Intro to Arduino

Maya Nasr
MIT-IIT Robotics Program 2017

What is an Arduino?

- The Arduino is a family of microcontroller boards used to easily design and experiment with electronics.

What is an Arduino?

- The Arduino is a family of microcontroller boards used to easily design and experiment with electronics.
- Open Source Physical Computing Platform

What is an Arduino?

- The Arduino is a family of microcontroller boards used to easily design and experiment with electronics.
- Open Source Physical Computing Platform
- Use it as brains for your robot.

What is an Arduino?

- The Arduino is a family of microcontroller boards used to easily design and experiment with electronics.
- Open Source Physical Computing Platform
- Use it as brains for your robot.
- Tiny computer you can program

What is an Arduino?

- The Arduino is a family of microcontroller boards used to easily design and experiment with electronics.
- Open Source Physical Computing Platform
- Use it as brains for your robot.
- Tiny computer you can program
- Arduinos contain a microcontroller — that's a complete computer with CPU, RAM, Flash memory, and input/output pins,all on a single chip.

What is an Arduino?

- It's designed to attach all kinds of sensors, LEDs, small motors and speakers, servos, etc. directly to these pins, which can read in or output digital or analog voltages between 0 and 5 volts.

What is an Arduino?

- It's designed to attach all kinds of sensors, LEDs, small motors and speakers, servos, etc. directly to these pins, which can read in or output digital or analog voltages between 0 and 5 volts.
- The Arduino connects to your computer via USB, where you program it in a simple language (C/C++) inside the free Arduino IDE by uploading your compiled code to the board.

What is an Arduino?

- It's designed to attach all kinds of sensors, LEDs, small motors and speakers, servos, etc. directly to these pins, which can read in or output digital or analog voltages between 0 and 5 volts.
- The Arduino connects to your computer via USB, where you program it in a simple language (C/C++) inside the free Arduino IDE by uploading your compiled code to the board.
- Once programmed, the Arduino can run with the USB link back to your computer, or stand-alone without it —no keyboard or screen needed, just power.

But how do you program it?

- Write programs on your PC

But how do you program it?

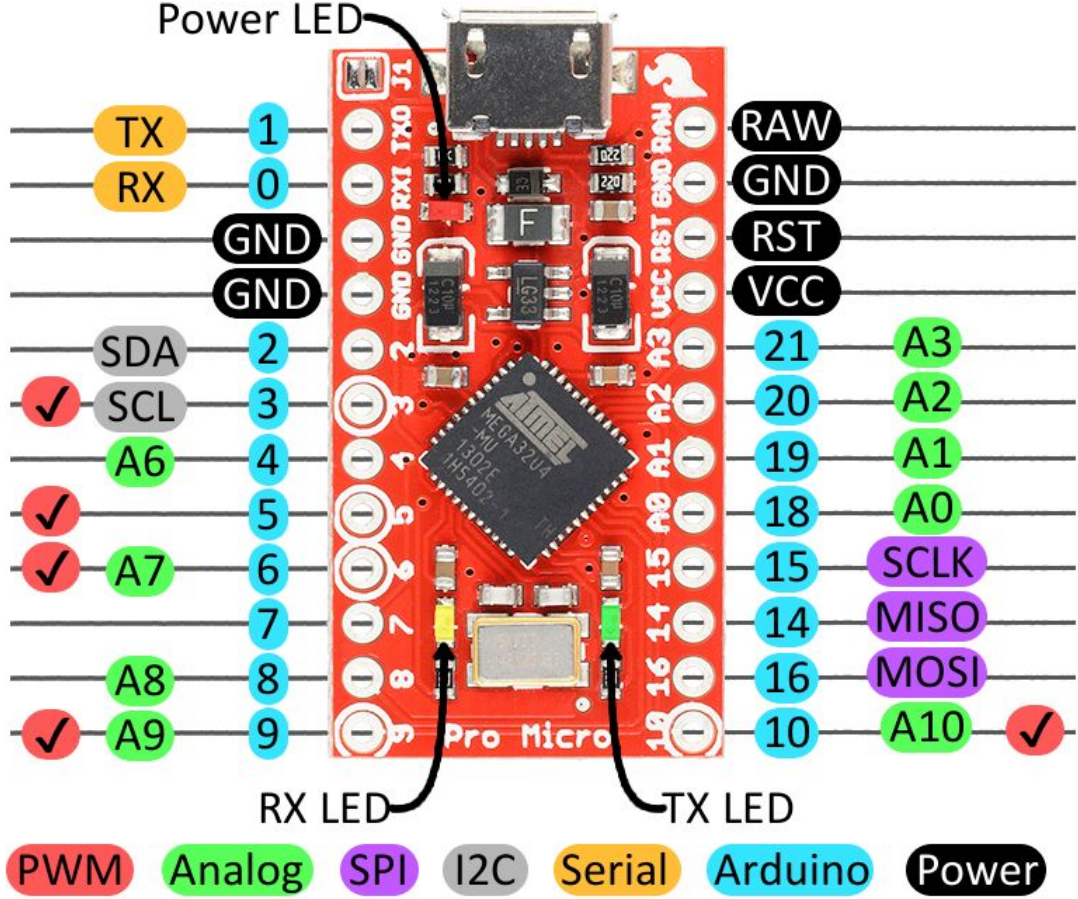
- Write programs on your PC
- Download them into the Arduino board

But how do you program it?

- Write programs on your PC
- Download them into the Arduino board
- Arduino board can then be used by itself

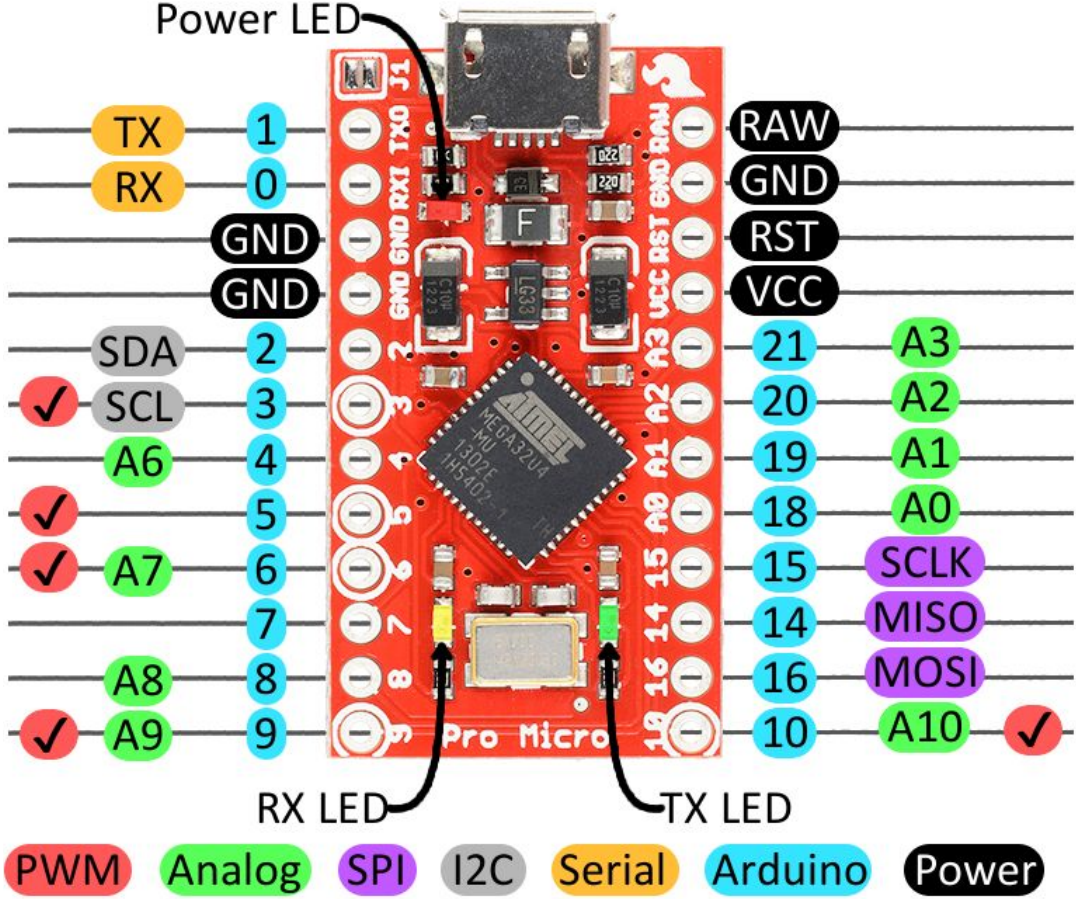
Arduino Pro Micro Board

- **RAW** is the unregulated voltage input for the Pro Micro. If the board is powered via USB, the voltage at this pin will be about 4.8V.



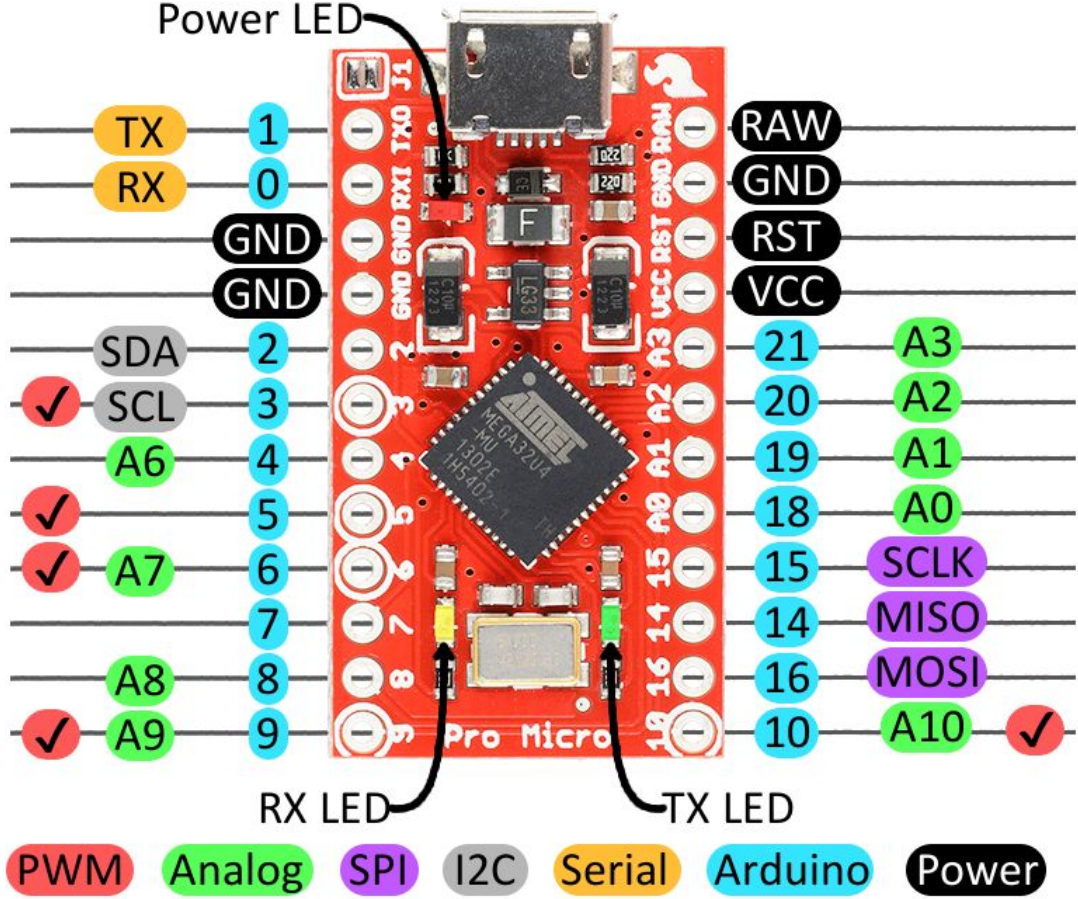
Arduino Pro Micro Board

- **RAW** is the unregulated voltage input for the Pro Micro. If the board is powered via USB, the voltage at this pin will be about 4.8V.
- **VCC** is the voltage supplied to the on-board ATmega32U4. This voltage will depend on whether you're using a 3.3V/8MHz Pro Micro or a 5V/16MHz version, it'll be either 3.3V or 5V respectively.



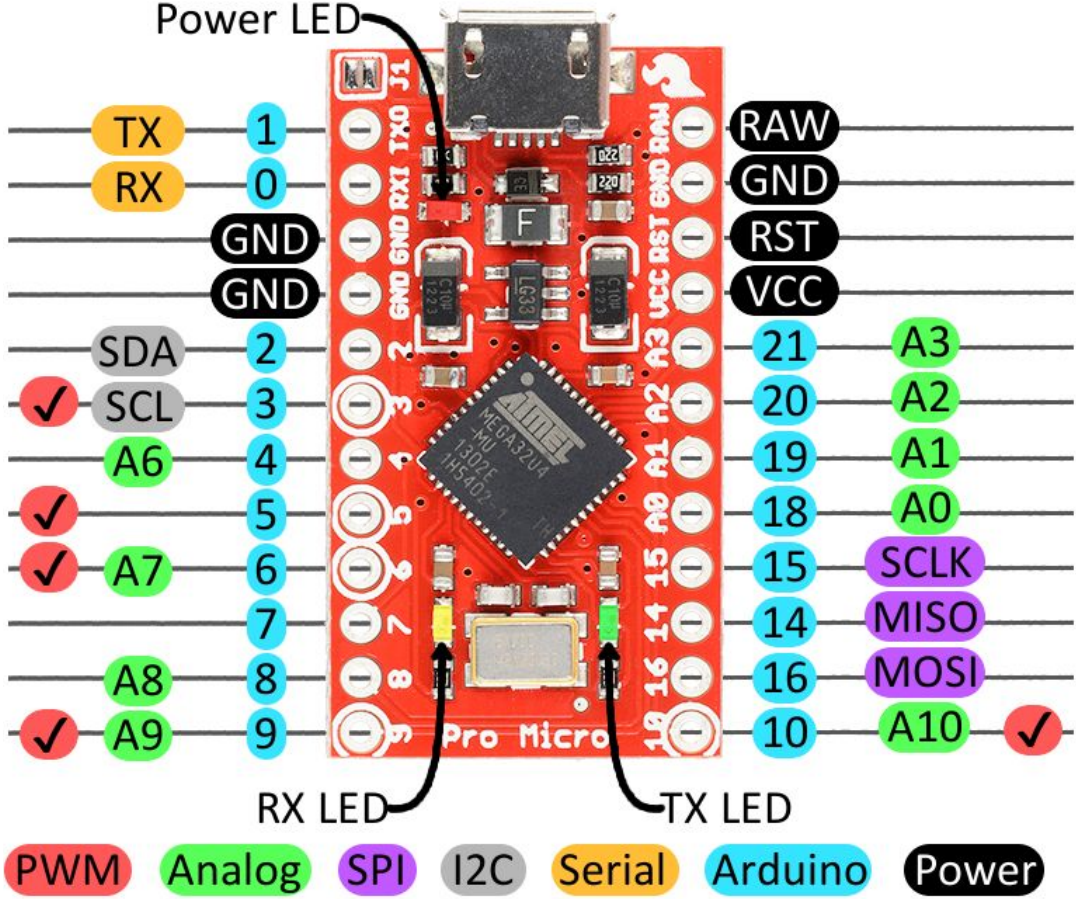
Arduino Pro Micro Board

- **RAW** is the unregulated voltage input for the Pro Micro. If the board is powered via USB, the voltage at this pin will be about 4.8V.
- **VCC** is the voltage supplied to the on-board ATmega32U4. This voltage will depend on whether you're using a 3.3V/8MHz Pro Micro or a 5V/16MHz version, it'll be either 3.3V or 5V respectively.
- **RST** can be used to restart the Pro Micro.



Arduino Pro Micro Board

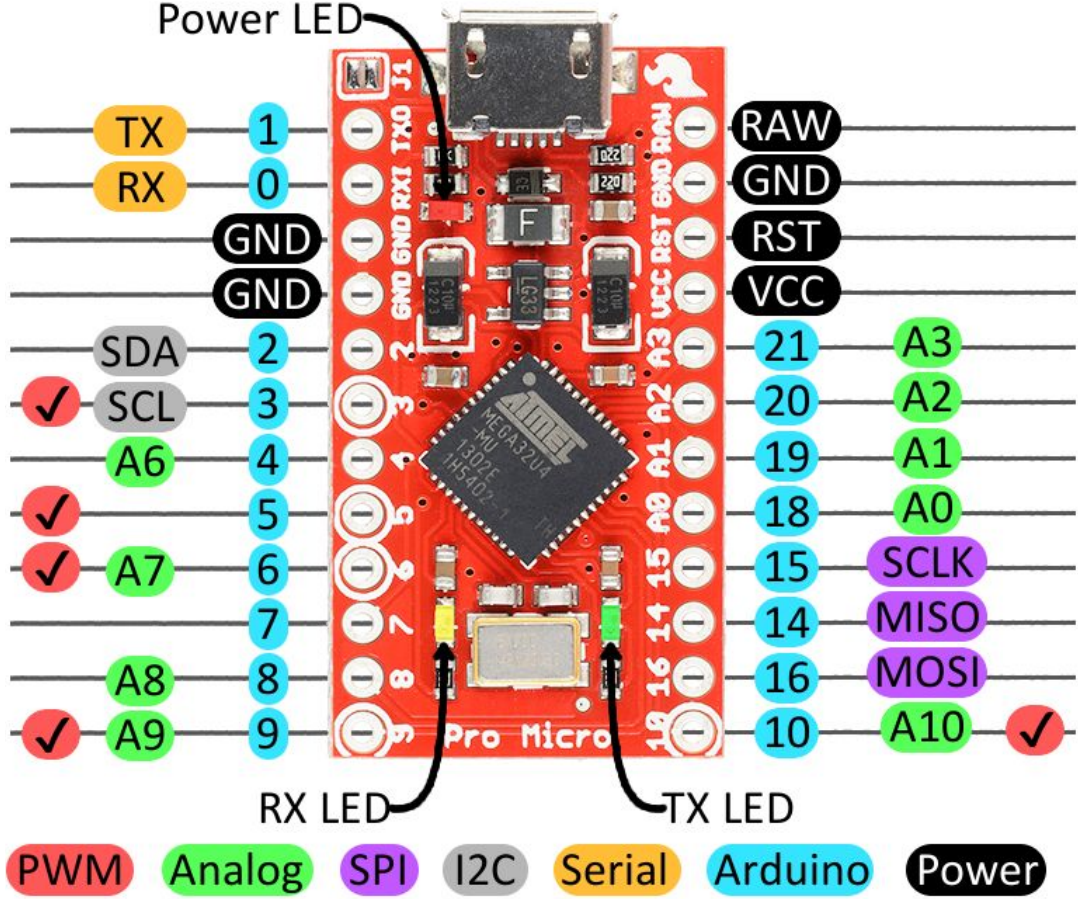
- **RAW** is the unregulated voltage input for the Pro Micro. If the board is powered via USB, the voltage at this pin will be about 4.8V.
- **VCC** is the voltage supplied to the on-board ATmega32U4. This voltage will depend on whether you're using a 3.3V/8MHz Pro Micro or a 5V/16MHz version, it'll be either 3.3V or 5V respectively.
- **RST** can be used to restart the Pro Micro.
- **GND** is the common, ground voltage (0V reference) for the system.



Arduino Pro Micro Board

On-Board LEDs

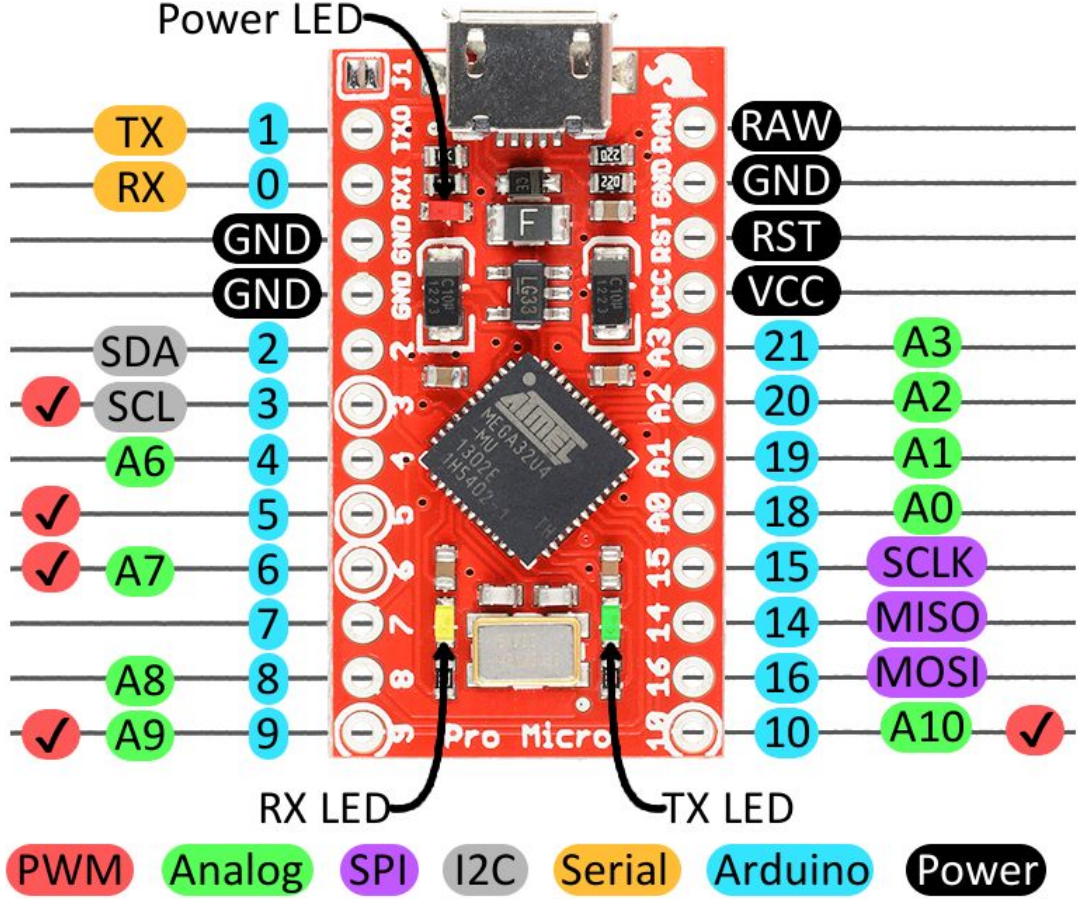
- There are three LEDs on the Pro Micro.



Arduino Pro Micro Board

On-Board LEDs

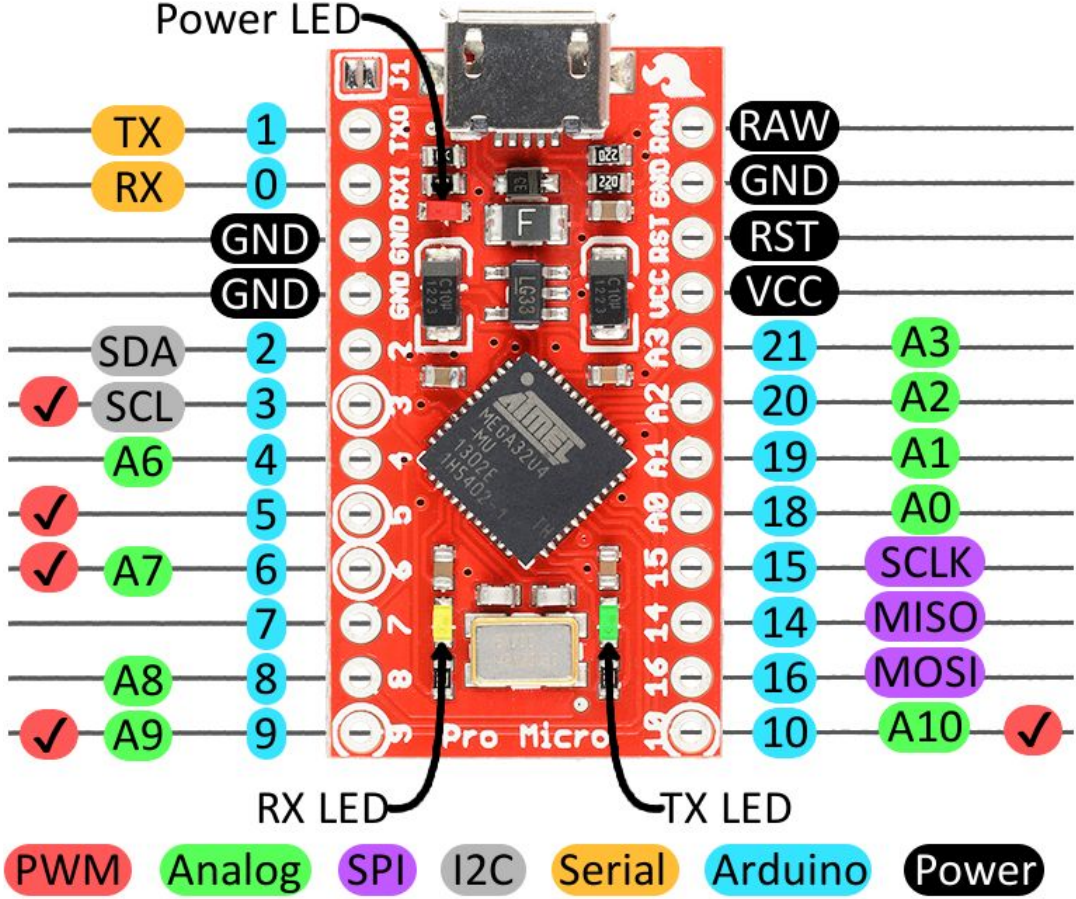
- There are three LEDs on the Pro Micro.
- One red LED indicates whether power is present.



Arduino Pro Micro Board

On-Board LEDs

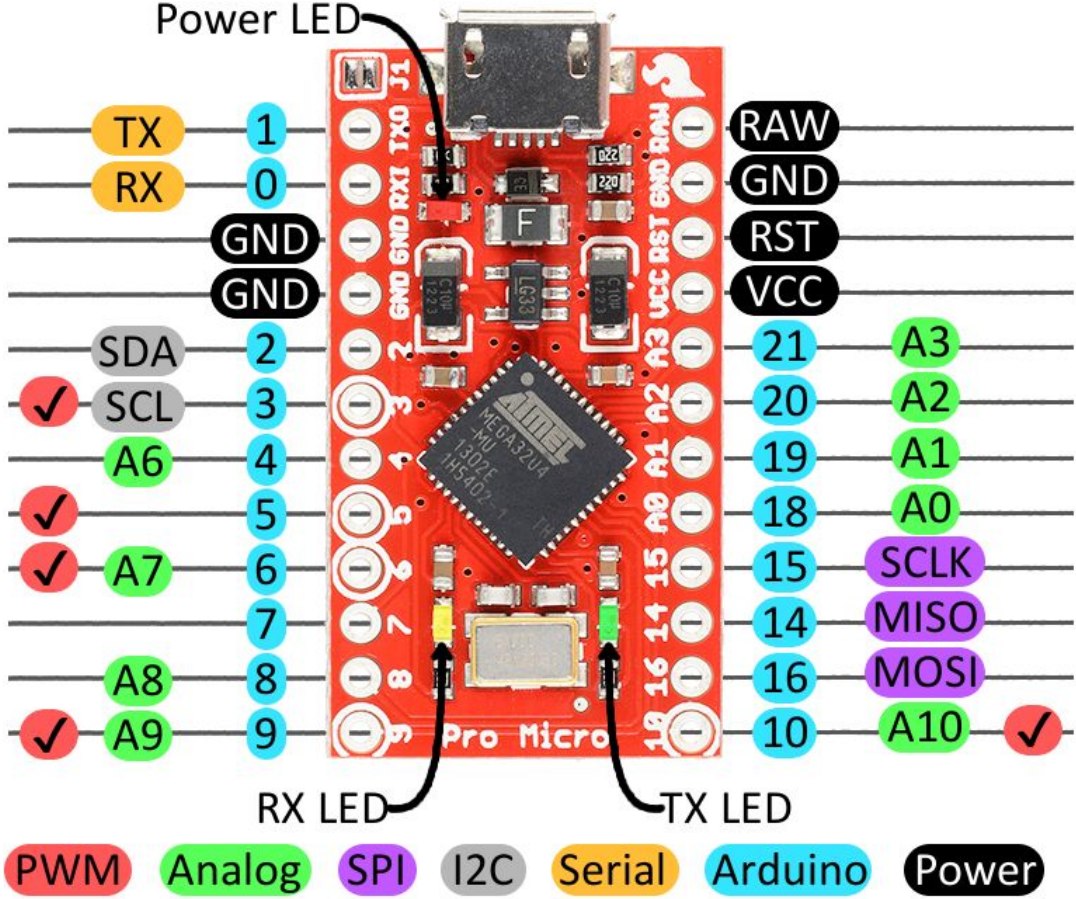
- There are three LEDs on the Pro Micro.
- One red LED indicates whether power is present.
- The other two LEDs help indicate when data is transferring over USB.



Arduino Pro Micro Board

On-Board LEDs

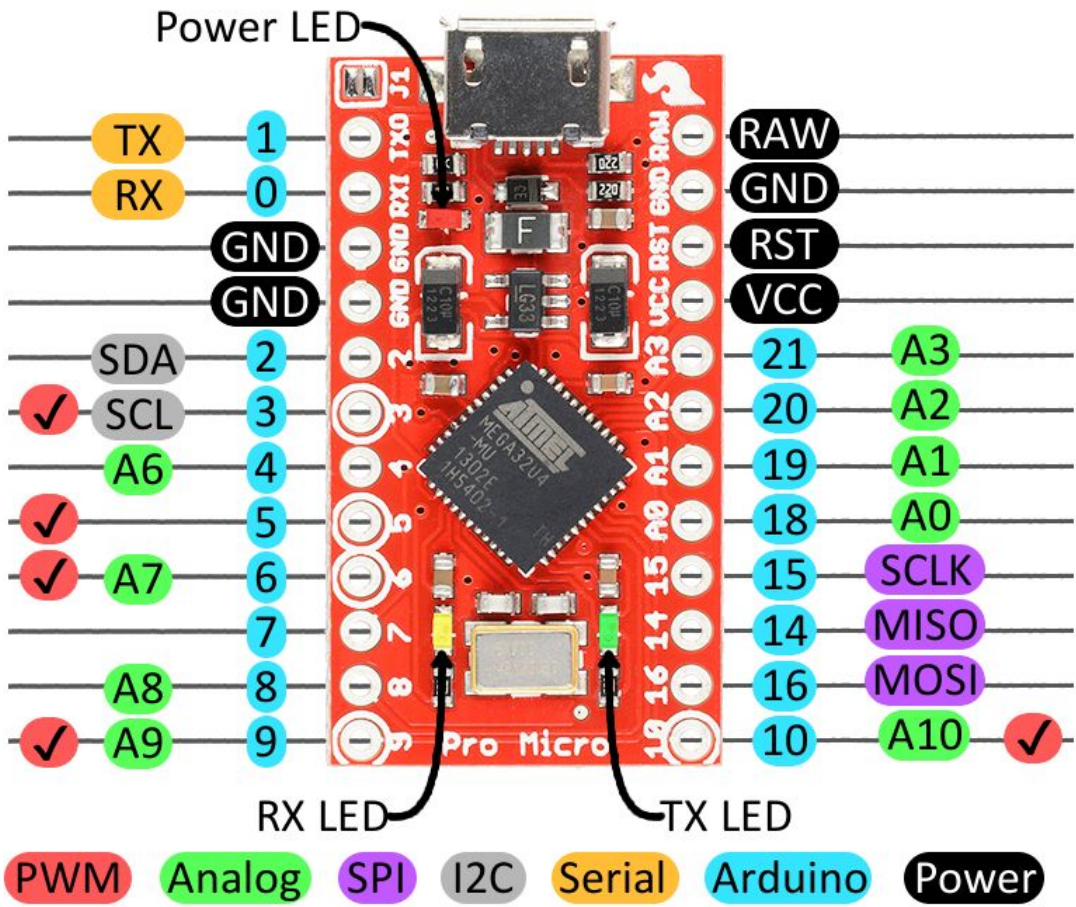
- There are three LEDs on the Pro Micro.
- One red LED indicates whether power is present.
- The other two LEDs help indicate when data is transferring over USB.
- A yellow LED represents USB data coming *into* (RX) the the Pro Micro, and a green LED indicates USB data going out (TX).



Arduino Pro Micro Board

I/O Pins

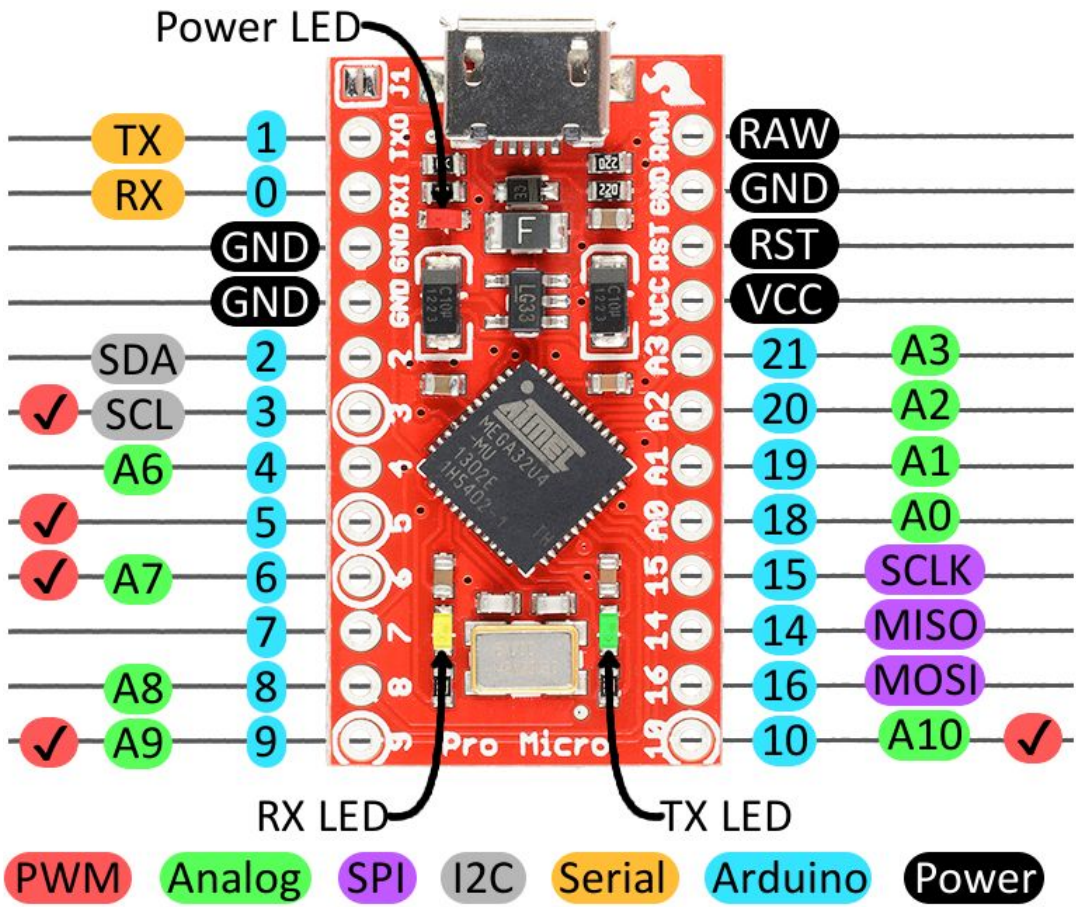
- The Pro Micro's I/O pins are 18 in all.



Arduino Pro Micro Board

I/O Pins

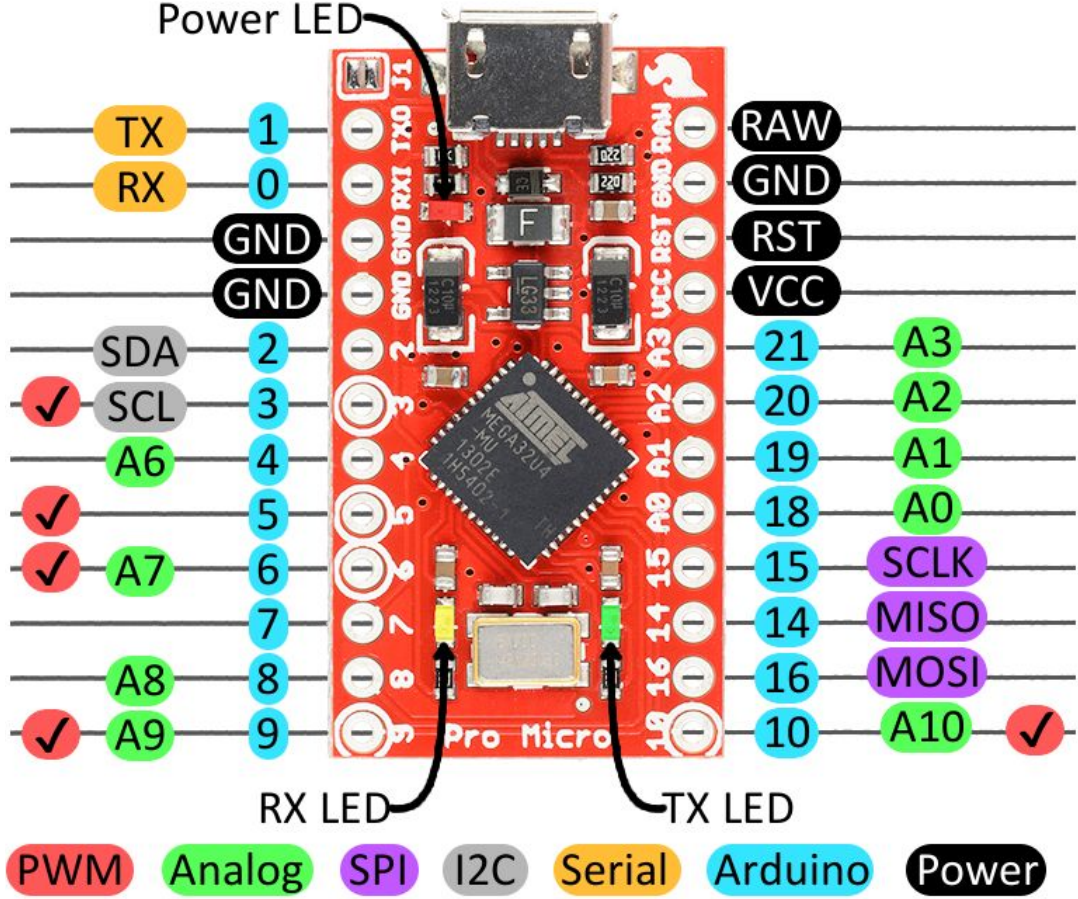
- The Pro Micro's I/O pins are 18 in all.
- Every pin can be used as a digital



Arduino Pro Micro Board

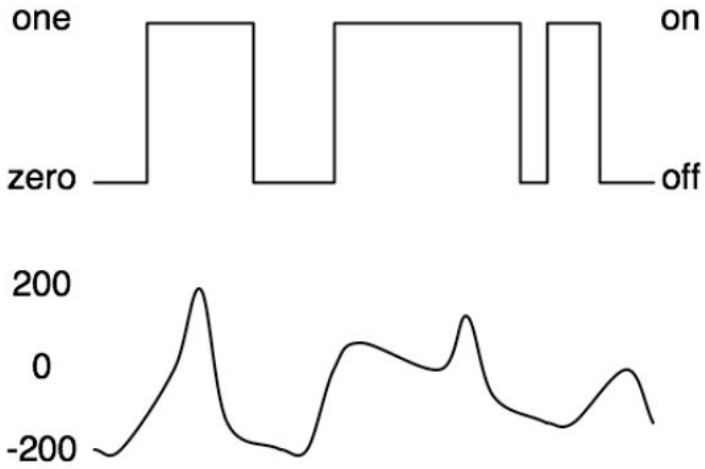
I/O Pins

- The Pro Micro's I/O pins are 18 in all.
- Every pin can be used as a digital input or output.
- These pins are referenced in the Arduino IDE via an integer value between 0 and 21.



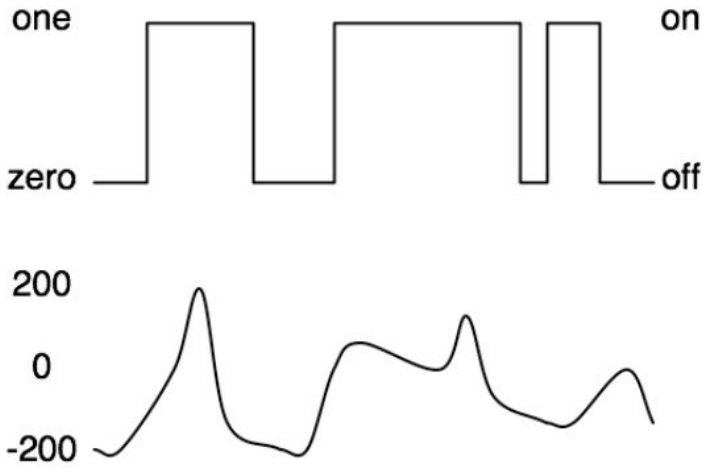
Digital? Analog?

- **Digital** – only has two values: on/off



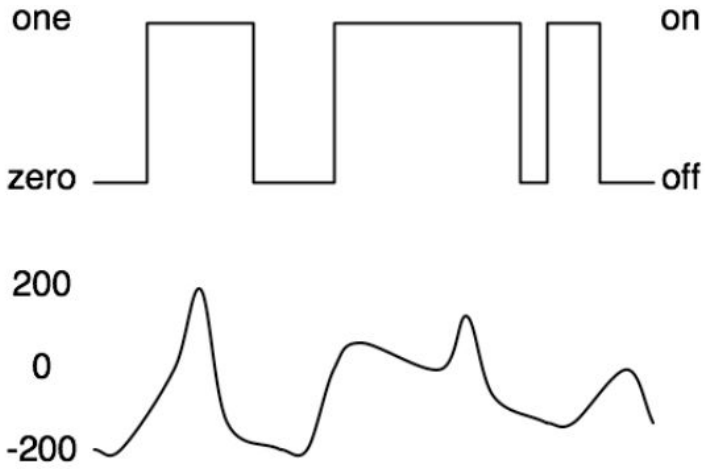
Digital? Analog?

- **Digital** – only has two values: on/off
- **Analog** – has many (infinite) values



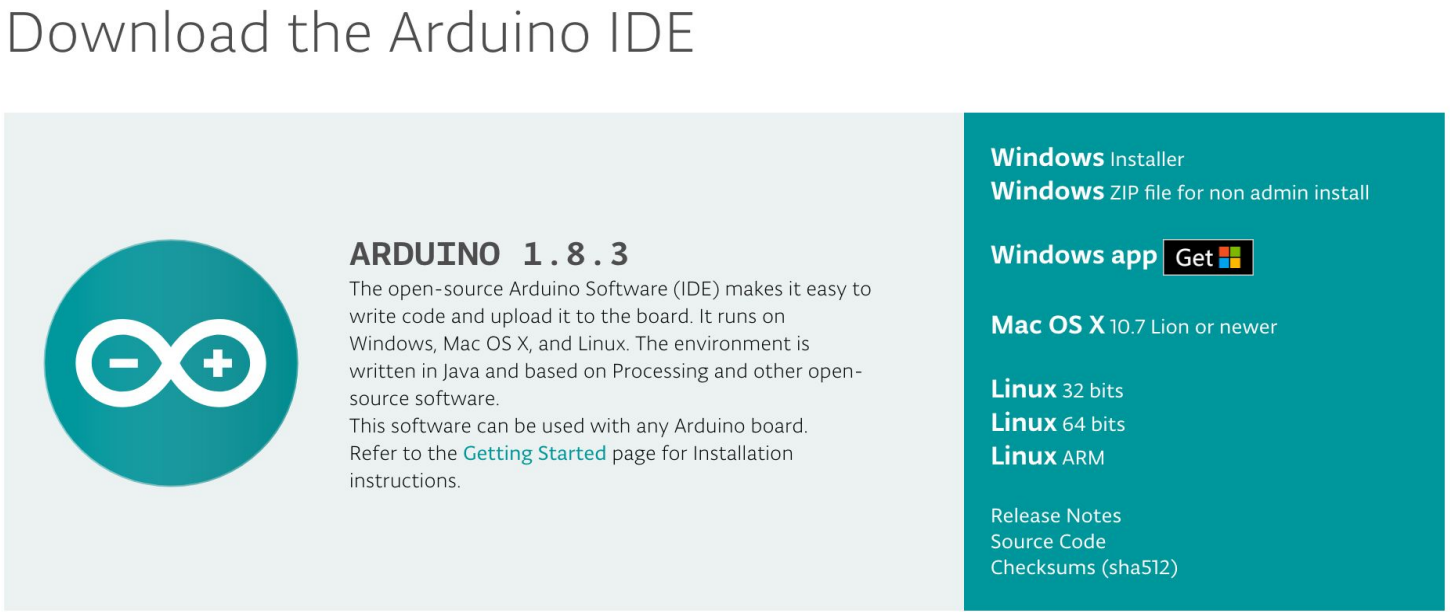
Digital? Analog?

- **Digital** – only has two values: on/off
- **Analog** – has many (infinite) values
- Computers don't really do analog
- So they fake it, with quantization (Quantization = breaking up the analog range into bins. The number of bins is the resolution.)



Installing Arduino

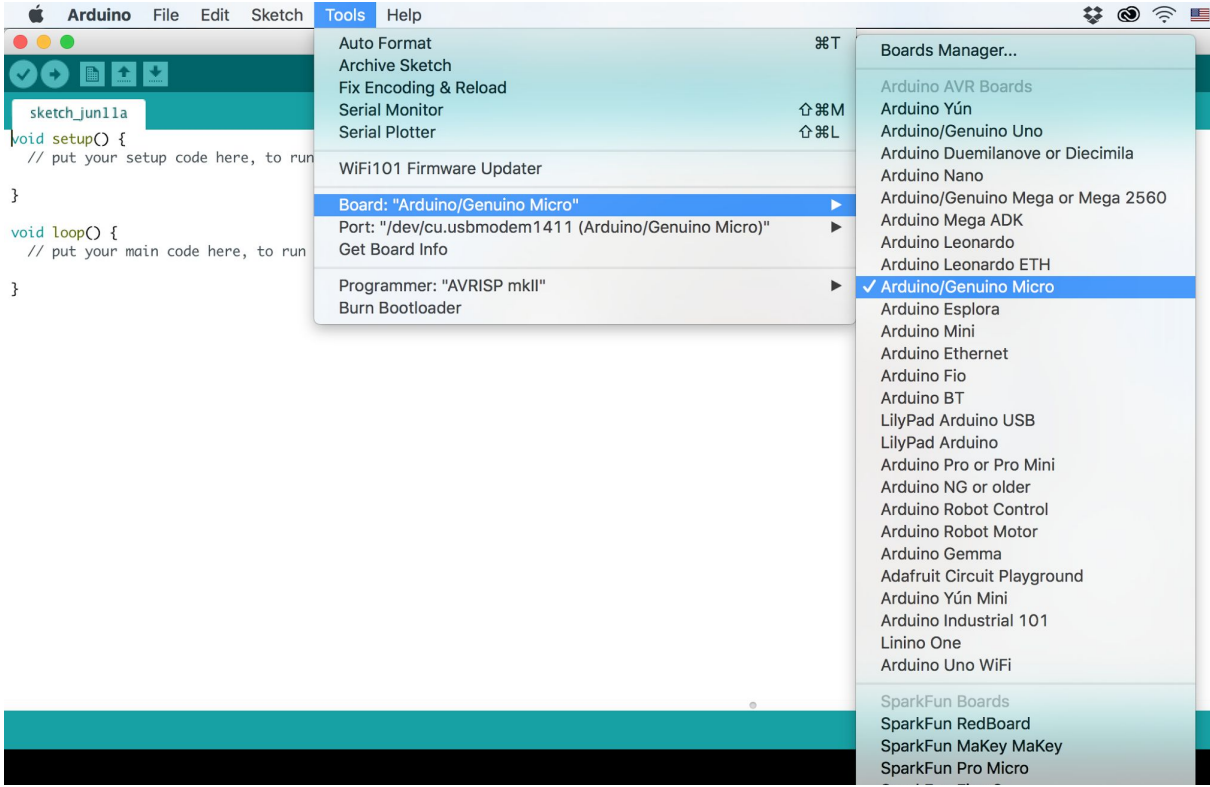
- Download software: https://www.arduino.cc/en/Main/Software



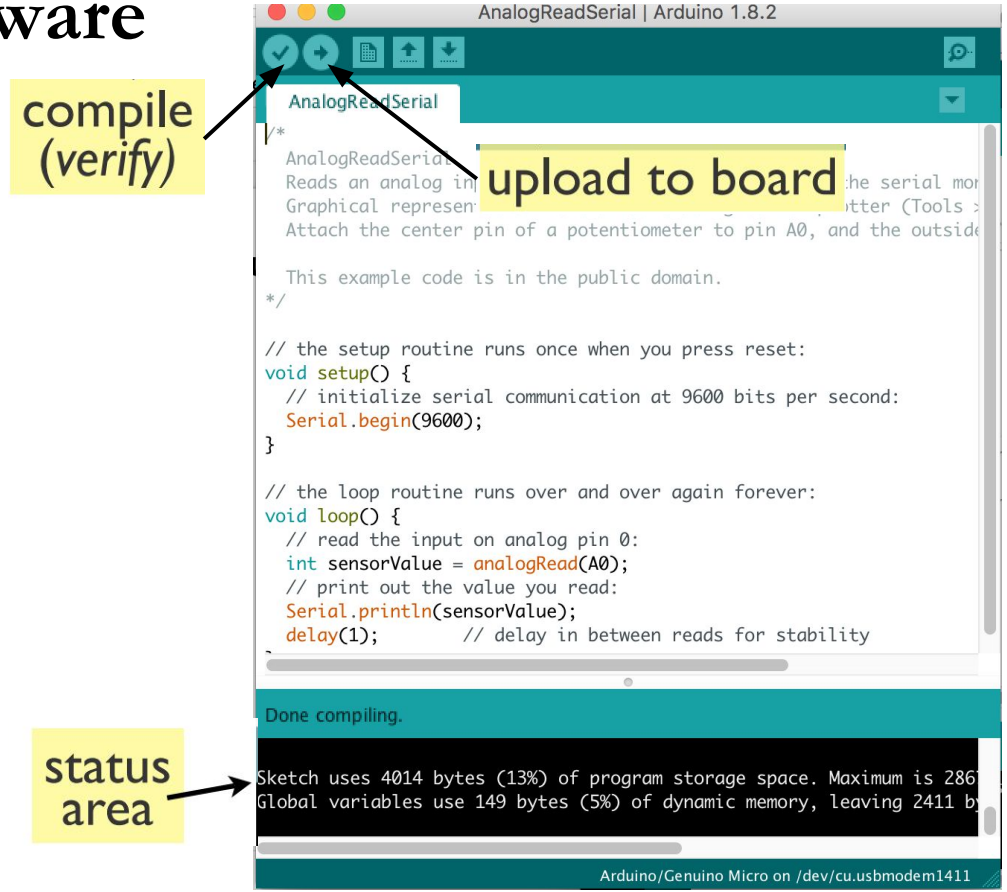
Installing Arduino

- Click OK. Then open the Board Manager by clicking Tools, then hovering over the Board selection tab and clicking Board Manager

- Click on Arduino/Genuino Micro



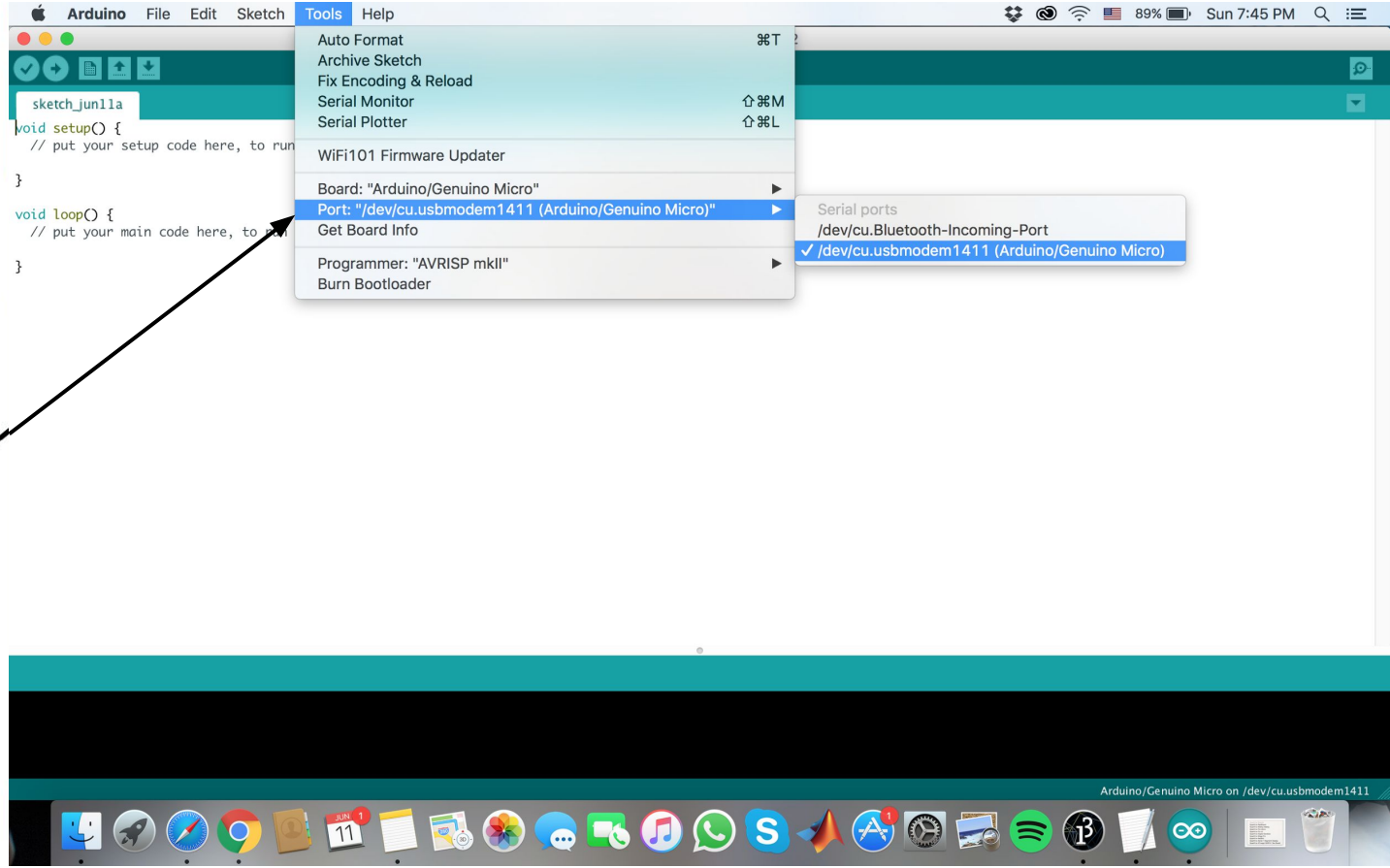
Arduino Software



Errors

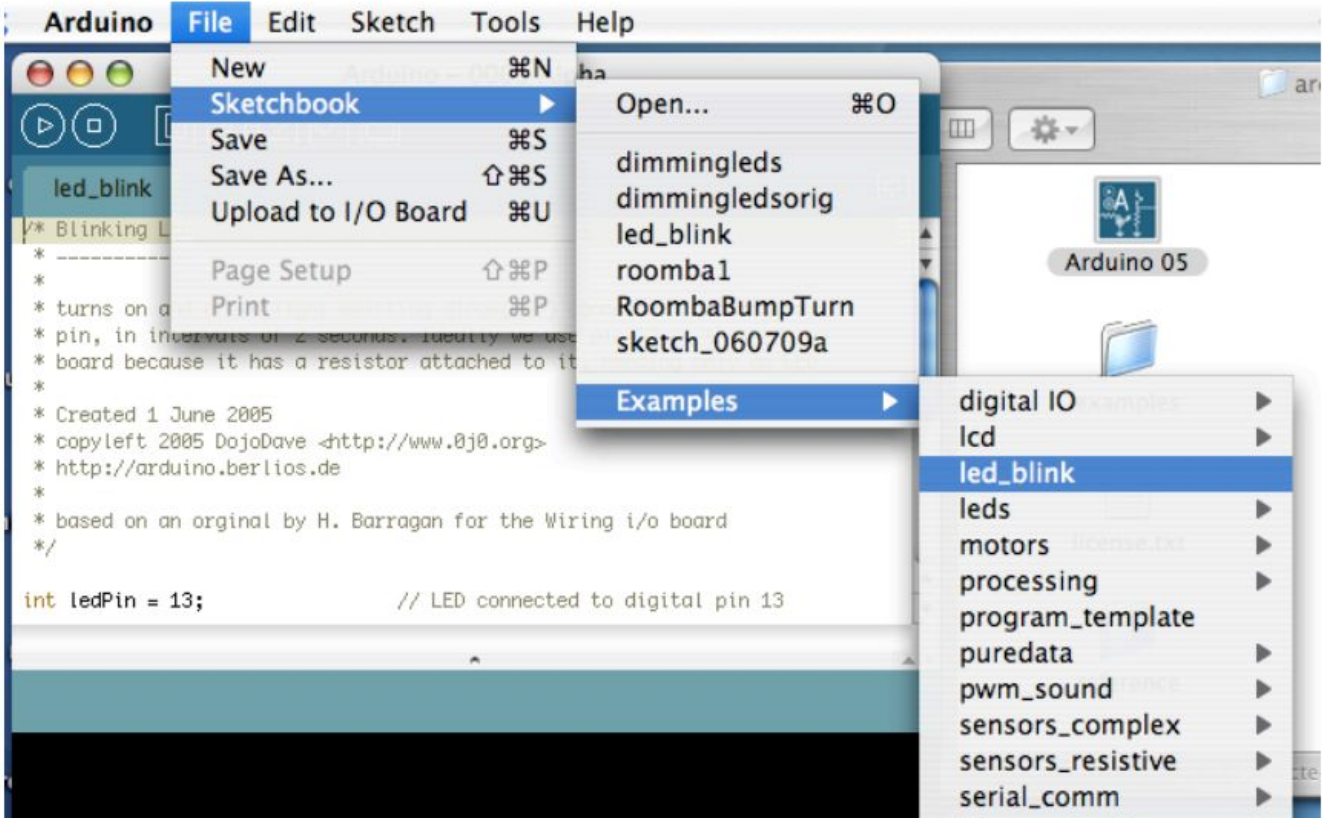
"Programmer is not responding"

Must select serial port



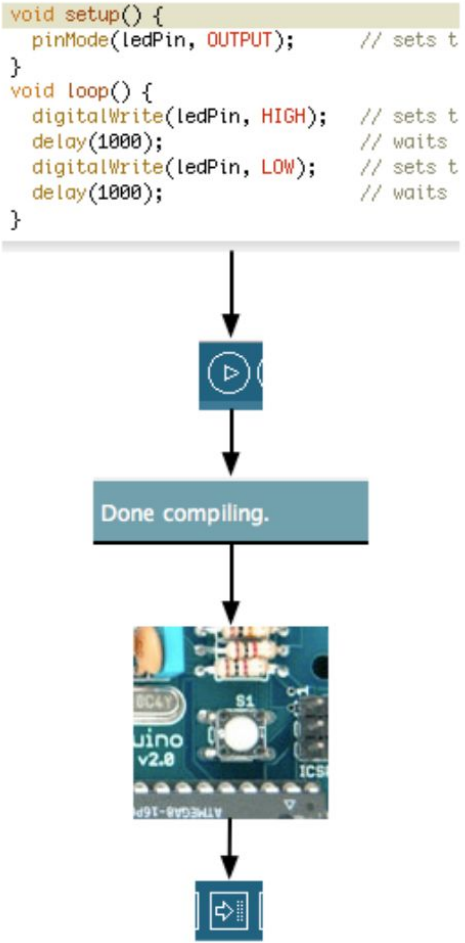
Using Arduino

Programs are called "**sketches**"



Using Arduino

- Write program
- Compile (check for errors)
- Reset board
- Upload to board



Arduino Sketch Structure

- Declare variables at top
- Initialize
    - `setup()` – run once at beginning, set pins

Arduino Sketch Structure

- Declare variables at top
- Initialize
    - `setup()` – run once at beginning, set pins
- Running
    - `loop()` – run repeatedly, after `setup()`

Arduino “Language”

Language is standard C, with lots of useful libraries, examples, and functions

- **`pinMode(pin, mode)`** – set a pin as `INPUT` or `OUTPUT`

Arduino “Language”

Language is standard C, with lots of useful libraries, examples, and functions

- **`pinMode(pin, mode)`** – set a pin as `INPUT` or `OUTPUT`
- **`digitalWrite(pin, value)`** – set a digital pin that is set as an `OUTPUT` as either `HIGH` (pulled to +5 volts) or `LOW` (pulled to ground).

Arduino “Language”

Language is standard C, with lots of useful libraries, examples, and functions

- **`pinMode(pin, mode)`** – set a pin as `INPUT` or `OUTPUT`
- **`digitalWrite(pin, value)`** – set a digital pin that is set as an `OUTPUT` as either `HIGH` (pulled to +5 volts) or `LOW` (pulled to ground).
- **`digitalRead(pin)`** – read a digital pin’s (that is set as an `INPUT`) state; returns whether it is `HIGH` (pulled to +5 volts) or `LOW` (pulled to ground).

More Functions

- **`analogWrite(pin, value)`** – write an "analog" PWM value. Some of the Arduino's pins support pulse width modulation. This turns the pin on and off very quickly making it act like an analog output. The value is any number between 0 (0% duty cycle ~0v) and 255 (100% duty cycle ~5 volts).

More Functions

- **`analogWrite(pin, value)`** – write an "analog" PWM value. Some of the Arduino's pins support pulse width modulation. This turns the pin on and off very quickly making it act like an analog output. The value is any number between 0 (~0v) and 255 (~5 volts).

- **`analogRead(pin)`** – read an analog pin. When the analog input pins are set to input you can read their voltage. A value between 0 (for 0 volts) and 1024 (for 5 volts) will be returned.

More Functions

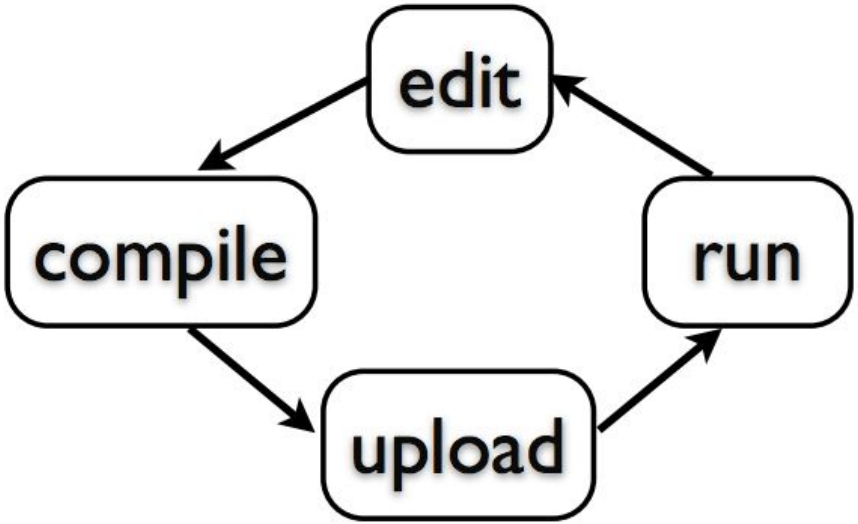
- **`analogWrite(pin, value)`** – write an "analog" PWM value. Some of the Arduino's pins support pulse width modulation. This turns the pin on and off very quickly making it act like an analog output. The value is any number between 0 (0% duty cycle ~0v) and 255 (100% duty cycle ~5 volts).

- **`analogRead(pin)`** – read an analog pin. When the analog input pins are set to input you can read their voltage. A value between 0 (for 0 volts) and 1024 (for 5 volts) will be returned.

- **`delay()`** – wait an amount of time

Development Cycle

Edit → compile → upload → run



RX/ TX Blink Lab

Write and upload a code uploaded to see the RX and TX LEDs take turns blinking on and off every second.

RX/ TX Blink Lab Understanding

- The RX LED is tied to Arduino's pin 17.
- You can control it just as you would any other digital pin.
- Set it as an `OUTPUT`, and `digitalWrite([pin], [level])` it `HIGH` or `LOW`.
- The TX LED was not provided as an Arduino-defined pin, unfortunately, so you'll have to use a pair of macros to control it. `TXLED1` turns the LED on, and `TXLED0` turns the LED off.