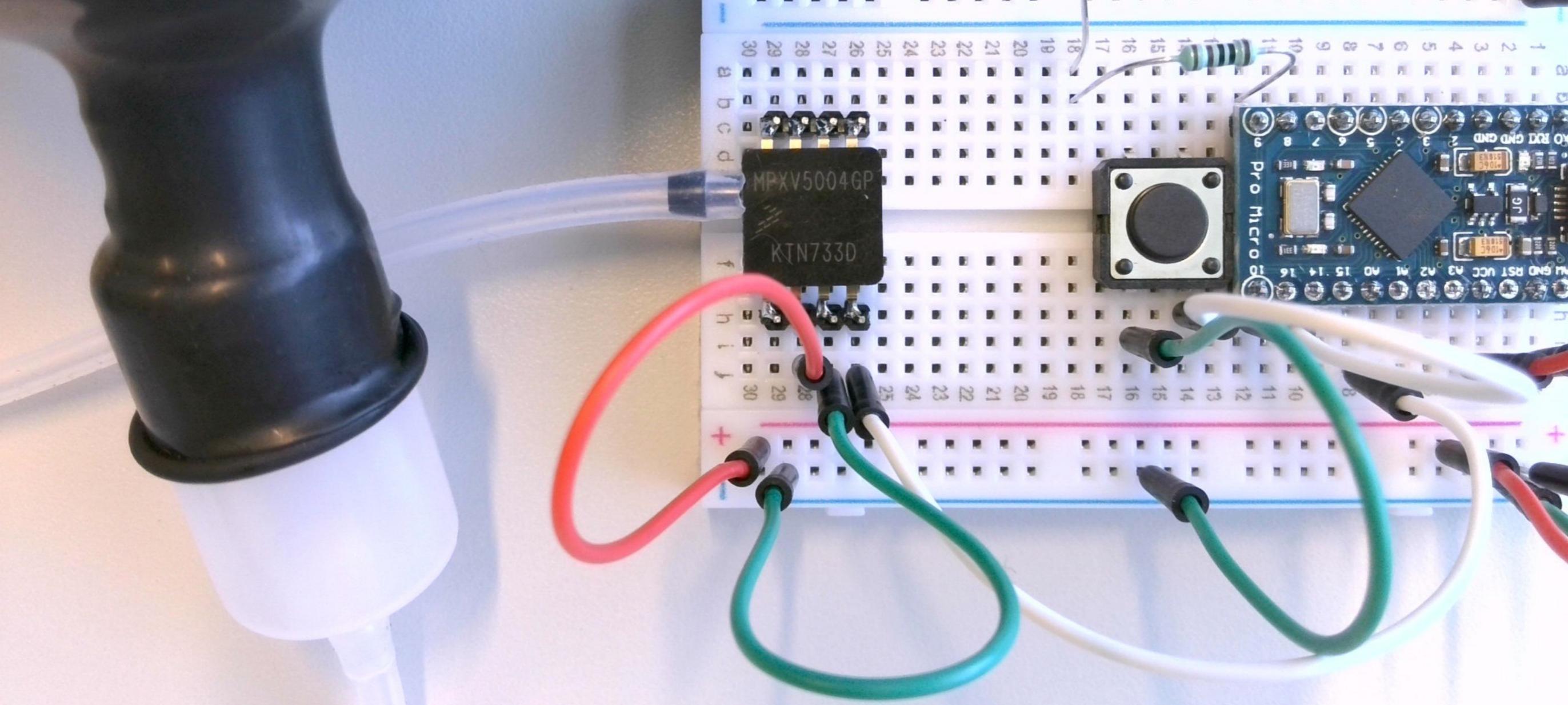


---

# DEFORMATION

*UWTMC*

*Matt Borland - 2019*



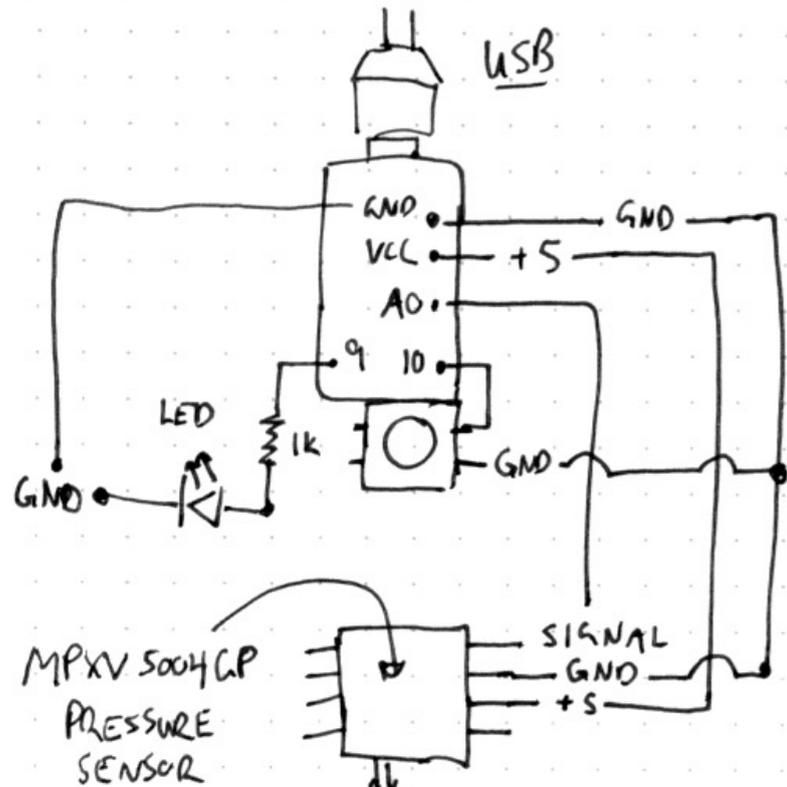
## WE'RE MAKING MUSICAL BALLOONS THAT USE DEFORMATION TO CREATE MUSIC

.....

*We'll use sensors to measure RATE of deformation and AMOUNT of deformation, then convert those to MIDI messages which your computer will then turn into musical sounds!*

*Check out a fancier version of this here: <https://vimeo.com/326637479>*

# UWTMC DEFORMATION

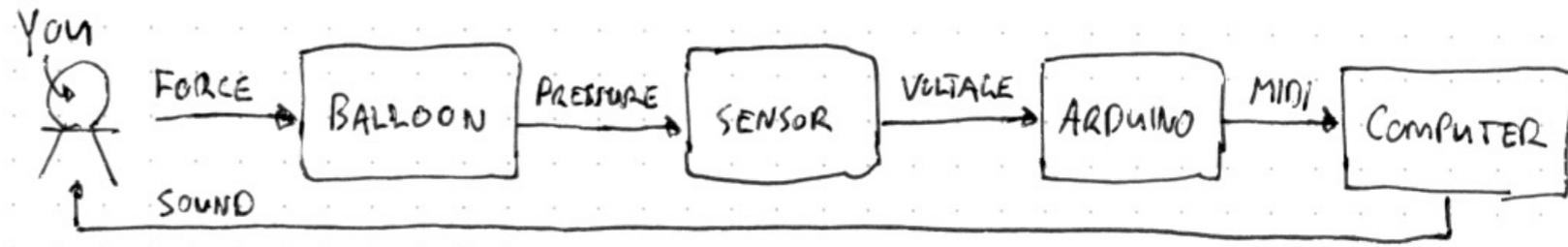


BE CAREFUL!  
HOOKING THIS  
UP WRONG WILL  
RUIN THE SENSOR!

MEASURES  
"Gauge Pressure"

$$P_{Gauge} = P_{BALLOON} - P_{ATMOSPHERIC}$$

CONVERTED TO AN ANALOG  
VOLTAGE (0-5V) READ BY  
THE ARDUINO AND TURNED  
INTO VALUES (0-1023) WHICH  
ARE USED TO DECIDE  
WHICH MIDI MESSAGES TO SEND



## MIDI: MUSICAL INSTRUMENT DIGITAL INTERFACE

WE SEND TWO TYPES OF MESSAGES TO OUR  
COMPUTERS TO BE TURNED INTO SOUNDS.

### ① NOTE ON

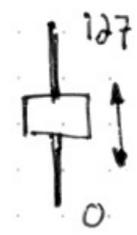
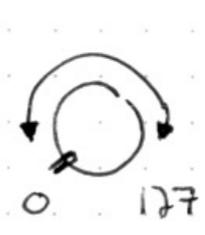


### NOTE OFF



CHANNEL  
1-16  
VELOCITY  
0-127  
PITCH  
0-127

### ② CONTROL CHANGE



POSITION OF A  
KNOB OR A  
SLIDER.

CHANNEL , CC NUM , VALUE  
1-16 , 0-127 , 0-127

128 KNOBS!

# DOWNLOADS

---

*You need two pieces of software. Both are free and multi-platform!*

Download the Arduino IDE



The screenshot shows the Arduino IDE download page. On the left, there is a teal circle with a white infinity symbol containing a minus and a plus sign. To its right, the text reads: **ARDUINO 1.8.10**. Below this, it says: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions." On the right side of the page, there are download links for Windows (installer and ZIP file), Windows app, Mac OS X, and Linux (32 bits, 64 bits, ARM 32 bits, ARM 64 bits). At the bottom right, there are links for Release Notes, Source Code, and Checksums (sha512).



*Arduino IDE - a software platform used to program your microcontroller.*

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

*Helm - a software synthesizer to make musical sounds with your computer.*

<https://tytel.org/helm/>

# FILES

Workshops this term are at CML! All sessions run 3:30-5:30pm at Critical Media Lab, located in Communitech, 151 Charles St. W., Kitchener.

Sept. 18th: Deformation - MIDI Balloons: [Deformation Files.zip](#)

Oct. 2nd: Gravity - Shruti Box

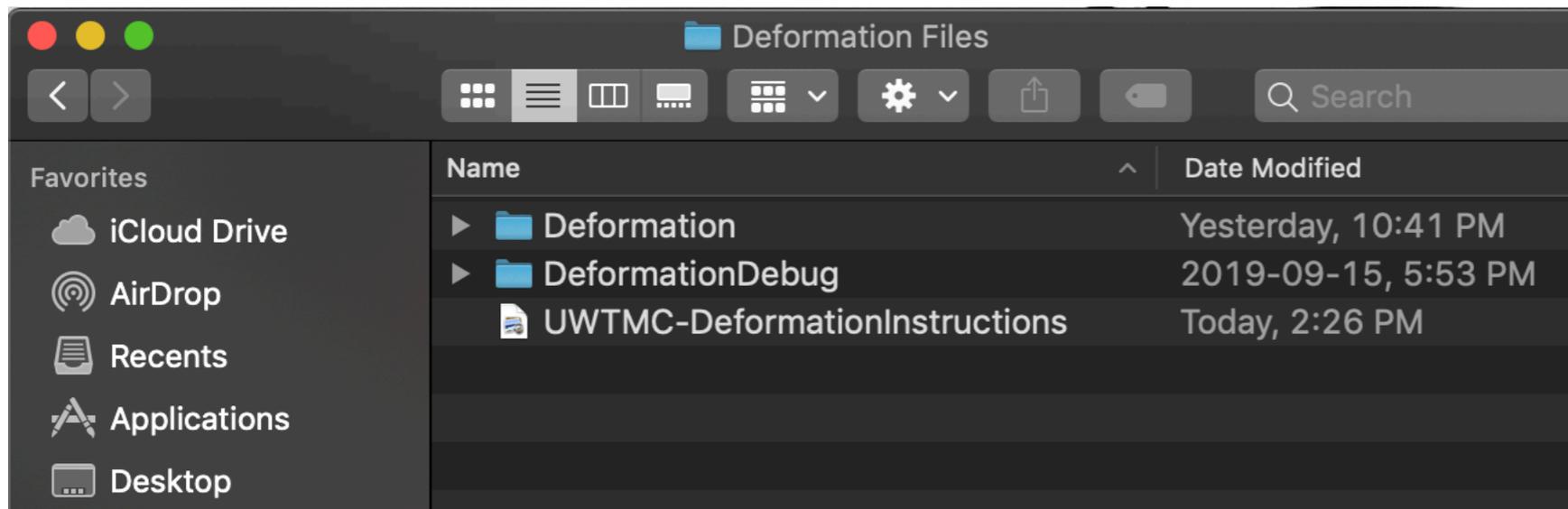
Oct. 23rd: Continuity - Seaboard and Bop Pad

Nov. 6th: Complexity - Modular Synthesis

Nov. 20th: Exploration - Co-play Patch Tables



Files are available as a ZIP at <https://uwtmc.com>



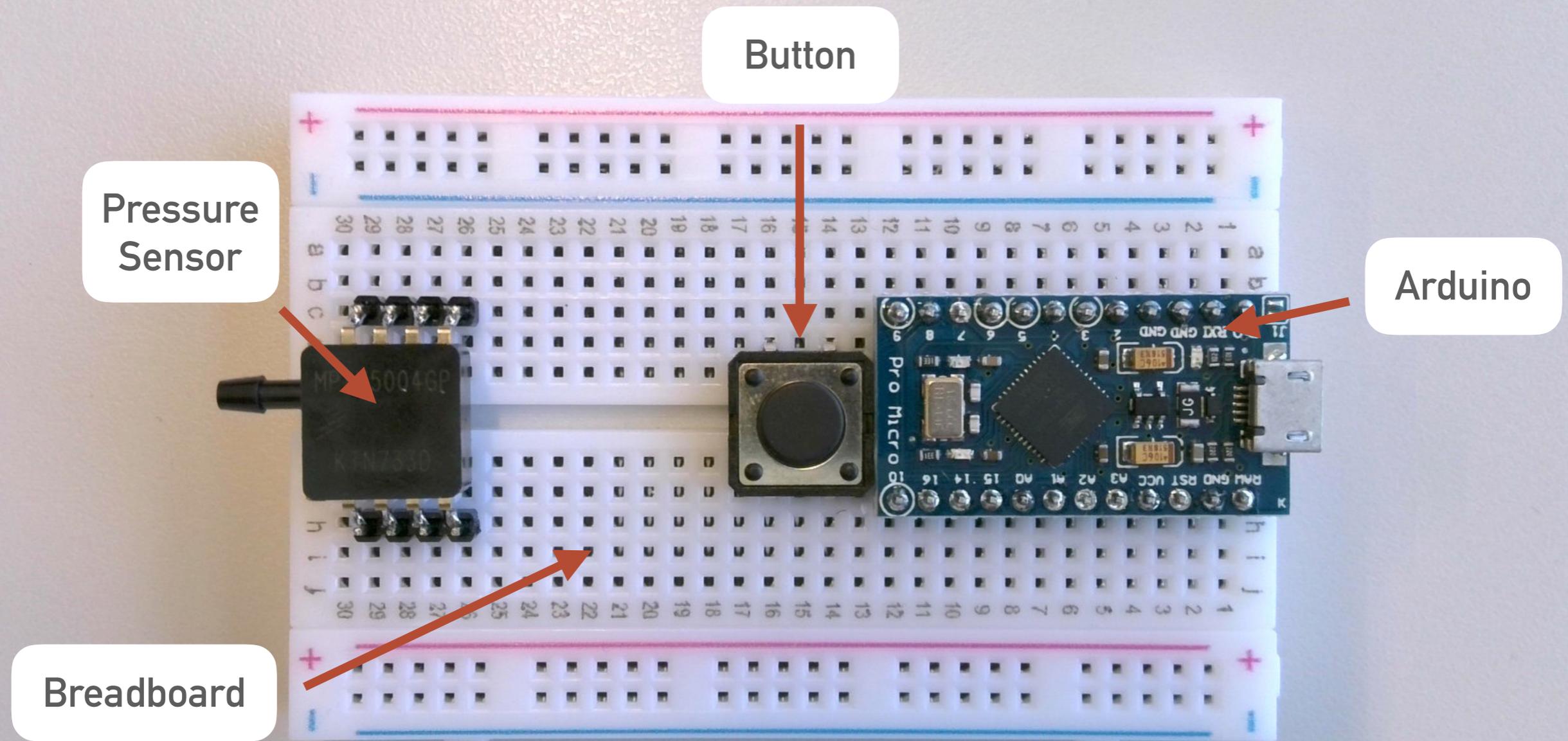
LET'S DO IT!

---

*Matt will explain everything, so follow along with him on the big screen. Ask questions if you're unsure - other people are probably in the same boat.*

# HOOKING UP YOUR SENSORS! FOLLOW MB ON THE PROJECTOR

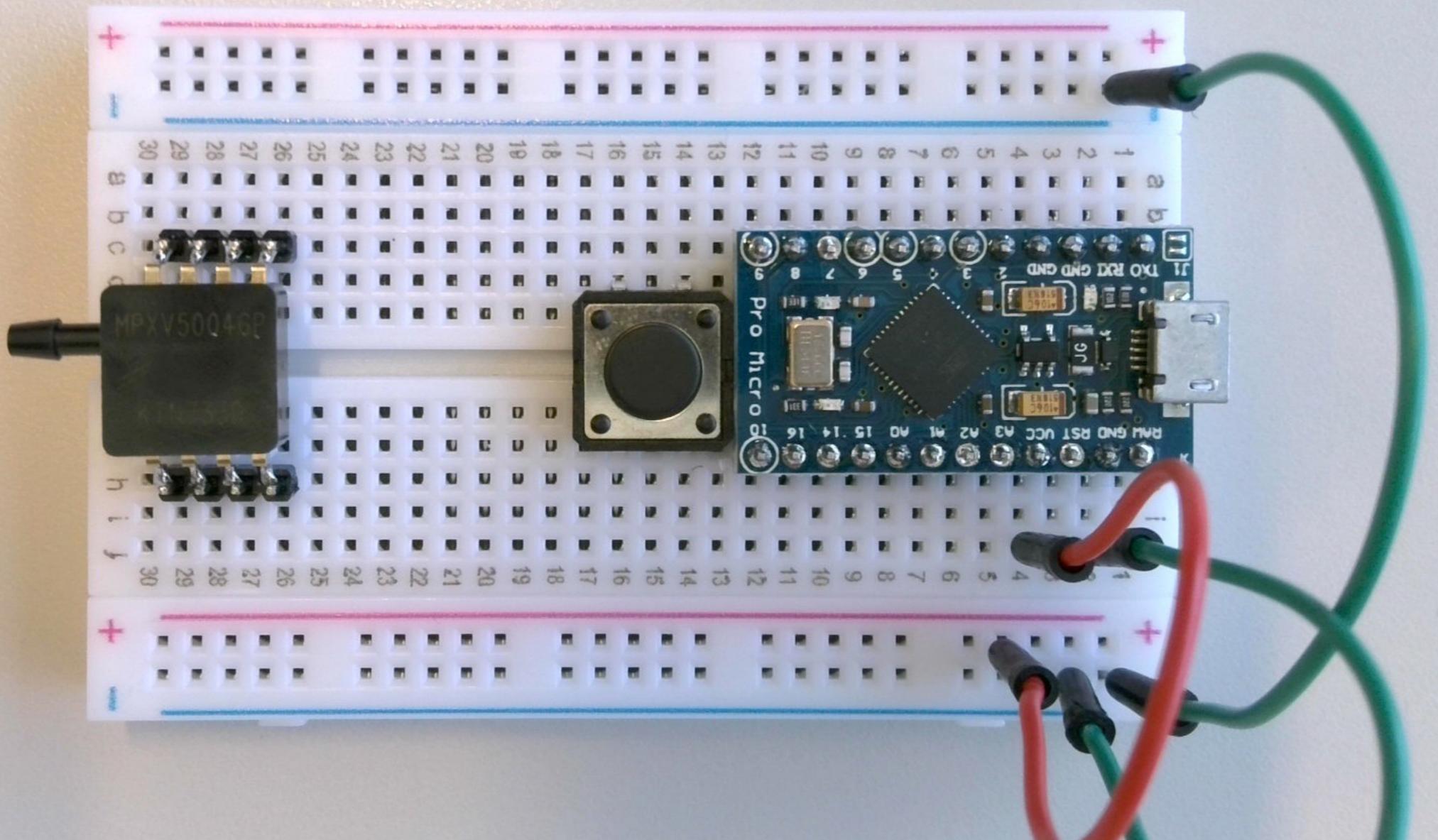
---



# HOOKING UP YOUR SENSORS! POWER

---

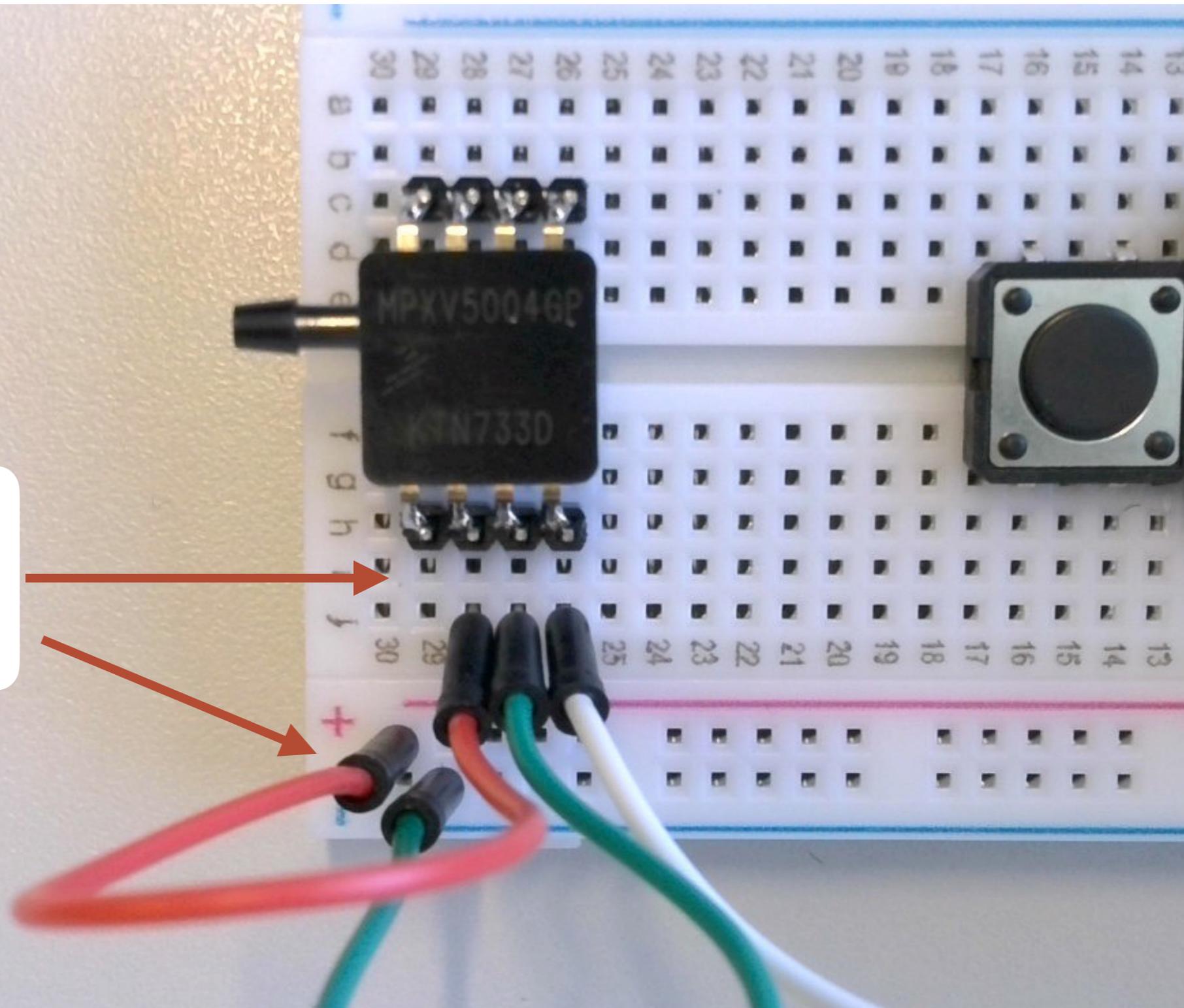
GND to the -ve Blue Row  
VCC to the +ve Red Row



# HOOKING UP YOUR SENSORS! PRESSURE

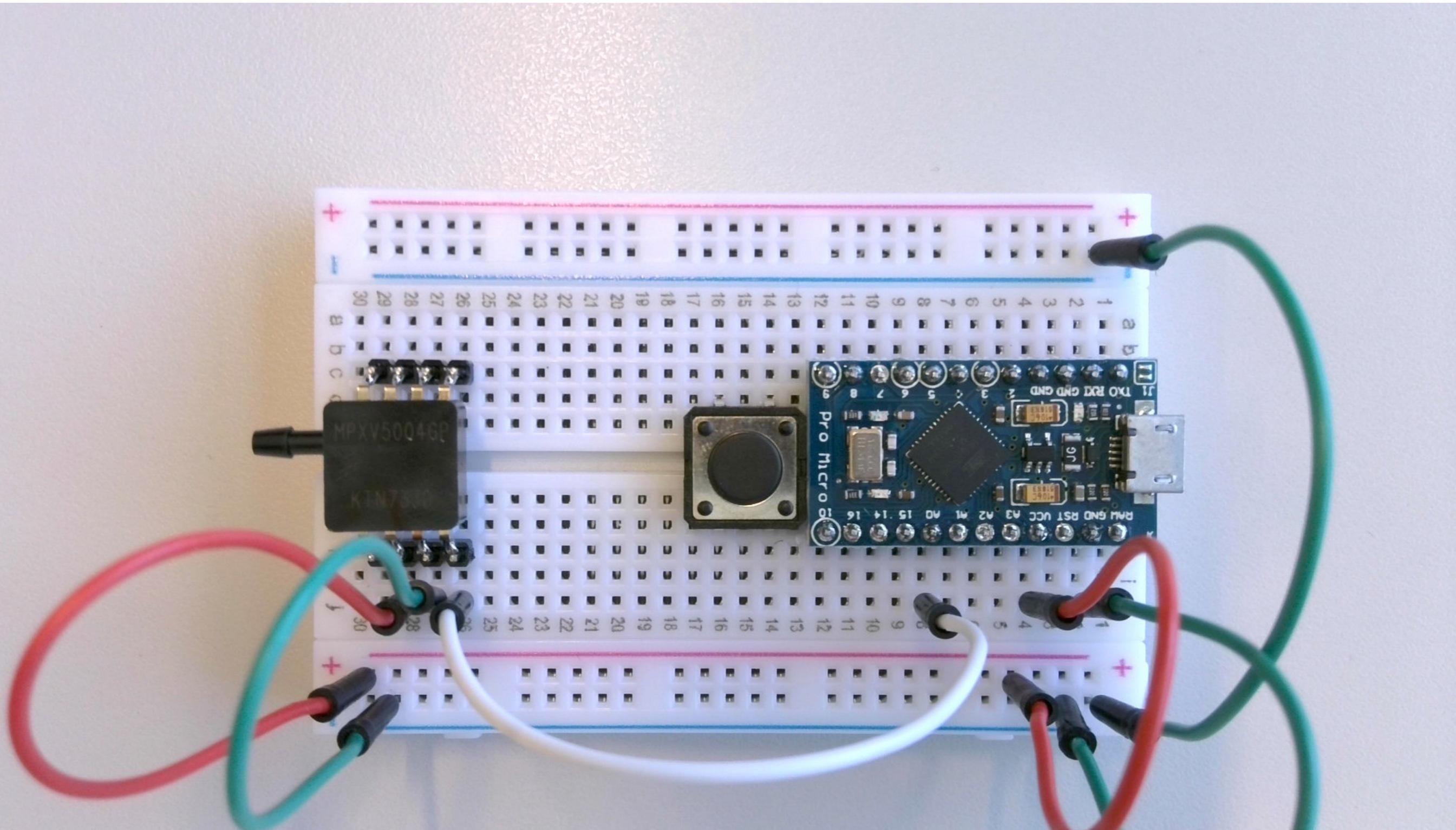
---

Make sure this is correct!  
If you hook it up wrong you  
will break the \$30 sensor!



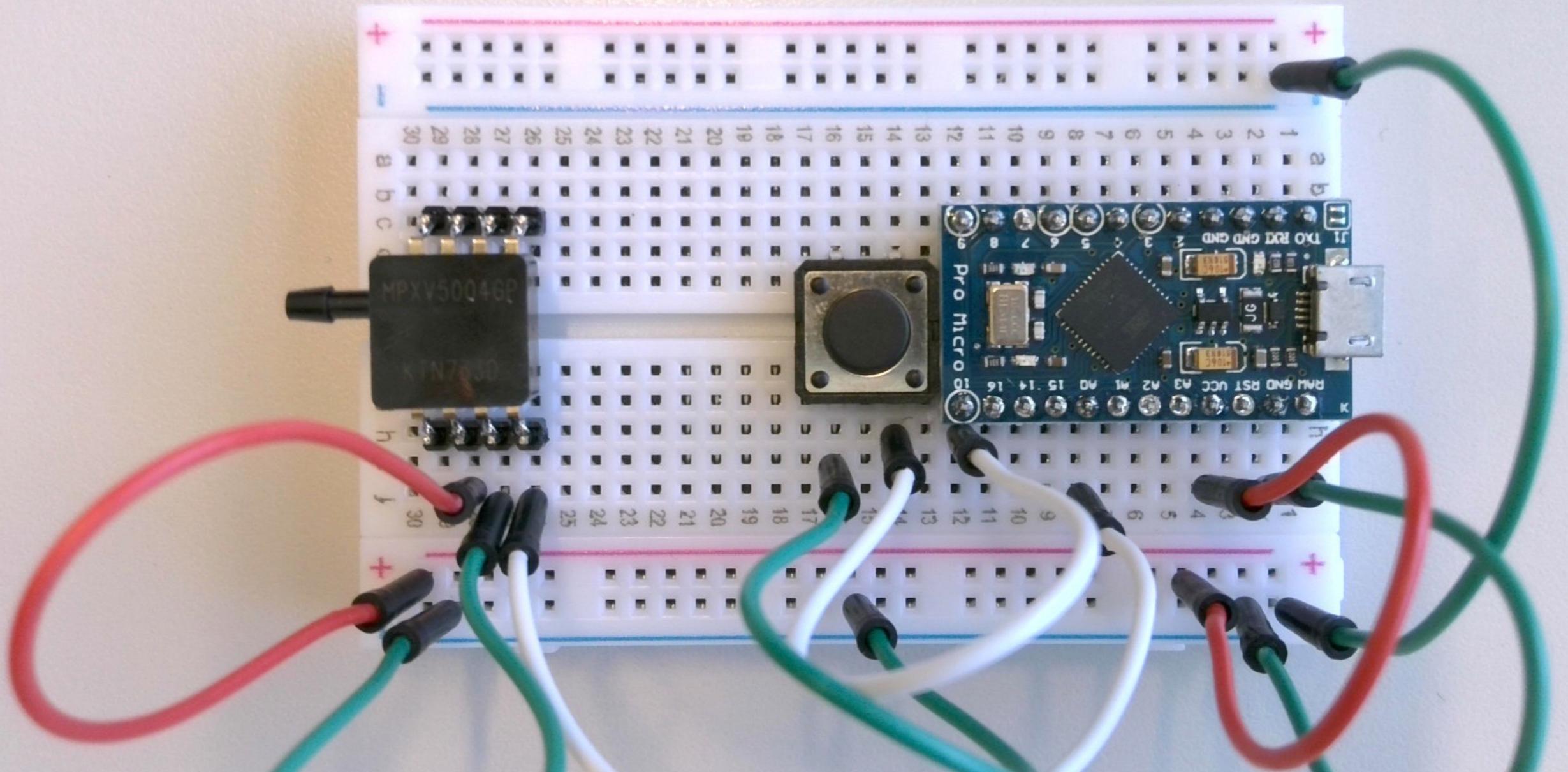
# HOOKING UP YOUR SENSORS! PRESSURE SENSOR

---



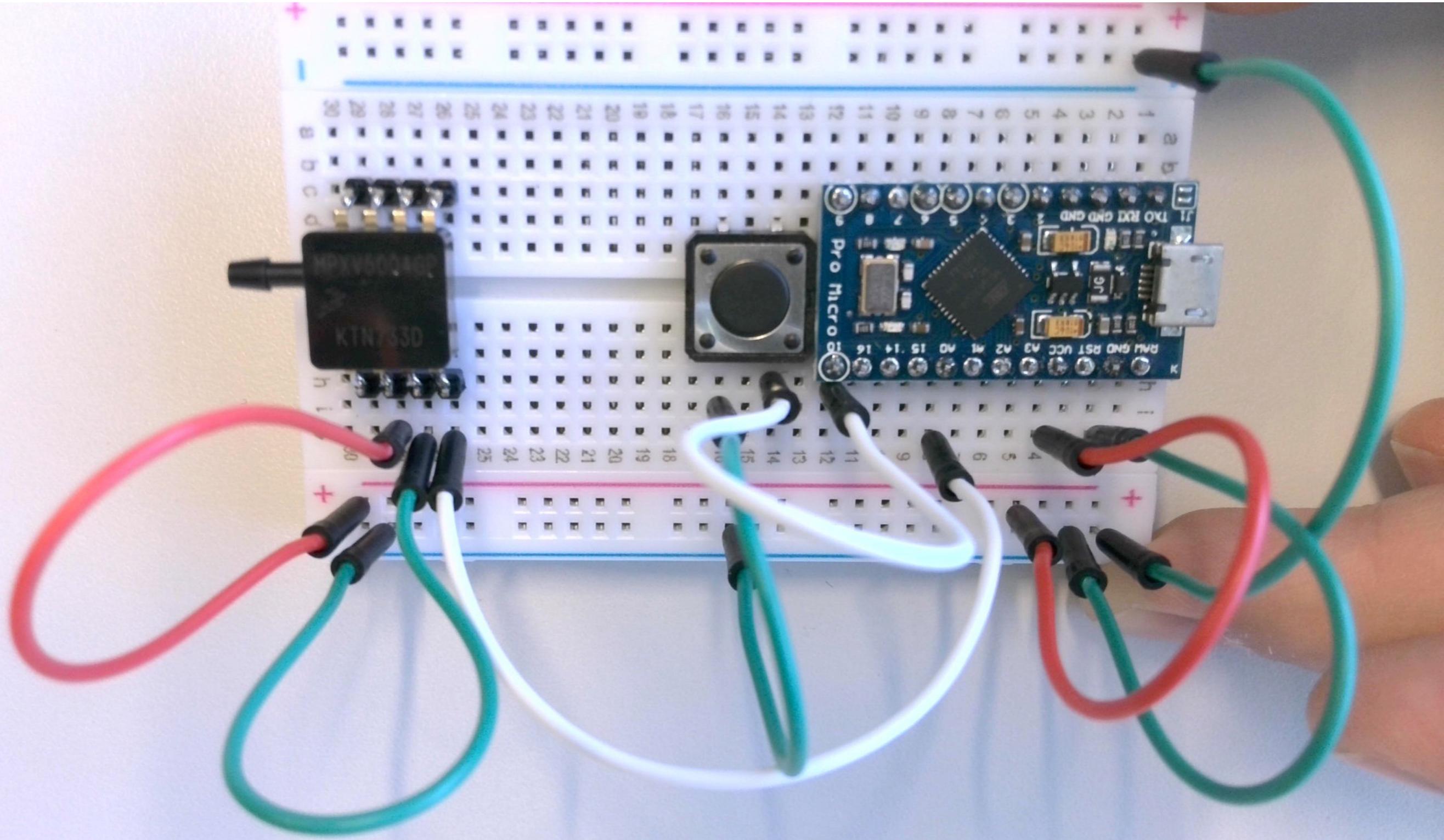
# HOOKING UP YOUR SENSORS! BUTTON

---



# HOOKING UP YOUR SENSORS! BUTTON

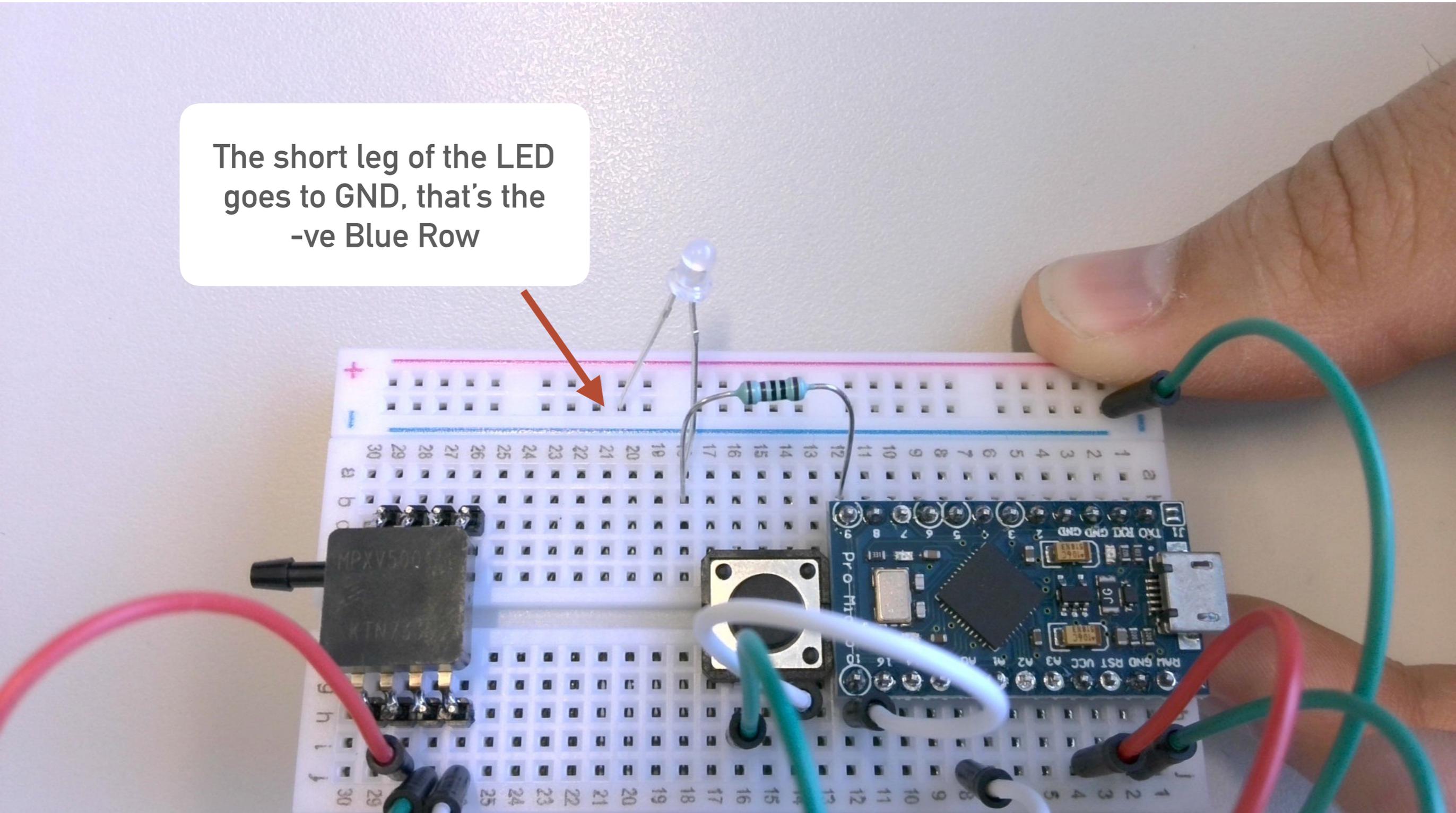
---



# HOOKING UP YOUR SENSORS! LED

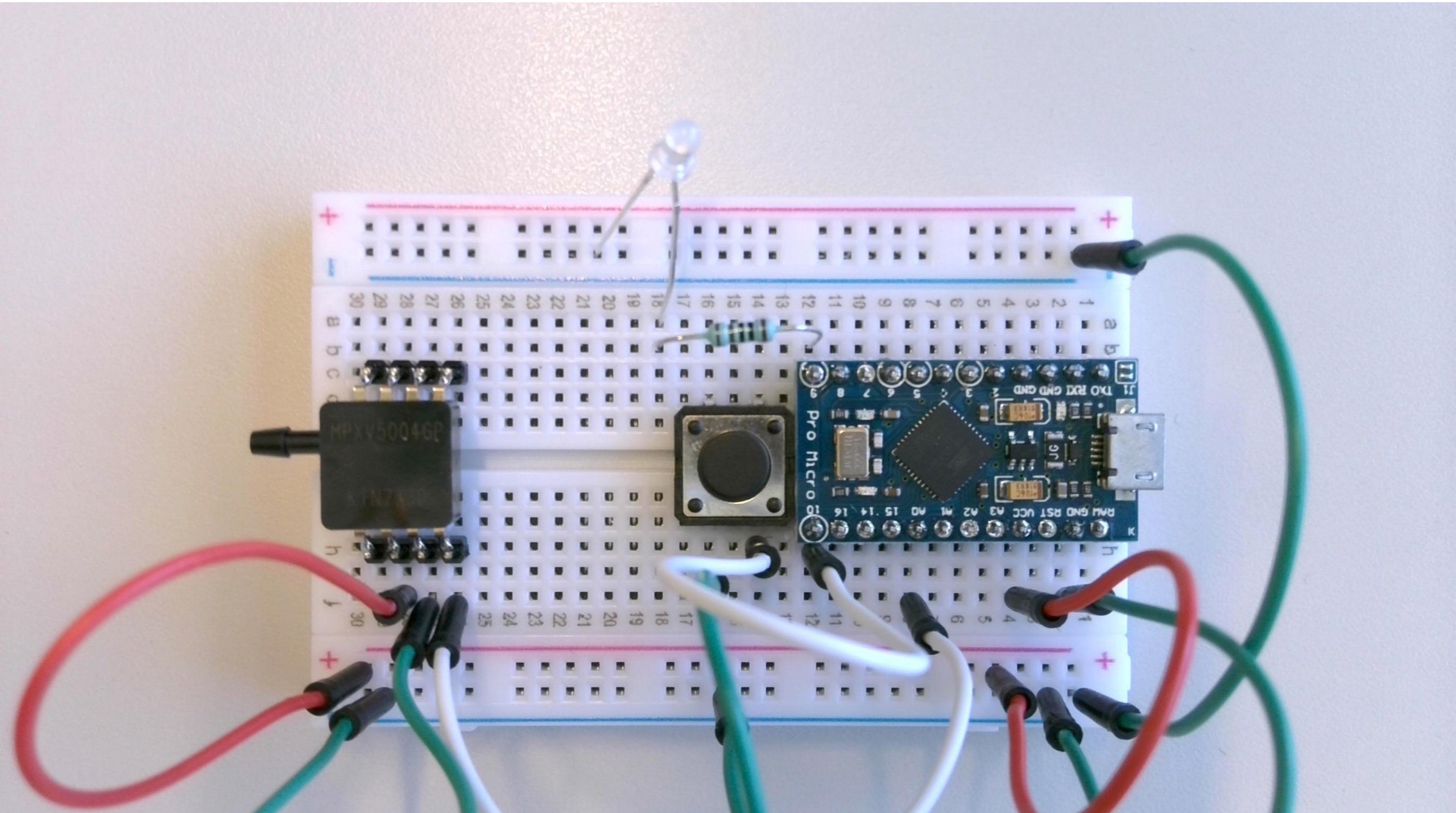
---

The short leg of the LED goes to GND, that's the -ve Blue Row



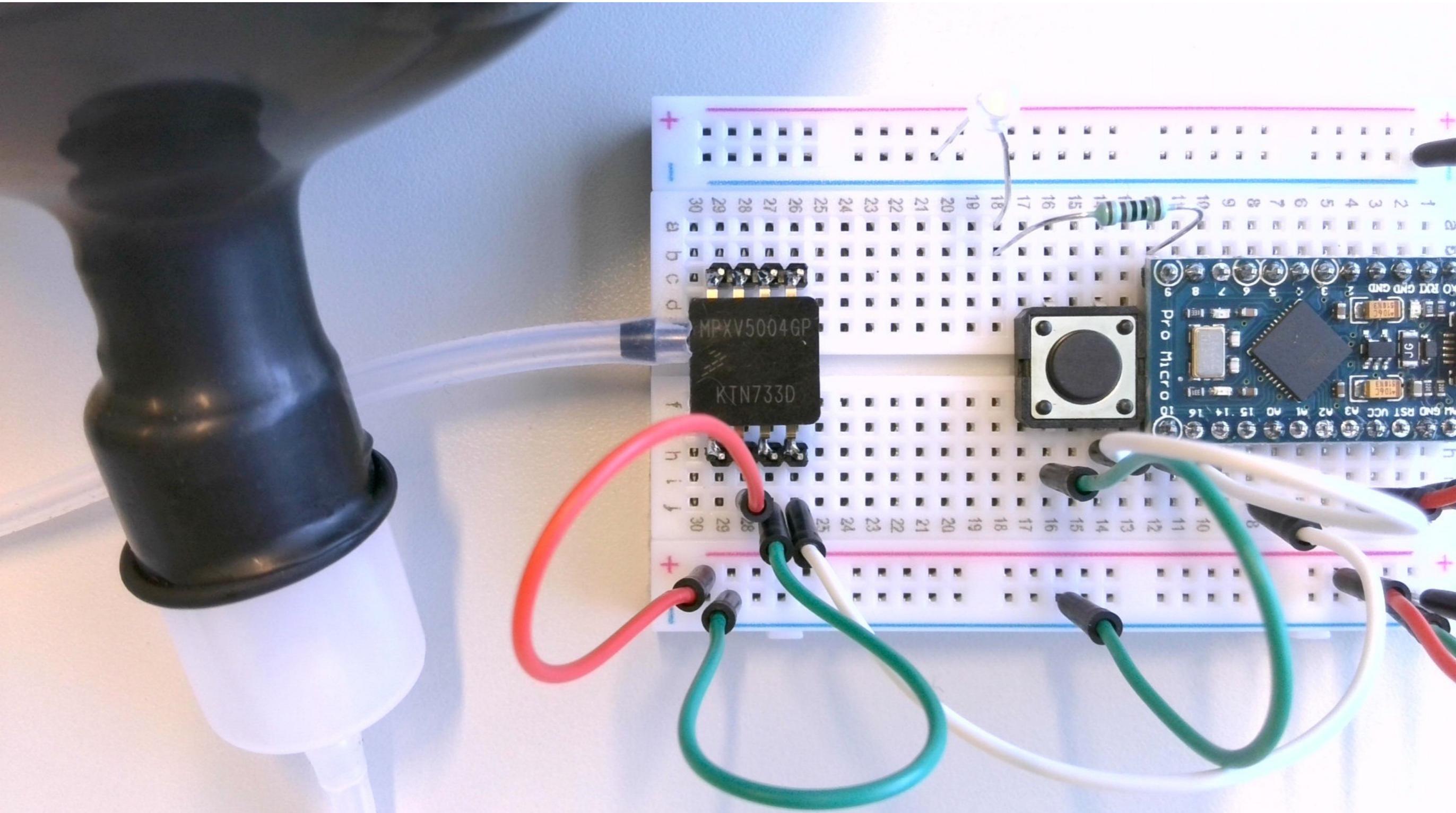
# HOOKING UP YOUR SENSORS! LED

---



# HOOKING UP YOUR SENSORS! ATTACH A BALLOON

---



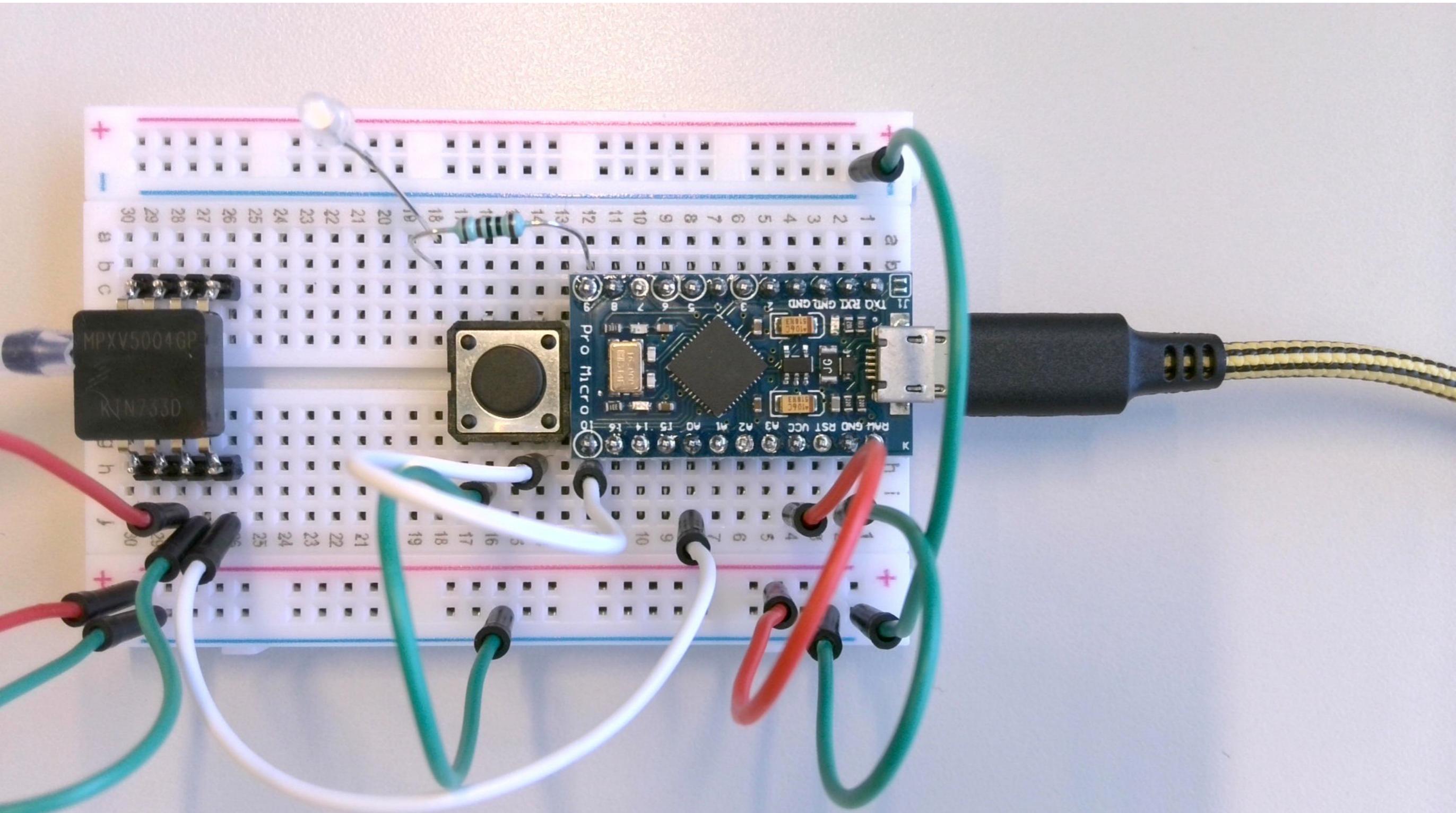
## **SAFETY CHECK!**

---

***BEFORE PLUGGING THE  
USB CABLE INTO YOUR  
COMPUTER GET MATT TO  
CHECK YOUR BOARD SO  
YOU KNOW IT'S SAFE AND  
NOTHING WILL BREAK!***

# HOOKING UP YOUR SENSORS! PLUG IN USB

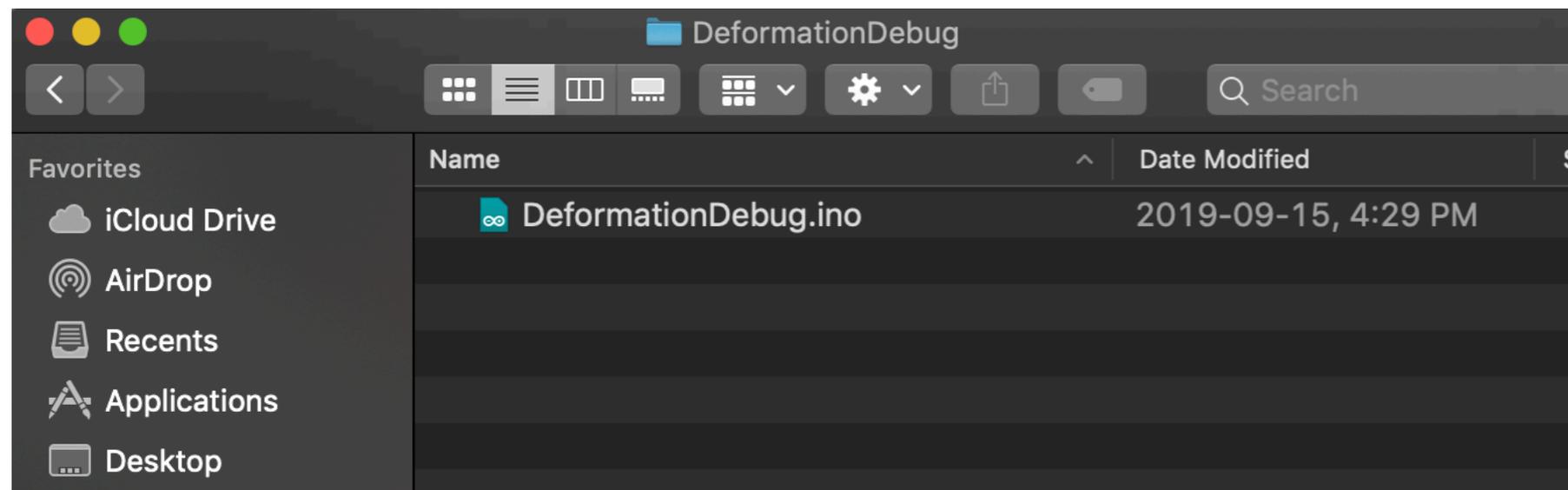
---



# ON YOUR COMPUTER

---

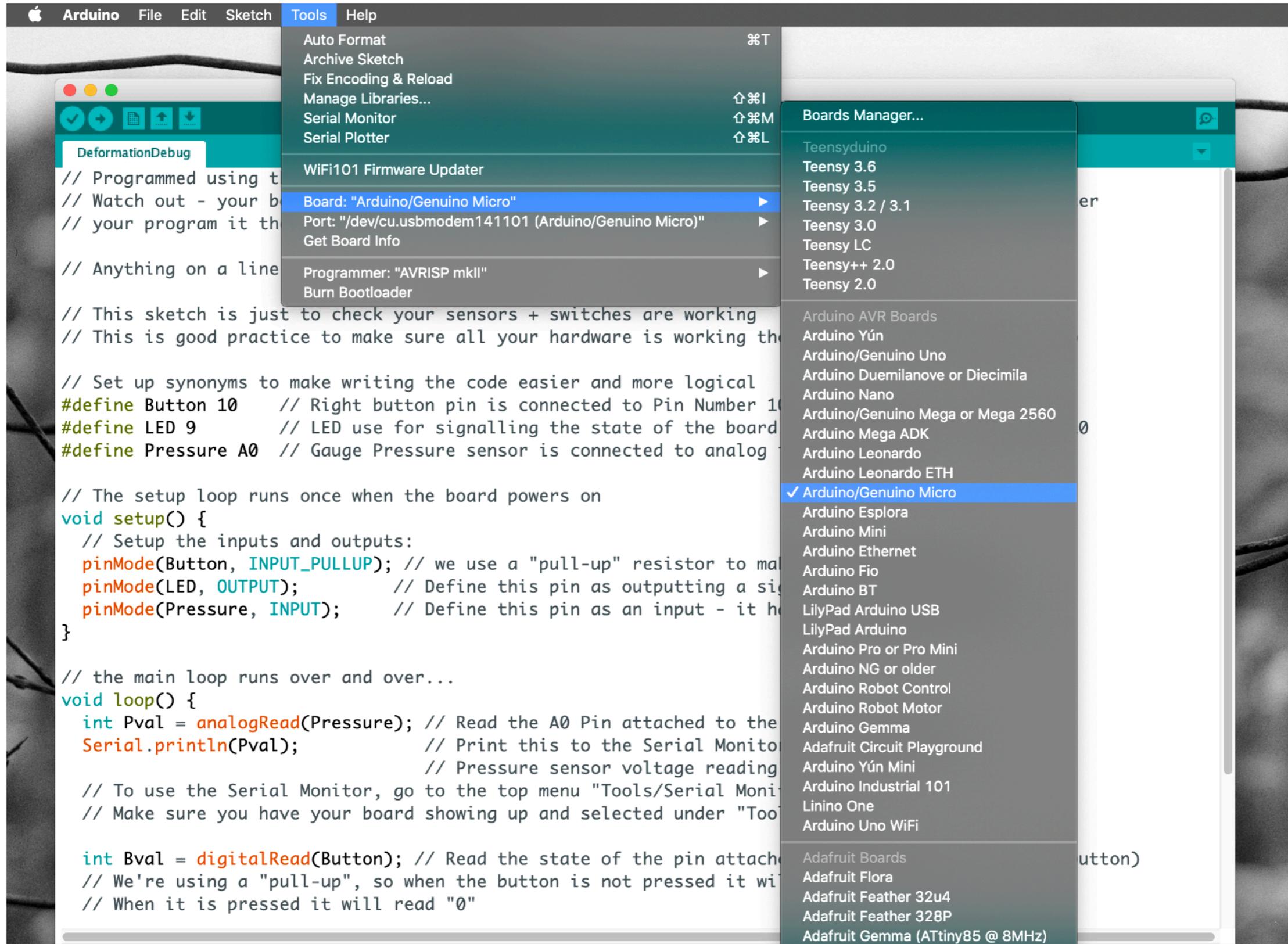
*Launch Arduino and open “DeformationDebug.ino”*



*Arduino IDE - a software platform used to program your microcontroller.*

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

# PROGRAM YOUR DEVICE – BOARD TYPE



The screenshot shows the Arduino IDE interface. The Tools menu is open, displaying various options. The Boards Manager window is also open, showing a list of boards. The 'Arduino/Genuino Micro' board is selected in both the Tools menu and the Boards Manager.

```
Arduino File Edit Sketch Tools Help
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... ⇧⌘I
Serial Monitor ⇧⌘M
Serial Plotter ⇧⌘L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Micro"
Port: "/dev/cu.usbmodem141101 (Arduino/Genuino Micro)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
```

Boards Manager...

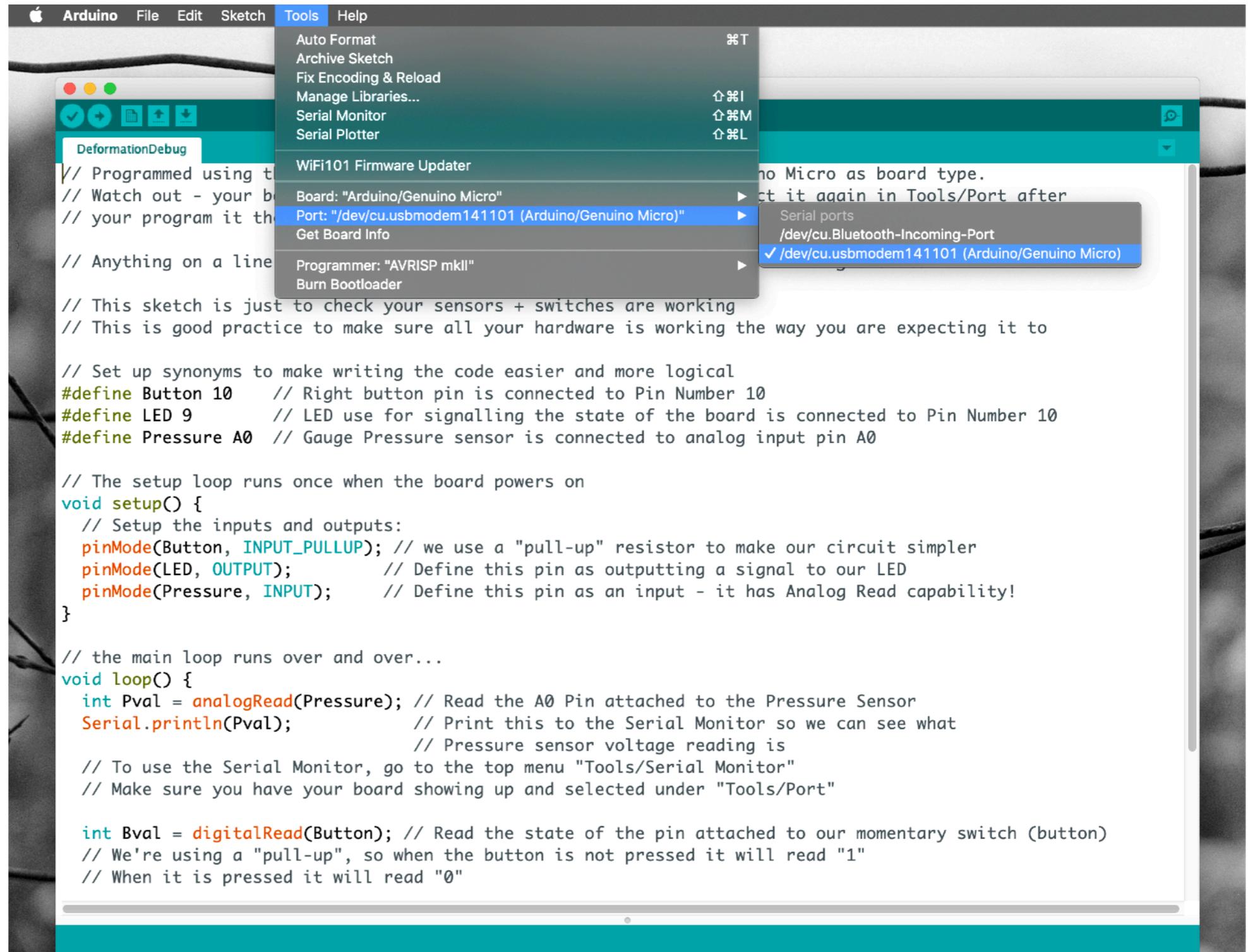
- Teensyduino
  - Teensy 3.6
  - Teensy 3.5
  - Teensy 3.2 / 3.1
  - Teensy 3.0
  - Teensy LC
  - Teensy++ 2.0
  - Teensy 2.0
- Arduino AVR Boards
  - Arduino Yún
  - Arduino/Genuino Uno
  - Arduino Duemilanove or Diecimila
  - Arduino Nano
  - Arduino/Genuino Mega or Mega 2560
  - Arduino Mega ADK
  - Arduino Leonardo
  - Arduino Leonardo ETH
  - ✓ Arduino/Genuino Micro
  - Arduino Esplora
  - Arduino Mini
  - Arduino Ethernet
  - Arduino Fio
  - Arduino BT
  - LilyPad Arduino USB
  - LilyPad Arduino
  - Arduino Pro or Pro Mini
  - Arduino NG or older
  - Arduino Robot Control
  - Arduino Robot Motor
  - Arduino Gemma
  - Adafruit Circuit Playground
  - Arduino Yún Mini
  - Arduino Industrial 101
  - Linino One
  - Arduino Uno WiFi
- Adafruit Boards
  - Adafruit Flora
  - Adafruit Feather 32u4
  - Adafruit Feather 328P
  - Adafruit Gemma (ATtiny85 @ 8MHz)

```
DeformationDebug
// Programmed using t
// Watch out - your b
// your program it th
// Anything on a line
// This sketch is just to check your sensors + switches are working
// This is good practice to make sure all your hardware is working th
// Set up synonyms to make writing the code easier and more logical
#define Button 10 // Right button pin is connected to Pin Number 1
#define LED 9 // LED use for signalling the state of the board
#define Pressure A0 // Gauge Pressure sensor is connected to analog
// The setup loop runs once when the board powers on
void setup() {
  // Setup the inputs and outputs:
  pinMode(Button, INPUT_PULLUP); // we use a "pull-up" resistor to ma
  pinMode(LED, OUTPUT); // Define this pin as outputting a sig
  pinMode(Pressure, INPUT); // Define this pin as an input - it h
}
// the main loop runs over and over...
void loop() {
  int Pval = analogRead(Pressure); // Read the A0 Pin attached to the
  Serial.println(Pval); // Print this to the Serial Monito
  // Pressure sensor voltage reading
  // To use the Serial Monitor, go to the top menu "Tools/Serial Moni
  // Make sure you have your board showing up and selected under "Too
  int Bval = digitalRead(Button); // Read the state of the pin attach
  // We're using a "pull-up", so when the button is not pressed it wi
  // When it is pressed it will read "0"
```

# PROGRAM YOUR DEVICE – PORT

*Sometime this name changes after programming, so you may need to reselect this from time to time.*

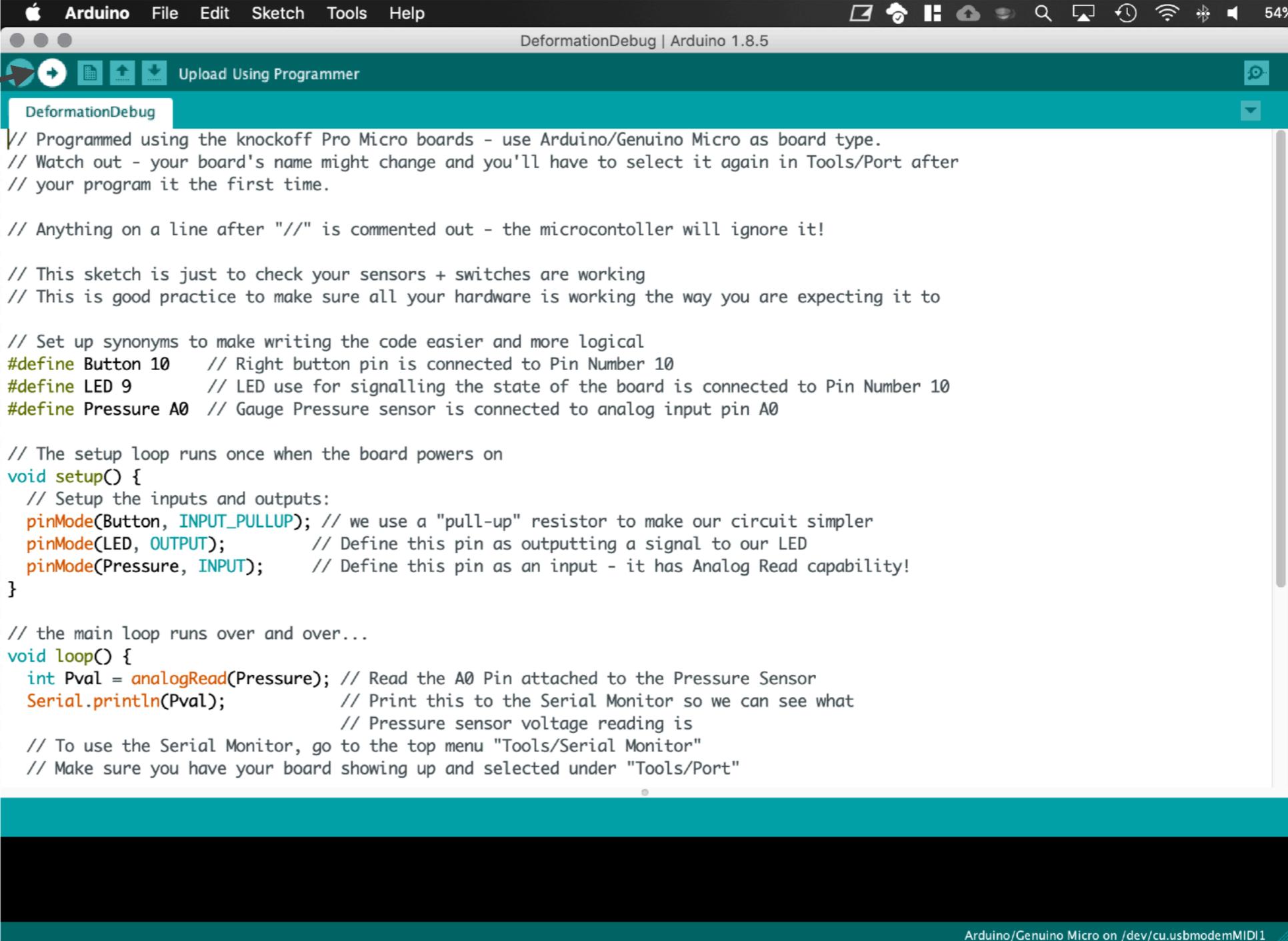
*If you get an error that says “board not found” you should check it’s plugged in and that your port is selected correctly.*



# PROGRAM YOUR DEVICE – DEFORMATIONDEBUG.INO

---

*With your arduino plugged in, press the ARROW button to compile the code and send it to the arduino.*



```
Arduino File Edit Sketch Tools Help
DeformationDebug | Arduino 1.8.5
Upload Using Programmer
DeformationDebug
// Programmed using the knockoff Pro Micro boards - use Arduino/Genuino Micro as board type.
// Watch out - your board's name might change and you'll have to select it again in Tools/Port after
// your program it the first time.

// Anything on a line after "//" is commented out - the microcontoller will ignore it!

// This sketch is just to check your sensors + switches are working
// This is good practice to make sure all your hardware is working the way you are expecting it to

// Set up synonyms to make writing the code easier and more logical
#define Button 10 // Right button pin is connected to Pin Number 10
#define LED 9 // LED use for signalling the state of the board is connected to Pin Number 10
#define Pressure A0 // Gauge Pressure sensor is connected to analog input pin A0

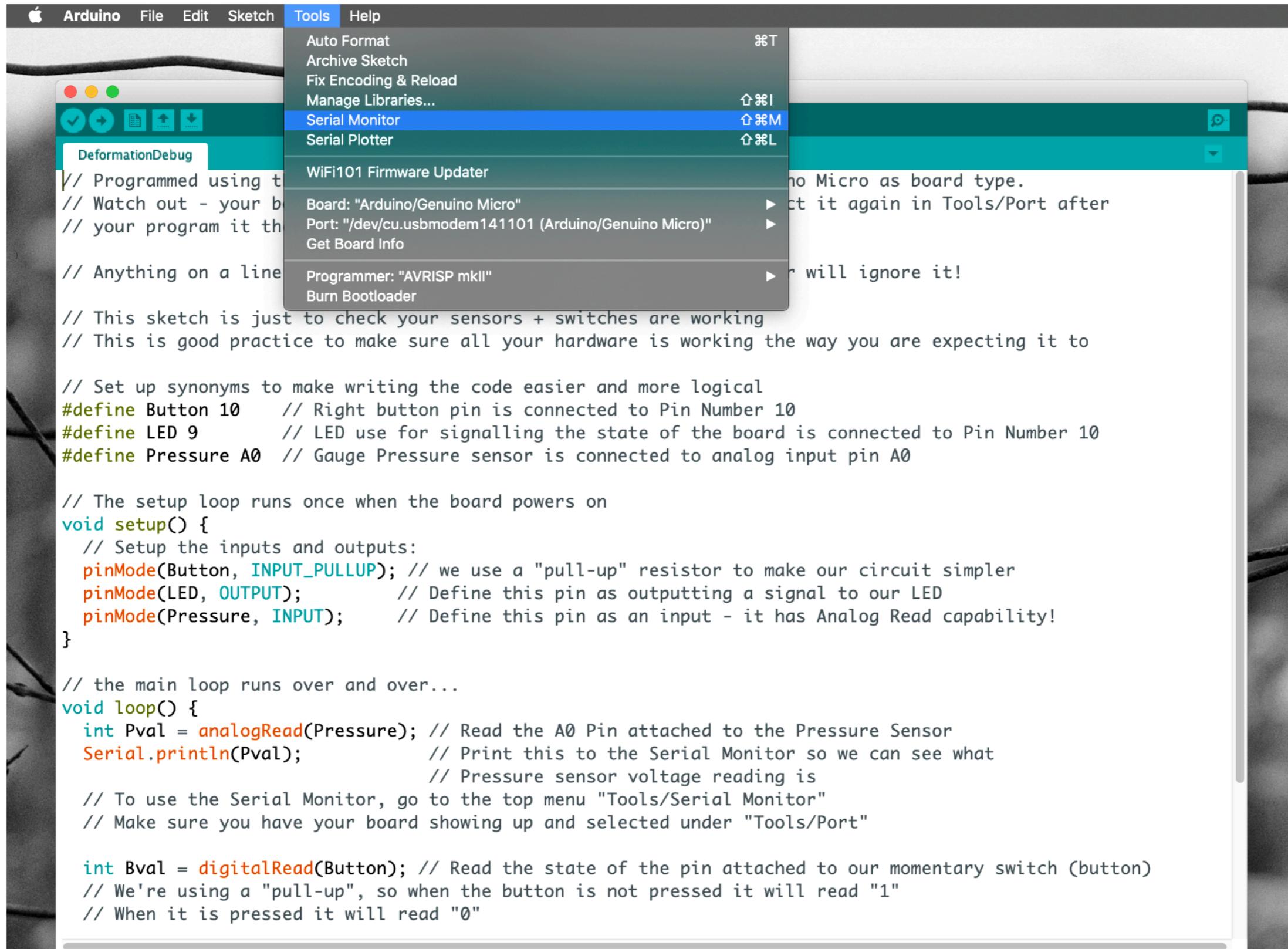
// The setup loop runs once when the board powers on
void setup() {
  // Setup the inputs and outputs:
  pinMode(Button, INPUT_PULLUP); // we use a "pull-up" resistor to make our circuit simpler
  pinMode(LED, OUTPUT); // Define this pin as outputting a signal to our LED
  pinMode(Pressure, INPUT); // Define this pin as an input - it has Analog Read capability!
}

// the main loop runs over and over...
void loop() {
  int Pval = analogRead(Pressure); // Read the A0 Pin attached to the Pressure Sensor
  Serial.println(Pval); // Print this to the Serial Monitor so we can see what
  // Pressure sensor voltage reading is

  // To use the Serial Monitor, go to the top menu "Tools/Serial Monitor"
  // Make sure you have your board showing up and selected under "Tools/Port"

Arduino/Genuino Micro on /dev/cu.usbmodemMIDI1
```

# PROGRAM YOUR DEVICE – MONITOR THE SERIAL PORT



The screenshot shows the Arduino IDE interface. The 'Tools' menu is open, and 'Serial Monitor' is selected. The Serial Monitor window is open, displaying the output of a C++ sketch. The sketch code is visible in the background, showing comments and code for setting up pins and reading sensor data.

```
// Programmed using t
// Watch out - your b
// your program it th

// Anything on a line

// This sketch is just to check your sensors + switches are working
// This is good practice to make sure all your hardware is working the way you are expecting it to

// Set up synonyms to make writing the code easier and more logical
#define Button 10 // Right button pin is connected to Pin Number 10
#define LED 9 // LED use for signalling the state of the board is connected to Pin Number 10
#define Pressure A0 // Gauge Pressure sensor is connected to analog input pin A0

// The setup loop runs once when the board powers on
void setup() {
  // Setup the inputs and outputs:
  pinMode(Button, INPUT_PULLUP); // we use a "pull-up" resistor to make our circuit simpler
  pinMode(LED, OUTPUT); // Define this pin as outputting a signal to our LED
  pinMode(Pressure, INPUT); // Define this pin as an input - it has Analog Read capability!
}

// the main loop runs over and over...
void loop() {
  int Pval = analogRead(Pressure); // Read the A0 Pin attached to the Pressure Sensor
  Serial.println(Pval); // Print this to the Serial Monitor so we can see what
  // Pressure sensor voltage reading is

  // To use the Serial Monitor, go to the top menu "Tools/Serial Monitor"
  // Make sure you have your board showing up and selected under "Tools/Port"

  int Bval = digitalRead(Button); // Read the state of the pin attached to our momentary switch (button)
  // We're using a "pull-up", so when the button is not pressed it will read "1"
  // When it is pressed it will read "0"
```

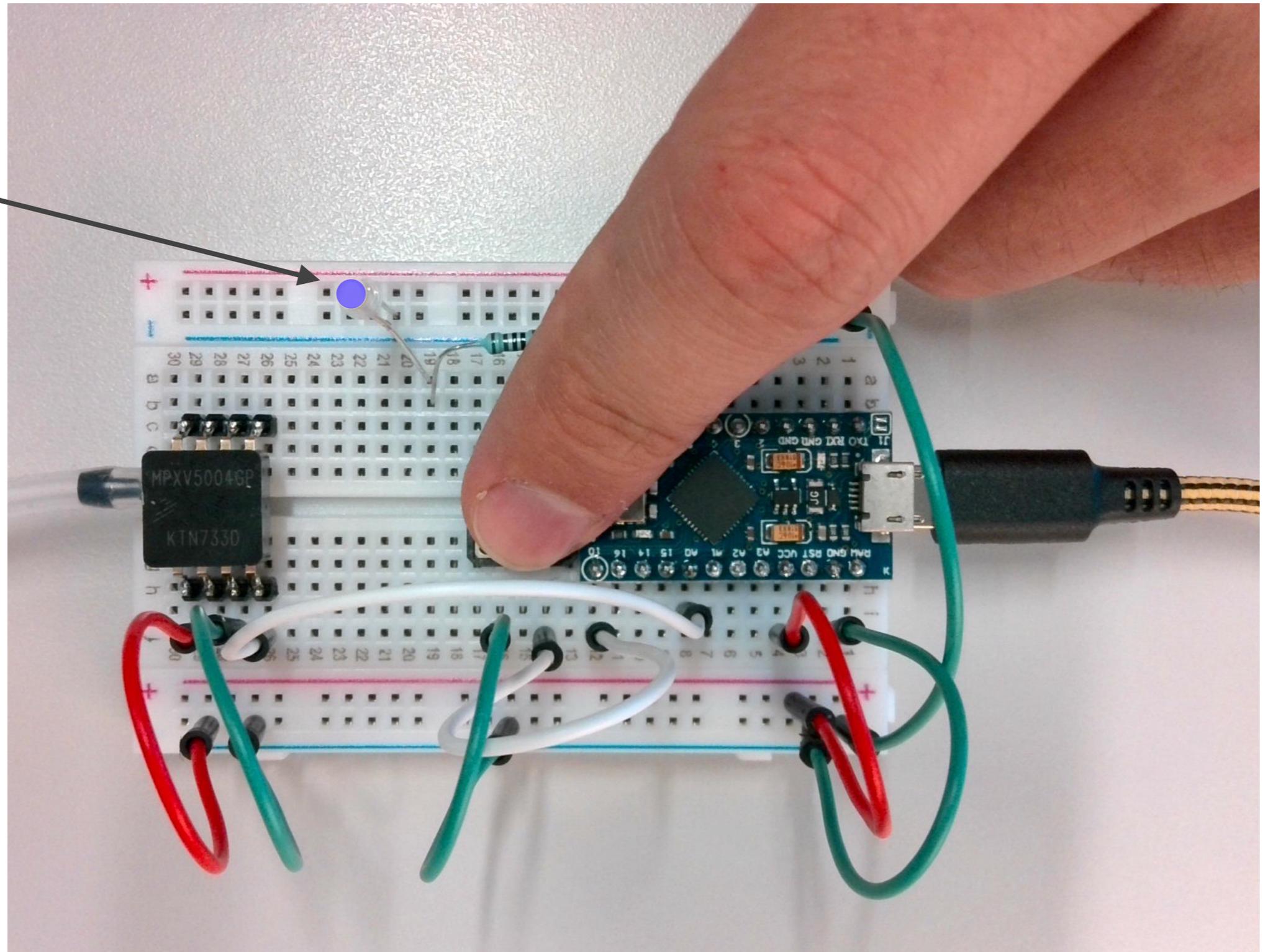


# PROGRAM YOUR DEVICE – CHECK BUTTON AND LED

---

*Without the button pressed the LED should be bright blue.*

*When you press the button the LED should turn off.*



## PROGRAM YOUR DEVICE – DEFORMATIONDEBUG.INO

---

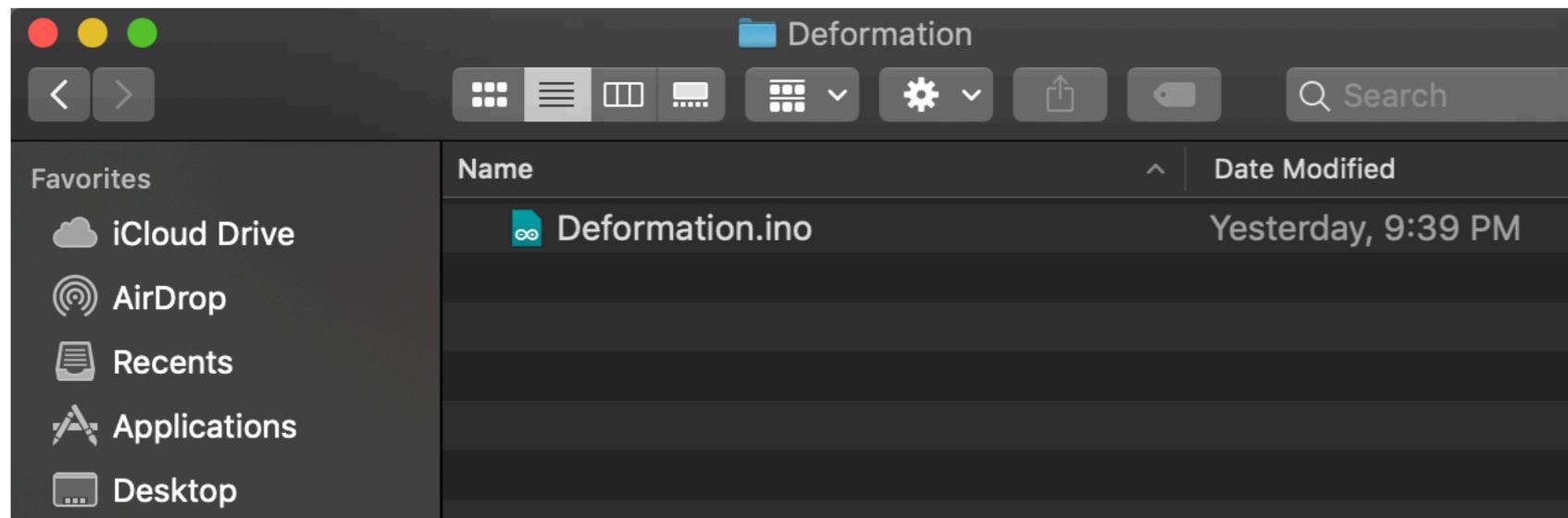
*If your button changed the LED from on to off, and your serial monitor values changed when you squeezed your balloon, you're good to go!*

*If it didn't, get some help to figure out what's wrong with how you hooked things up on your breadboard!*

# OPEN A FILE

---

*Launch Arduino and open “Deformation.ino”*

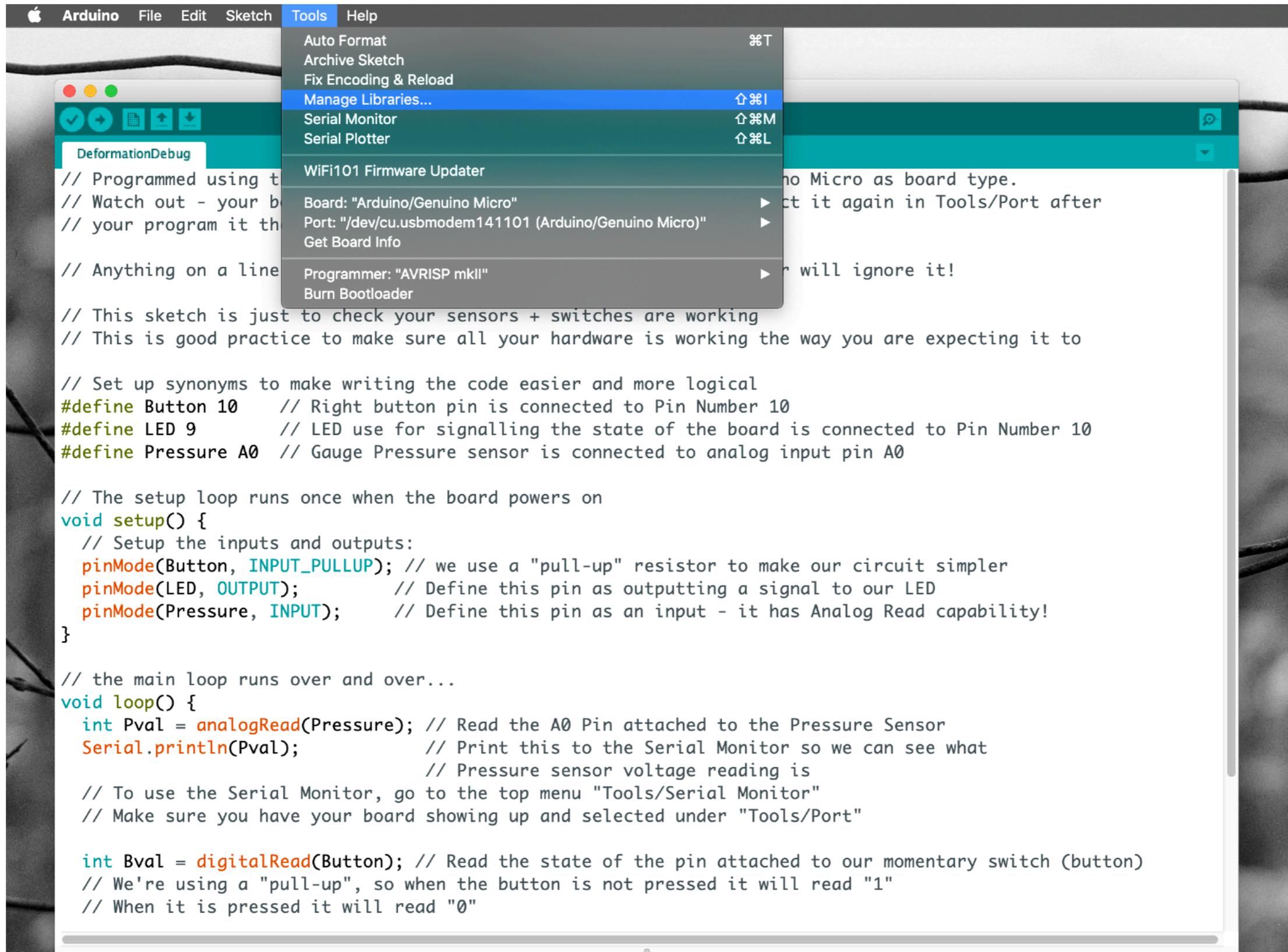


*Arduino IDE - a software platform used to program your microcontroller.*

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

# LIBRARIES – INSTALL THESE TO ADD FUNCTIONS TO YOUR PROGRAM

---



The screenshot shows the Arduino IDE interface. The 'Tools' menu is open, and 'Manage Libraries...' is highlighted. The code editor in the background contains the following C++ code:

```
// Programmed using t
// Watch out - your b
// your program it th

// Anything on a line

// This sketch is just to check your sensors + switches are working
// This is good practice to make sure all your hardware is working the way you are expecting it to

// Set up synonyms to make writing the code easier and more logical
#define Button 10 // Right button pin is connected to Pin Number 10
#define LED 9 // LED use for signalling the state of the board is connected to Pin Number 10
#define Pressure A0 // Gauge Pressure sensor is connected to analog input pin A0

// The setup loop runs once when the board powers on
void setup() {
  // Setup the inputs and outputs:
  pinMode(Button, INPUT_PULLUP); // we use a "pull-up" resistor to make our circuit simpler
  pinMode(LED, OUTPUT); // Define this pin as outputting a signal to our LED
  pinMode(Pressure, INPUT); // Define this pin as an input - it has Analog Read capability!
}

// the main loop runs over and over...
void loop() {
  int Pval = analogRead(Pressure); // Read the A0 Pin attached to the Pressure Sensor
  Serial.println(Pval); // Print this to the Serial Monitor so we can see what
  // Pressure sensor voltage reading is

  // To use the Serial Monitor, go to the top menu "Tools/Serial Monitor"
  // Make sure you have your board showing up and selected under "Tools/Port"

  int Bval = digitalRead(Button); // Read the state of the pin attached to our momentary switch (button)
  // We're using a "pull-up", so when the button is not pressed it will read "1"
  // When it is pressed it will read "0"
```

# LIBRARIES - ELAPSEDMILLIS

The image shows the Arduino IDE interface. The main window displays a sketch titled "DeformationDebug" with the following code:

```
// Programmed using the knockoff Pro Micro boards - use Arduino/Genuino Micro as board type.
// Watch out - your board's name might change and you'll have to select it again in Tools/Port after
// your program it the first time.

// Anything on a line after "//" is commented out - the microcontoller will ignore it!

// This sketch is just to check your sensors + switches are working
// This is good practice to make sure all your hardware is working the way you are expecting it to

// Set up synony
#define Button 1
#define LED 9
#define Pressure

// The setup loop
void setup() {
  // Setup the i
  pinMode(Button
  pinMode(LED, O
  pinMode(Pressu
}

// the main loop
void loop() {
  int Pval = ana
  Serial.println

// To use the
// Make sure you have your board showing up and selected under

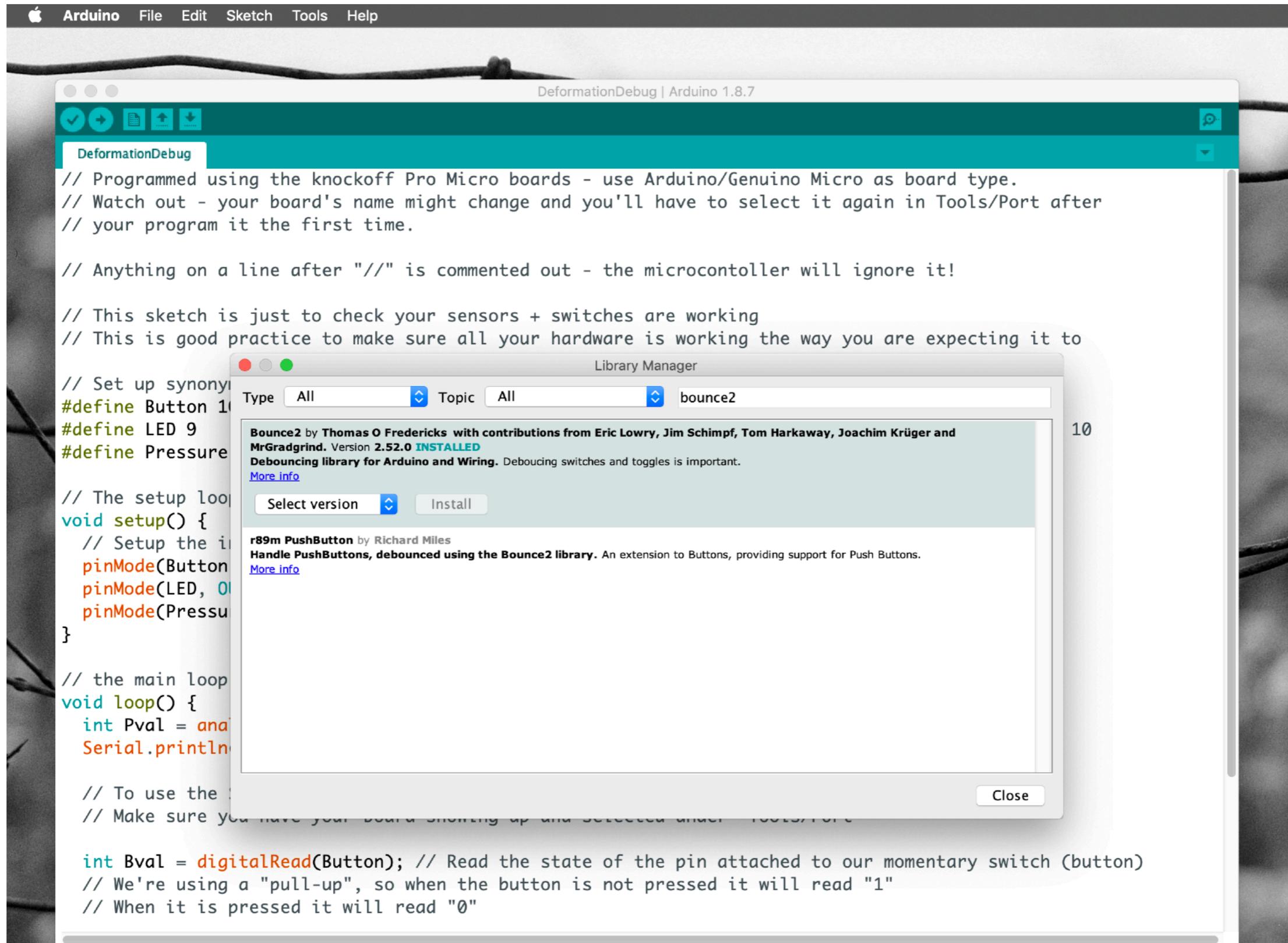
int Bval = digitalRead(Button); // Read the state of the pin attached to our momentary switch (button)
// We're using a "pull-up", so when the button is not pressed it will read "1"
// When it is pressed it will read "0"
```

The Library Manager window is open, showing search results for "elapsedMillis". The search filters are set to "All" for both Type and Topic. The search results list two libraries:

- DMXUSB** by DaAwesomeP: DMXUSB emulates an ENTTEC-compatible DMXKing USB to DMX serial device with one or two universes. DMXUSB implements the ENTTEC DMX USB Pro Widget API Specification 1.44 on any serial port. DMXUSB can emulate a single DMX port/universe device like the DMXKing USB ultraDMX Micro or a two port/universe device like the DMXKing ultraDMX Pro. Both devices are compatible with the ENTTEC standard. DMXUSB works with the Open Lighting Architecture (OLA) as a usbserial device. This library requires the elapsedMillis library for all boards except the PJRC Teensy. [More info](#)
- elapsedMillis** by Paul Stoffregen: Version 1.0.5 **INSTALLED**. Makes coding responsive sketches easier. When using delay(), your code can not (easily) respond to user input while the delay is happening (unless you use interrupts or complex timer code). This library makes this easy by allowing you to create variables (objects) that automatically increase as time elapses. It is easy to check if a certain time has elapsed, while your program performs other work or checks for user input. [More info](#)

The "elapsedMillis" library is highlighted, and the "Install" button is visible. The "Close" button is also present at the bottom right of the Library Manager window.

# LIBRARIES – BOUNCE2



The image shows a screenshot of the Arduino IDE interface. The main window displays a sketch titled "DeformationDebug" with the following code:

```
// Programmed using the knockoff Pro Micro boards - use Arduino/Genuino Micro as board type.
// Watch out - your board's name might change and you'll have to select it again in Tools/Port after
// your program it the first time.

// Anything on a line after "//" is commented out - the microcontoller will ignore it!

// This sketch is just to check your sensors + switches are working
// This is good practice to make sure all your hardware is working the way you are expecting it to

// Set up synony
#define Button 1
#define LED 9
#define Pressure

// The setup loop
void setup() {
  // Setup the i
  pinMode(Button
  pinMode(LED, O
  pinMode(Pressu
}

// the main loop
void loop() {
  int Pval = ana
  Serial.println

// To use the
// Make sure you have your board showing up and selected under Tools/Port

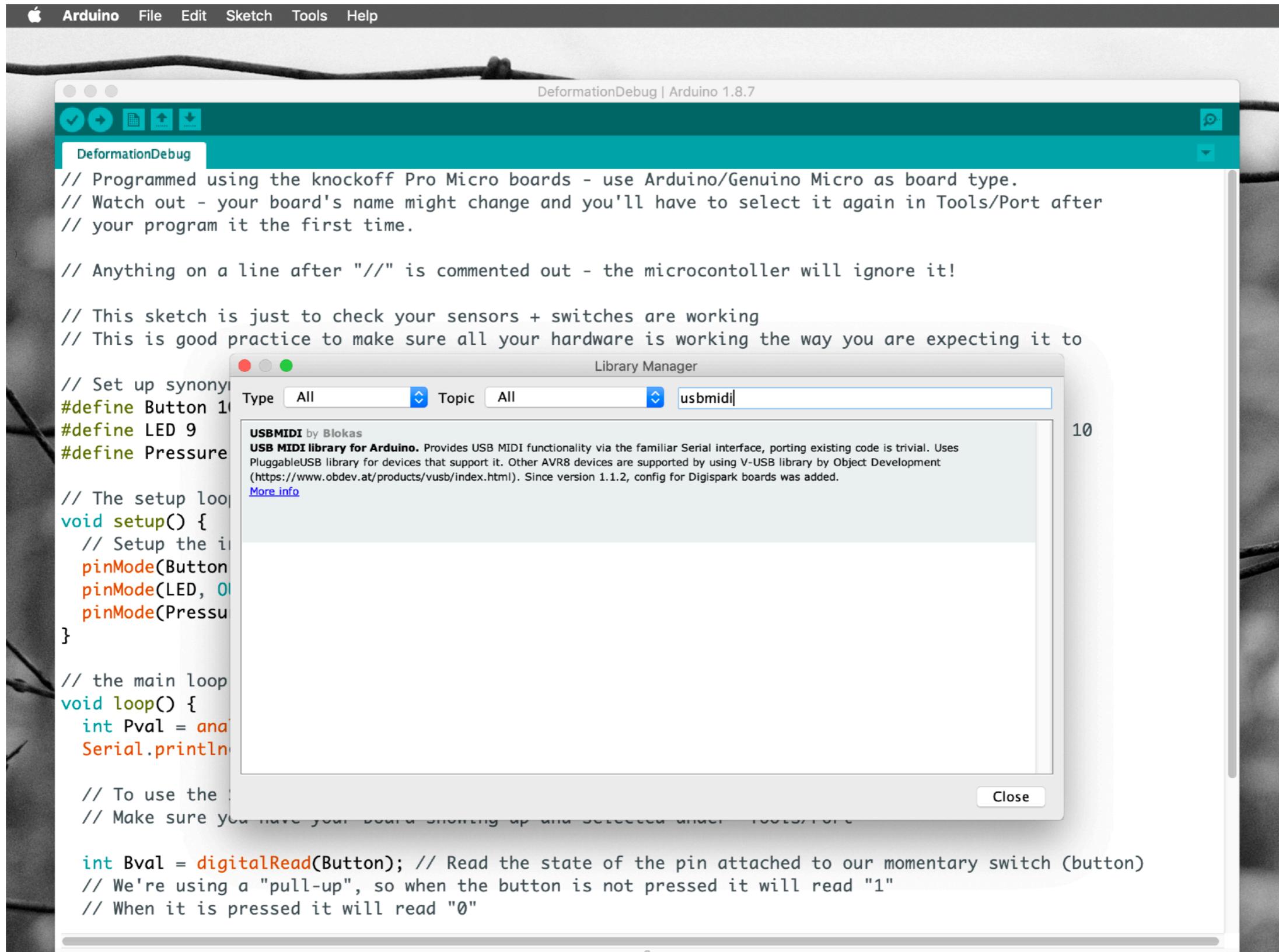
int Bval = digitalRead(Button); // Read the state of the pin attached to our momentary switch (button)
// We're using a "pull-up", so when the button is not pressed it will read "1"
// When it is pressed it will read "0"
```

The Library Manager window is open, showing search results for "bounce2". The search filters are set to "All" for both Type and Topic. The results list includes:

- Bounce2** by Thomas O Fredericks with contributions from Eric Lowry, Jim Schimpf, Tom Harkaway, Joachim Krüger and MrGradgrind. Version 2.52.0 **INSTALLED**. Debouncing library for Arduino and Wiring. Debouncing switches and toggles is important. [More info](#)
- r89m PushButton** by Richard Miles. Handle PushButtons, debounced using the Bounce2 library. An extension to Buttons, providing support for Push Buttons. [More info](#)

The Library Manager window has a "Close" button at the bottom right.

# LIBRARIES – USBMIDI



The image shows a screenshot of the Arduino IDE interface. The main window displays a sketch titled "DeformationDebug" with the following code:

```
// Programmed using the knockoff Pro Micro boards - use Arduino/Genuino Micro as board type.
// Watch out - your board's name might change and you'll have to select it again in Tools/Port after
// your program it the first time.

// Anything on a line after "//" is commented out - the microcontoller will ignore it!

// This sketch is just to check your sensors + switches are working
// This is good practice to make sure all your hardware is working the way you are expecting it to

// Set up synony
#define Button 1
#define LED 9
#define Pressure

// The setup loop
void setup() {
  // Setup the i
  pinMode(Button
  pinMode(LED, 0
  pinMode(Pressu
}

// the main loop
void loop() {
  int Pval = ana
  Serial.println

// To use the
// Make sure you have your board showing up and selected under Tools/Port

int Bval = digitalRead(Button); // Read the state of the pin attached to our momentary switch (button)
// We're using a "pull-up", so when the button is not pressed it will read "1"
// When it is pressed it will read "0"
```

The Library Manager window is open, showing the search results for "usbmidi". The search filters are set to "All" for both Type and Topic. The search results show the "USB MIDI" library by Blokas. The description of the library is:

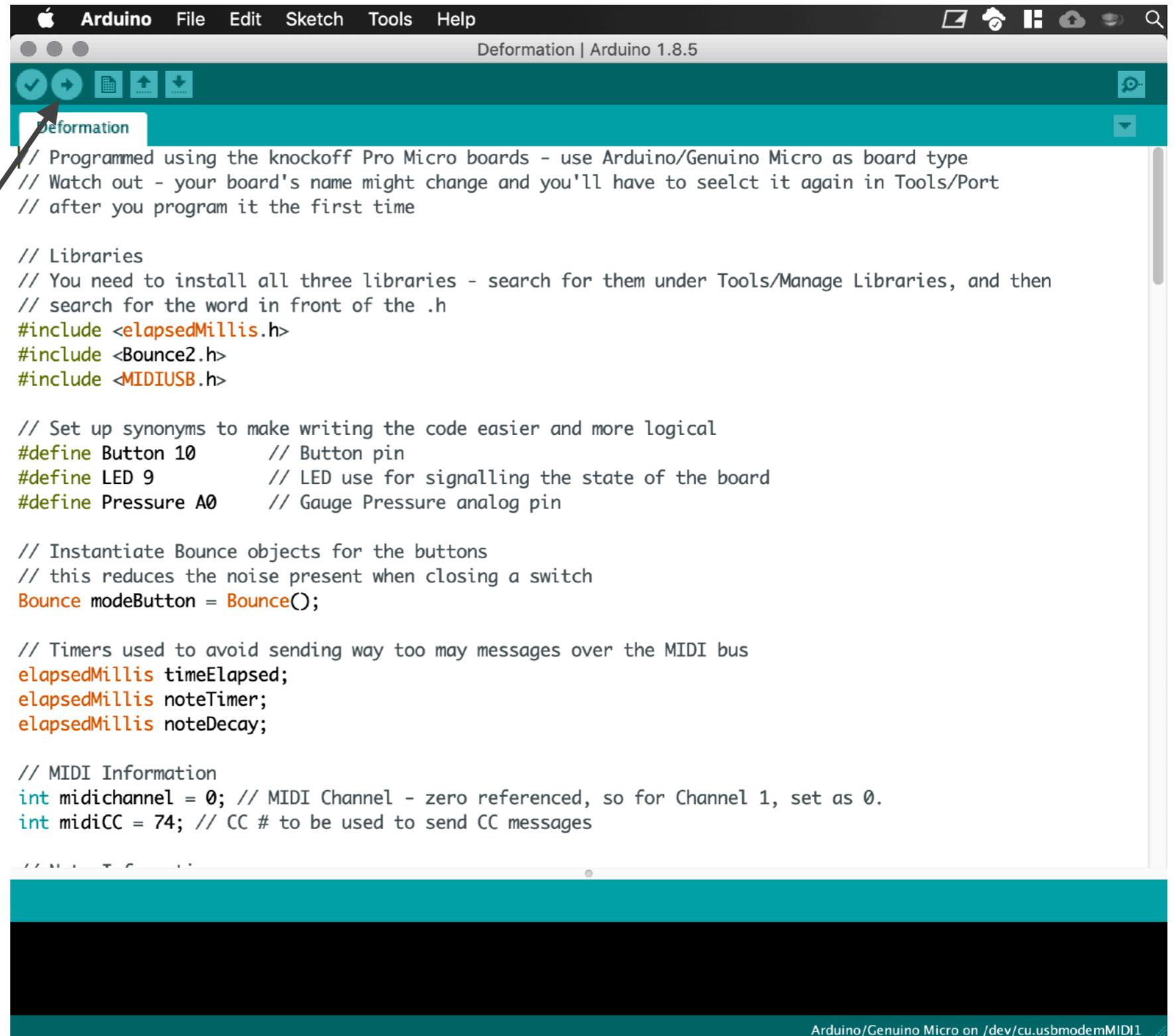
**USB MIDI** by Blokas  
**USB MIDI library for Arduino.** Provides USB MIDI functionality via the familiar Serial interface, porting existing code is trivial. Uses PluggableUSB library for devices that support it. Other AVR8 devices are supported by using V-USB library by Object Development (<https://www.obdev.at/products/vusb/index.html>). Since version 1.1.2, config for Digispark boards was added.  
[More info](#)

The Library Manager window also shows a "Close" button at the bottom right.

# PROGRAM YOUR DEVICE – DEFORMATION.INO

---

*Now program your board with the actual code. This will send out MIDI messages to your computer that HELM will turn into sounds.*



```
Arduino  File  Edit  Sketch  Tools  Help
Deformation | Arduino 1.8.5

Deformation
// Programmed using the knockoff Pro Micro boards - use Arduino/Genuino Micro as board type
// Watch out - your board's name might change and you'll have to select it again in Tools/Port
// after you program it the first time

// Libraries
// You need to install all three libraries - search for them under Tools/Manage Libraries, and then
// search for the word in front of the .h
#include <elapsedMillis.h>
#include <Bounce2.h>
#include <MIDIUSB.h>

// Set up synonyms to make writing the code easier and more logical
#define Button 10      // Button pin
#define LED 9         // LED use for signalling the state of the board
#define Pressure A0   // Gauge Pressure analog pin

// Instantiate Bounce objects for the buttons
// this reduces the noise present when closing a switch
Bounce modeButton = Bounce();

// Timers used to avoid sending way too many messages over the MIDI bus
elapsedMillis timeElapsed;
elapsedMillis noteTimer;
elapsedMillis noteDecay;

// MIDI Information
int midichannel = 0; // MIDI Channel - zero referenced, so for Channel 1, set as 0.
int midiCC = 74; // CC # to be used to send CC messages

// ...
```

Arduino/Genuino Micro on /dev/cu.usbmodemMIDI1

# PROGRAM YOUR DEVICE - MODES

---

*Pressing your button will cycle through the modes:*

***MODE 0 - low LED***

*Send MIDI Notes*

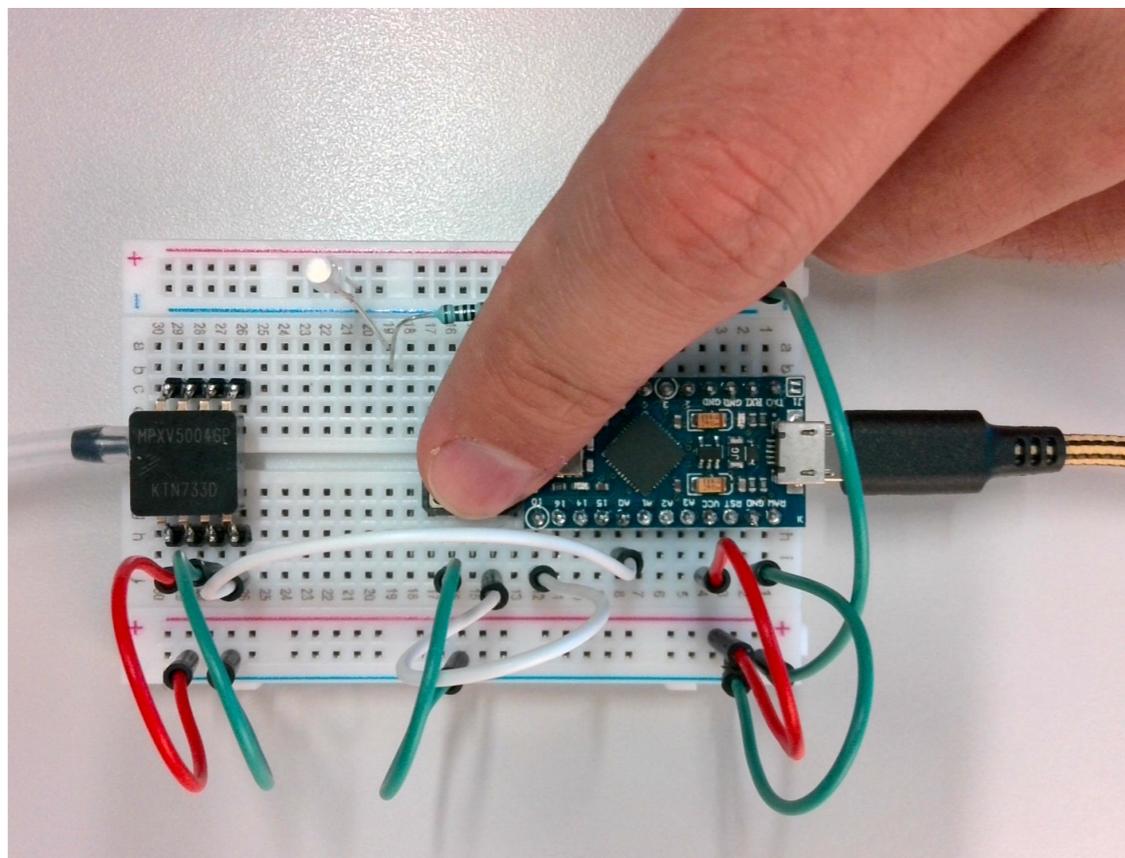
*Send MIDI CC*

***MODE 1 - med LED***

*Send MIDI CC*

***MODE 2 - high LED***

*Set the High Pressure value - make sure you squeeze the balloon in this mode so the pressure range is set properly.*



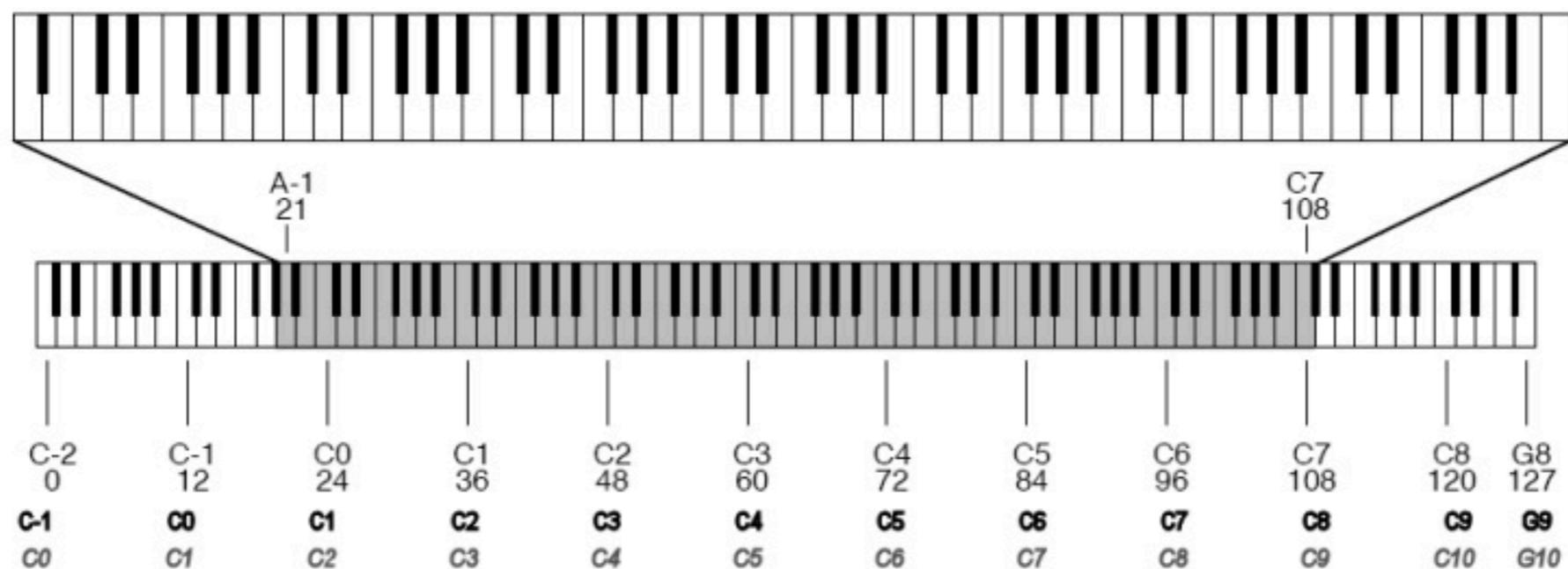
# CHANGE YOUR NOTES!

.....

*Each squeeze will trigger a note randomly from a set of pre-selected notes. You can change what those notes are and how many are in the set. If you make any changes you need to reprogram your arduino with the updated code.*

```
// Note Information
// Pitch defines notes using MIDI note numbers - 60 is middle C, valid pitches are 0-127
// 0 is super low, and 127 is super high! Be wise about the pitches you include!
byte pitch[] = {60, 64, 67, 71, 72}; // You can change which pitch plays by changing these MIDI Note Numbers
int numNotes = 5; // this variable MUST match the number of notes in the pitch array in the previous line
```

## STANDARD PIANO KEYBOARD

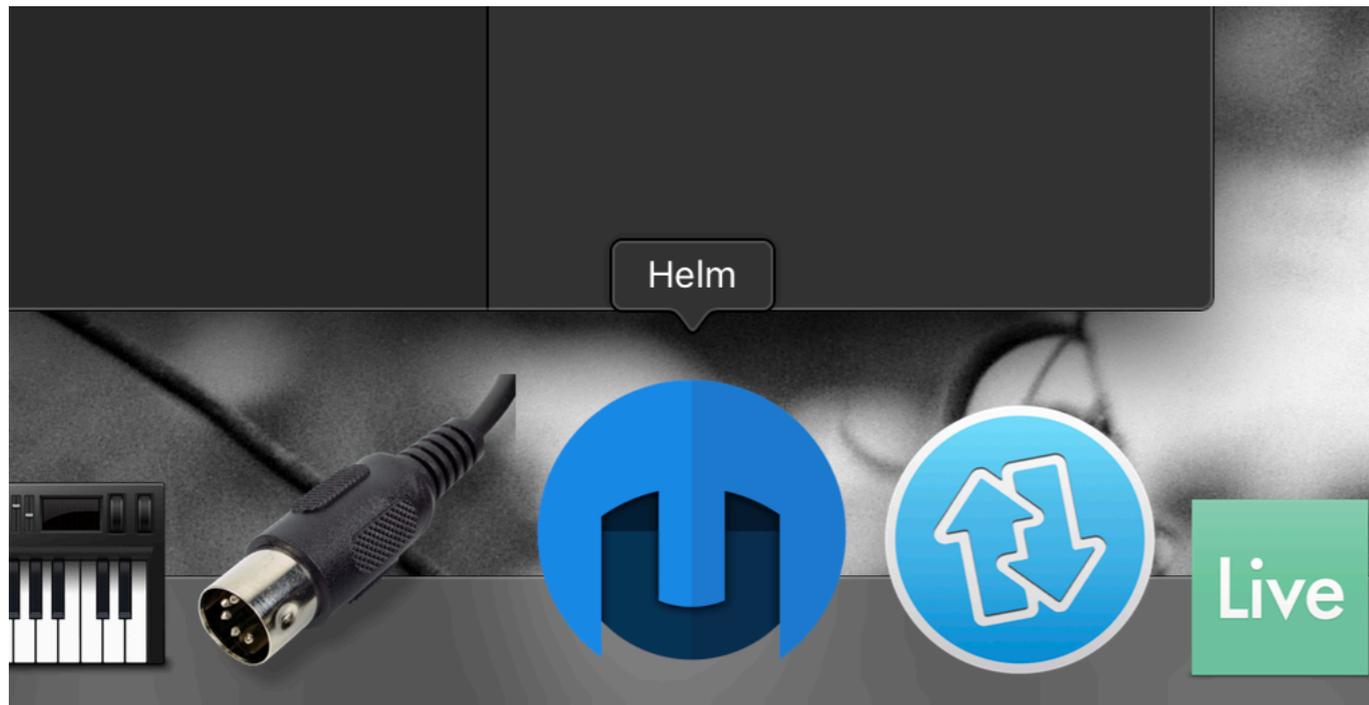


## MIDI NOTES

<https://en.wikipedia.org/wiki/MIDI>

# OPEN HELM – SOFTWARE SYNTHESIZER

---



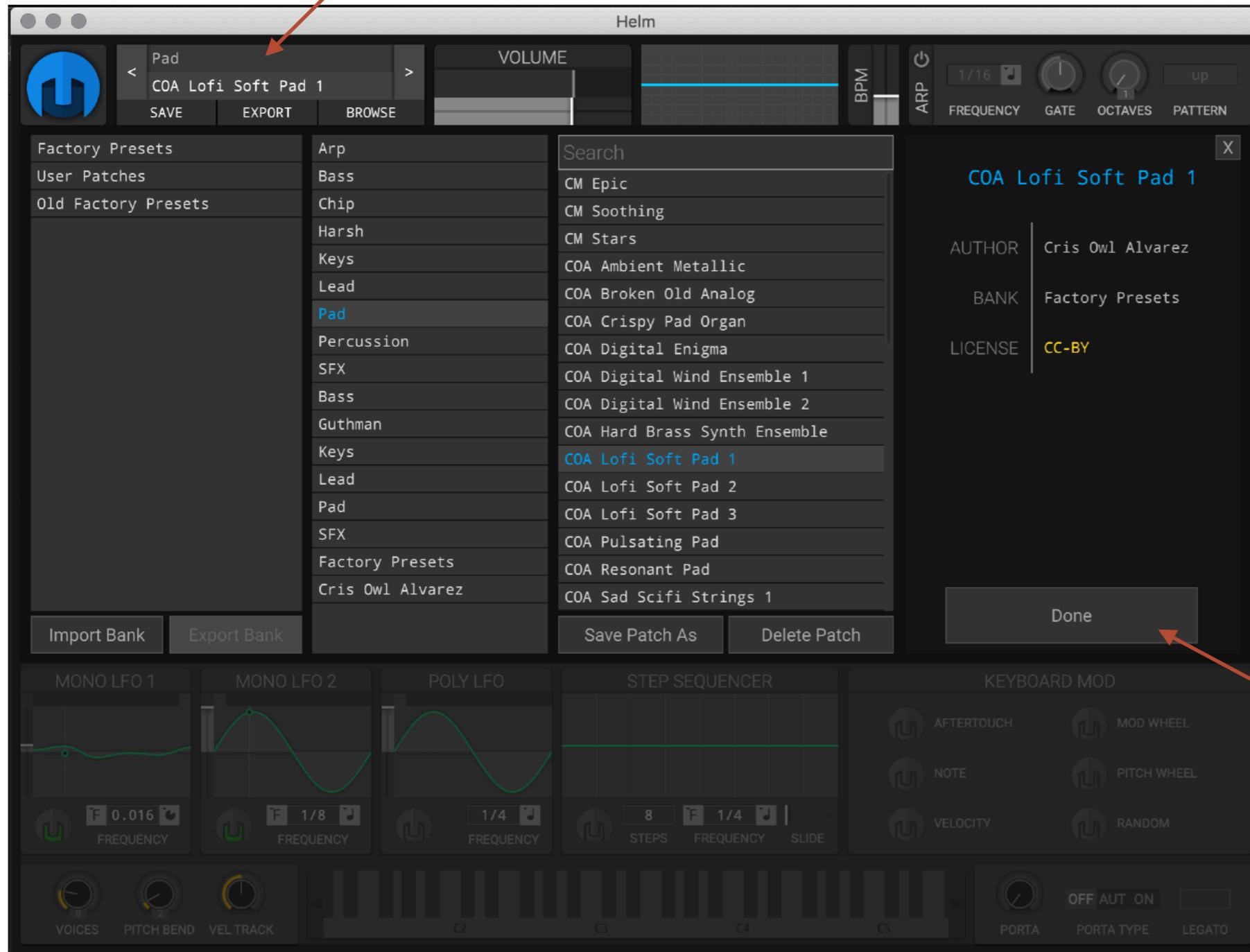
*Helm - a software synthesizer  
to make musical sounds with  
your computer.*

<https://tytel.org/helm/>

# SETUP HELM

*Click here to open the menu to select a sound*

*Make sure your laptop's speakers are on and turned up!*



*Squeeze your balloon quickly to send notes to Helm!*

*Try different sounds, then click done.*

# SETUP HELM



*Make sure your laptop's speakers are on and turned up!*

*Squeeze your balloon quickly to send notes to Helm!*

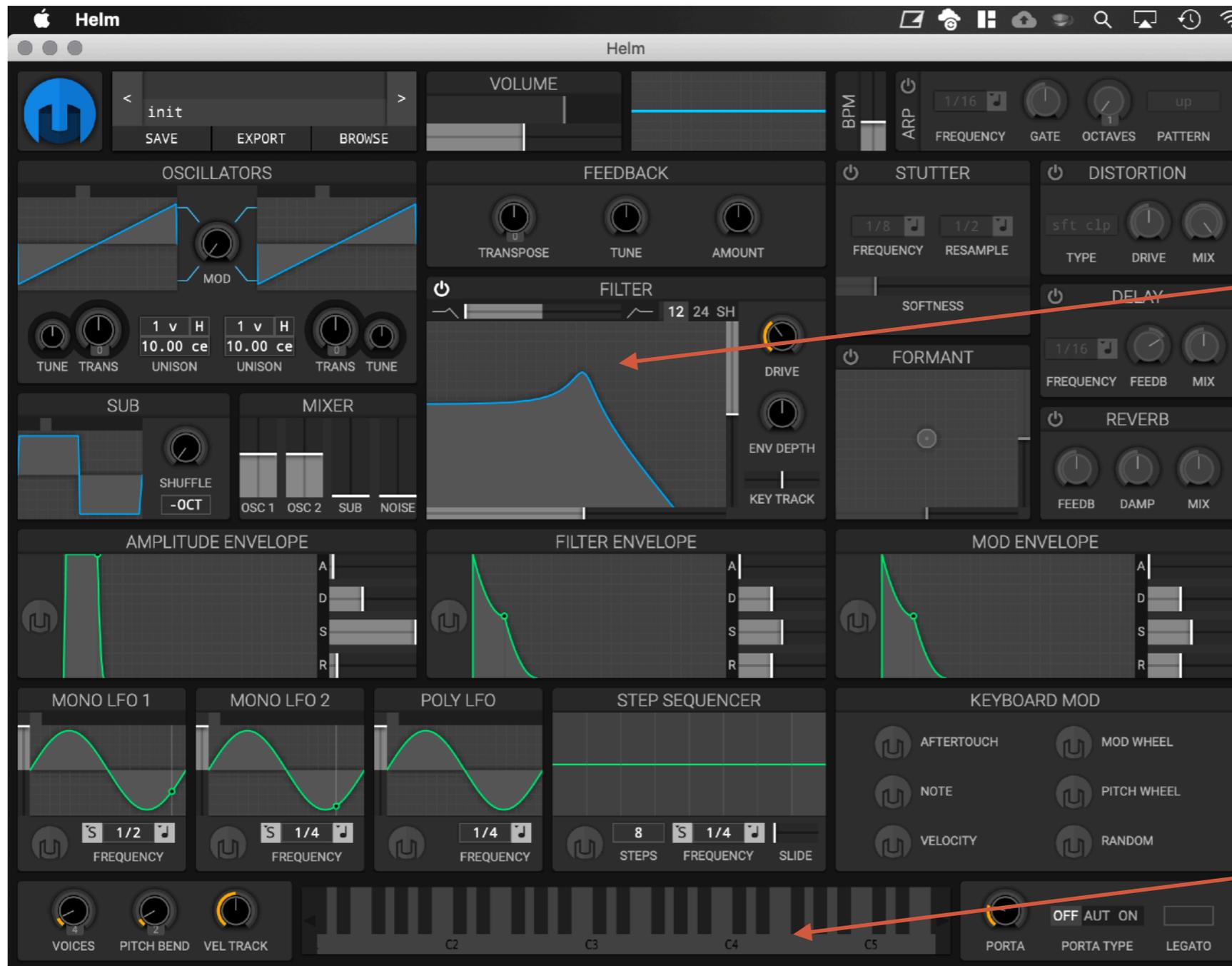
# SETUP HELM



*Make sure the filter is on - it shouldn't be greyed out. If it is greyed out, click the ON button here.*

*Right Click the bar below the filter, then click "Learn MIDI Assignment"*

# SETUP HELM

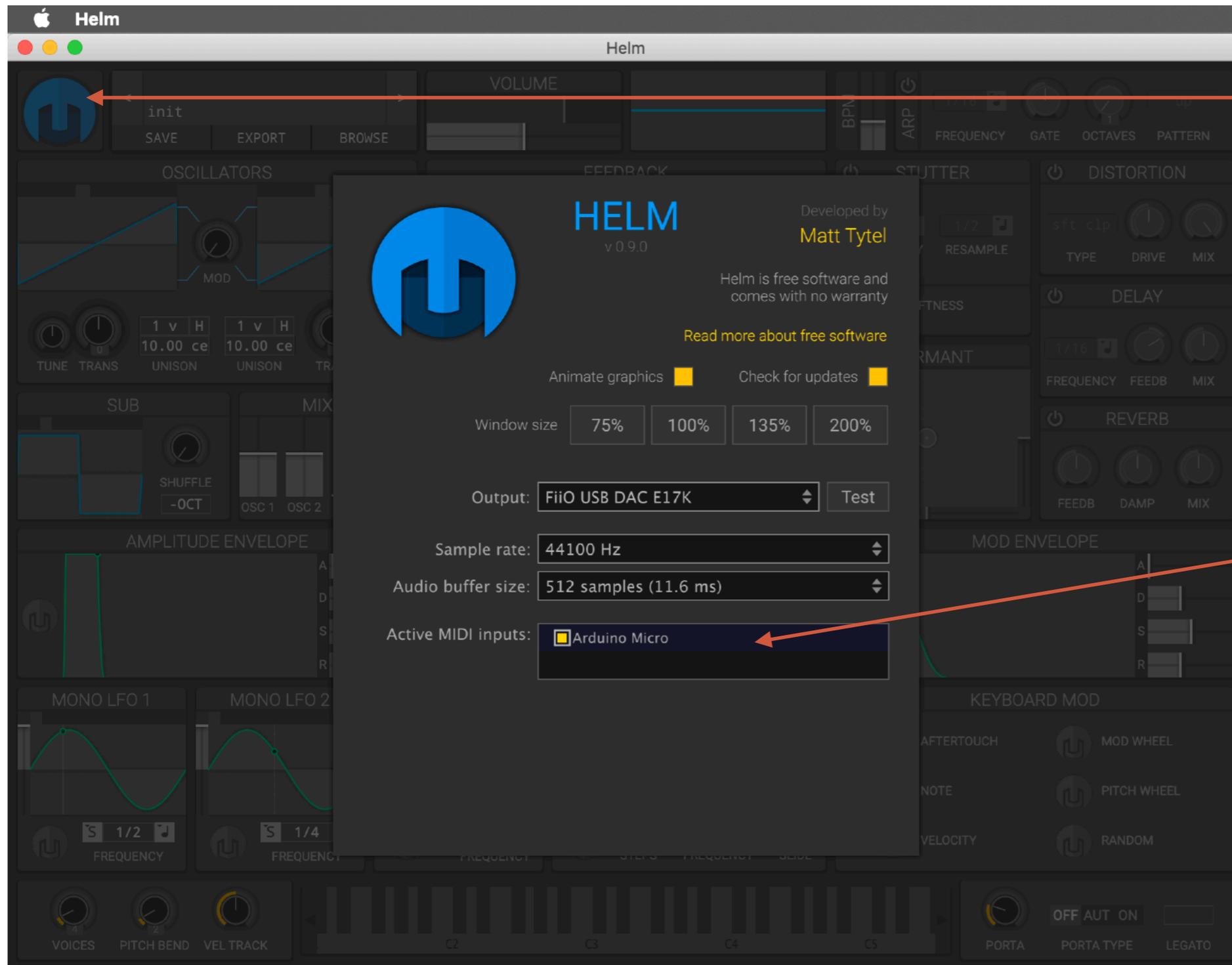


*You should see the filter peak move back and forth as you squeeze your balloon!*

*You should see the keys of the piano lighting up as you squeeze your balloon!*

# NOT WORKING? CHECK THE MIDI PORT

---



*Click the HELMet to get this dialog to open.*

*You should see "Arduino Micro" as a MIDI input, with a yellow box showing it is selected.*