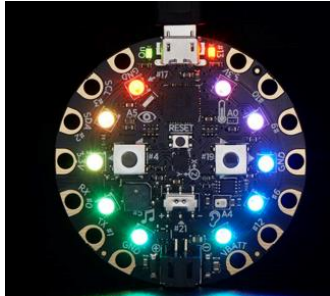


## Assignment 4: Introduction to Arduino. Colorful timer.

The purpose of this assignment is to help you get started with Arduino. Refer to slides from week 5 for more information about first steps with Arduino. Also, see the Appendix in this document for getting started with the software Arduino IDE and running your first program.

### Instructions



In this assignment, you are going to start working with an Arduino board. You will be using the Adafruit Circuit Playground. For this assignment, you will only need the Arduino board and the usb charger provided in class. You will not need the android phone. You will use the extra components that we distributed for the next (and last) assignment), and possibly for your final project.

### Submission

Submit your assignment using Canvas, under the Assignment section. The due date is shown in Canvas. You must submit a **.zip** file, containing the following items:

- Arduino Project folder
- README.txt file (see instructions below)
- video file with a demo.

The file named README.txt should contain your name and email address and a very brief description of it, along with any special instructions that the user might need to know in order to use it properly (if there are any). For example:

Laura Arjona <arjona@uw.edu>

Arduino Counter – I used the left button instead of the right just to confuse the instructor.

## Colorful Timer

In this project you are going to design a colorful timer. Every “time step”, one NeoPixel LED will light up with a different color (of your choice). When all the LEDs are lighted up, it means that the timer has expired. Then, play a tone, blink 2 times all the LEDs, and turn them all off.

1. Start the counter when the user presses the right button.
2. Define a total duration for your timer (for example, 10 seconds?)
3. Define a time step to light every LED. For example, if you had 10 LEDs and a counter of 10 seconds, one led will light every second. Once one LED has lighted up, it should remain lighted up until the timer expires.
4. When the timer expires (this means that all LEDs should be on), blink twice all the LEDs simultaneously, and turn them all off.
5. Bonus: Shake the board to restart the counter. See appendix for help function.

### To get started, follow the instructions in the appendix of this document:

1. Install Arduino IDE
2. Select the board and port in the IDE
3. Install the Circuit Playground library
4. And voila!

### Grading

- Up to 80 points: minimum requirements (functionalities exposed above), and overall performance.
- 90-100 points: shake detection to restart the counter.

**Some helping code for the shake detection**

```

    . . .
# define SHAKE_THRESHOLD    30    // Total acceleration threshold for shake detect
    . . .

void setup() {
    . . .
    . . .
    CircuitPlayground.begin();
    CircuitPlayground.clearPixels();
    CircuitPlayground.setAccelRange(LIS3DH_RANGE_8_G);
    . . .
    . . .
}

float getTotalAccel() {
    // Compute total acceleration
    float X = 0;
    float Y = 0;
    float Z = 0;
    for (int i=0; i<10; i++) {
        X += CircuitPlayground.motionX();
        Y += CircuitPlayground.motionY();
        Z += CircuitPlayground.motionZ();
        delay(1);
    }
    X /= 10;
    Y /= 10;
    Z /= 10;

    return (what should it return ???-> absolute value )
}
}

```

## Getting started with Arduino Circuit Playground

### Download Latest Arduino IDE

Download the latest Arduino IDE

[Download here](#)

### Install Drivers (**Windows 7 Only**)

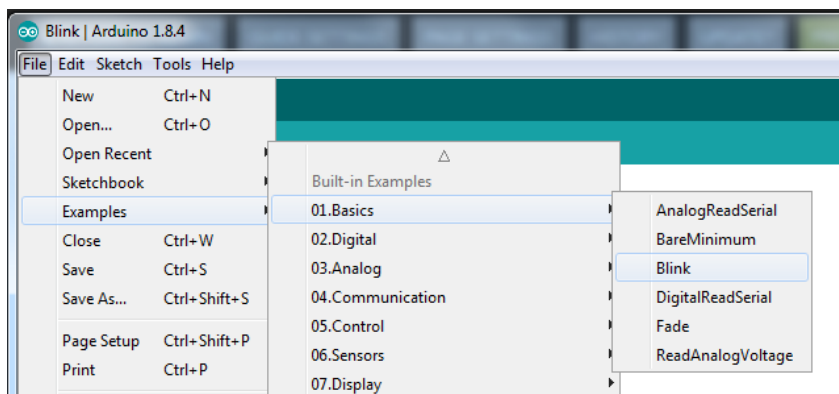
When you plug in the board, you'll need to possibly install a driver

[Click here to download our Driver Installer](#)

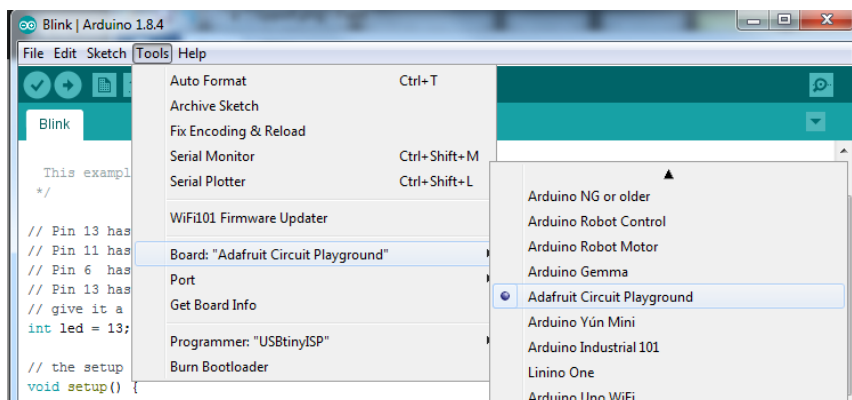
### Blink

Now you can upload your first blink sketch!

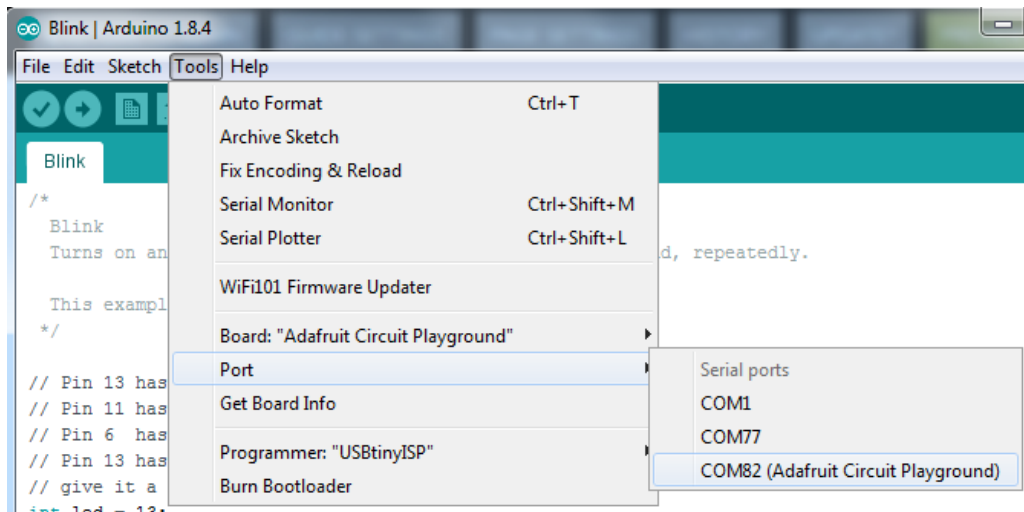
Open up the Blink example from the Arduino IDE



Select **Circuit Playground** from the Tools -> Board dropdown menu



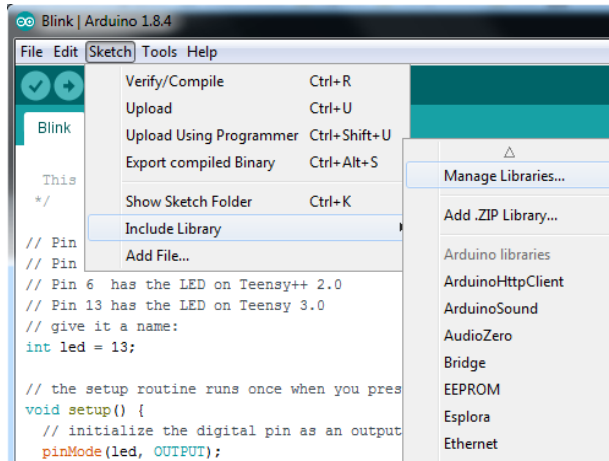
Plug in the Circuit Playground and wait for it to be recognized by the OS (just takes a few seconds). It will create a serial/COM port, you can now select it from the dropdown, it'll even be 'indicated' as a Circuit Playground board!



**And click upload!** That's it, you will be able to see the LED blink rate change as you adapt the **delay()** calls.

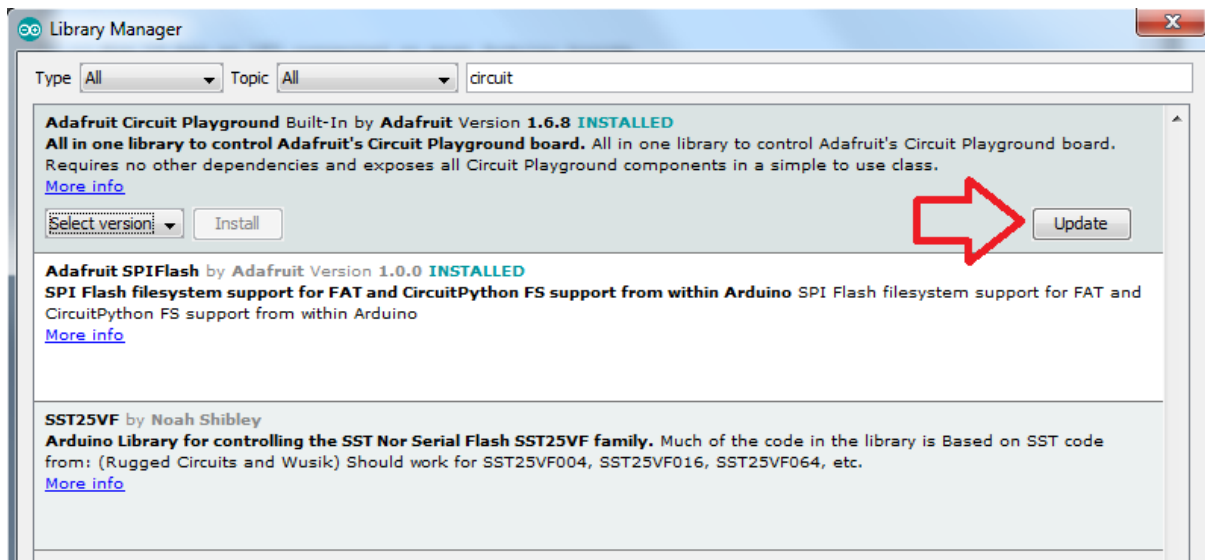
## Install the CircuitPlayground Library

### Installing Via Library Manager



- In the menubar click "Sketch", then "Include Library"
- At the top, click "Manage Libraries. . ."

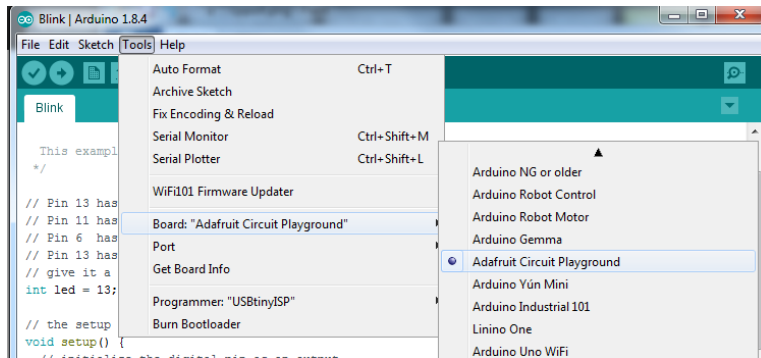
Type **Circuit** in the search box. You should see **Adafruit Circuit Playground** listed. Then click **Update** to get the very latest version!



Run the sensor demo that you can find in Canvas under the assignment 4 folder

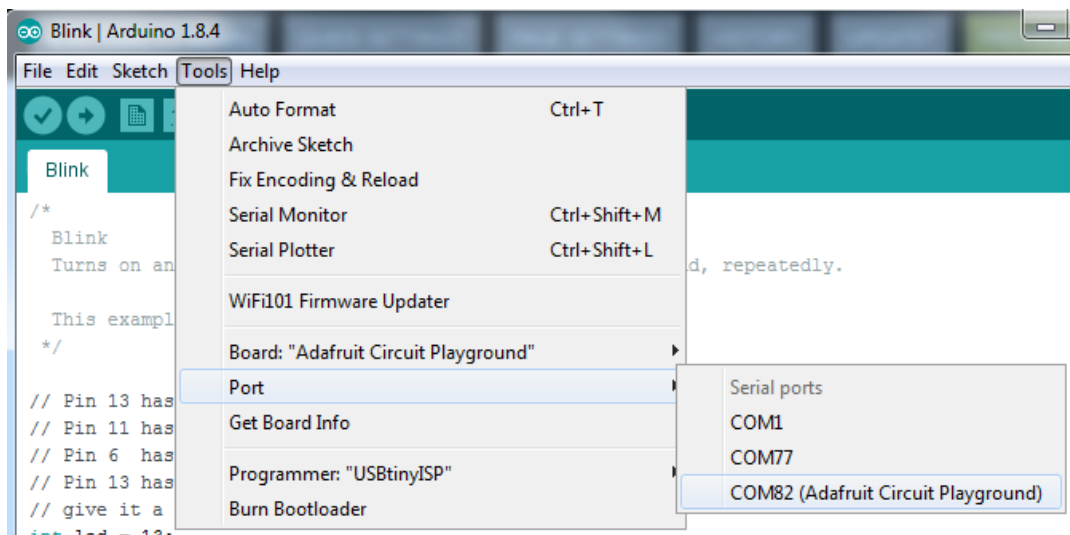
### Select the Circuit Playground Board

Under the **Tools -> Board** submenu, pick **Adafruit Circuit Playground**

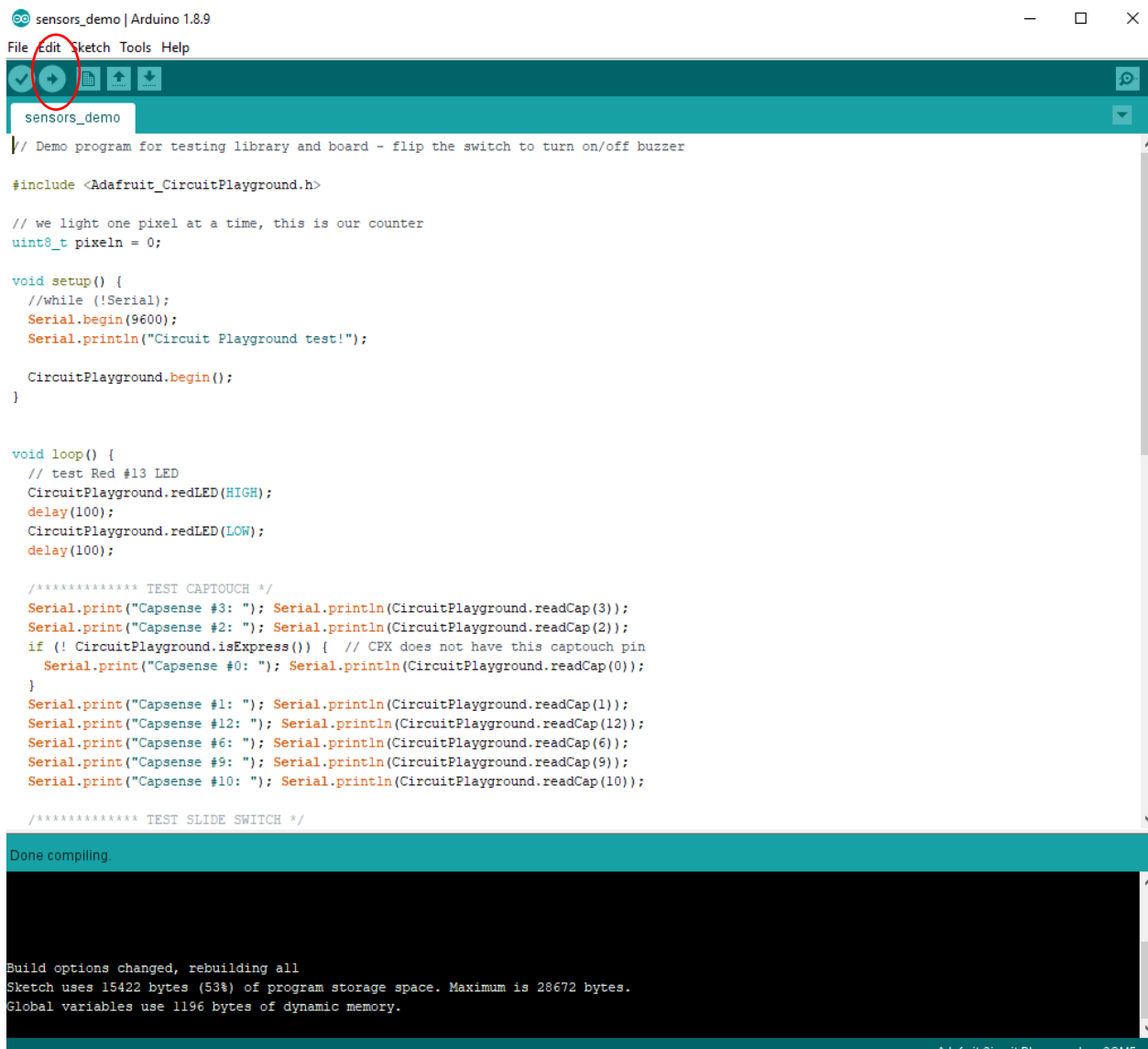


### Select the matching Port

Under **Tools->Port** select the port that is labeled (Circuit Playground)



## Load the program on the board



```
sensors_demo | Arduino 1.8.9
File Edit Sketch Tools Help

sensors_demo

// Demo program for testing library and board - flip the switch to turn on/off buzzer

#include <Adafruit_CircuitPlayground.h>

// we light one pixel at a time, this is our counter
uint8_t pixeln = 0;

void setup() {
  //while (!Serial);
  Serial.begin(9600);
  Serial.println("Circuit Playground test!");

  CircuitPlayground.begin();
}

void loop() {
  // test Red #13 LED
  CircuitPlayground.redLED(HIGH);
  delay(100);
  CircuitPlayground.redLED(LOW);
  delay(100);

  /***** TEST CAPTOUCH */
  Serial.print("Capsense #3: "); Serial.println(CircuitPlayground.readCap(3));
  Serial.print("Capsense #2: "); Serial.println(CircuitPlayground.readCap(2));
  if (! CircuitPlayground.isExpress()) { // CPX does not have this captouch pin
    Serial.print("Capsense #0: "); Serial.println(CircuitPlayground.readCap(0));
  }
  Serial.print("Capsense #1: "); Serial.println(CircuitPlayground.readCap(1));
  Serial.print("Capsense #12: "); Serial.println(CircuitPlayground.readCap(12));
  Serial.print("Capsense #6: "); Serial.println(CircuitPlayground.readCap(6));
  Serial.print("Capsense #9: "); Serial.println(CircuitPlayground.readCap(9));
  Serial.print("Capsense #10: "); Serial.println(CircuitPlayground.readCap(10));

  /***** TEST SLIDE SWITCH */

Done compiling.

Build options changed, rebuilding all
Sketch uses 15422 bytes (53%) of program storage space. Maximum is 28672 bytes.
Global variables use 1196 bytes of dynamic memory.
```

Make sure you get **"Done compiling."** and no errors



You can now run the serial console (**Tools-> Serial Monitor**) to get data output:

