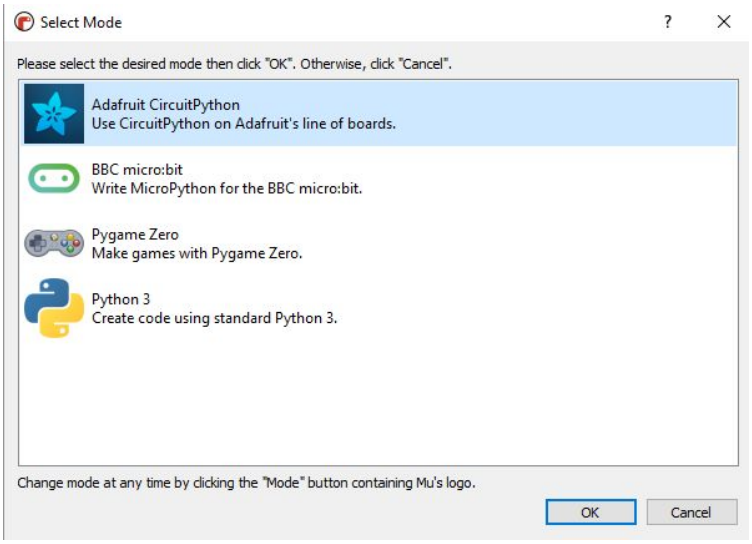
Circuit Building and Programming with The Adafruit Itsy Bitsy M0 Express

Nick Marchuk
9/6/2018

A digital version of this document can be found at
http://hades.mech.northwestern.edu/images/0/01/CircuitsAndProgrammingWithCircuitPythonM0-2018.pdf

Tutorial Parts List:
Breadboard
6 colors of 24 gauge solid core wire
Wire strippers
Itsy Bitsy M0 Express with pins already soldered on
Micro USB cable
(USB C to USB2 adapter)
Neopixel breakout board
10k potentiometer
330Ω, 10kΩ resistors
Red, green LEDs
Micro RC Servo with header pins stuck in the plug
Accelerometer
Speaker

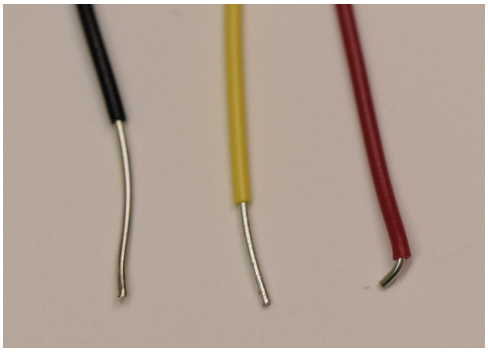
Software for the Itsy Bitsy M0:
Install Mu. With your board plugged into your computer, open Mu, and select “Adafruit CircuitPython”.



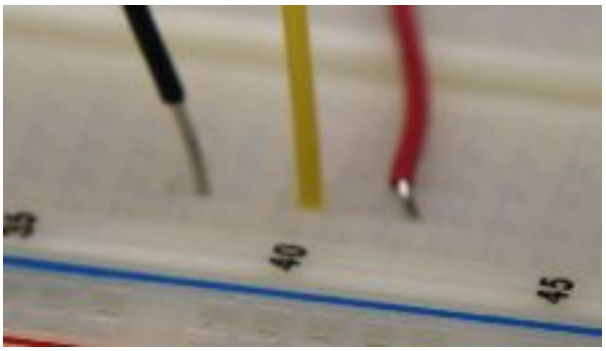
Immediately copy the main.py file from the file drive that appears when you plug the USB cable in, save it somewhere you’ll remember it, just in case you need to refer back to it later.

Electronics: Cut and strip a wire

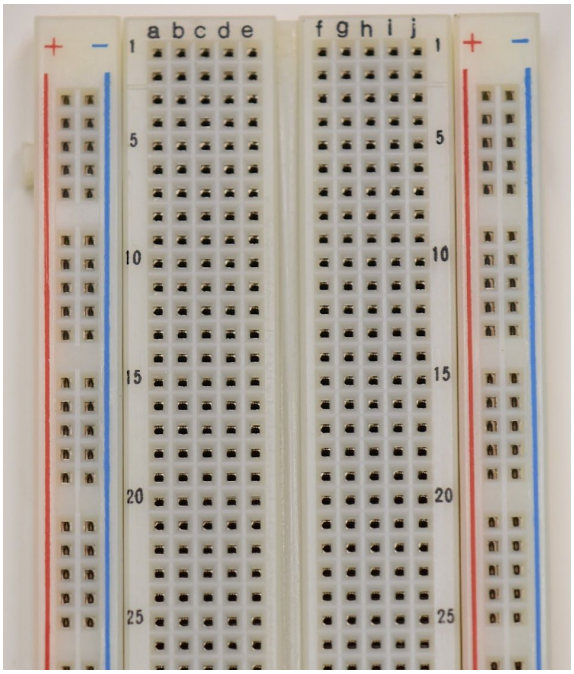
Practice using your wire strippers to cut a few 1” long wires. Remove about ¼” of insulation from both ends of the wire, using the 26 AWG stripper hole. ¼” of exposed wire is about the right length to be plugged into the breadboard without leaving any exposed wire to accidentally touch other exposed wires (the breadboard is about ¼” deep).

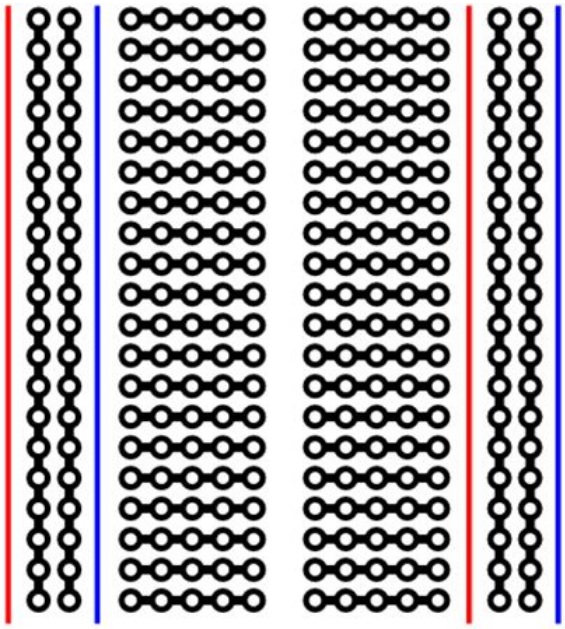


Too much wire exposed (left), too little (right), goldilocks (center)



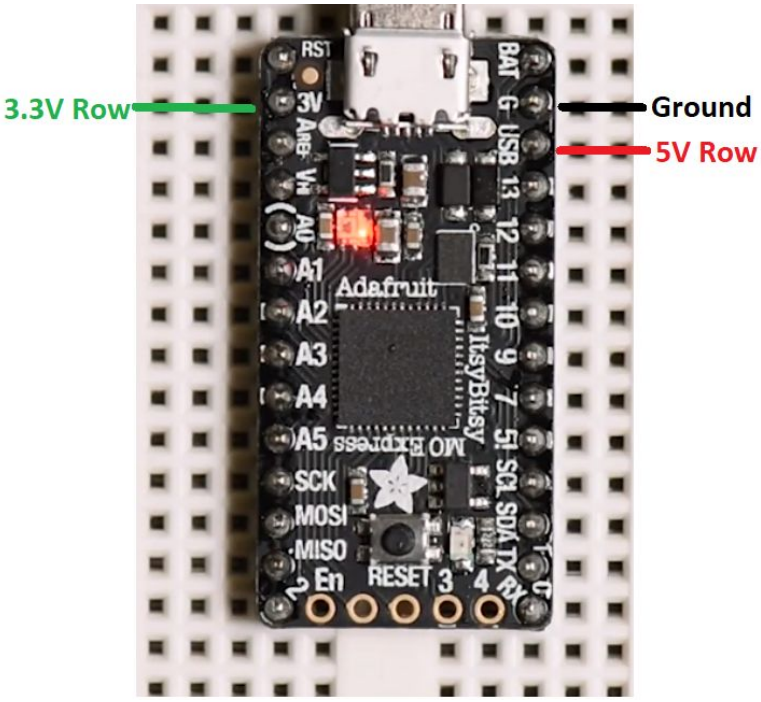
Electronics: How a breadboard works





A breadboard is a collection of conducting clips that make it easy to connect wires to components like resistors, LEDs, and Itsy Bitsy Express pins. Along the outer edges of the breadboard are long columns of clips called rails. In the middle section of the breadboard are rows of 5 connected holes. The left and right side of the center channel are not connected.

When a component like the Itsy Bitsy Express is plugged into the breadboard, the rows become connected to the microcontroller pins and give several access points to each pin.

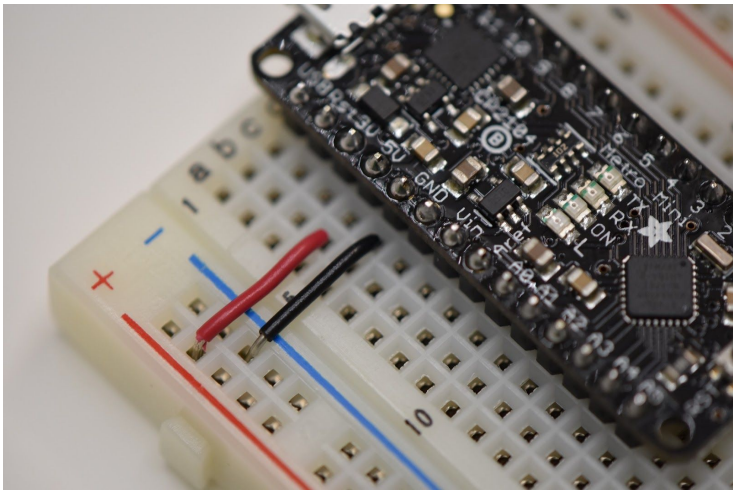


Circuits usually need many access points to power and ground, so wire is used to connect the rails to the power rows to gain more power and ground holes. Make some wires to connect the left **+** rail to **3.3V**, the right **+** rail to **5V,** and the right **-** rail to **GND** (0V). The color convention is to use **red wire** for 5V, **green wire** for 3.3V, and **black wire** for Ground, so that you can quickly look at a circuit and debug it later.

Note that the Itsy Bitsy Express is supplied with 5V from the computer using the USB cable, and now the breadboard has access to the computer's 5V line. The computer would be very unhappy if you shorted it's 5V (no damage, but the USB port would be disabled and only reenabled by a reboot). Applying a voltage to the 5V connection, say from a 6V pack of AA batteries, would damage the computer's motherboard, so extra care should always be taken when using power from both the computer and an external source.

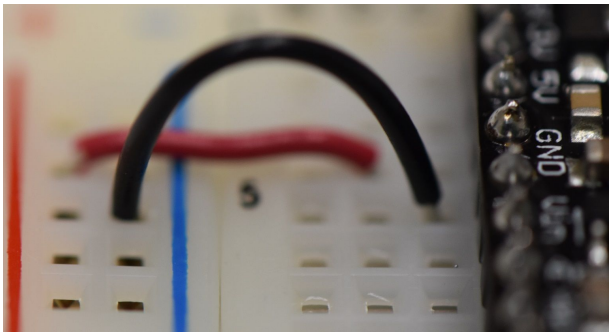
Also note that the Itsy Bitsy itself runs on 3.3V, not 5V, so be careful to only apply a maximum of 3.3V to any pin.

Bonus points for wires that are flush to the board.



Flush = good

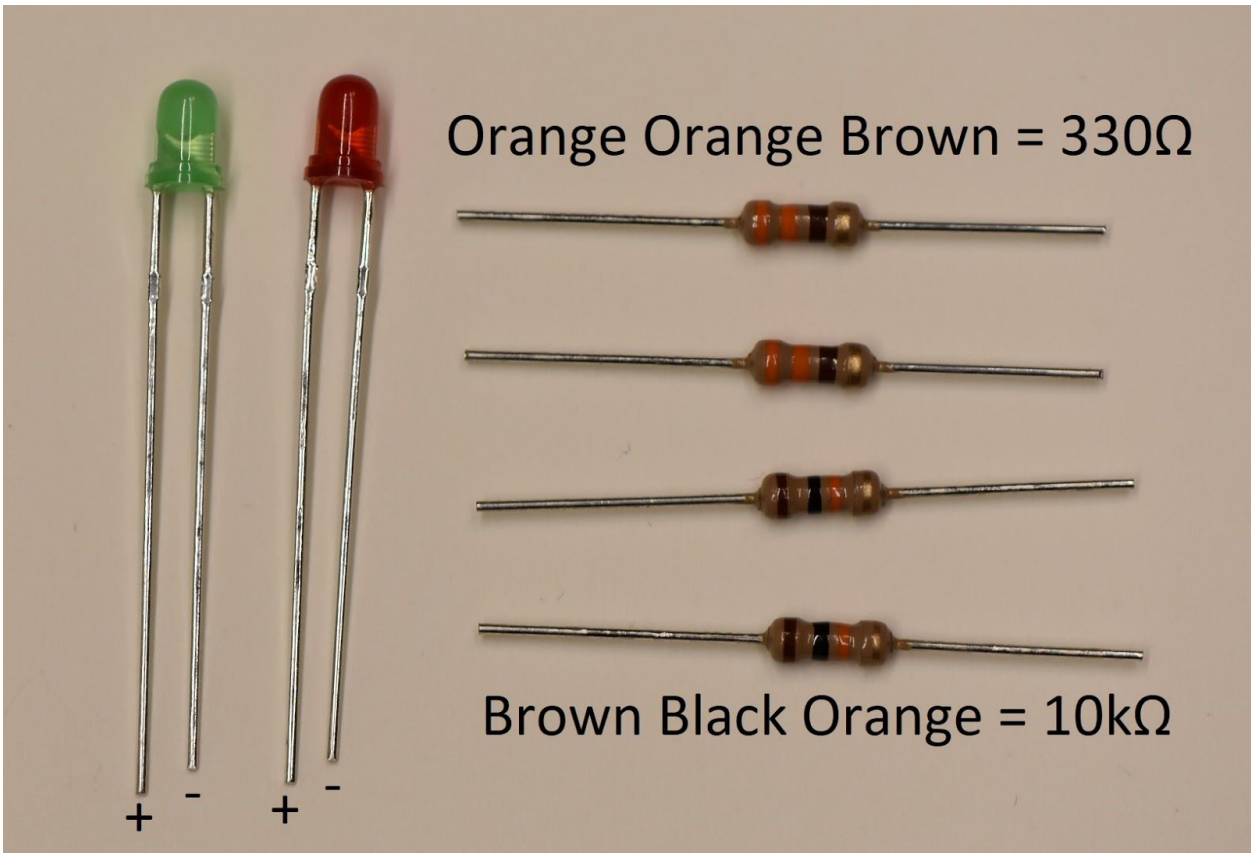
vs



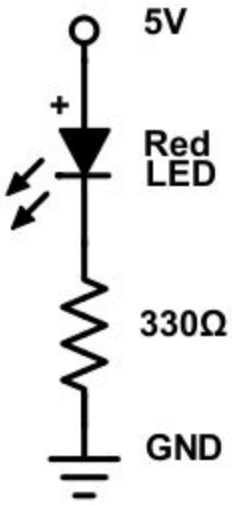
Loopy = ok

Extra bonus points for connecting the left and right **-** rails along the bottom of the board, so that you have access to Ground on both sides!

Electronics: Power an LED



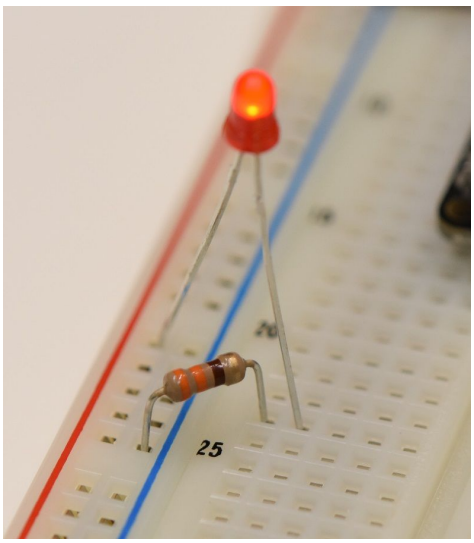
An LED will light when current flows through it, from the positive leg (longer wire) to the negative leg (shorter wire). A resistor in series with the LED is necessary to reduce the current through the LED and prevent it from overheating.

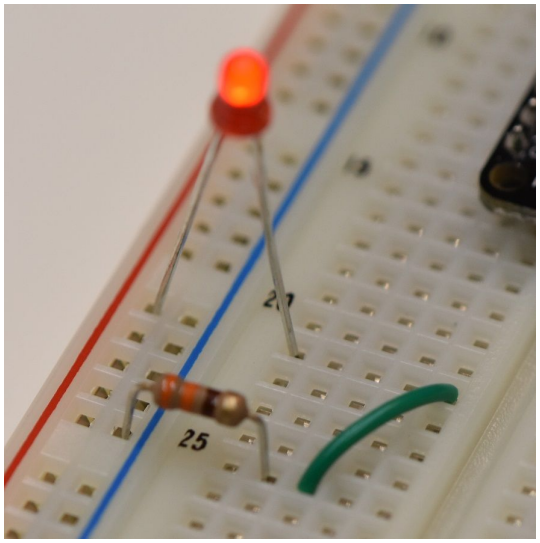


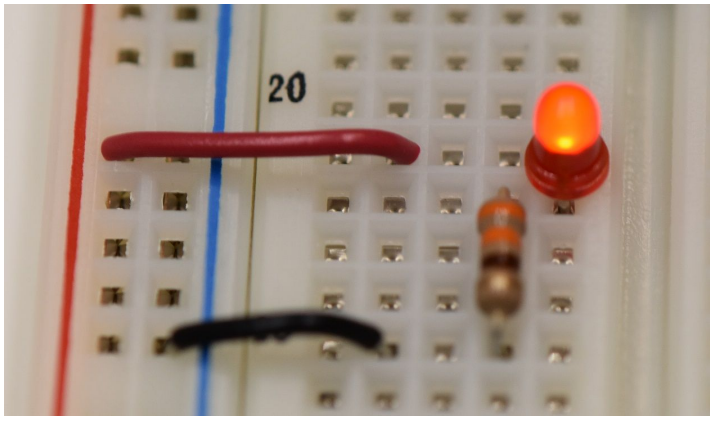
It is important to draw a circuit schematic before building the circuit. The schematic is the the reference for what you were supposed to build, and your guide when debugging.

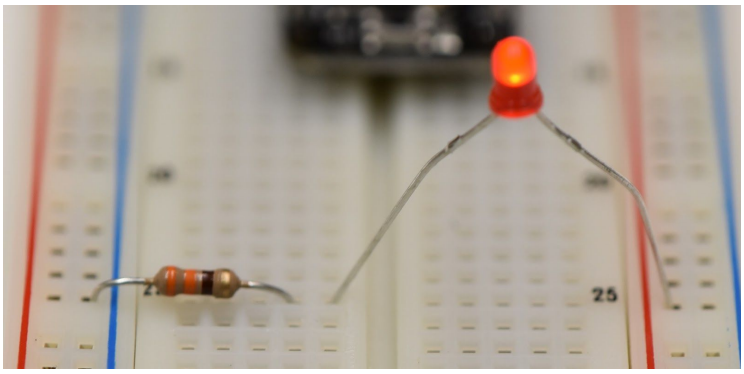
Build this circuit somewhere on your breadboard. What happens when you replace the 330Ω resistor with the 10kΩ?

The schematic does not tell you how to make the connections on the breadboard; there are infinite ways to do that. Use whatever construction you are comfortable with! You can also trim the legs of the LED and resistor to make them less likely to fall out, but try to leave the positive leg of the LED a little longer so you remember which way is positive.

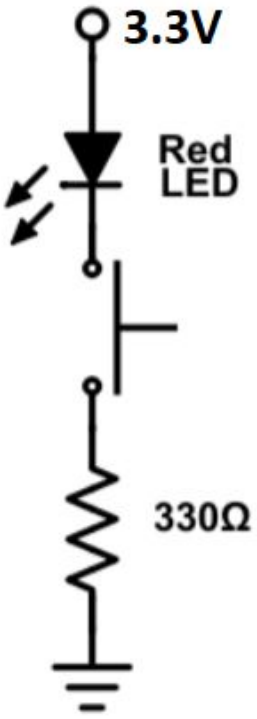








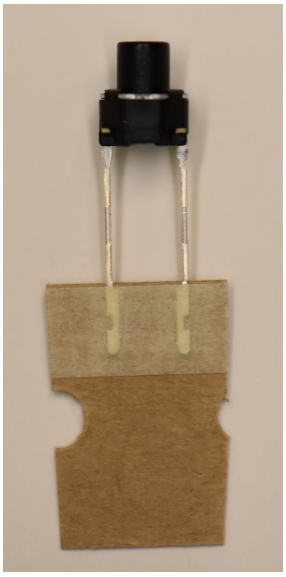
Electronics: Turn on an LED with a button

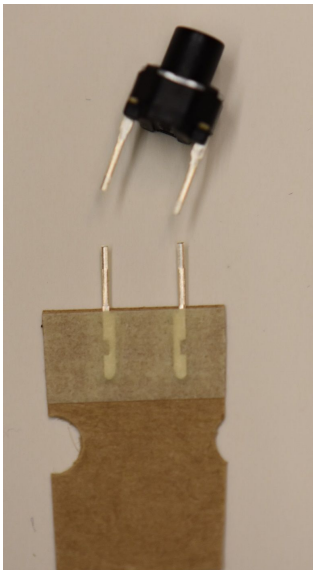


Edit your circuit so that the LED is on only when a push button is pushed.

Do the order of the components matter?

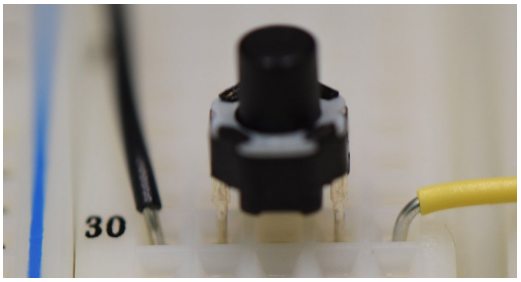
How could you redesign the circuit so that the LED is on only when the button is not pushed?

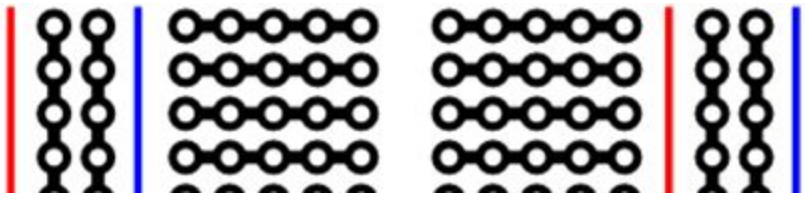




Note that the pushbutton has funny feet that do not fit in the breadboard. Cut them off!

Why does the following circuit not function?





Circuit Python programming: Using REPL

We prefer not to build circuits like the one above because they are inflexible. To get different functionality, the circuit must be redesigned. A more flexible approach is to build simple circuits, linked through code and a microcontroller. Changing code is easy and free! This will vastly improve your prototyping ability.

Open Mu (“Moo”), set the Mode to Adafruit CircuitPython, and open main.py on the drive that appeared when you plugged in your Itsy Bitsy Express. This file has lots of examples in it, make a copy and save it somewhere on your computer, it will be helpful later.

What makes the Itsy Bitsy Express tick? It is running a Python interpreter, capable of running your main.py file, as well as controlling the physical input and output pins of the microcontroller. You can communicate with the board over Serial communication by clicking the Serial button. When you do, text will appear at the bottom of the screen, because the default code is already running.

On your keyboard, press Ctrl-C, and then any key, and you will enter REPL mode (Read-Eval-Print-Loop). Just like when running Python on your computer, you can interact with your code through REPL. When you want to go back to running main.py, press Ctrl-D.

An interesting thing you can do in REPL is find out which module libraries are installed. Type help(“modules”) to see what libraries are available for import. Type import math, then dir(math), to see all the available functions in the math library. Sometimes you can help the functions, like help(math.pi), but usually this does not yield any helpful information, but you can search online once you know the name of the function.

Try investigating some of the libraries, then exit REPL.

Circuit Python programming: Blink an LED

The default main.py contains lots of examples. Let's pare it down and start with the basics.

We start with importing the libraries necessary to make the board work:

- The pins are defined in the board module, so you need to import board.
- Time functions, like sleep, are in the time module, so you need to import time.
- Pin functions, like stating direction and getting value, are in the digitalio module, and we specifically want to set the pin that is being used and in which direction, so we need to import DigitalInOut and Direction from digitalio.

To blink the red LED that is already on the Itsy Bitsy Express pin D13, we need to make a variable to represent the pin:

- led = DigitalInOut(board.D13)

Then we need to set the pin to Output:

- led.direction = Direction.OUTPUT

Now we create in infinite loop, turn the LED on, then wait, then turn the LED off, then wait, and continue:

While True:
    Led.value = True
    time.sleep(1) # in seconds
    Led.value = False
    time.sleep(1)

Overall, it looks like:

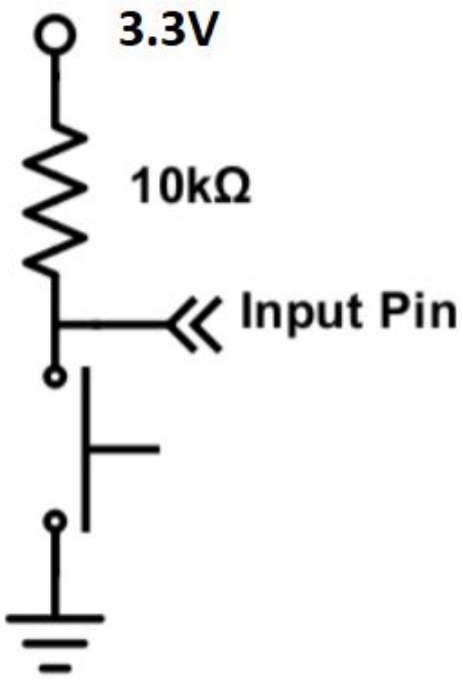
```
import board # pin definitions
import time # sleep function
from digitalio import DigitalInOut, Direction

# Built in red LED on D13
led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT

########################## MAIN LOOP ##############################
while True:
    led.value = True
    time.sleep(1) # in seconds
    led.value = False
    time.sleep(1)
```

Save the code, and make sure the LED blinking rate corresponds to the code. Change the delay time and make sure the board updates.

Circuit Python programming: Read a button and turn on an LED



The following push button circuit is the typical button built for a microcontroller. The input pin is by default True, and becomes False when pushed. The Itsy Bitsy Express contains the 10k resistor inside of the pin, and it can be turned on in code, by also importing the Pull function from digitalio. Edit your code so that you have a variable called button, on pin D9, that is an input pin, and the pullup resistor is enabled using button.pull = Pull.UP.

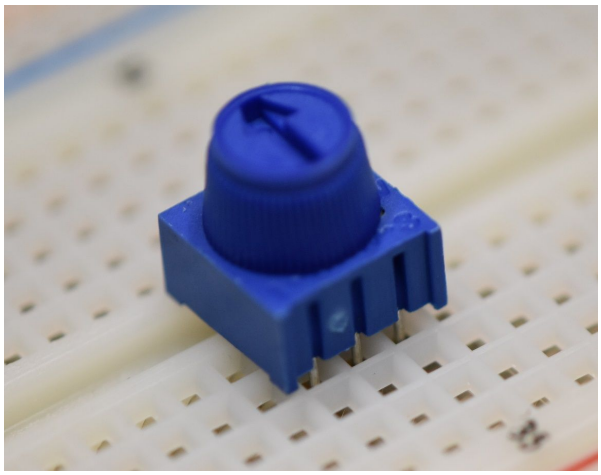
In the infinite loop, add an if statement that determines if D9 is tapped to Ground using a wire, and if it is, turn on the LED, otherwise turn off the LED. Remove any delays so the code loops as often as possible.

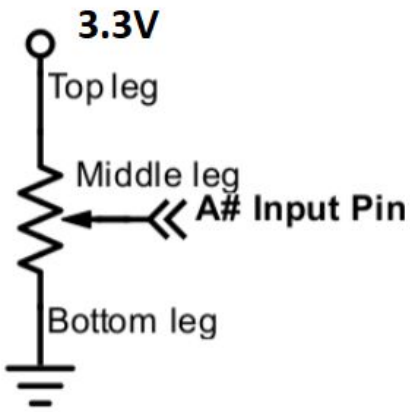
How fast is the code running? One way to try to find out is to make a variable, add one to it every time the loop iterates and the button is pushed, and print the value if the button is pushed. Then, when you tap the button, you will see how fast the variable increments, and have a better idea for how fast the loops runs.

Declare a variable called i and set it to zero before the infinite loop starts. In your if statement, set i = i + 1 if the button is pushed, and print the value using print(“i = “ + str(i)). Notice that you could use print(i), but not print(“i = “ + i). Why is that?

Arduino programming: Read a potentiometer and print the voltage to the computer

A potentiometer is a variable resistor knob with three legs. When the bottom and top legs are attached to Ground and 3.3V, the middle leg will output a voltage from 0V to 3.3V that is proportional to the angle of the knob. Potentiometers are usually limited to 180 degrees of rotation.





The Itsy Bitsy Express can read voltages from 0V to 3.3V on pins that begin with A (A1 to A5). Build the potentiometer circuit and connect the output voltage to pin A1.

To read an analog value, import AnalogIn from the module analogio. Create a variable for the pin, and set it to pin A1. This should look like:

```
from analogio import AnalogIn
analog1in = AnalogIn(board.A1)
```

Edit the infinite loop so that if the button is pushed, the value of the analog pin, analog1in.value, is printed to the computer (without any extra text). What is the range of numbers as you turn the knob? Can you add math so that the voltage is printed instead?

The Itsy Bitsy Express is so fast that the numbers are choking the computer. Add a 0.01s delay to the inifinte loop to slow things down.

In addition to the Serial monitor, Mu has a plotting tool. To see the number plotted, the print statement can't just be analog1in.value, it must be (analog1in.value,). If you want to plot several numbers, you can add them to the list, and end it with a comma.

Circuit Python programming: Read a potentiometer and set the brightness of an LED

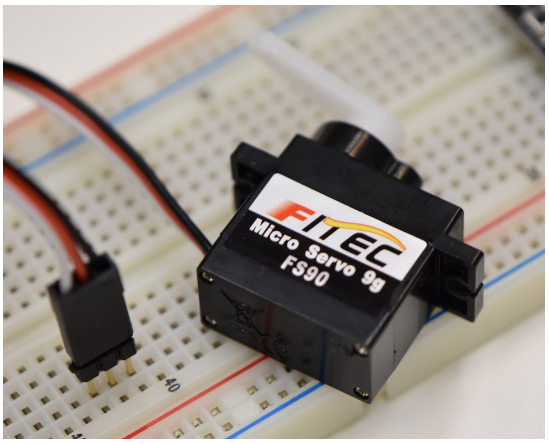
Most microcontroller boards lack the ability to output analog voltages (voltages that are between 0V and 3.3V. Actually the Itsy Bitsy Express can do this on pin A0, but it is meant for low power music signals). Instead they actually output a square wave at a high frequency, blinking the pin so fast that it appears to be analog (this is typically called pulse width modulation, or PWM). You can use PWM on the pins that have a white dash next to them on the board (5, 7, 9, 10, 11, 12, 13).

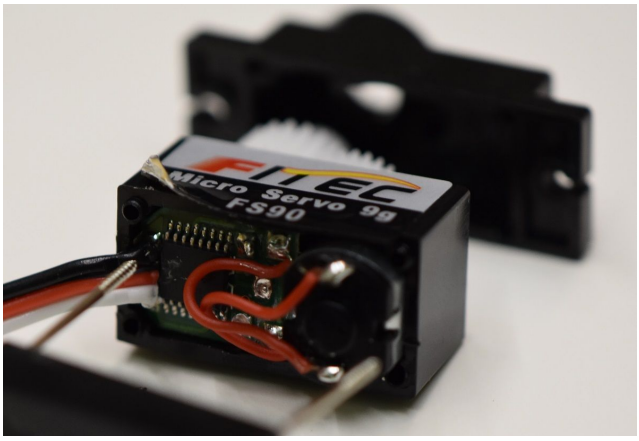
The module is called pulseio, and we'll reuse the red LED on pin 13 to try it out. Comment out the LED part of your code, and replace the pin initialization with led = pulseio.PWMOut(board.D13, frequency=5000, duty_cycle=0), and the value with something like led.duty_cycle = analog1in or led.duty_cycle = 0. The duty_cycle is the percentage that the pin is on, where 65535 is 100% on.

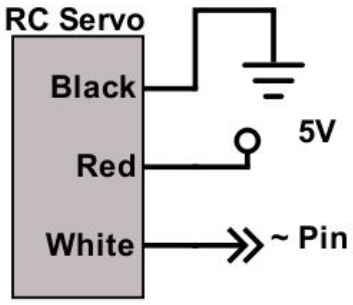
Edit the code and turn the knob while pressing the button, does the red LED change intensity with the knob angle?

Circuit Python programming: Control the position of an RC servo

A Remote Control style Servomotor is an inexpensive way to get position control of a motor. The RC servo has a built in controller, brushed DC motor, gears, and potentiometer. It reads its own position and forces it to match the commanded position told to it by the Itsy Bitsy Express.







Connect the black wire of the servo to Ground and the red wire to 5V. The orange/white wire goes to a PWM capable pin on the Itsy Bitsy Express. Larger RC servos can draw more current than USB can supply and need an external power supply, usually 4.8-6V.

The servo module comes from the manufacturer of the Itsy Bitsy Express

```
from adafruit_motor import servo
```

It uses PWM at 50Hz, and a special servo type of variable:

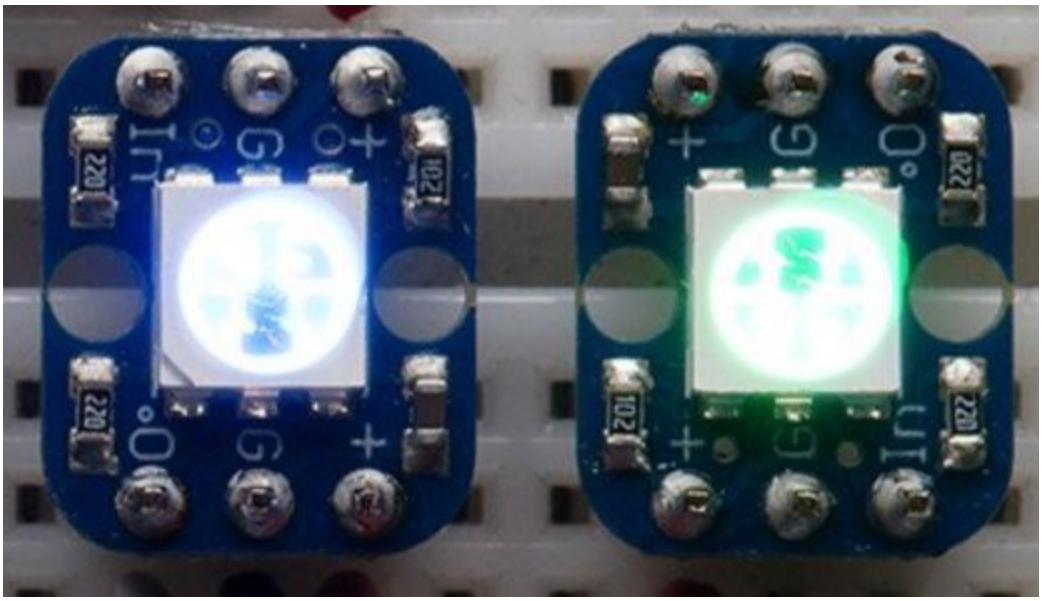
```
servo_pwm = pulseio.PWMOut(board.D12, frequency=50)
servo = servo.Servo(servo_pwm)
```

Then, you can set servo.angle anywhere in you code, from 0 to 180, and the motor will match and hold that angle.

Edit your code so that the servo follows the angle of your knob.

Circuit Python programming: Control the color of addressable RGB LEDs

The WS2812B is an addressable RGB LED that is commonly known as a NeoPixel. Using one pin from the Itsy Bitsy Express, you can control the brightness and color of hundreds of NeoPixels. We only have two, but you can purchase NeoPixels on flexible tape, stiff lines, arcs, panels, or a variety of different configurations, and make cool displays and lighting effects.



Each NeoPixel requires a connection to 5V on +, Ground on G, and a pin to the In pin. NeoPixels can be chained from their O pin to the next In pin, and so on. Connect your NeoPixels together.

Import the neopixel module, and make a variable to control the colors:

```
neopixels = neopixel.NeoPixel(board.D7, 2, brightness=0.2, auto_write=False)
```

In this case, the first NeoPixel is connected to pin 7, and there are 2 NeoPixels on the strip.

To set their colors, use something like:

```
neopixels[0] = [0,int(analog1in.value/256),0] # the first pixel on the strip
neopixels[1] = [0,0,int(analog1in.value/256)] # the second pixel
```

The values are [RED, GREEN, BLUE], from 0-255.

Circuit Python programming: Playing Sound

The Itsy Bitsy Express can play a short audio clip on pin A0. Connect A0 to a 1uF capacitor (labeled 105z), the capacitor to one end of the speaker, and the other end of the speaker to Ground. The sound will not be very loud. You can add a variety of different audio amplifiers to get more sound.

The servo module seems to have a bug that prevents the audio from working, so comment out the servo parts of your code.

Import the audioio module, and add a variable for the .wav files already on the Itsy Bitsy Express:

```
audiofiles = ["rimshot.wav", "laugh.wav"]
```

Copy this function that will play a .wav file:

```
def play_file(filename):
    print("")
    print("----------------------------------")
    print("playing file "+filename)
    f = open(filename, "rb")
    a = audioio.AudioOut(board.A0, f)
    a.play()
    while a.playing:
        pass
    print("finished")
    print("----------------------------------")
```

In your infinite loop, you can call the function and the sound will play (but the code will wait there until the sound is done):

```
play_file(audiofiles[0])
play_file(audiofiles[1])
```

More information:

See the examples on the Adafruit website