Woo
5  */
6
7  int button = 2;
8  int redLED = 3;
9  int greenLED = 4;
10
11 void setup() {
12   pinMode(button, INPUT);
13   pinMode(redLED, OUTPUT);
14   pinMode(greenLED, OUTPUT);
15 }
16
17 void loop() {
18   if (digitalRead(button) == HIGH) {
19     digitalWrite(redLED, LOW);
20     digitalWrite(greenLED, HIGH);
21   } else {
22     digitalWrite(redLED, HIGH);
23     digitalWrite(greenLED, LOW);
24   }
25 }
```

At the bottom of the editor, a status bar shows the upload progress: 'Success: Done uploading Button'. Below this, a terminal window displays the following output:

```
avrdude: verifying ...
avrdude: 924 bytes of flash verified
avrdude done. Thank you.
```

Libraries / Plugins

The screenshot displays the Arduino Editor web interface. On the left is a teal sidebar with navigation links: EDITOR, Sketchbook, Examples, Libraries (selected), Monitor, Reference, Help, Preferences, and Features usage. The main area is divided into two panes. The left pane is the 'LIBRARY MANAGER' with a search bar and tabs for DEFAULT, FAVORITES, and CUSTOM. It lists several libraries: ARDUINO UNO WIFI DEV ED LIBRARY, ARDUINOBEARSSL, ARDUINOCLOUDTHING, and ARDUINODMX. The right pane shows a code editor for a sketch named 'Button'. The code includes comments and defines pins for a button and two LEDs. At the bottom, a status bar shows a success message: 'Success: Done uploading Button'. Below that, a terminal window displays the upload progress: 'avrdude: verifying ...', 'avrdude: 924 bytes of flash verified', and 'avrdude done. Thank you.'

Arduino Editor

create.arduino.cc/editor/dannewoo/6e2f442a-859d-4a1f-85c2-1dbd6268f945

Apps Bookmarks Ambient Weather Personal ITP Scripts Materials Design DV CUNY CFS Other Bookmarks Reading List

EDITOR

- Sketchbook
- Examples
- Libraries**
- Monitor
- Reference
- Help
- Preferences
- Features usage

LIBRARY MANAGER

SEARCH LIBRARIES

SHOWING LIBRARIES FOR UNO

DEFAULT FAVORITES CUSTOM

This tab lists read-only libraries authored and maintained by the Arduino Team and its partners.

ARDUINO UNO WIFI DEV ED LIBRARY

This library allows users to use network features like rest and mqtt. Includes some tools for the ESP8266. [More info.](#)

> Examples

ARDUINOBEARSSL

Port of BearSSL to Arduino. [More info.](#)

> Examples

ARDUINOCLOUDTHING

Easily connect your Arduino/Genuino board to the Arduino Cloud. [More info.](#)

ARDUINODMX

Control DMX lights with your Arduino. [More info.](#)

Button

UPGRADE PLAN

Arduino Uno /dev/cu.usbmodem1421301

Button.ino ReadMe.adoc

```
1 /*
2  Queens College, Physical Computing Week 3
3  Button
4  Danne Woo
5  */
6
7  int button = 2;
8  int redLED = 3;
9  int greenLED = 4;
10
11 void setup() {
12   pinMode(button, INPUT);
13   pinMode(redLED, OUTPUT);
14   pinMode(greenLED, OUTPUT);
15 }
16
17 void loop() {
18   if (digitalRead(button) == HIGH) {
19     digitalWrite(redLED, LOW);
20     digitalWrite(greenLED, HIGH);
21   } else {
22     digitalWrite(redLED, HIGH);
23     digitalWrite(greenLED, LOW);
24   }
25 }
```

Success: Done uploading Button

avrdude: verifying ...
avrdude: 924 bytes of flash verified
avrdude done. Thank you.

Serial Monitor

The screenshot displays the Arduino IDE interface with the Serial Monitor open. The left sidebar shows the 'Monitor' tab selected. The main editor area displays the 'Button' sketch code. The Serial Monitor window at the bottom shows the upload progress and a success message.

Arduino IDE Interface:

- Top Bar:** Arduino Editor, create.arduino.cc/editor/dannewoo/6e2f442a-859d-4a1f-85c2-1dbd6268f945
- Left Sidebar:** EDITOR, Sketchbook, Examples, Libraries, Monitor (selected), Reference, Help, Preferences, Features usage.
- Serial Monitor Window:**
 - PORT: Arduino Uno at /dev/cu.usbmodem1421301
 - Settings: Newline, 9600 baud
 - SEND button
 - Autoscroll checkbox (checked)
 - DISCONNECT button
- Code Editor:**
 - File: Button
 - Buttons: Checkmark, Arrow, Arduino Uno, /dev/cu.usbmodem1421301, More options
 - Tabs: Button.ino (selected), ReadMe.adoc
 - Code:

```
1 /*
2  * Queens College, Physical Computing Week 3
3  * Button
4  * Danne Woo
5  */
6
7 int button = 2;
8 int redLED = 3;
9 int greenLED = 4;
10
11 void setup() {
12   pinMode(button, INPUT);
13   pinMode(redLED, OUTPUT);
14   pinMode(greenLED, OUTPUT);
15 }
16
17 void loop() {
18   if (digitalRead(button) == HIGH) {
19     digitalWrite(redLED, LOW);
20     digitalWrite(greenLED, HIGH);
21   } else {
22     digitalWrite(redLED, HIGH);
23     digitalWrite(greenLED, LOW);
24   }
25 }
```
- Serial Monitor Output:**
 - Success: Done uploading Button
 - avrdude: verifying ...
 - avrdude: 924 bytes of flash verified
 - avrdude done. Thank you.

Language Reference

The screenshot displays the Arduino IDE interface. On the left, a teal sidebar contains navigation links: EDITOR, Sketchbook, Examples, Libraries, Monitor, Reference (selected), Help, Preferences, and Features usage. The main workspace is divided into two panes. The left pane, titled 'Button', shows a search bar and tabs for FUNCTIONS, VARIABLES, and STRUCTURE. The FUNCTIONS tab is active, displaying a list of functions categorized by board family (Digital I/O, Analog I/O, Zero, Due & MKR Family, Advanced I/O, Time). The right pane shows the code for the 'Button' library, which includes a header file 'Button.h' and a source file 'Button.cpp'. The code defines a 'Button' class with a constructor and a 'loop()' function. The code is as follows:

```
1 /*
2  * Queens College, Physical Computing Week 3
3  * Button
4  * Danne Woo
5  */
6
7 int button = 2;
8 int redLED = 3;
9 int greenLED = 4;
10
11 void setup() {
12   pinMode(button, INPUT);
13   pinMode(redLED, OUTPUT);
14   pinMode(greenLED, OUTPUT);
15 }
16
17 void loop() {
18   if (digitalRead(button) == HIGH) {
19     digitalWrite(redLED, LOW);
20     digitalWrite(greenLED, HIGH);
21   } else {
22     digitalWrite(redLED, HIGH);
23     digitalWrite(greenLED, LOW);
24   }
25 }
```

At the bottom of the IDE, a status bar shows the upload progress: 'Success: Done uploading Button'. Below this, a terminal window displays the output of the upload process:

```
avrdude: verifying ...
avrdude: 924 bytes of flash verified
avrdude done. Thank you.
```

Text Editor

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

Programming Arduino

Functions

setup()

loop()

pinMode()

digitalRead()

digitalWrite()

analogRead()

analogWrite()

delay()

map()

Serial.begin() // BPS

Examples:

```
pinMode(3, OUTPUT);
```

```
digitalWrite(3, HIGH);
```

```
map(inputValue, 300, 750, 0, 255);
```

Variables

int (whole numbers)

float (decimals)

boolean (true or false)

string (text)

```
int variableName = 0;
```

HIGH

LOW

INPUT

OUTPUT

true

false

Programming Arduino

Conditional Statements

```
if (input == HIGH) {  
    // Do something  
} else {  
    // Do something else  
}
```

```
if (input == 1) {  
    // Do something for 1  
} else if (input == 2) {  
    // Do something for 2  
} else {  
    // Do something else  
}
```

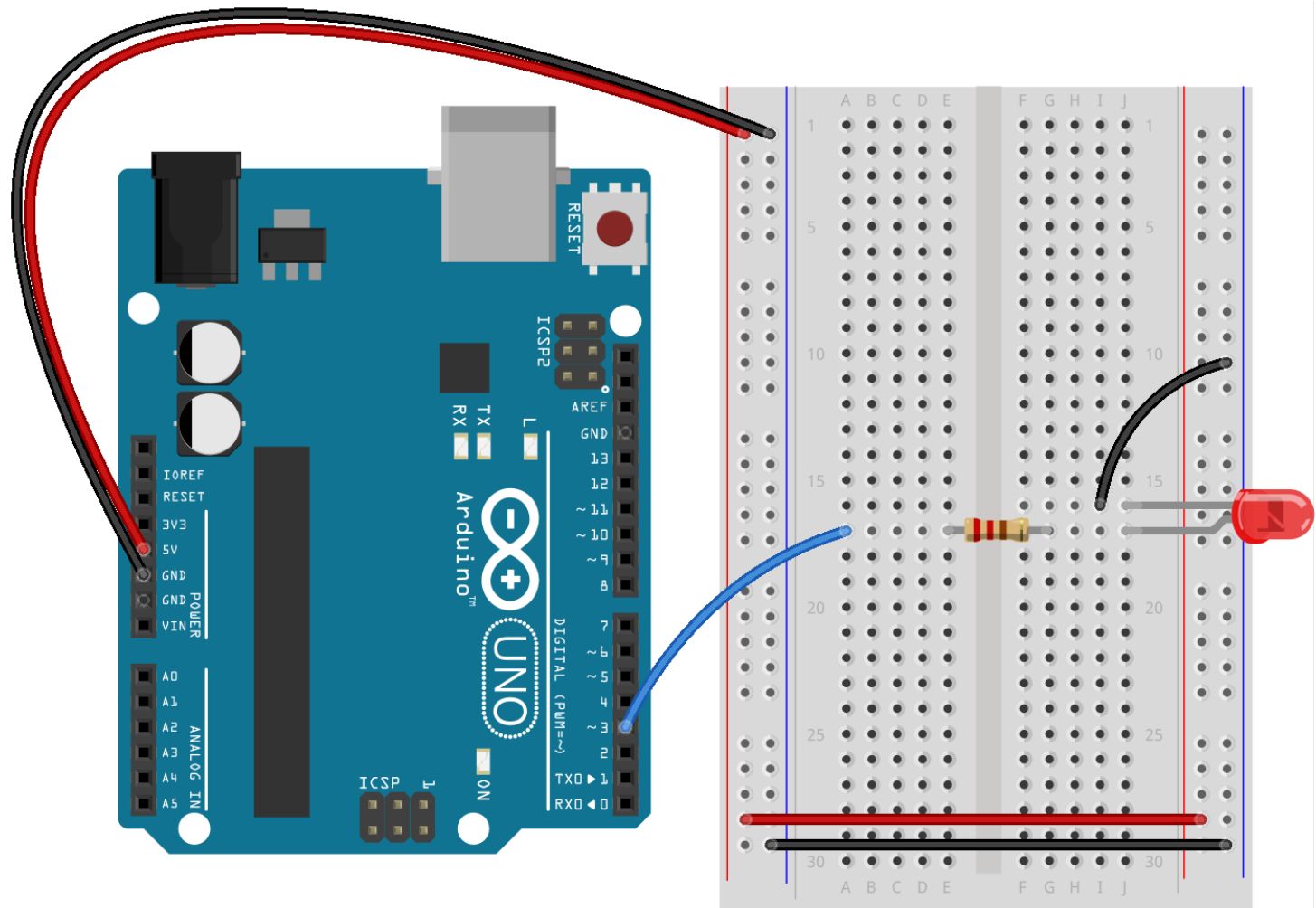
Comparison Operators

!= (not equal to)
< (less than)
<= (less than or equal to)
== (equal to)
> (greater than)
>= (greater than or equal to)

Arithmetic Operators

% (modulo)
* (multiplication)
+ (addition)
- (subtraction)
/ (division)
= (assignment operator)

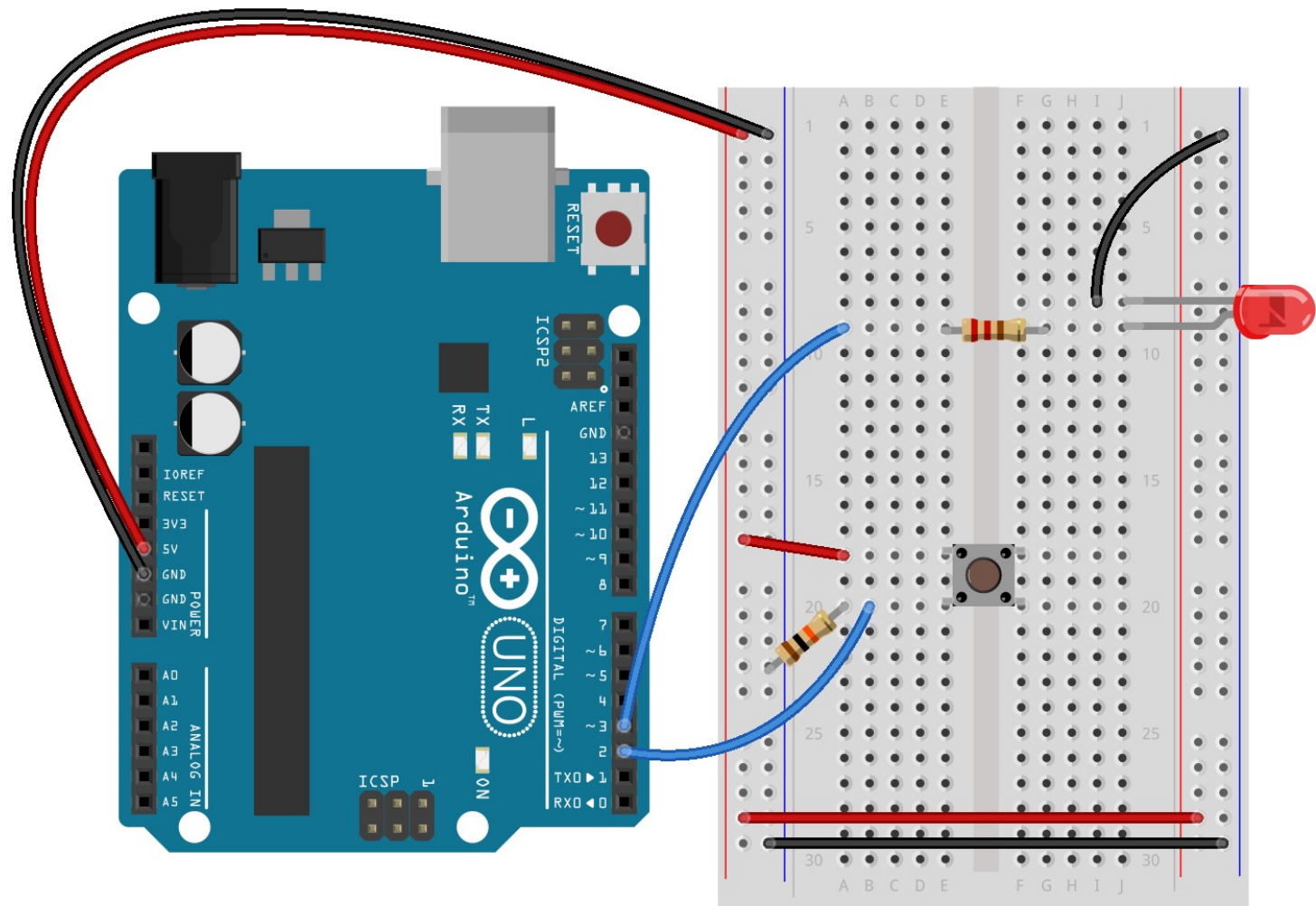
Digital Write



Digital Write

```
void setup() {  
  // Define which pin you are using and if it is input or output.  
  pinMode(3, OUTPUT);  
}  
  
void loop() {  
  // Send power to pin 3  
  digitalWrite(3, HIGH);  
  // Delay for 1000 milliseconds or 1 second  
  delay(1000);  
  // Turn off power to pin 3  
  digitalWrite(3, LOW);  
  // Delay for 200 milliseconds  
  delay(200);  
}
```

Digital In and Digital Out

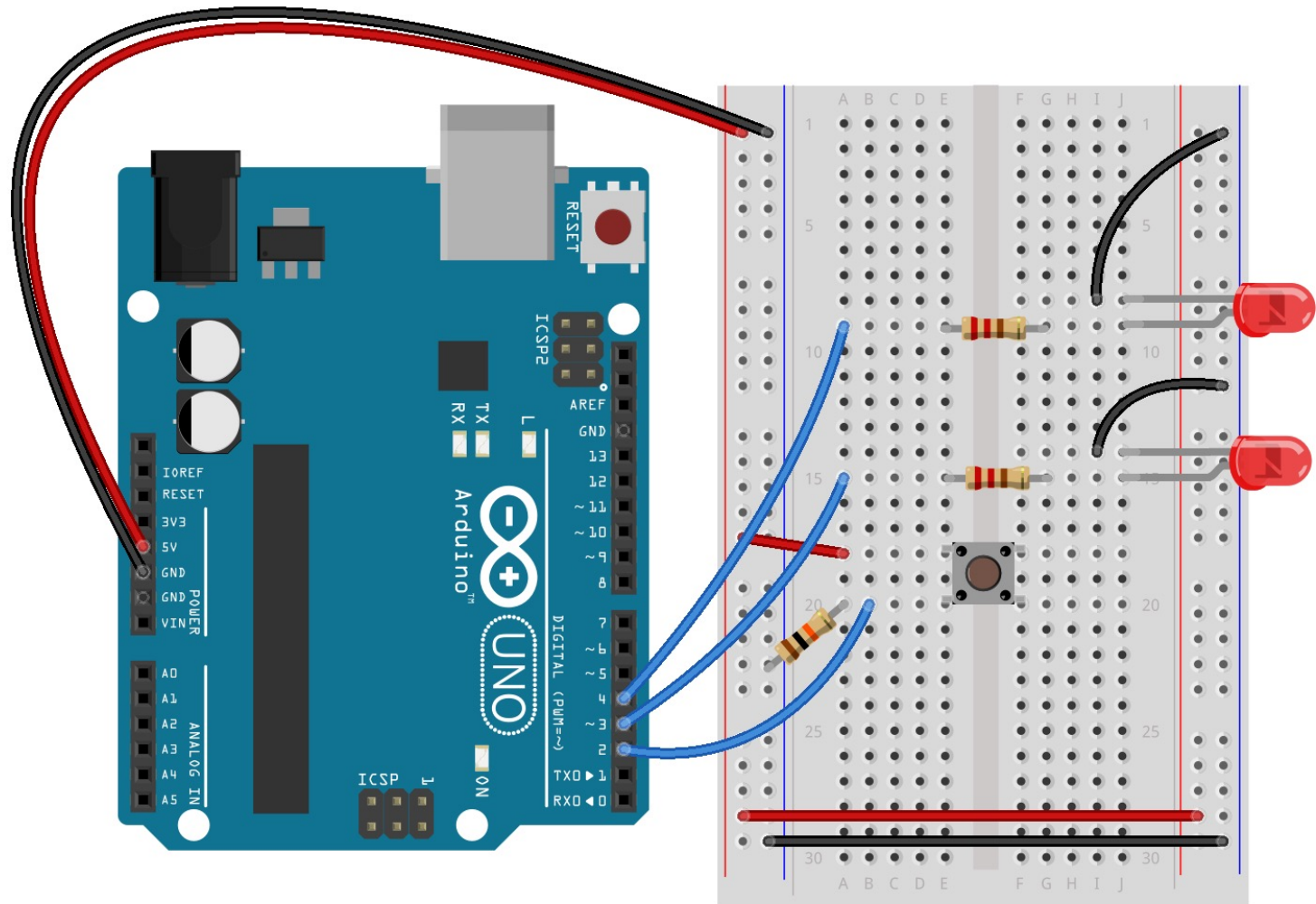


fritzing

Digital In and Digital Out

```
void setup() {  
  pinMode(2, INPUT);  
  pinMode(3, OUTPUT);  
}  
  
void loop() {  
  if (digitalRead(2) == HIGH) {  
    digitalWrite(3, HIGH);  
  } else {  
    digitalWrite(3, LOW);  
  }  
}
```

Digital In and Digital Out



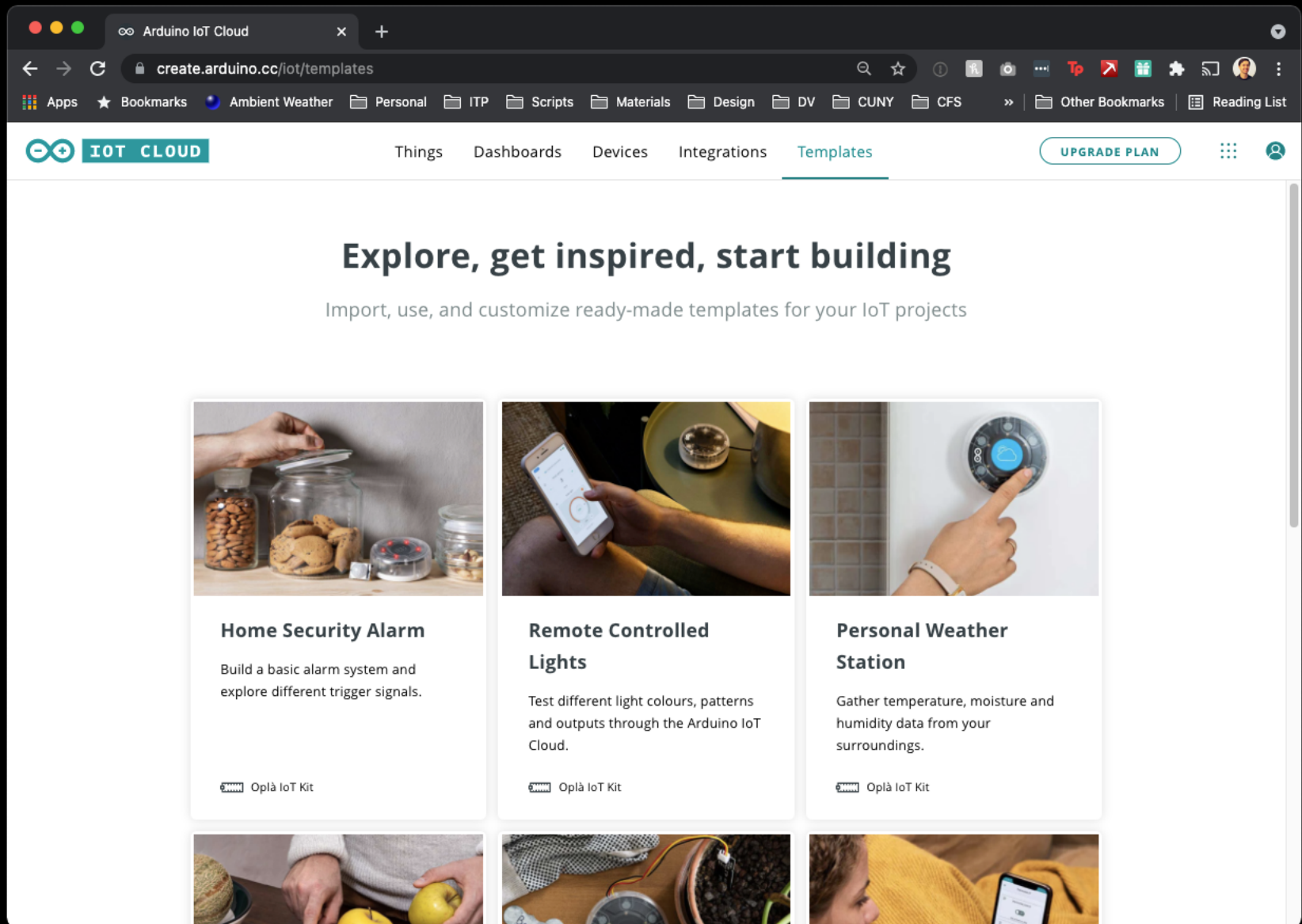
Digital In and Digital Out

```
void setup() {  
  pinMode(2, INPUT);  
  pinMode(3, OUTPUT);  
  pinMode(4, OUTPUT);  
}  
  
void loop() {  
  if (digitalRead(2) == HIGH) {  
    digitalWrite(3, HIGH);  
    digitalWrite(4, LOW);  
  } else {  
    digitalWrite(3, LOW);  
    digitalWrite(4, HIGH);  
  }  
}
```

Hack Existing Products



Arduino IoT Cloud



The screenshot shows the Arduino IoT Cloud website interface. The browser address bar displays 'create.arduino.cc/iot/templates'. The navigation bar includes 'IOT CLOUD' and links for 'Things', 'Dashboards', 'Devices', 'Integrations', and 'Templates'. A 'UPGRADE PLAN' button is visible on the right. The main heading is 'Explore, get inspired, start building', followed by the subtitle 'Import, use, and customize ready-made templates for your IoT projects'. Below this, three project templates are featured:

- Home Security Alarm**: Build a basic alarm system and explore different trigger signals. Includes an 'Oplà IoT Kit' logo.
- Remote Controlled Lights**: Test different light colours, patterns and outputs through the Arduino IoT Cloud. Includes an 'Oplà IoT Kit' logo.
- Personal Weather Station**: Gather temperature, moisture and humidity data from your surroundings. Includes an 'Oplà IoT Kit' logo.

Below these, the tops of three more templates are visible: one with fruit, one with a plant, and one with a person using a smartphone.

In Class/Homework

Review the labs that were covered in class this week, and come up with a simple yet creative application using digital input and output. Post your experiments to your blog. Include videos and images.

Physical Computing

Professor Danne Woo

danne.woo@qc.cuny.edu

pcomp.dannewoo.com