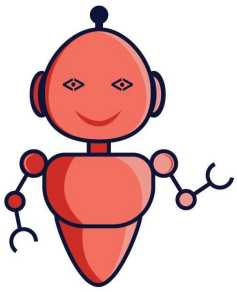
2018



KidzCanCode

IGNITE, INSPIRE, INNOVATE

www.kidzcancode.com

Blinking Multiple LEDs + Resistors + Switch

KIDZCANCODE

TREVOR WARREN

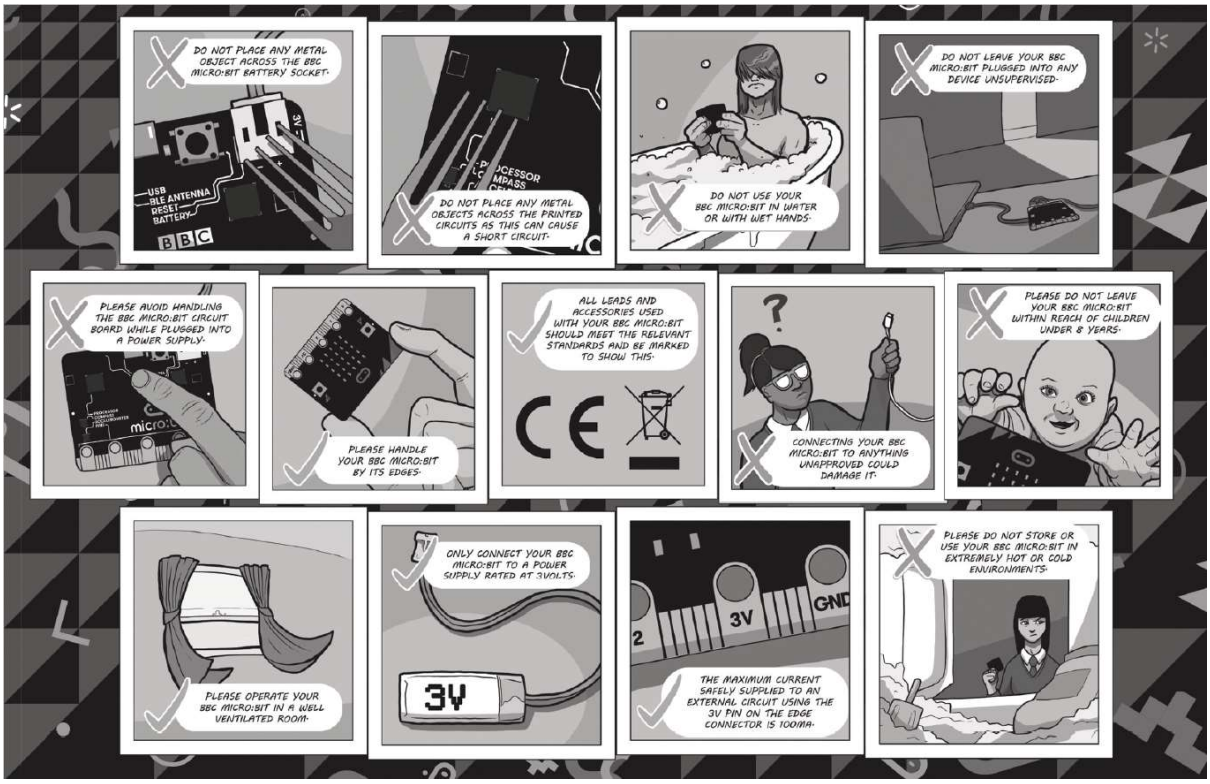
Tutorial 4d4 – Blinking Multiple LED's, Additional Resistors and a Switch

1. Safety Warnings

THE MICRO:BIT IS AN EXPOSED BOARD, TO BE USED WITH CARE PLEASE RETAIN THIS INFORMATION FOR FUTURE REFERENCE. You can read the detailed document at - http://microbit.org/guide/safety-advice/

a. General Safety Warnings

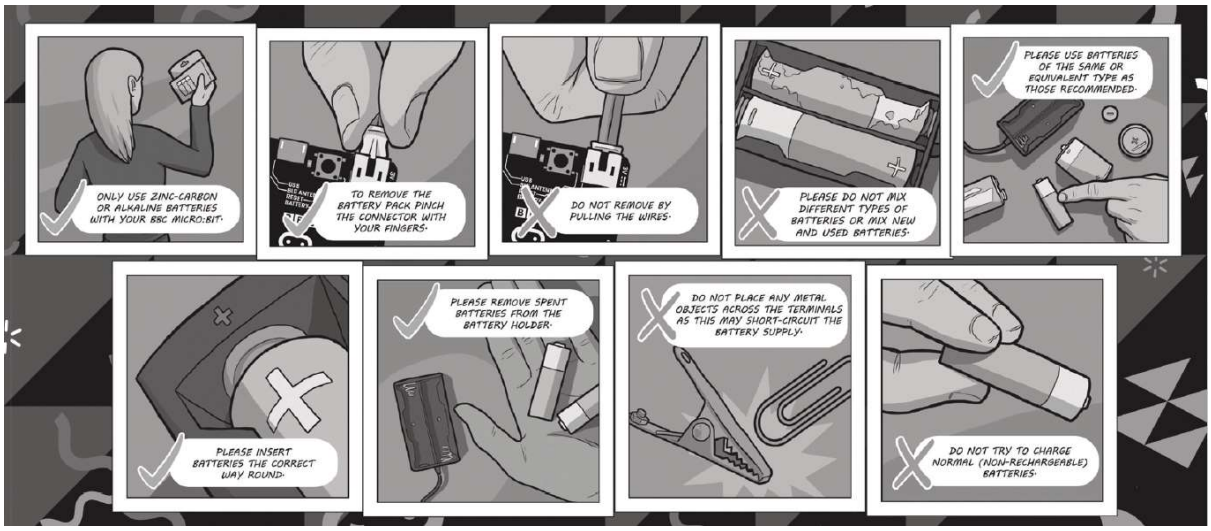
Using the BBC micro:bit is easy to use but is designed to have all the electrical parts on display. This does mean there's a small risk that the parts can be damaged and even overheat with a risk of injury but a little bit of care and caution will ensure you and your micro:bit will stay fit and healthy.



1. Always keep your BBC micro:bit in the anti-static bag when not in use. It's good practice for students to earth themselves before handling it.
2. Please handle your BBC micro:bit by its edges. This minimises the risk of damage through an electrostatic discharge.
3. Please use the battery pack and the USB lead provided to power your micro:bit. Do not use portable battery chargers or USB charging ports (often marked with a lightning bolt or 'SS'), to power your micro:bit. Using these may damage your micro:bit and stop it working properly.
4. Please avoid handling the BBC micro:bit circuit board while plugged into a power supply.
5. All peripherals (for example: USB cable, battery holder, sensors) used with your BBC micro:bit should comply with the relevant standards and should be marked accordingly.
6. Connecting your BBC micro:bit to any unapproved peripherals could damage your BBC micro:bit
7. Please do not attempt to keep using faulty micro:bits. If a school-issued micro:bit develops a fault, contact the vendor immediately.

8. The maximum current safely supplied to an external circuit using the 3V pin on the edge connector is 100mA. Please make sure this limit is not exceeded.
9. Please do not store or use your BBC micro:bit in extremely hot or cold environments.
10. Do not place any metal objects across the printed circuits on the board as this can cause a short circuit damaging your BBC micro:bit. This can cause risk of burn or fire.
11. Do not use your BBC micro:bit in water or with wet hands.
12. Do not leave your BBC micro:bit plugged into a computer or any other device unsupervised.
13. Please do not leave your BBC micro:bit within reach of children under 8 years of age.
14. Please operate your BBC micro:bit in a well ventilated room To remove the battery pack, pinch the connector with your fingers. Do not remove by pulling the wires.

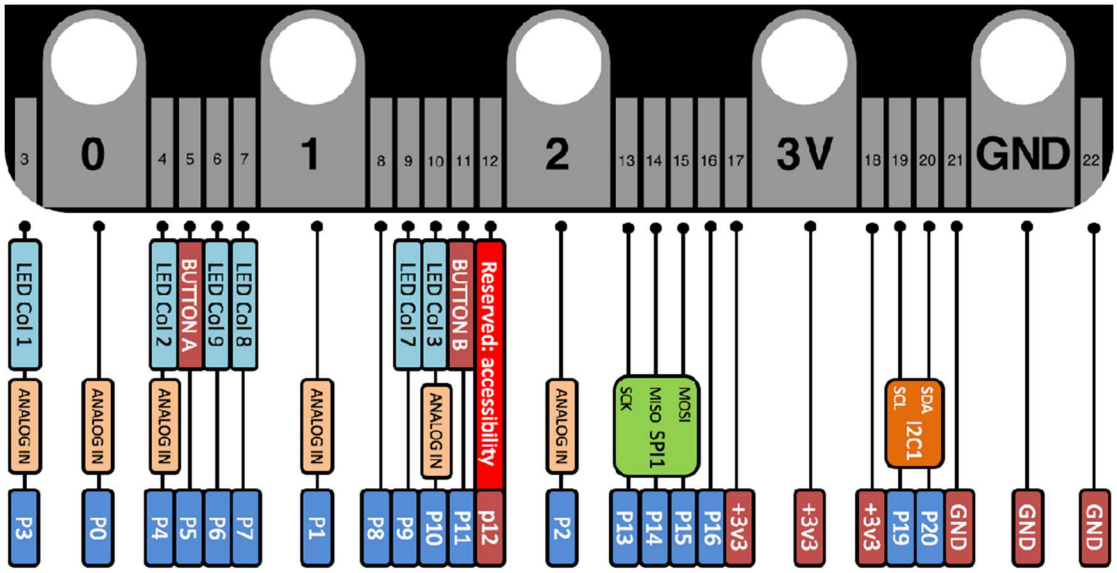
b. Battery Warnings



1. Do not try to charge normal (non-rechargeable) batteries
2. Please do not mix different types of batteries or mix new and used batteries.
3. Please use batteries of the same or equivalent type as those recommended.
4. Please insert batteries the correct way round (with the correct polarity).
5. Please remove spent batteries from the battery holder.
6. Do not short-circuit the battery supply terminals, for example by placing a metal object across the terminals.
7. Only use Zinc or Alkaline batteries with your BBC micro:bit.
8. Please do not use rechargeable batteries

2. BBC micro:bit – A Short Introduction

The BBC micro:bit has 25 external connections on the edge connector of the board, which we refer to as 'pins'. The edge connector is the grey area on the right side of the figure above. There are five large pins that are also connected to holes in the board labelled: 0, 1, 2, 3V, and GND. And along the same edge, there are 20 small pins that you can use when plugging the BBC micro:bit into an edge connector.



Large pins - You can easily attach crocodile clips or 4mm banana plugs to the five large pins.

The first three, labelled 0, 1 and 2 are flexible and can be used for many different things - which means they are often called 'general purpose input and output' (shortened to GPIO). These three pins also have the ability to read analogue voltages using something called an analogue-to-digital converter (ADC). They all have the same function:

- **0**: GPIO (general purpose digital input and output) with analogue to digital convertor (ADC).
- **1**: GPIO with ADC
- **2**: GPIO with ADC

The other two large pins (3V and GND) are very different! The pins labelled 3V and GND relate to the power supply of the board, and they should NEVER be connected together.

If the BBC micro:bit is powered by USB or a battery, then you can use the 3V pin as a *power output* to power peripherals with.

3V: *3 volt power output* or *power input*. (1) *power output*: If the BBC micro:bit is powered by USB or a battery, then you can use the 3V pin as a power output to power peripherals with; (2) *power input*: If the BBC micro:bit is not being powered by USB or battery, you can use the 3V pin as a power input to power the BBC micro:bit

GND: attaches to ground in order to complete a circuit (required when using the 3V pin)

If you hold the 'GND' pin with one hand, you can program the BBC microbit to detect yourself touching the 0,1 or 2 pins with your other hand, giving you three more buttons to experiment with (you just used your body to complete an electrical circuit).

Small pins - There are 20 small pins numbered sequentially from 3-22 (these pins are not labeled on the BBC micro:bit, however, they are labelled in the picture above).

Unlike the three large pins that are dedicated to being used for external connections, some of the small pins are shared with other components on the BBC micro:bit board. For example, pin 3 is shared with some of the LEDs on the screen of the BBC micro:bit, so if you are using the screen to scroll messages, you can't use this pin as well. See below for details of the other pins on the micro:bit.

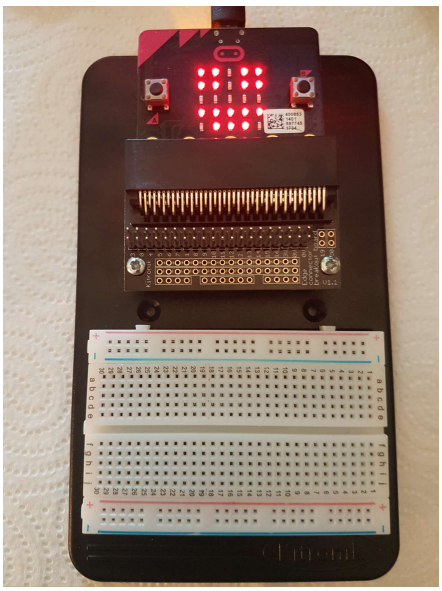
- **pin 3**: GPIO shared with LED Col 1 of the LED screen; can be used for ADC and digital I/O when the LED screen is turned off.
- **pin 4**: GPIO shared with LED Col 2 of the LED screen; can be used for ADC and digital I/O when the LED screen is turned off.
- **pin 5**: GPIO shared with Button A. This lets you trigger or detect a button "A" click externally. This pin has a pull-up resistor, which means that by default it is at voltage of 3V. To replace button A on the BBC micro:bit with an external button, connect one end of the external button to pin 5 and the other end to GND. When the button is pressed, the voltage on pin 5 is pulled down to 0, which generates a button click event.
- **pin 6**: GPIO shared with LED Col 9 of the LED screen; can be used for digital I/O when the LED screen is turned off.
- **pin 7**: GPIO shared with LED Col 8 of the LED screen; can be used for digital I/O when the LED screen is turned off.
- **pin 8**: Dedicated GPIO, for sending and sensing digital signals.
- **pin 9**: GPIO shared with LED Col 7 of the LED screen; can be used for digital I/O when the LED screen is turned off.
- **pin 10**: GPIO shared with LED Col 3 of the LED screen; can be used for ADC and digital I/O when the LED screen is turned off.
- **pin 11**: GPIO shared with Button B. This lets you trigger or detect a button "B" click externally.
- **pin 12**: Dedicated GPIO, for sending and sensing digital signals.
- **pin 13**: GPIO that is conventionally used for the serial clock (SCK) signal of the 3-wire Serial Peripheral Interface (SPI) bus.
- **pin 14**: GPIO that is conventionally used for the Master In Slave Out (MISO) signal of the SPI bus.
- **pin 15**: GPIO that is conventionally used for the Master Out Slave In (MOSI) signal of the SPI bus.
- **pin 16**: Dedicated GPIO (conventionally also used for SPI 'Chip Select' function).
- **pins 17 and 18**: these pins are wired to the 3V supply, like the large '3V' pad.
- **pins 19 and 20**: implement the clock signal (SCL) and data line (SDA) of the I2C bus communication protocol. With I2C, several devices can be connected on the same bus and send/read messages to and from the CPU. Internally, the accelerometer and the compass are connected to i2c.
- **pins 21 and 22**: these pins are wired to the GND pin and serve no other function

3. Pre-requisites

If you need assistance with the assembly of the micro:bit, Edge connector breakout board, mounting board and the breadboard please speak to one of the volunteers on duty.

To be able to perform this tutorial you will need the following components –

1. Parts required –
    a. 1 x BBC Micro:bit
    b. 1 x Mounting Plate
    c. 1 x Edge connector breakout board
    d. 1 x Bread board
    e. 2 x LED
    f. 2 x 2.2K (2200) Ohm Resistor
    g. 2 x 47 Ohm Resistor
    h. 1 x NPN transistor
    i. 1 x push button switch
    j. 3 x Male – Female Dupont wires
    k. 2 x Male – Male Dupont wires
2. Assembly required –
    a. Bread board mounted on top of the mounting plate
    b. BBC Micro:bit inserted into the Edge Connector breakout board



Before proceeding please check your setup and confirm that all the required parts are configured as demonstrated in the above picture.

4. Learning Objectives

The objectives of this tutorial is to extend the basic concepts of electricity, circuits, LED's including using code to turn an LED on/off and listening for user input to perform a given set of actions. This tutorial builds upon concepts introduced in previous tutorials so please make sure you have covered the previous tutorials before you dive into this one. The previous tutorial in this series i.e. 4d1, 4d2, 4d3 used code to turn an LED on/off and this one extends that tutorial a bit further using pushing buttons to turn the LED on/off while also integrating learning from a previous tutorial where we forward bias an NPN transistor.

So overall this tutorial intends to build up concepts learnt in previous tutorials exploring new concepts using code to listen for input (from a switch in this case) and then trigger agreed actions.

Explore the following concepts through making –

1. Connecting elements in a circuit
2. Understanding the flow of current through the elements in the circuit
3. Understanding the concepts of voltage across different components in a circuit
4. Understanding the concept of elements connected in parallel in a circuit
5. Understanding the approach to using code to create switches
6. Understanding the approach to using code to listen for input from a switch and perform an agreed set of actions.
7. Understanding the approach to creating multiple switches and creating lighting patterns using LEDs
8. Understanding the impact of connecting additional resistors in series with existing resistors in a circuit
9. Understand the circuit to forward bias an NPN BJT transistor
10. Understand how NPN transistors behave when they are in a forward biased mode

This experiment differs from the previous one i.e. 4d2, 4d3 in the respect that it now includes an additional transistor setup in a forward bias mode plus the use of a switch which will be programmed to provide user input. Code will be used as usual to listen for user actions and then trigger an agreed set of responses.

5. Exploring components

If you haven't been introduced to the following components it's worth reading through this section –

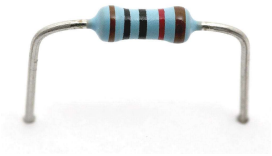
1. Resistors
2. LED's
3. NPN/PNP Bi-Junction Transistors or also called BJT's

Let's briefly look at what resistors and LED's are all about before we dive into actually making the circuit.

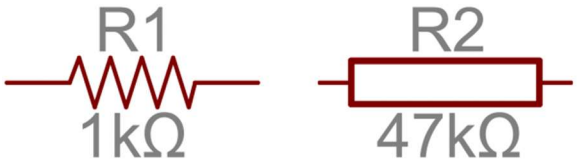
a. What Are Resistors

"What is a resistor?" the student asked.
"It's a component that resists the flow of current" I said.
"Hmm.. I don't get it. What does it do to my circuit?" the student asked.
"Well, it doesn't actually do anything actively, all a resistor does is resists the flow of current through a given circuit" I said. Have a look at what the resistor might look like.

Resistors are electronic components which have a specific, never-changing electrical resistance. The resistor's resistance limits the flow of electrons through a circuit. They are passive components, meaning they only consume power (and can't generate it). Resistors are usually added to circuits where they complement active components like op-amps, microcontrollers, and other integrated circuits. Commonly resistors are used to limit current, divide voltages, and pull-up I/O lines.



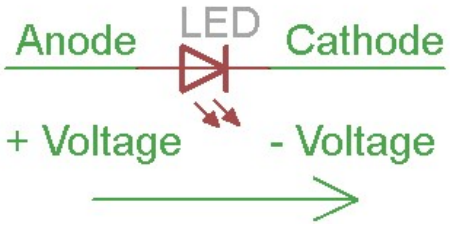
The schematic symbol used for resistors in circuits is –



b. What Are LED's

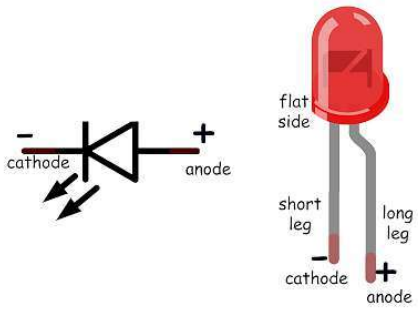
LEDs are so common, they come in dozens of different shapes and sizes. The LEDs you are most likely to use are the through hole LEDs with two legs. There are lots of LEDs that are small and hard to solder but these are easy to use with a breadboard because they have long wires we can stick in. The clear or clearish bulb is what protects the light emitter (that's where the magic happens). In fact, the first two letters of LED stand for Light Emitting.

A really nice thing about LEDs is that they are very simple. Unlike some chips that have dozens of pins with names and special uses, LEDs have only two wires. One wire is the anode (positive) and another is the cathode (negative). The two wires have different names because LEDs only work in one direction and we need to keep track of which pin is which. One goes to the positive voltage and the other goes to the negative voltage. Electronic parts that only work in 'one direction' like this are called Diodes, that's what the last letter of LED stands for.

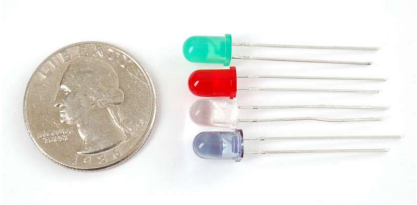


- The longer lead goes to the more-positive voltage
- Current goes in one direction, from the anode (positive) to the cathode (negative)
- LEDs that are 'backwards' won't work - but they won't break either

The schematic representation for LED's in circuits is as follows –



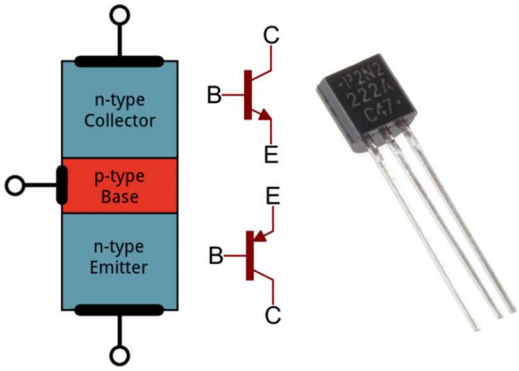
It's all a little confusing - we often have to think about which is which. So to make it easy, there's only one thing you need to remember and that's the LED wont light up if you put it in backwards. If you're ever having LED problems where they are not lighting, just flip it around. Its very hard to damage an LED by putting it in backwards so don't be scared if you do



One of the best things about modern LEDs is all the colors they come in. It used to be that LEDs were only red or maybe yellow and orange, which is why early electronics from the 70s and 80s only had red LEDs. The color emitted from an LED has to do with what type of material they are made of. So red, for example, is made with Gallium Arsenide. Since then, scientists have experimented with many other materials and figured out how to make other colors such as green and blue, as well as violet and white.

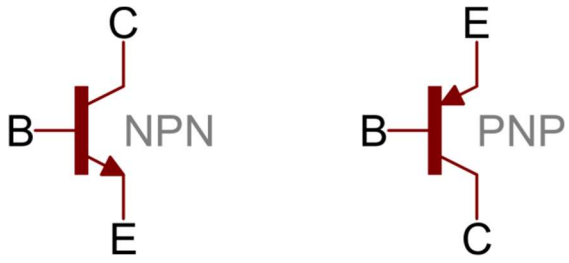
c. What Are Bi-Polar Junction Transistor or BJT's

Transistors make our electronics world go 'round. They're critical as a control source in just about every modern circuit. Sometimes you see them, but more-often-than-not they're hidden deep within the die of an integrated circuit. In this tutorial we'll introduce you to the basics of the most common transistor around: the bi-polar junction transistor (BJT). In small, discrete quantities, transistors can be used to create simple electronic switches, digital logic, and signal amplifying circuits. In quantities of thousands, millions, and even billions, transistors are interconnected and embedded into tiny chips to create computer memories, microprocessors, and other complex ICs.

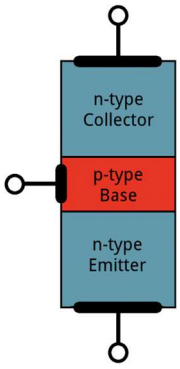


There are two types of basic transistor out there: bi-polar junction (BJT) and metal-oxide field-effect (MOSFET). In our tutorials we'll focus on the BJT, because it's slightly easier to understand and work with. Digging even deeper into transistor types, there are actually two versions of the BJT: NPN and PNP. We'll turn our focus even sharper by limiting our early discussion to the NPN. By narrowing our focus down – getting a solid understanding of the NPN – it'll be easier to understand the PNP (or MOSFETS, even) by comparing how it differs from the NPN.

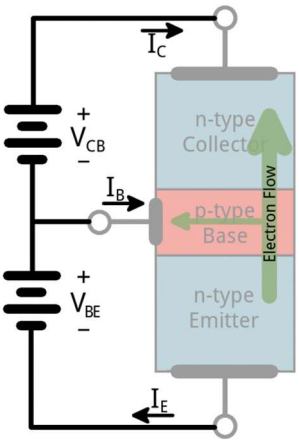
Transistors are fundamentally three-terminal devices. On a bi-polar junction transistor (BJT), those pins are labelled collector (C), base (B), and emitter (E). The circuit symbols for both the NPN and PNP BJT are below:



The only difference between an NPN and PNP is the direction of the arrow on the emitter. The arrow on an NPN points out, and on the PNP it points in. Transistors are built by stacking three different layers of semiconductor material together. Some of those layers have extra electrons added to them (a process called "doping"), and others have electrons removed (doped with "holes" – the absence of electrons). A semiconductor material with extra electrons is called an n-type (n for negative because electrons have a negative charge) and a material with electrons removed is called a p-type (for positive). Transistors are created by either stacking an n on top of a p on top of an n, or p over n over p. See the diagram below -



Simplified diagram of the structure of an NPN. Notice the origin of any acronyms?



With some hand waving, we can say electrons can easily flow from n regions to p regions, as long as they have a little force (voltage) to push them. But flowing from a p region to an n region is really hard (requires a lot of voltage). But the special thing about a transistor – the part that makes our two-diode model obsolete – is the fact that electrons can easily flow from the p-type base to the n-type collector as long as the base-emitter junction is forward biased (meaning the base is at a higher voltage than the emitter).

The NPN transistor is designed to pass electrons from the emitter to the collector (so conventional current flows from collector to emitter). The emitter "emits" electrons into the base, which controls the number of electrons the emitter emits. Most of the electrons emitted are "collected" by the collector, which sends them along to the next part of the circuit. A PNP works in a same but opposite fashion. The base still controls current flow, but that current flows in the opposite direction – from emitter to collector. Instead of electrons, the emitter emits "holes" (a conceptual absence of electrons) which are collected by the collector.

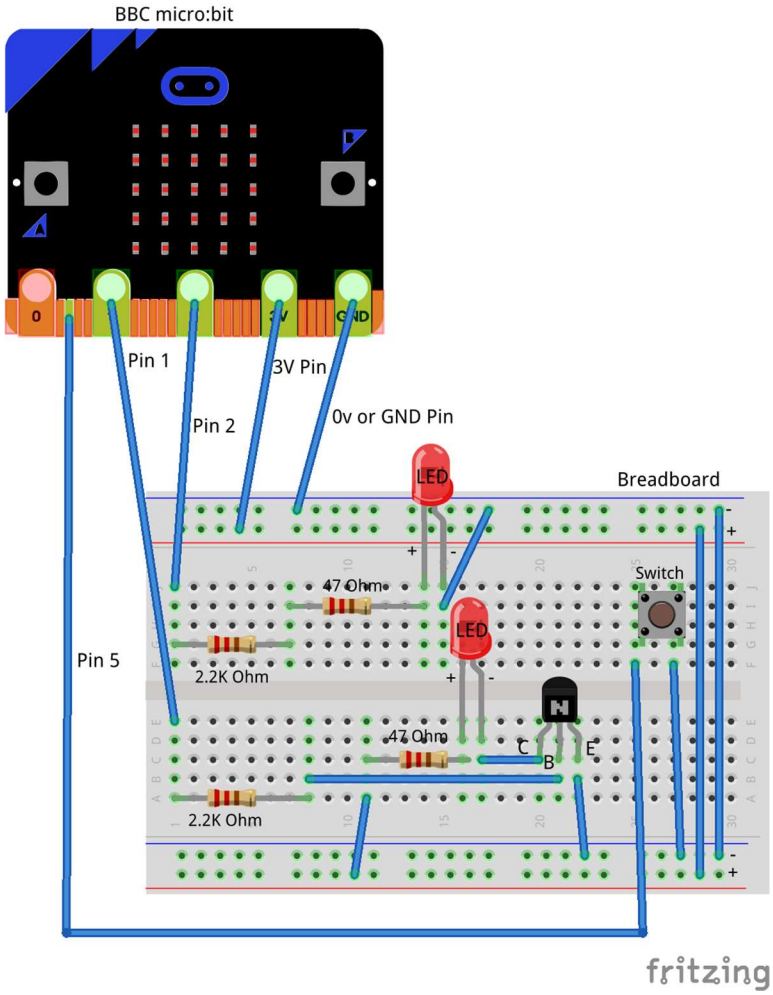
The transistor is kind of like an electron valve. The base pin is like a handle you might adjust to allow more or less electrons to flow from emitter to collector.

6. Activity

Before we get started, let's make sure that all the required components are configured as required.

a. Step 1 - Build

Let's now connect up the relevant components on our micro:bit.



Please note that on the breadboard, between the horizontal blue and red lines all the holes are horizontally connected to each other. In the middle of the board however all the holes are vertically connected to each other. In in doubt please speak to a volunteer/mentor to clarify the approach.

The connections should be made as follows:

1. Connect the Male – Female Dupont wire from the Ground (0v) pin on the micro:bit (using the Edge Connector Breakout board) to the ground rail on the breadboard.
2. Connect the Male – Female Dupont wire from the 3v pin on the micro:bit (using the Edge Connector Breakout board) to the positive rail on the breadboard.
3. Connect a Male – Female Dupont wire from Pin2 pin on the micro:bit (using the Edge Connector Breakout board) to 2.2K Ohm resistor on the breadboard as shown.
4. Connect up the 2.2K Ohm resistor in series to the 47 Ohm resistor as show.
5. Connect the 47 Ohm resistor to the Anode (Positive, Large leg) of the LED. Use a Male to Male dupont cable if required.

6. Connect the Cathode (Negative, Shorter led) of the LED to the Ground (0v) using a Male to Male dupont cable.

Now let's connect up the second pair of LEDs and resistor so as to be able setup the NPN transistor in a forward bias mode.

7. Connect a Male – Female Dupont wire from Pin1 pin on the micro:bit (using the Edge Connector Breakout board) to 2.2K Ohm resistor on the breadboard as shown
8. Connect up the 2.2K Ohm resistor in series with the Base (B) of the NPN transistor
9. Connect up power or 3v to the 47 Ohm resistor and then to the Anode (Positive, Large leg) of the LED. Use a Male to Male dupont cable if required.
10. Connect the Cathode (Negative, Shorter led) of the LED to the Collector (C) of the NPN transistor. The LED in conjunction with the resistor will form the load on that the NPN transistor will drive.
11. Connect up the Emitter (E) of the NPN transistor to the ground (0v) rail of the breadboard.

This circuit you will note is very similar to the circuit we used for our series experiences i.e. 4a1, 4a2 and 4a3. The difference here is that we have introduced an additional active element called an NPN transistor with the existing components and will be using code to turn the LED (connected to the transistor) on and off.

This simple circuit consists of two similar sets of components each comprising of, a 2.2K Ohm, a 47 Ohm resistor and an LED. The 2.2K resistor in series with the LED is required to limit the current flowing through the LED and preventing the LED from burning up. The 2.2K resistor in series with the Base of the NPN transistor is used to limit the current flowing into the base of the transistor.

We will be extending the code written in the earlier tutorial and automate the turning on and off of the two LED's. We will capture user input provided using buttons and perform a series of automated actions that include turning on/off the LED's. You can also play around with the timing and change the rate of blinking of the LEDs using code.

Self-Assessment

- ☐ I have the breadboard mounted onto the mounting plate.
- ☐ I have the micro:bit plugged into the Edge connector breakout board.
- ☐ I understand each of the components in the circuit i.e. LED, resistor, Male – Female Dupont wires, etc.
- ☐ I have each of the elements in the circuit wired up using the breadboard
- ☐ I have checked the connections on the circuit and have not powered it on
- ☐ I understand how to use the male – male and male – female dupont cables to create connections using the breadboard
- ☐ I understand what the positive / 3v and ground / 0v rails (top and bottom) of the breadboard do and how they are connected
- ☐ I understand how the central rows in the breadboard are lined up together and how they are connected.
- ☐ I understand the difference between 3 Volt (3V) and Ground (0 Volt of 0V)
- ☐ I understand the impact of the addition of the 2.2K Ohm resistor with regards to the current flowing through the base of the transistor and the impact of choosing a resistor too low in value.

This now completes your circuit.

PLEASE DO NOT POWER ON YOUR micro:bit UNLESS YOU HAVE HAD YOUR CIRCUIT VALIDATED BY A FACILITATOR/MENTOR/Volunteer.

b. Step 2 - Explore

Discuss the following concepts with the learning facilitator

1. Which components in this circuit are in series?
2. Are there any components in this circuit which are connected in parallel?
3. What are the paths for current flowing through this parallel circuit?
4. Which are passive and active components in a parallel circuit?
5. What does it mean to setup the transistor in a forward bias mode?
6. What does it mean to capture user input provided through click of a button?

Self-Assessment

- ☐ I can explain what a circuit means, what are components connected in series and components connected in parallel.
- ☐ I can explain the differences between active and passive components in circuits.
- ☐ I understand how components can be connected in series and parallel.
- ☐ I understand the use of the 2.2K resistor in the circuit to limit the current flowing into the base of the transistor
- ☐ I understand the impact of having a resistor too large in value or too small in value in series with the Base of the transistor
- ☐ I understand the behaviour of an NPN transistor when it's forward baised

c. Step 3 – Validate

Please call a facilitator to check your circuit before you power it on. It is important that you get a learning facilitator/mentor/volunteer to check the connections on your circuit before powering it on.

Incorrect connections might result in a dead (bricked, burnt, short-circuited, etc.) Micro:bit requiring you to purchase another Micro:bit for $30 AUD.

d. Step 4 – Power On

Awesome, you are almost at the point where you can power on your circuit and watch the magic un-fold. Please call a facilitator to check your circuit before you power it on. It is important that you get a learning facilitator to check the connections on your circuit before powering it on.

Once the learning facilitator has verified your circuit you should now proceed and power on the Micro:bit. If you have not had your learning facilitator verify the circuit PLEASE GO BACK To step 3 and have one of the learning facilitators verify your connections.

Powering on the Micro:bit requires that you connect the USB power cord for your Micro:bit (plugged into the Micro:bit board) into your laptop.

Incorrect connections might result in a dead (bricked, burnt, short-circuited, etc.) Micro:bit requiring you to purchase another Micro:bit for $30 AUD.

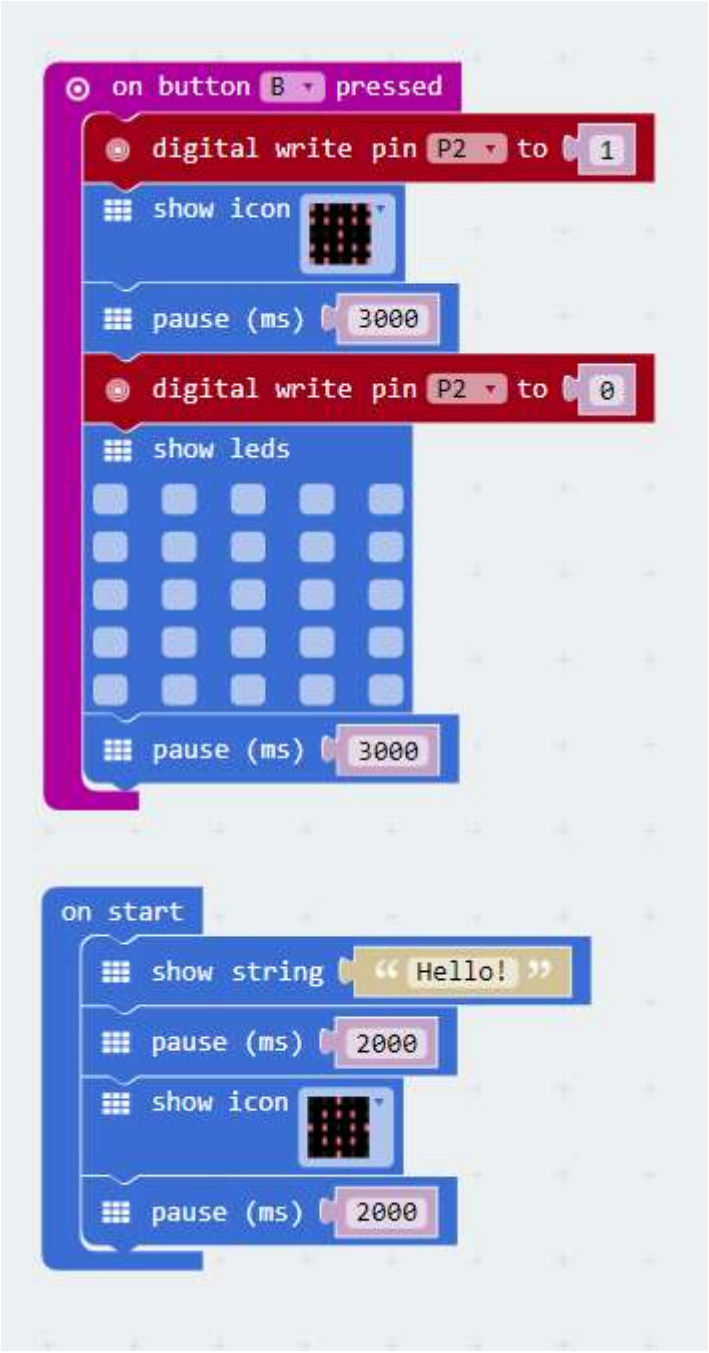
Self-Assessment

- ☐ I am able to wire the series circuit with all the elements shown
- ☐ I understand the flow of current through a series circuit
- ☐ I understand the difference between 3 Volt (3V) and Ground (0 Volt of 0V)
- ☐ I have validated that the circuit is now wired correctly and safe to be powered on

7. Let's write some code

Now that we have put together the circuit it's time to write some code and get the circuit working. So let's head over to the micro:bit block code editor page (https://makecode.microbit.org/) and get coding!!!

In the following section we will dive into the code you will put together for this tutorial –



The code used in this tutorial is similar to the code we've written for our previous tutorials where we wrote code to automate the turning on and off of two LEDs. You are encouraged to play around with the timings (pause command) and change the rate of blinking of the LEDs using code or even change the pins used to power on the LED and see if that makes any difference.

Tutorial 4d4 – Blinking Multiple LED's, Additional Resistors and a Switch

As mentioned before the aim of this tutorial is to create a software based switch i.e. writing commands in software and asking the micro:bit to turn the LED on and off. We will extend the code written in the previous tutorial and create two sets of software switches with a difference i.e. listen for user input provided by a switch and then turn the LED's on/off. Also by varying the "pause" time i.e. the amount of time we request the micro:bit to go to sleep we are able to create different types of lighting patterns.

The software based switch involves the use of the following command, "digitial write pin2" and "digitial write pin1". In the digital world computers only understand 0's and 1's. So if we were to turn on a pin i.e. to send a signal and switch on a particular pin on the micro:bit you have to send it a 1 and if we intended to turn off a pin i.e. send it a signal and switch off a particular pin on the micro:bit you would send it a 0.

In the first set of code that we write we'll only focus on listening to user input provided on Button B. So let's dive into the code and get started.

On start code block -

1. We display a welcome message to the user.
2. We pause for 2s. This essentially tells the micro:bit to go to sleep for 2s.
3. We then display a pattern using LED's. You can change this to whatever you like.
4. Finally we have another set of pause commands so that people can appreciate the LED's being lit up.
    - Not having the pause command will mean that the micro:bit (our little computer) still lights up the LED's but does it so quickly that we just don't get an opportunity to appreciate the work it's done.
    - Similar to the work we've done in scratch we will inject pause statements (only where required) asking the computer to pause so that the human beings interacting with it can appreciate the transition from one mode of operation to another.
5. We finally pause a bit before continuing with rest of the code.

On Button B Pressed block –

1. The On Button B pressed code block on the bbc micro:bit is designed to run code only when the user ever presses the Button B on the micro:bit.
2. The first line now includes the "digital write pin2" command we discussed a bit earlier to turn on the LED by sending it a 1
3. We then light up the LED's.
4. The third line then pauses the code for 3 seconds. Just like in scratch the pause command is really important causes it gives the human beings time to see, understand and perceive what is happening. The computer doesn't need any time as such and without the pause/sleep command it would have happily continued going about its business. The pause commands makes the transition from on to the off state visible to the individual working on the micro:bit.
5. The fourth line then uses the "digital write pin2" command again to turn the pin off by sending it a 0.
6. The fifth line clears up the LED's and displays nothing on the screen giving it the illusion of the screen turning off creating a flashing on/off effect.

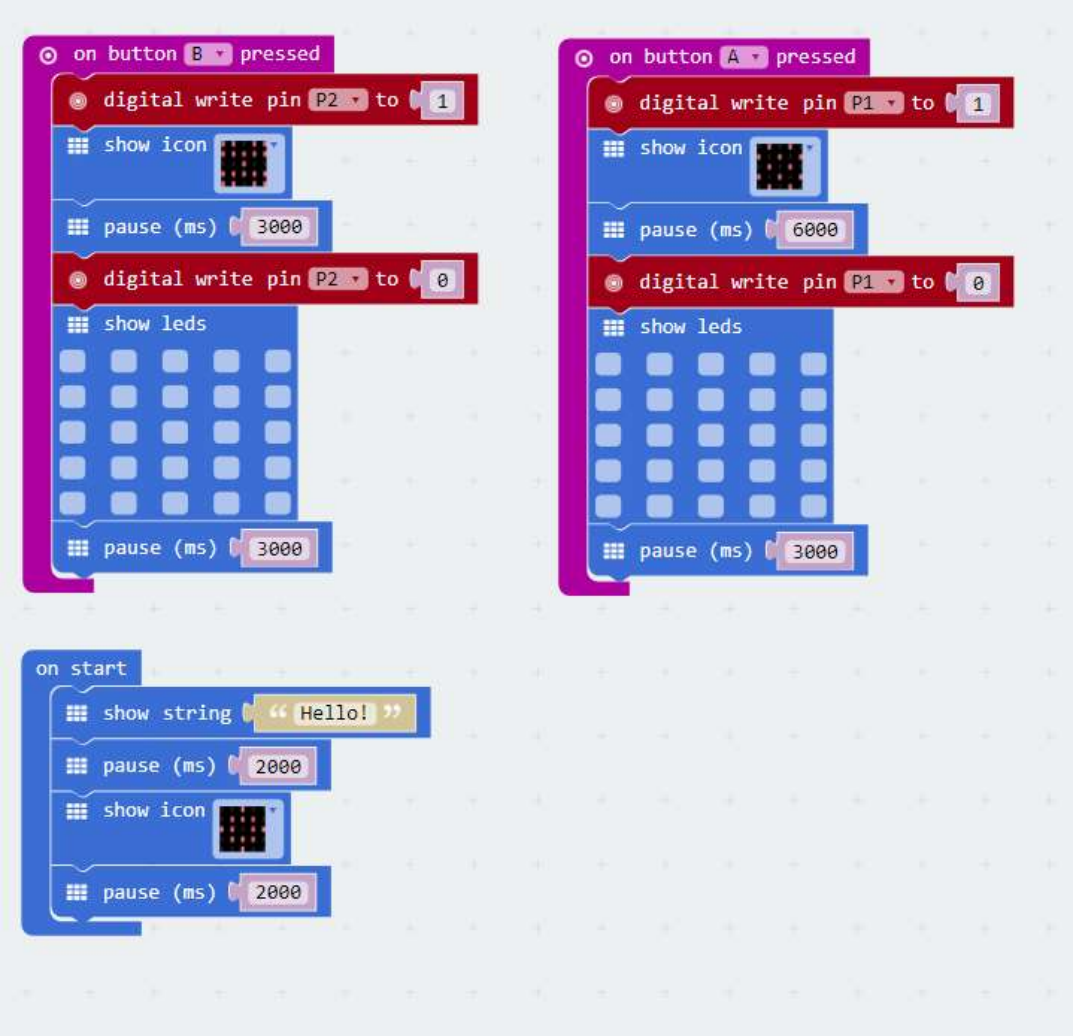
7. The sixth line of the simply pauses the entire code block for 3 seconds. Just like in scratch the pause command is really important causes it gives the human beings time to see, understand and perceive what is happening. The computer doesn't need any time as such and without the pause/sleep command it would have happily continued going about it's business. The pause commands makes the transition from on to the off state visible to the individual working on the micro:bit.

The previous code blocks gave us the capability to look for pressing of button B and then performing a given set of actions. In the coding blocks below we introduce the use of code to monitor for button A being pressed and performing similar set of actions. The different here now is that button A provides a signed on Pin 1 which drives the forward bias of the NPN transistor. When the transistor is in forward bias mode current flowing from the Collector through to the Emitter should light up LED2 connected to the transistor.

On Button A Pressed block –

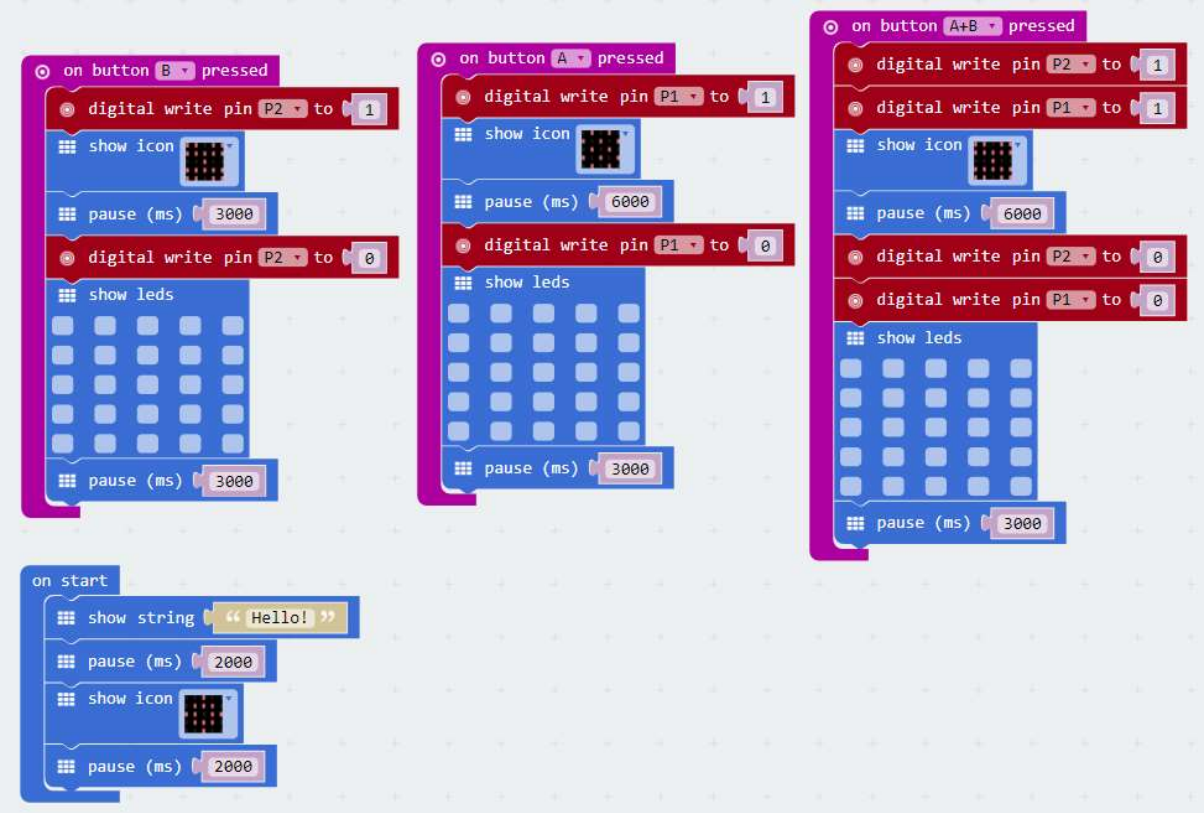
1. The On Button B pressed code block on the bbc micro:bit is designed to run code only when the user ever presses the Button B on the micro:bit.
2. The first line now includes the "digital write pin1" command we discussed a bit earlier to turn on the LED by sending it a 1
3. We then light up the LED's.
4. The third line then pauses the code for 6 seconds. Just like in scratch the pause command is really important causes it gives the human beings time to see, understand and perceive what is happening.
    - The computer doesn't need any time as such and without the pause/sleep command it would have happily continued going about its business.
    - The pause commands makes the transition from on to the off state visible to the individual working on the micro:bit.
5. The fourth line then uses the "digital write pin1" command again to turn the pin off by sending it a 0.
6. The fifth line clears up the LED's and displays nothing on the screen giving it the illusion of the screen turning off creating a flashing on/off effect.
7. The sixth line of the simply pauses the entire code block for 3 seconds. Just like in scratch the pause command is really important causes it gives the human beings time to see, understand and perceive what is happening.
    - The computer doesn't need any time as such and without the pause/sleep command it would have happily continued going about its business.
    - The pause commands makes the transition from on to the off state visible to the individual working on the micro:bit.

Tutorial 4d4 – Blinking Multiple LED's, Additional Resistors and a Switch



The final addition to the code blocks below look for press of button A + B and then execute a combined set of actions previously used to trigger both the LED's i.e. LED1 and LED 2 connected to the Collector of the NPN transistor.

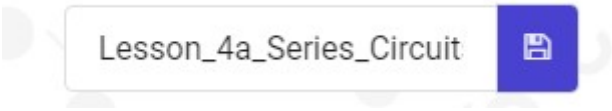
Tutorial 4d4 – Blinking Multiple LED's, Additional Resistors and a Switch



We encourage you to play around with the code, try increasing/decreasing the value of the pause time including changing the LED patterns displayed.

8. Downloading Your Code To The micro:bit

Once you have completed the program, enter a name for your program using the option provided below i.e. in the text box adjacent to the small save button.



Now that you have given your program a name and saved it you can download it your micro:bit. But before we do that let's confirm what drive your micro:bit shows up as. On most machines the micro:bit will show up as an additional USB drive. So head over into windows explorer and confirm what drive name (D:, E:, F:. G:, etc.) the micro:bit shows up as. You need to absolutely be sure what drive the micro:bit shows up as. Once you've confirmed what drive the micro:bit shows up as on your machine you can select the right drive when downloading the code to the micro:bit. If in doubt please ask the volunteer/mentor/session facilitator helping out.

To download the newly written code to the micro:bit, hit the download button shown above. You should now see a dialog box open up and you will be asked to save the file somewhere on your machine. Please choose the drive your micro:bit shows up as i.e. D: or E: or F: or whatever it shows up as on your machine. A sign of success is when you see the lights on the rear (orange) lighting up in quick succession suggesting that the code is being written to the micro:bit. On completion the micro:bit reboots and you should now see the code in action on the micro:bit.

If hitting the download button shown above does not open up a dialog box asking you to save to the micro:bit please save the file (you will have a <filename.hex> file) to your desktop. Then open up windows explorer and drag that file onto the drive which is your micro:bit. A sign of success is when you see the lights on the rear (orange) lighting up in quick succession suggesting that the code is being written to the micro:bit. On completion the micro:bit reboots and you should now see the code in action on the micro:bit.

Please feel free to customize the code blocks, have a play. Add your own custom code and re-download the code to the micro:bit. Give yourself a tap on the back, you've just completed your first circuit!!!!!

9. Challenges

Well done for completing the tutorial. There's a lot of ground we have covered in this tutorial so please feel free to make notes, come back to the tutorial at some point down the line and ask your learning facilitator any questions or doubts you might have on the concepts covered this far.

For those who want to stretch it a bit further -

1. Can you try changing the value of the pause commands to vary the time the LED's are put to sleep?
2. Can you try different combinations of the pause commands to create an on/off lighting effect?
3. What if you clicked the button on the breadboard along with button A? Why does that happen?
4. What would happen if you connected up your button on the breadboard to Pin 11 instead of Pin 5?
5. Can you write the code for the above circuit in python?