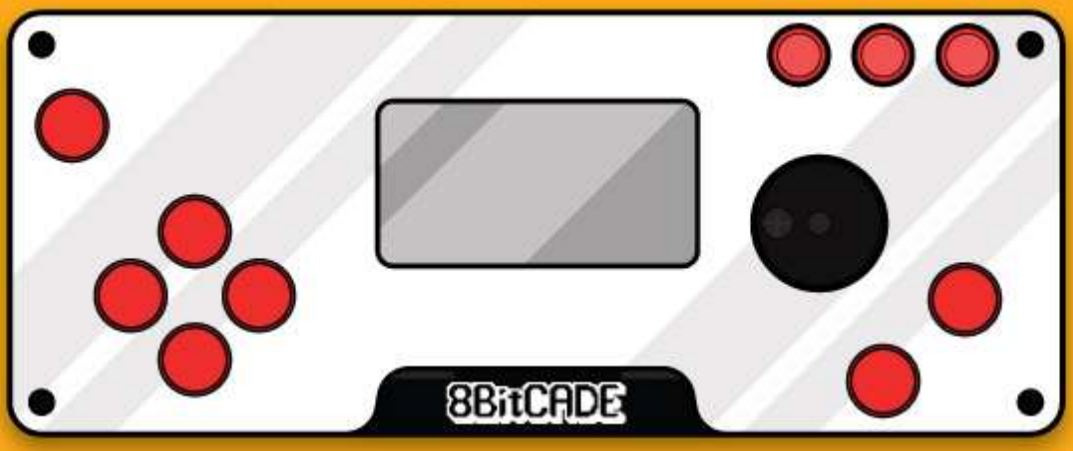
MAKE

8BitCADE

Make Guide

With 8BitCADE



8BitCADE
Make Learn Game

A Welcome From 8BitCADE

"Hello maker! First, give yourself a pat on the back, you just took your first step in learning about electronics, design and programming - well done, and we are pleased you chose us! Thank you! Lets not mess around, lets bring out the maker in you."
From the 8BitCADE Team

The 8BitCADE Original

So you chose the 8BitCADE! Great choice, our first console released! Some call it the 8Bit-CADE Original. Its the perfect balance, how all things should be, of gaming, building and programming. Today we are going to delve into making, lets get started.

What You Will Learn

After doing this tutorial, you'll learn about all the components: what they are; why they are used and understand some basic electronics. You'll also pick up a couple of skills and tips on the way, like soldering!

Estimated Build Time of 1 Hour

How To Use This Guide

Each stage of the making is broken into steps, TAKE YOUR TIME, follow these steps according to the description and photos. **Before** you solder a component in, compare it with the "Parts Required" thumbnail photos to see if its the **right part** and **correctly inserted**. Feel free to print this guide off or download it and use it digitally.

If you need support, email: ***Support@8bitcade.com***

Top Tip: Watch out for the "Top Tip" box that will give you additional guidance regarding a particular step. These are shown in gray boxes like this one and will give you tips to help improve your build, reduce any errors and build time.

Checking Your Parts

Before your start, check your parts with the "8BitCADE Visual Parts Guide" on the website at *8bitcade.com/make* and use the table below, to ensure that you have everything you need to get started!
Missing a part? Email: Support@8bitcade.com with the part your missing and your order number and we will handle the rest.

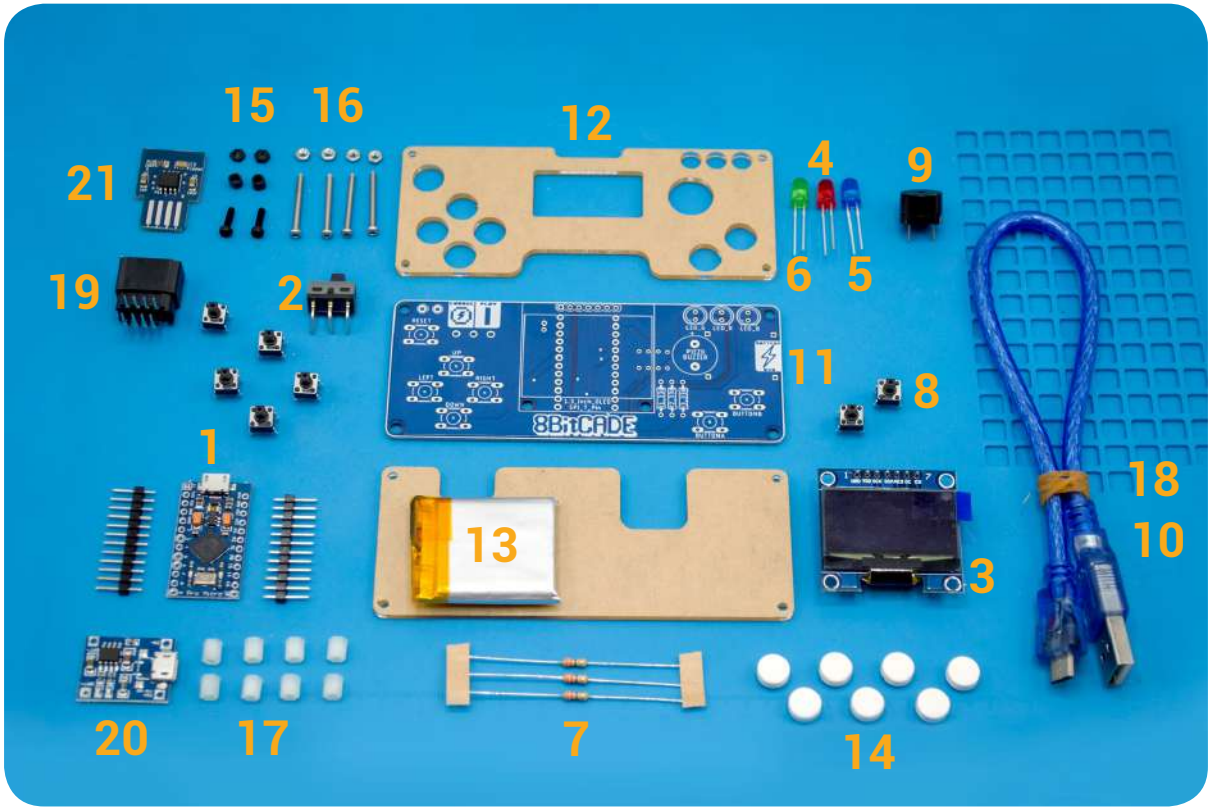
Component Name	Description	Quan	#
Pro Micro	ATmega32U4 5V 16MHz	1	1
Toggle Switch	Right Angle Toggle Switch 3PIN	1	2
1.3 inch OLED SPI Screen	1.3 inch OLED SPI 7 Pin White (Driver SSD1306 or SH1106) Resolution 128*64	1	3
Red LED	LED 5mm dia Red	1	4
Blue LED	LED 5 mm dia Blue	1	5
Green LED	LED 5mm dia Green	1	6
330Ω Resistor	330Ω	3	7
Control Buttons	6mm x 6mmx 4.3mm tactile micro switch 4 leg	7	8
Buzzer	Piezo speaker	1	9
Lead	Programming Lead	1	10
PCB Board	8BitCADE PCB	1	11
Case	Acrylic case	1	12
Battery	Lithium Polymer	1	13
Button Caps	8mm Button caps	1	14
M2 3mm Nylon Spacers	For the Screen (**Older Models**)	2	15
M2 8mm Nylon Screws	For the Screen (**Older Models**)	2	
M2 Nylon Nuts	For the Screen (**Older Models**)	2	
M2 20mm Bolts	For the Case	4	16
M2 Nuts	For the Case	4	
M2 6mm Nylon Spacers	For the Case	8	17
Elastic Band	Construction	1	18
Cartidge Connector	90 Degree Connector	1	19
TP 4056 charging module	TP 4056, USB Micro B	1	20
Optional: Game Cartridge	8BitCADE GC	1	21

Bringing out the tools

These 3 icons will be used throughout to indicate what tool you need to be using. The main tool is the soldering iron (green) however we are also going to use wire cutters (red) and a screwdriver (blue).



Whats in the kit?



Understanding Your Parts

At the back of the booklet, you can find your part dictionary. This will guide you through each major component that you will be using when building your console. You can flick to this if you don't understand what something is/does and it will give a short description. Not sure the name of the part? Compare the part with the photos, do be aware that the parts in the photos might vary slightly from your part, due to changes in suppliers. We highly recommend reading this before your start so you have a rough understanding of the parts you will be using, what their role is in the console and what they do.

Before We Begin

Read this section before you start making!

Soldering 101

This section aims to bring you up to speed on how to solder. We highly recommend reading this - even if you have soldered before. Lets learn a new skill!

What is Soldering?

Soldering is used to bond two pieces of metal together using a filler metal. Welding might come to mind however welding is where you melt the work pieces together, here we use something called solder (the filler material) to bond the work pieces together. Firstly, we melt the solder, allowing it to flow between the workpieces and cool to create a bond. In our project, the work pieces are the PCB and components. We are going to be doing "through hole soldering". Before you start, ***check out this video here***

https://www.youtube.com/watch?v=QKbJxytERvg

So What Tools Will We Be Using?

Soldering Iron

A soldering iron will be the heat source; it is used to heat up the solder. You don't need any fancy kit, a soldering iron that can reach temperatures between 300 to 400 degree Celsius. The one that we use and recommend can be found on the ***8BitCADE store*** and is a great 80W, temperature adjustable soldering iron. We recommend an iron that can be adjusted as it allows for greater versatility and safety when soldering components such as IC chips that are heat sensitive – this one has a brilliant LCD screen built in that allows you to see and adjust the temperature.

https://8bitcade.com/shopfront

Solder

We advise solder that has a ratio of 60/40. This means that it is made up of 60% tin and 40% lead. Low quality solder can cause bad solder joints and endless amounts of frustration – get good solder! The one that we recommend can be purchased at the ***8BitCADE Store***. Its 60/40 and 0.5mm in diameter, forcing you to be conservative with your soldering!

https://8bitcade.com/shopfront

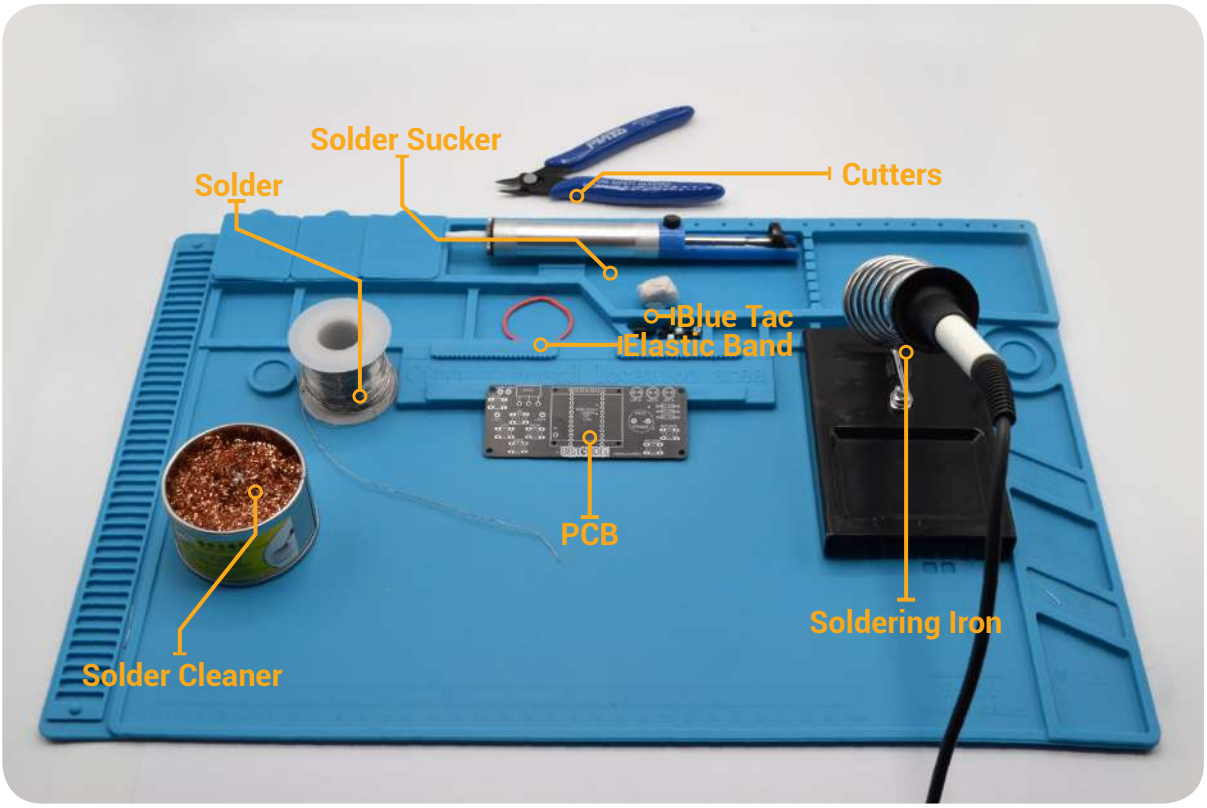
Solder Cleaner

This is used to clean the tip of your solder from both excess solder, flux and any oxidation that might have occurred on the tip. **Simply wet the sponge until it expands. Rub the tip of the iron until clean.**

Solder Safety

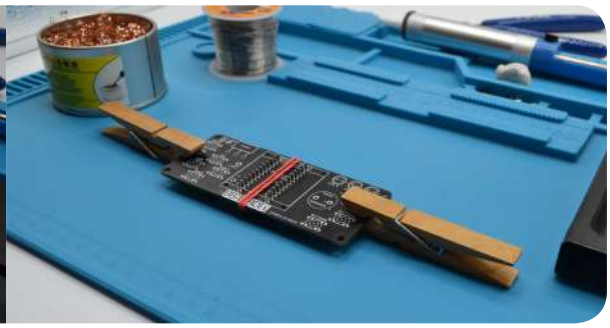
- Do not touch the end of the soldering iron while it is on/cooling down – they are very hot!
- After soldering a joint, DO NOT touch the joint as it will still be hot.
- Soldering does produce fumes, and for most can be nauseating if directly inhaled over long solder periods. We advise you to take 5 minute breaks every 25 minutes of soldering and to solder in a well ventilated area. DO NOT breathe in the fumes directly – a fan or ventilator can help remove fumes or simply moving your head to the side, not directly above.
- NEVER place your solder on the workstation, ALWAYS place it in the soldering stand/station.

Sample Soldering Station



Understanding Your Work Station

While we have broken down each core tool we will be using, for those that want to get into soldering, we recommend the setup like above. A polymer mat helps stop burning solder from marking your table and also organizes your work station. We can also see we have some basic tools to help secure our work piece, such as elastic bands and Blue Tac - see more below.

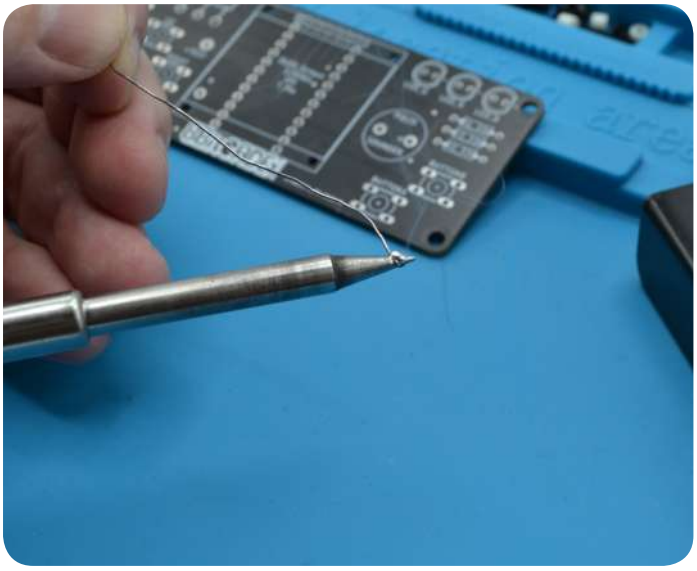


Holding the Work Piece

For those starting out with soldering, and don't have a “helping-hand” station, then we advise you grab some of your mums pegs and peg your work piece! It raises the PCB up enough for the work piece to be secure, flat and for the components underneath to not be touched. On the contrary, for those who are looking to solder more often, we advise investing in a “helping-hand” station that securely holds your work piece using crocodile clips.

How to Solder!

Read this section **before you start making!** ***Do not solder any parts from the kit yet!***



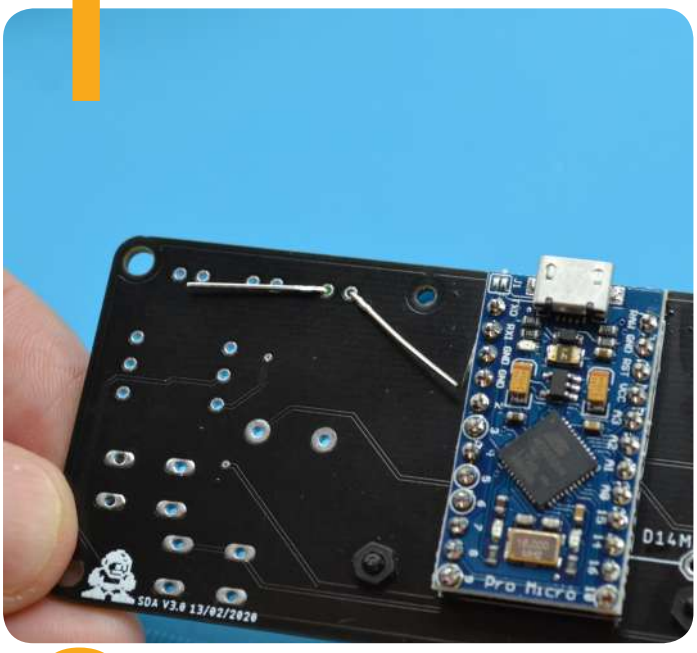
Tinning The Tip

What does "tinning the tip" mean? To put it simply, its covering the solder tip with solder. We can use this before we start soldering and when we are finished. Before packing away your soldering iron, you should tin the tip to increase the life of the tip. The tip of your soldering iron oxidizes quickly, as it is typically made out of copper plated with iron. By melting solder around the tip, we are stopping the tip from oxidizing (as an oxidized tip is inefficient at transferring heat). However anther use for tinning the tip is before you solder a fresh joint. By having a bit of solder on the tip, you can spot-solder the joint as the solder from the tip cools and fixed the joint in place while you heat up the joint. It also aids to the efficiency of transferring heat from the tip to the joint.

Soldering Components to a PCB

Known as "through hole component soldering" it will be the main soldering we use in this build guide. This is a tedious job because we are dealing with heat sensitive components. Its crucial to not over heat the parts/pads. But do not worry! We will take you, step by step, through the process and you'll be a soldering genius in no time – remember, you can always desolder! And you'll learn that too!

1

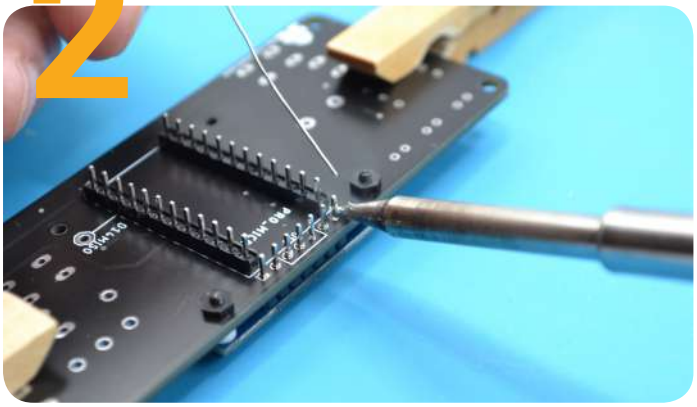


Mount Component

Check with the guide photos while doing this as you don't want to be soldering and Desoldering a part because you put it in the wrong way around! For components like resistors, you can slightly bend the legs out, allowing you to flip the PCB around and solder the joint. For other components, masking tape can be used to hold the component in place. For more advanced uses, a "helping hand" can be used. These units utilize crocodile clips, on arms, to hold the PCB and component in place – it's up to you how you want to approach this but blue tac and masking tape can go a long way!

Top Tip: Before soldering, ensure your work pieces are secure and wont shift mid solder - that can extremely frustrating. Major takeaway? Utilize masking tape and blue tac!

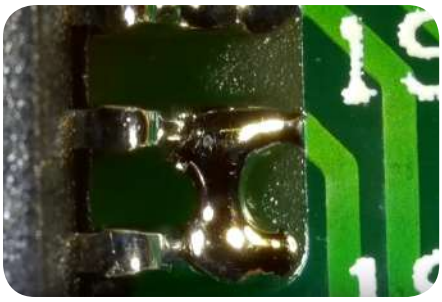
2



Heat The Joint

Apply heat on the component leg and conductor pad. The aim is for the pad and component to melt the solder, not the soldering iron tip. This ensures you do not get a "cold" joint (see reference picture below).

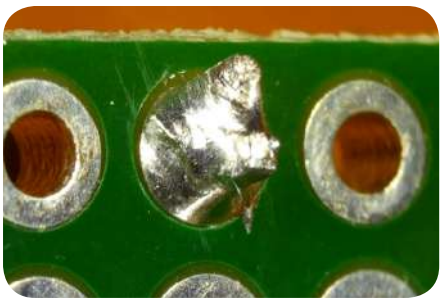
Check. Check. Check! Its crucial to check your joints. When soldering, you can run into a field of issues that can be easily fixed. Below is some of the most common mistakes and how to fix them!



Source: Pimoroni, Youtube-Androkavo

Bridging

When soldering joints that are very close to eachother, the solder can sometimes flow and melt together. This can cause a short circuit and damage your board. Its important to check for any “solder bridges”. Simply drag the tin down the middle, like a knife, to melt the solder and remove it. Sometimes this isn't enough and using a “solder sucker” can be a easy way to remove the excess solder. That's why this problem occurred in the first place, because of too much solder.

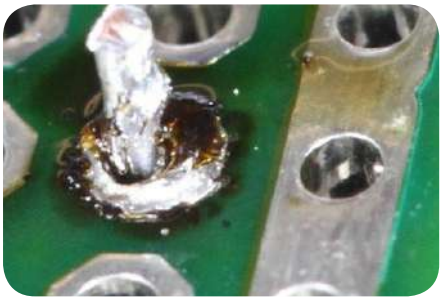


Source: Androkavo Youtube

Cold Joints

Is where the solder cooled too quickly and didn't get enough time to seep into the joint crevices. These generally look lumpy and rough with their strength being unreliable. If you get a cold joint, simply reheat the joint allowing the solder to flow better. Another reason for getting a cold joint is using too much solder, the excess solder can be drawn out by your tip or sucked out using a “solder sucker” that we will discuss later on.

Top Tip for Cold Joints: When reheating the joint, put some solder onto your tip. As the solder has cooled down, it has oxidized on the joint and can be harder to heat up. Adding solder onto your tip can help reduce this problem.



Source: Pimoroni, Youtube-Androkavo

Overheated Joints

Not enough heat? Well don't over heat it! This can cause issues as well. Not only will the heat get conducted up into the component and potentially damage the component, it can also damage the pad on the PCB – usually seen through black burnt marks and/or an orange tint on the solder. To prevent this, use a clean tinned tip with the correct heat settings (350 degrees Celsius).

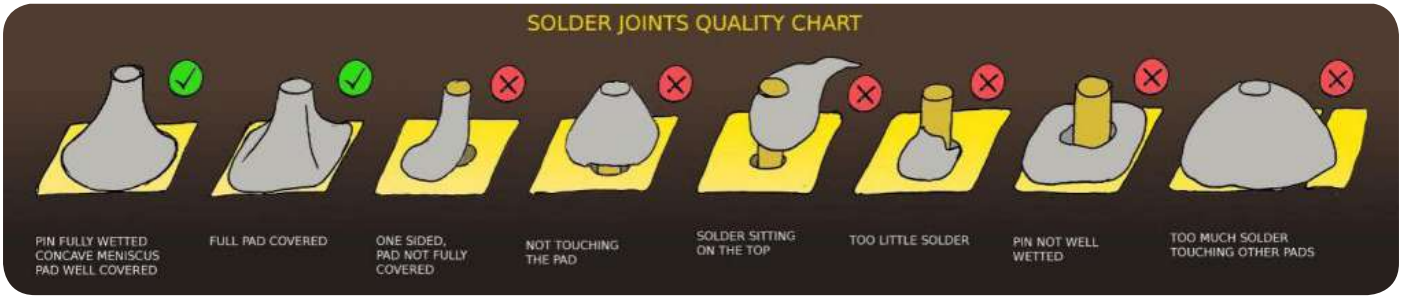
Source: SeedStudio - Soldering Problems

Insufficient Wetting

A term commonly used in reference to soldering. It is to do with how well the solder melts/bonds to something, e.g. the pads on the PCB. For us we are concerned with the solder properly wetting (being bonded to) the pads and legs of components. A joint that has insufficient wetting would be seen as the solder won't ‘stick' to it. To solve this, resolder the joint with more solder, ensuring to heat the part that the solder didn't bond to.

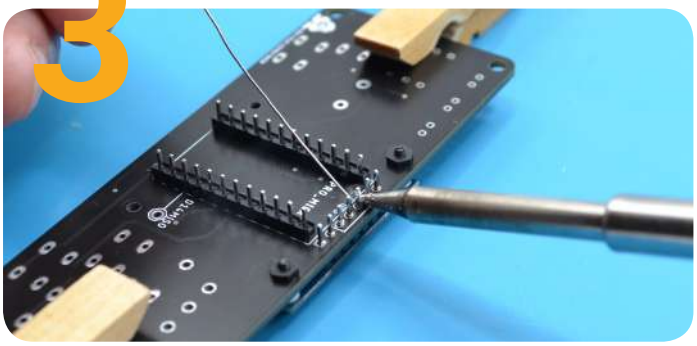
Wetting could be seen as using too little solder – but you can also use too much! The perfect solder join is one that arcs up into the pin from the PCB, as shown in the photo. The aim is to make it look almost like a volcano – with the pin erupting from the middle.

Top Tips: You should be heating each joint for about 2 to 3 seconds and then applying the solder. A Temperature of 350 degrees on your soldering iron is recommend as too much heat applied can cause the component to be damaged.



Source: gaudi.ch

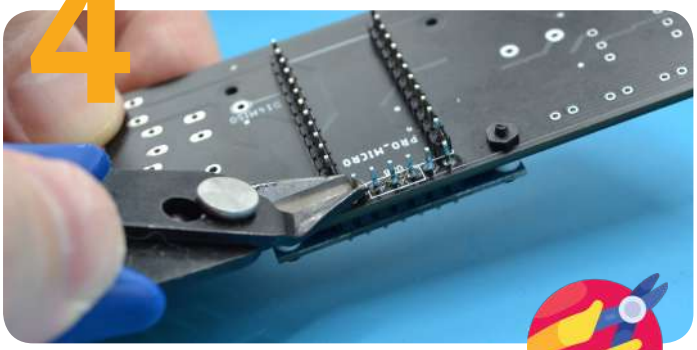
3



Apply Solder To Joint

Dip the solder between the component leg and pad, allowing the joint to melt the solder and fill the crack. If it does not then the temperature is not right, there are three general reasons for this: You soldered too early; The soldering iron temperature is too low and should be turned up or the soldering tip is not transferring heat effectively, meaning the tip could be corroded and/or too small for the joint.

4

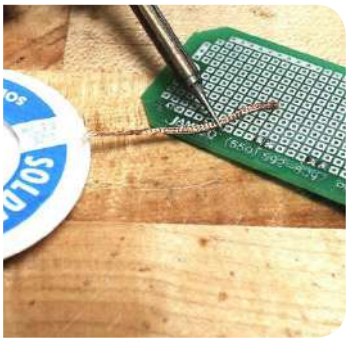


Snip Protruding Material

When trimming the legs of components like resistors, as a health safety rule, you should either hold the part being cut off, or cover it with your hand. This is as when you cut it, the pin will fly off and could land in your eye. To trim the legs, simply put the flat side of your cutters on the top of the solder and squeeze (covering it with your hand!).

Summary Tips:

- Do not heat the component for too long – as this could damage the component/pad
- Tin the Soldering iron tip beforehand to remove oxidization and make it easier to solder.
- The perfect solder joint is one that is shiny and looks like a volcano (a cone shape a concaved surface)



Source: printeraction Instructable

Desoldering

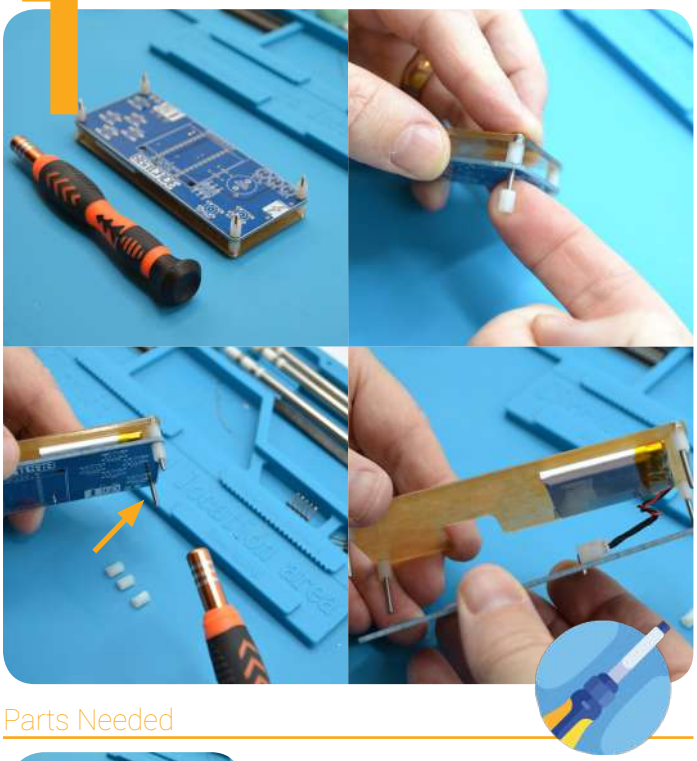
To desolder a part, we can use a solder suck to quite literally suck melted solder from the joint, or we can use Desoldering braid/wick. Desoldering braid/wick is braided copper wire that will remove the solder from your joint by 'absorbing' the solder.

LET'S GET MAKING

Lets get started! Disassembly

Follow the below steps to prepare your kit!

1



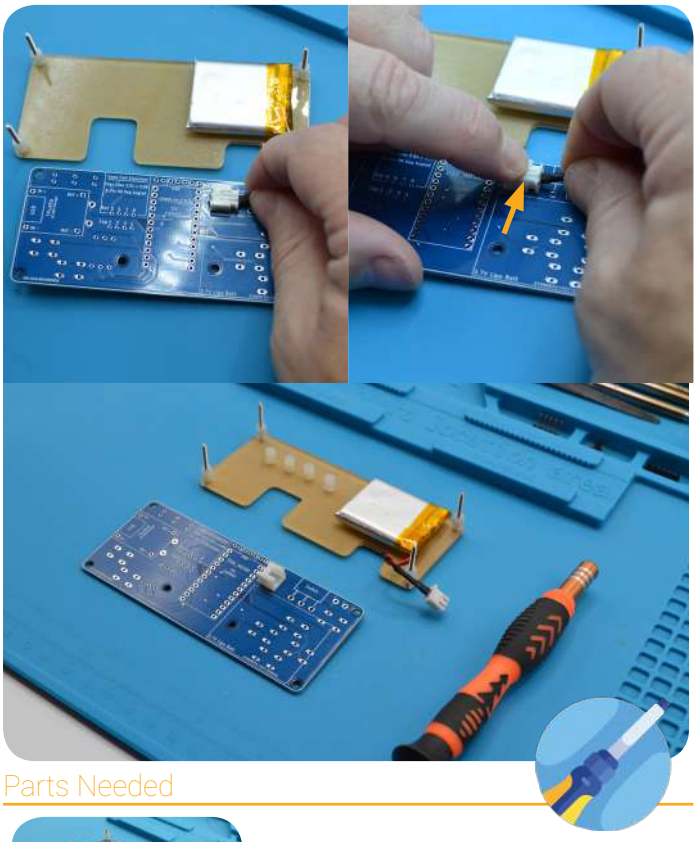
Disassembling

Before we get started, we need to firstly disassemble the PCB and case assembly so we have access to the PCB for soldering.

Grab your hex screwdriver and begin unscrewing the spacers on the TOP side of the PCB - by holding the spacer while unscrewing the 20mm bolt on the other side.

Do this for all spacers so we can take off the PCB.

Parts Needed



Unplug the battery

Unplug the battery from the PCB. To do this, hold the female PCB-fitted connector with one hand while pulling on the male battery connector with your other hand.

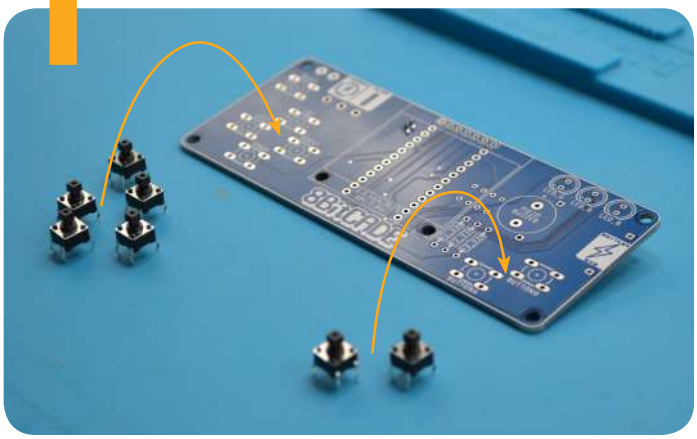
Do not pull from the wires but rather from the connector itself to not damage the battery or connections.

Parts Needed

Lets Start Making!

Follow the below steps to make your kit!

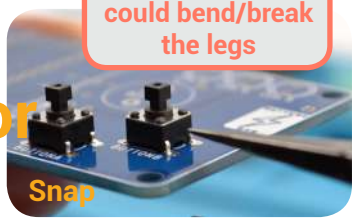
1 Soldering & Trimming the Buttons



Buttons must be soldered in on the side with the 8BitCADE Logo

The first thing we need to solder are the buttons! We are going to solder each button, one by one, to make sure your soldering skills are up to scratch!

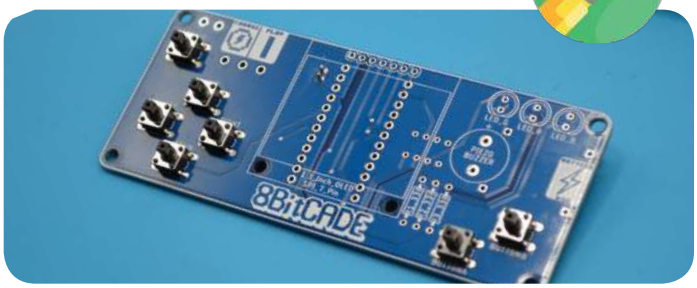
or



Do not push on the button to hard is it could bend/break the legs

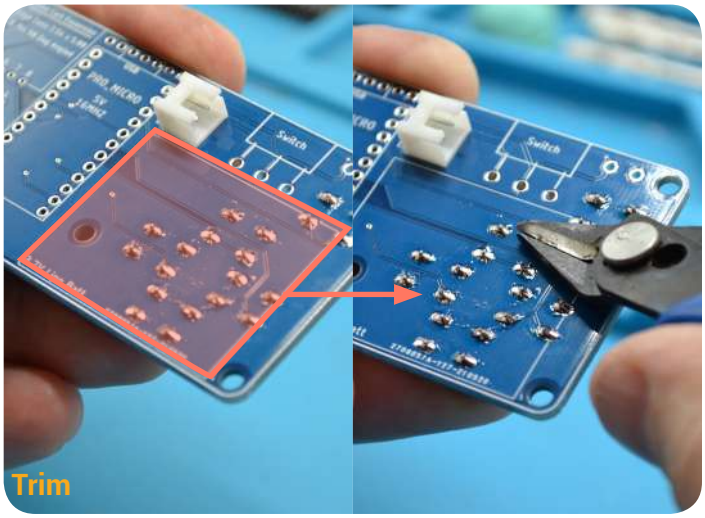
Pick up any button and place it in the marked button holes on console (see photo). Then, firmly press down to push the button into position. (Do not push to hard as it could bend the button legs). If this does not work, place one side of the button pins inside the holes, then, using the edge of the peg, push and bend the other side of the pins inside the holes - the button should click into place!

Its important to solder the buttons flat - so check that your buttons are pressed up against the PCB before you solder them! A good idea is to press your button a couple times to push it down into the PCB. Once fitted, solder the buttons in place. If their pins poke through too much, trim them!

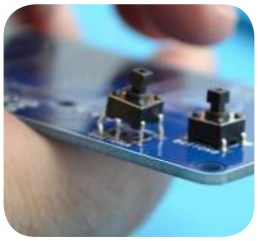


Now go ahead and repeat this procedure for each button on the console! All 7 buttons!

1. Place the button in the correct position.
2. Push or use tweezers to guide the button into place
3. Turn the PCB over, and solder the joints.
4. If the button pins point out alot, trim them to shorten them.

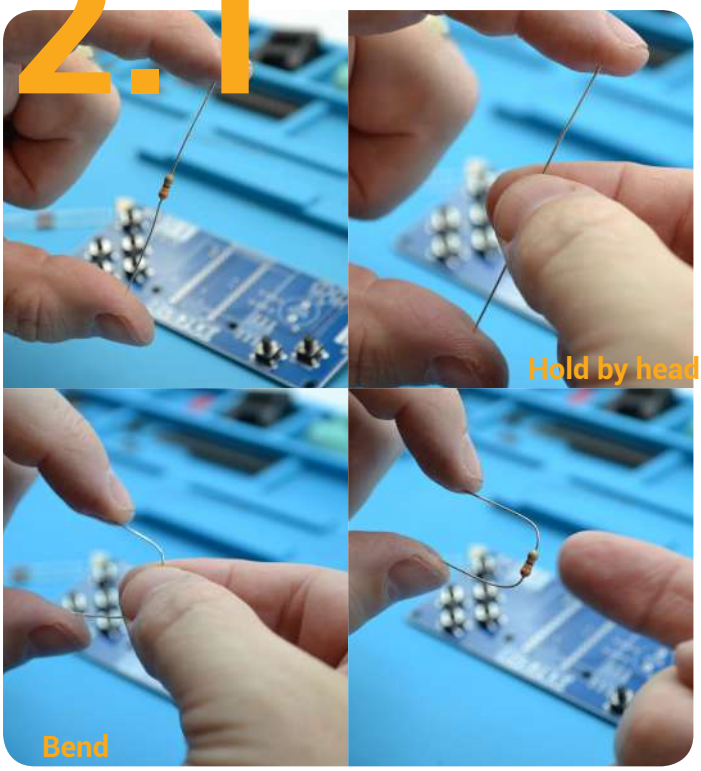


Trim Your Pins! Community Feedback:
Some 8BitCADERS have advised us that if the pins are to long/not trimmed short, then the pins are very close to the battery and you can risk puncturing the battery. Please make sure to trim all solder joints in the white battery square on the rear side of the PCB short!



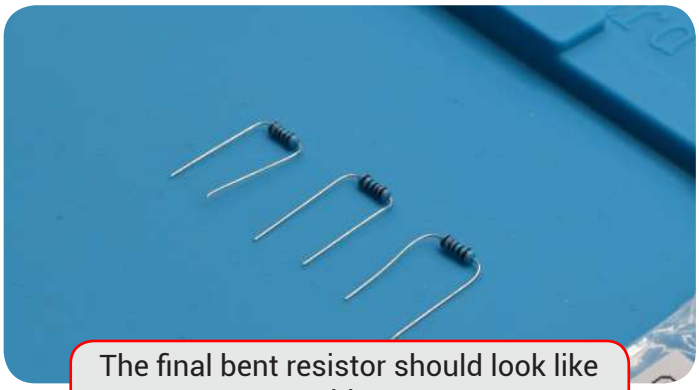
Make sure the buttons are soldered in flat!

2.1



How to prepare Resistors/ Diodes for soldering

This sub step should be conducted whenever you are soldering in a resistor or diode. Here we are bending the legs to allow them to be properly placed in the PCB. Here, you place your index and thumb on the resistor/diode and bend the legs so they are square (90 degrees).

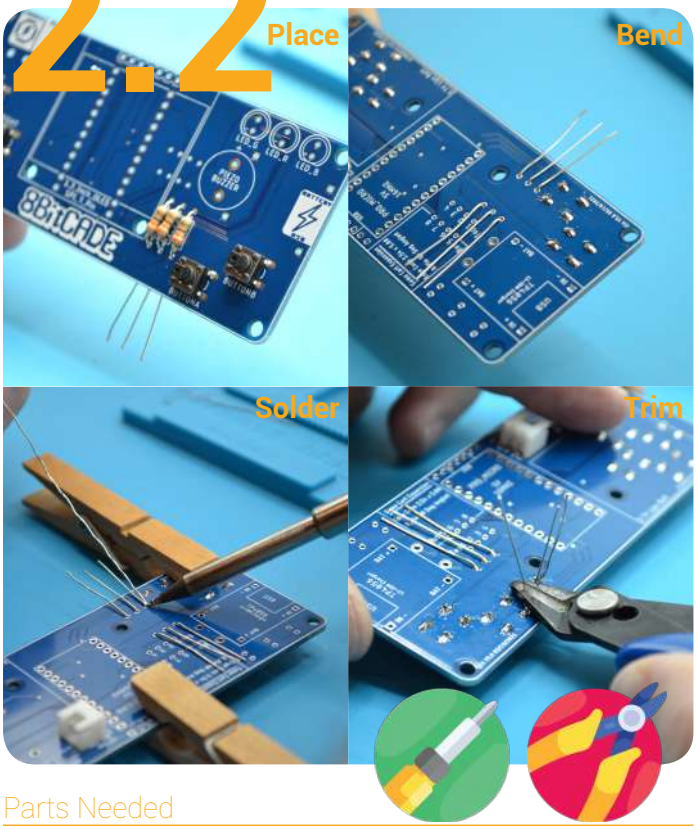


The final bent resistor should look like this!

Parts Needed

Any resistor

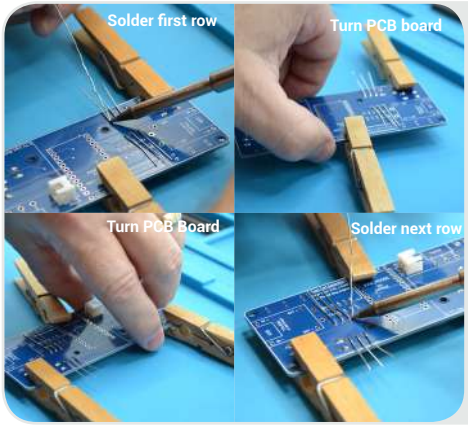
2.2



Solder & Trim the 330 Resistors

Fit the 330 resistors in their labeled holes on the PCB (orientation is not a problem). Please check that you are using the correct resistors and that they are being fitted inside the correct place on the board.

Refer to step 5.1 on how to bend the legs of a resistor. Once bent, place the resistor in the PCB, turn it over and bend the legs. Then solder the legs in place. Once complete, pry the resistor legs up and trim the ends of the legs off just above the solder joint.

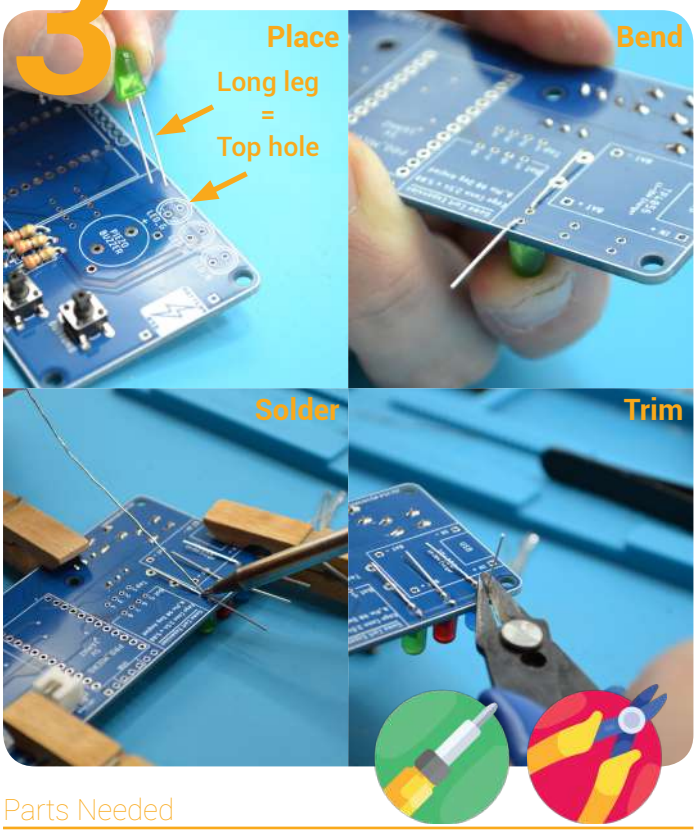


Make sure to turn the PCB to make soldering easier! You shouldn't be soldering over components as you may accidentally melt /damage a part!

Parts Needed

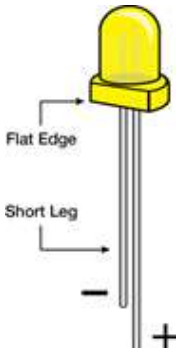
Colour code:
Orange, Orange, Yellow, Gold

3



Solder & Trim the LED'S

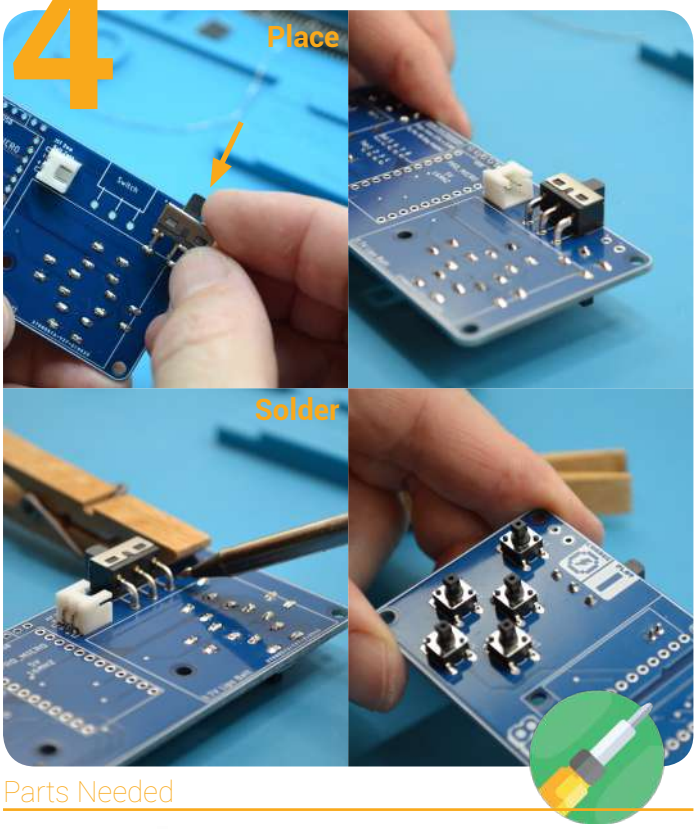
Fit the LEDs into their respective holes (they are labeled on the PCB: LED_G = LED green), in the correct orientation (flat edge facing right, short leg in the negative hole - the PCB has this labeled with a minus and the actual flat edge). Bend the legs of the LED to hold it flat to the surface of the board. Solder each leg and trim the legs, just above the solder joint. Repeat this process for the 3 different LEDS.



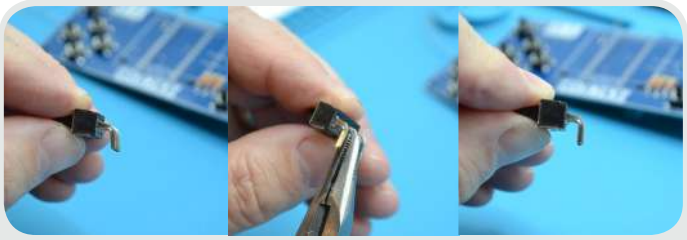
Parts Needed

Make sure to put the cut offs from the resistors & LEDS aside as these will be used for the TP4056 later on!

4



Place & Solder the Switch



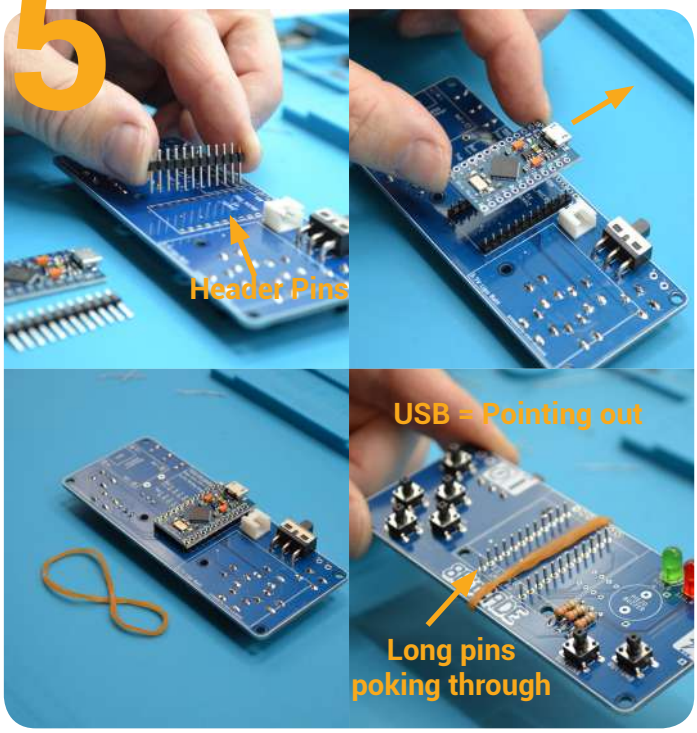
Before soldering in the switch, to ensure it sits flat, make sure that the pins are at a 90 degree angle. Here you can see we had to slightly bend the pins with some pliers to adjust them to a 90 degree angle!

Flip the board onto its rear and fit the Switch in its marked position - push the pins into the sockets firmly. Solder the component in place. If the pins poke through too far, trim them - however this is generally not necessary.

Before you move on, to be sure your solder joints are strong and correctly soldered - switch the button on and off to test how your solder joint deals with the pressure. If it is weak/moves, check your solder joints

Parts Needed

5

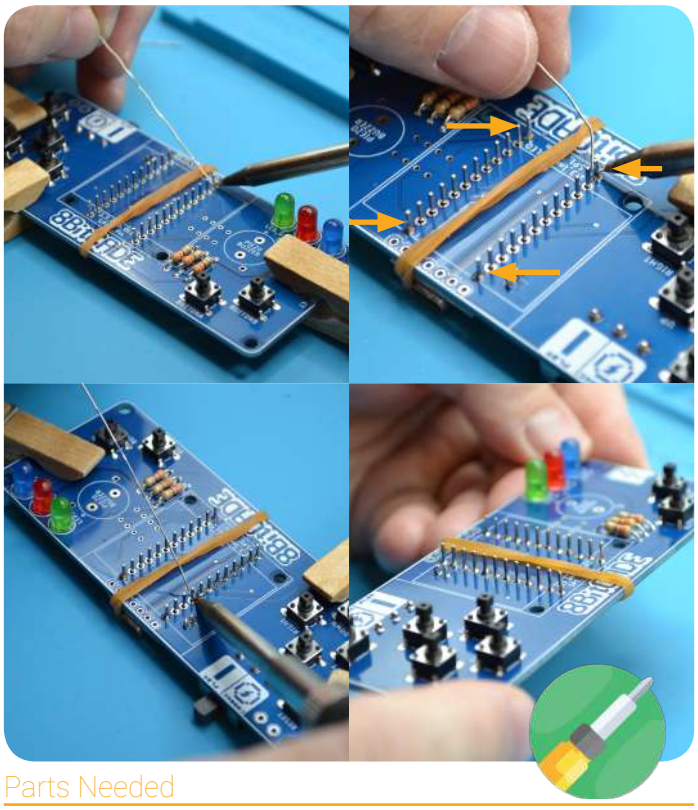


Assembling The Pro Micro

Place the Pro Micro on the 8BitCADE OG Board (rear side) with the **long side of the header pins** passing through the board and USB connector aiming away from the board. Place an elastic band on the assembly to hold the Pro Micro in place. This will help us in the next step as it will hold everything in the correct position.

Be sure that the USB (on the micro) is pointing away from the board, aiming at the top of the board. You will know it is the top of the board, as if you look on the front of the PCB, the 8BitCADE Logo is at the bottom.

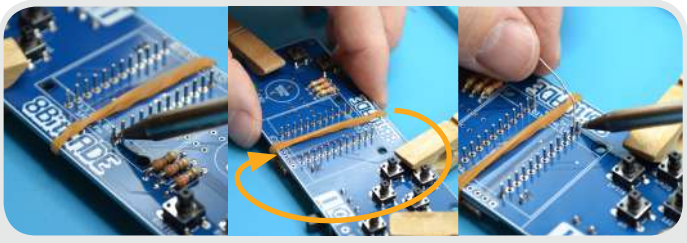
Parts Needed



Solder the Pro Micro Pins

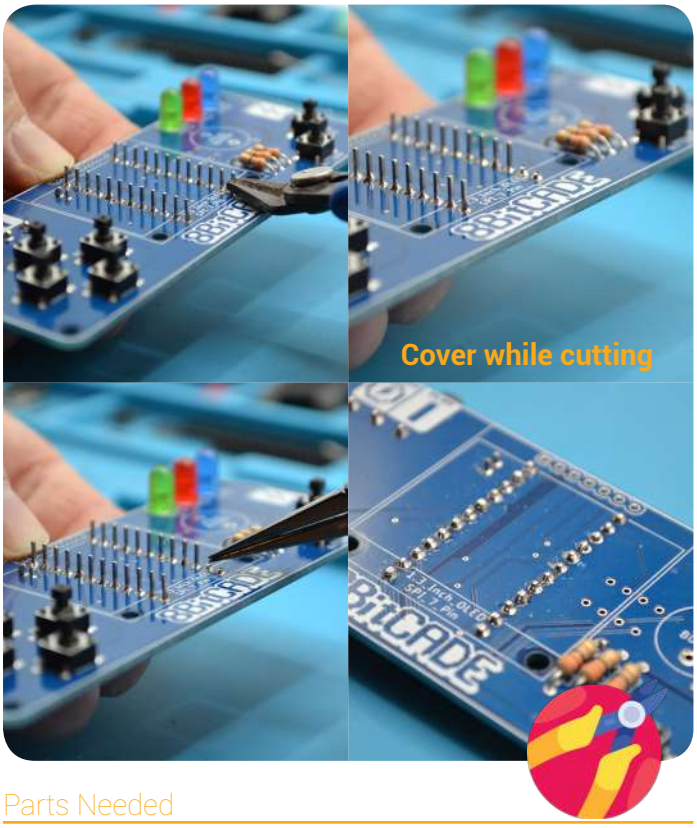
Solder the four corner pins first (see arrows), once done, remove the elastic band and solder the rest of the pins, working from left to right, towards the center pins to spread the heat load on the electronic device - as heat can damage your components, which is why you should not be spending too long applying heat to the joint. All pins must be soldered.

As these pins are close together, check them to make sure none of the pins are connected by a solder bridge. If so, use the tip of your soldering iron to split the connection.



When soldering, don't be afraid to move and rotate the PCB. Here we solder one row of the Pro Micro, rotate the PCB around and solder the other side, from the same angle. This is to make it more comfortable and reduce soldering over parts.

Parts Needed



Trimming the Pins

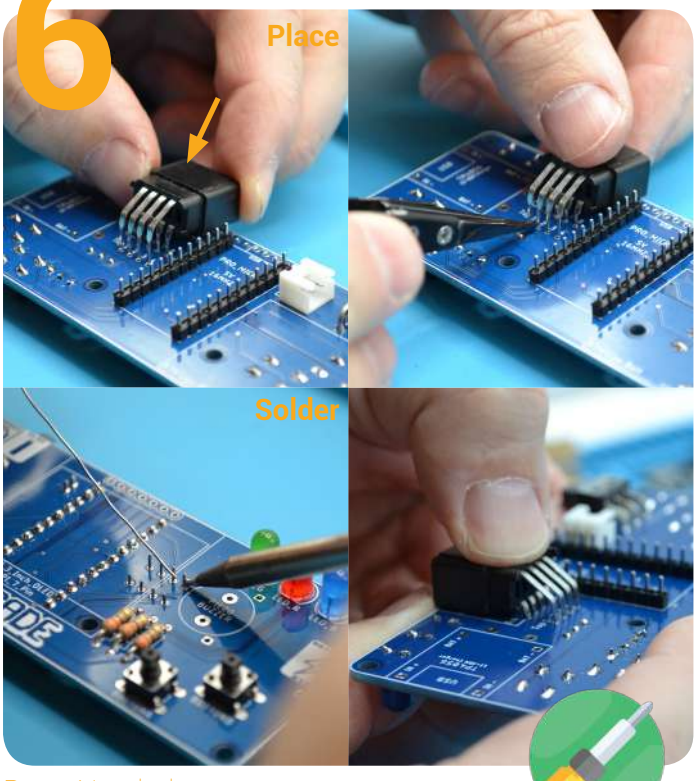
Once soldered, cut the ends of the pins with sharp cutters.

When you are trimming your pins, be sure to either point the cutter down when you cut it, or to cover the pin with your hand - both stop the pin from flying out and hitting your eye. See photos on how to do this.

While trimming your pins, be sure to place the cutters parallel/flat to ensure you don't scratch the PCB. Scratching the PCB can cause a multitude of issues, especially if you end up bridging it with a solder joint! So please be careful!

Parts Needed

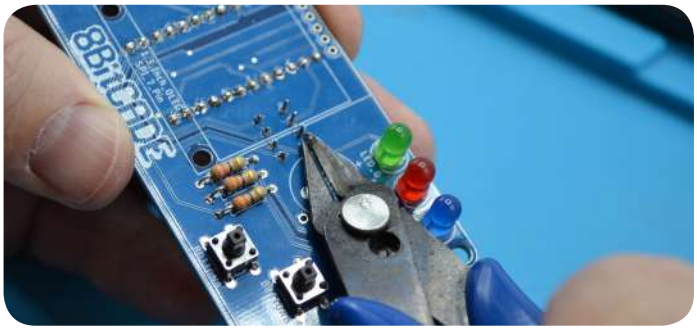
6



Solder the Cart Connector

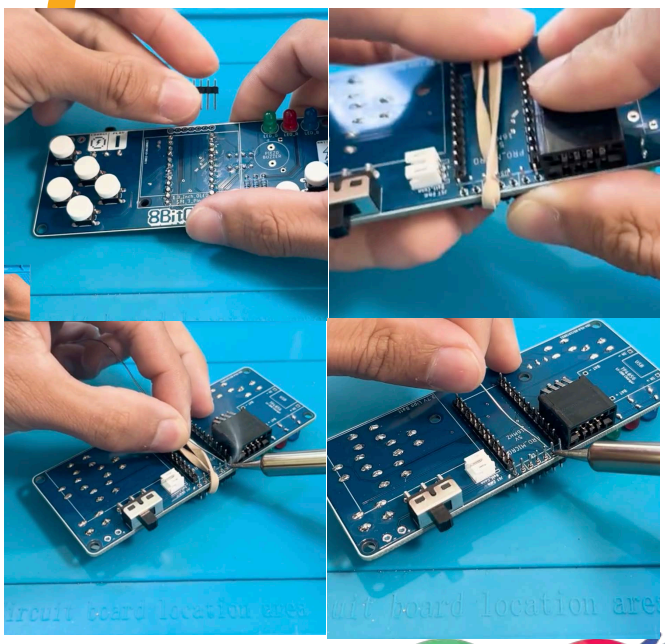
Flip the board onto its rear and fit the Cartridge Connector in its marked position. Make sure to push the pins into the sockets firmly - you may have to use tweezers to push the pins closest into their holes.

Once in place, solder the component in place. Be sure not to solder over components such as the buttons or LEDs to reduce the risk of melting them! Then we need to go ahead and trim the pins.



Parts Needed

7

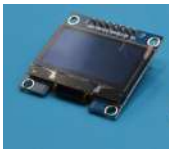


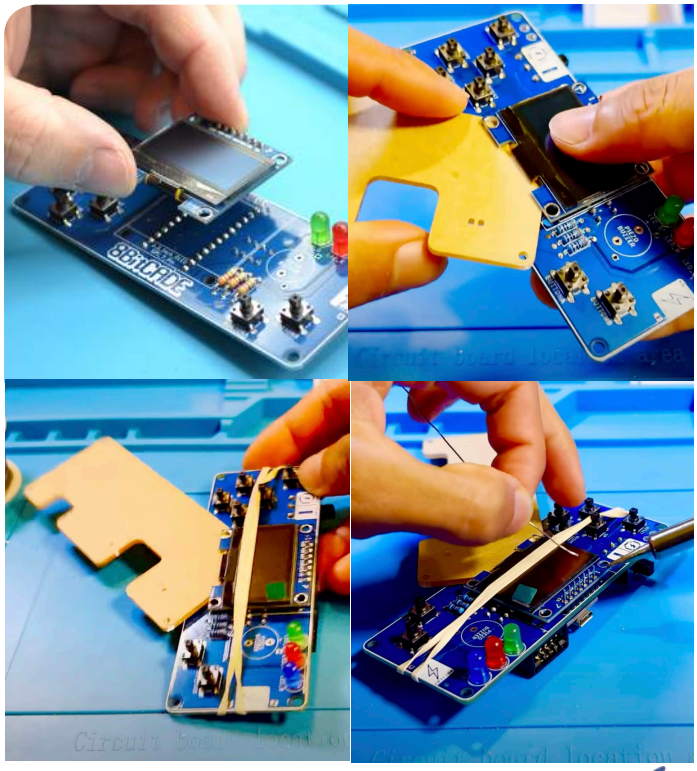
Placing Pins for the Screen!

Place the screen header pins in their designated spot above the PCB area. Make sure the pins are in the right orientation. The longer side should be sticking through the PCB towards the side with the switch. The shorter side with the black spacer should be on the side with the buttons.

Secure the header pins with your rubber band and begin by soldering the edge pins. Be very careful when soldering the pin next to the 90-degree connector so you do not melt the plastic. After you complete the edge pins, you may take off the rubber band and solder the rest of the pins.

Parts Needed





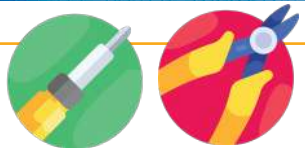
Solder the Screen in Place

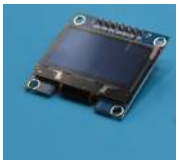
Place the screen onto the device with the screen side resting on the black spacers. It is very important that the spacers provide the gap to rest the screen on.

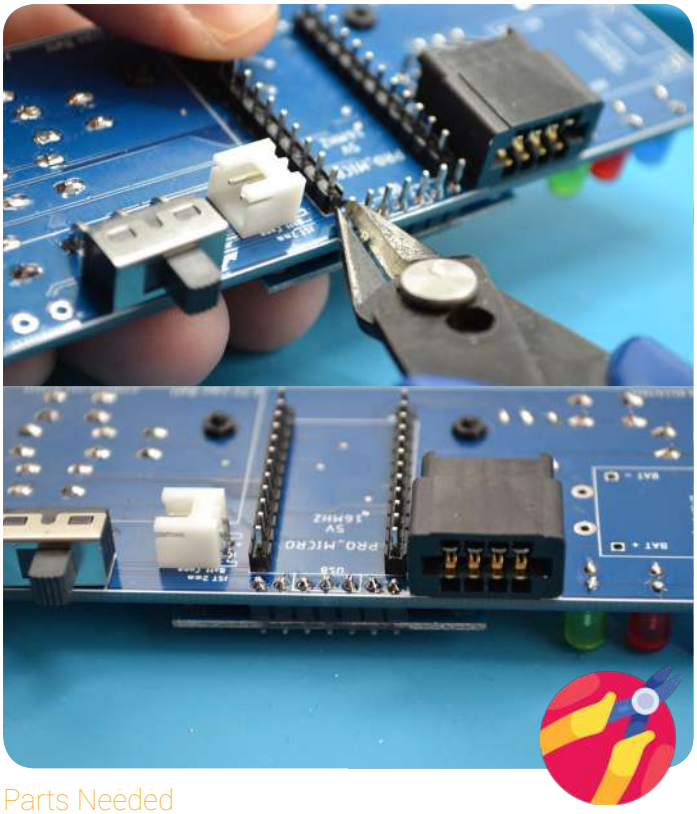
We will be using an acrylic case panel to act as a spacer when soldering on the screen. Insert the corner of the acrylic case between the screen and PCB and use rubber bands to secure everything in place. Carefully solder the joints above the screen.

Try to keep the screen as level as possible. Once you solder the screen on, there is no taking it off. After you finish soldering, you may undo the rubber band and remove the spacer.

Parts Needed



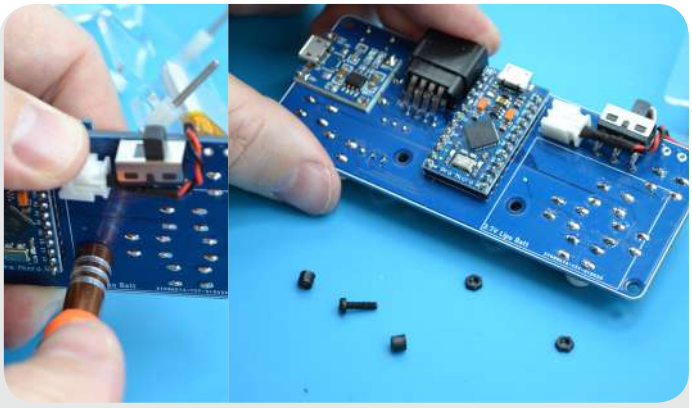




Trim the Screen

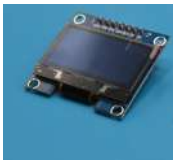
The final step for the screen is to use the cutters to trim the pins of the screen down to the top of the solder

When you are trimming your pins, be sure to either point the cutter down when you cut it, or to cover the pin with your hand - both stop the pin from flying out and hitting your eye.

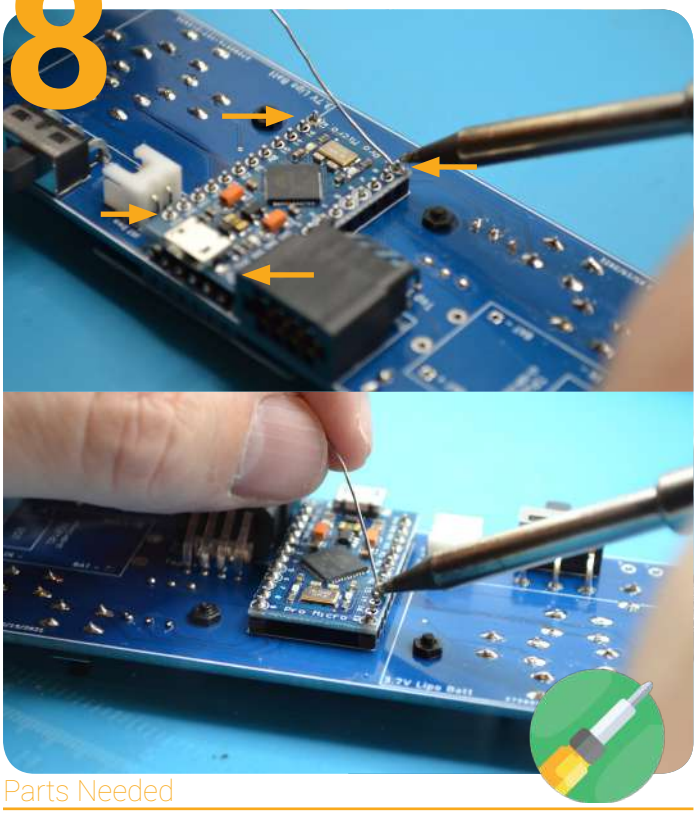


Once you have completed soldering in your screen, its important that you remove the temporary spacers. To do this, simply loosen off the bolt, remove the screw and then remove the spacer! Like so!

Parts Needed



8

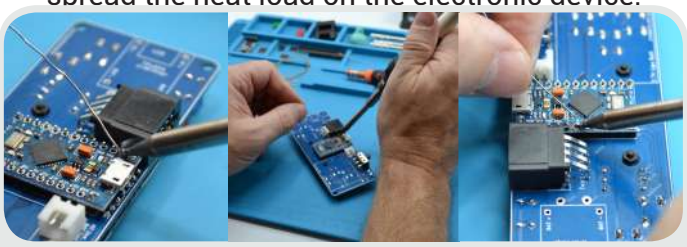


Soldering the Pro Micro

Flip the board to its rear to solder the Pro Micro to the header pins. Ensuring the USB is facing the correct orientation (USB aiming forward, as shown).

When soldering the micro controller, take great care. Be sure to follow precisely the below soldering pattern, to reduce the amount of heat we apply on this sensitives unit:

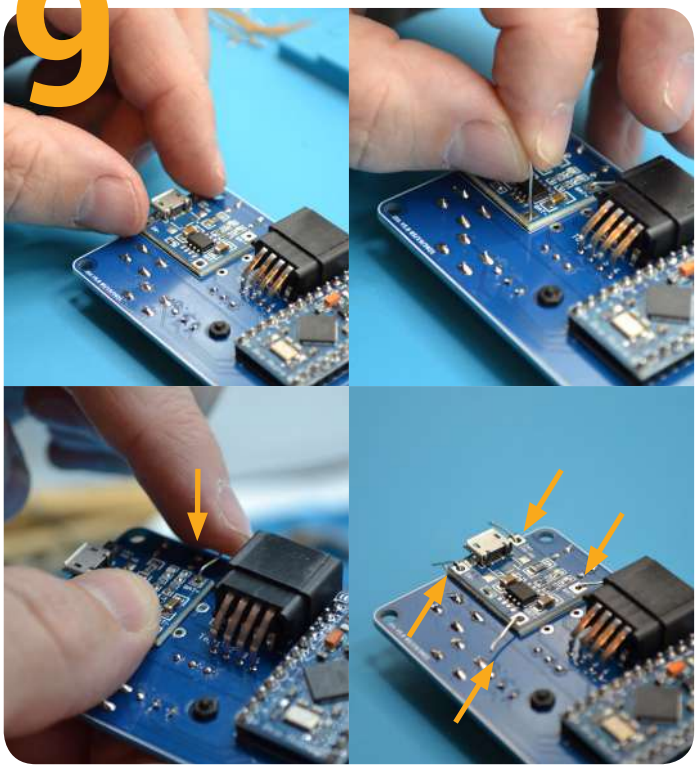
1. Solder the four corner pins first, this will lock the component in (see arrows).
2. Once done, solder the rest of the pins, working from left to right, heading towards the center, to spread the heat load on the electronic device.



The pins around the Cartridge Connector should be soldered with extra care. Make sure to solder them at an angle to ensure you dont touch the soldering iron on any other component! Notice how we are soldering at such a steep angle!

Parts Needed

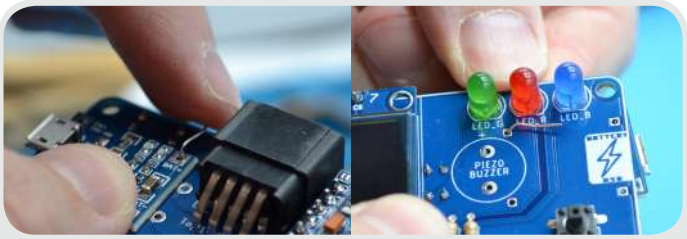
9



Place the Charge Protection Board

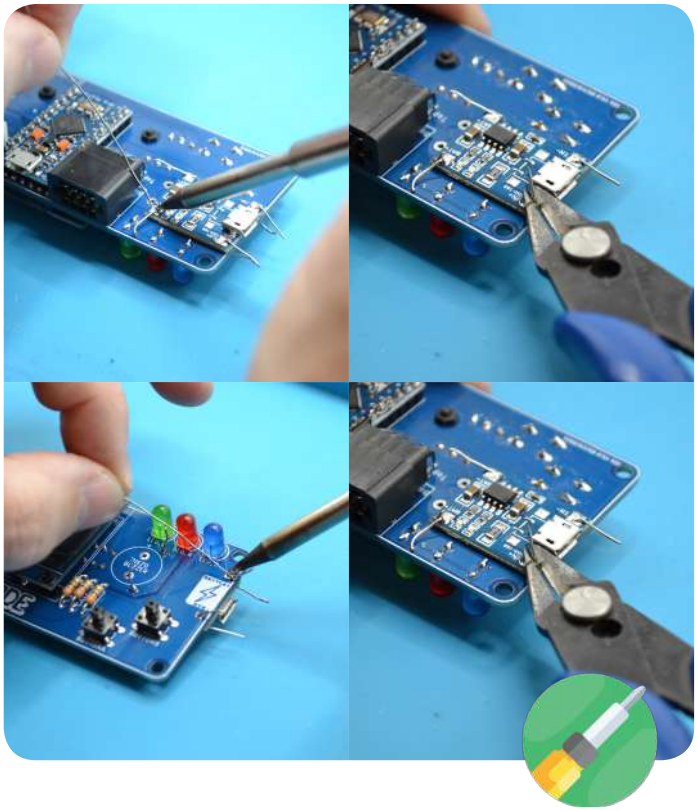
Place the tp4056 battery protection board in its outlined place on the PCB. Using the trimmed Resistor legs from before, place them in the 4 outer holes on the protection board - as seen in the photos. It can help to bend these so the board stays in place.

Ensure that the board is on the rear side of the PCB board and that the USB connector is facing outwards, away from the Cart Connector



When placing the trimmed legs, make sure to hole the TP4056 in place and bend the pins over on both sides of the board. This will secure the TP4056 in place!

Parts Needed

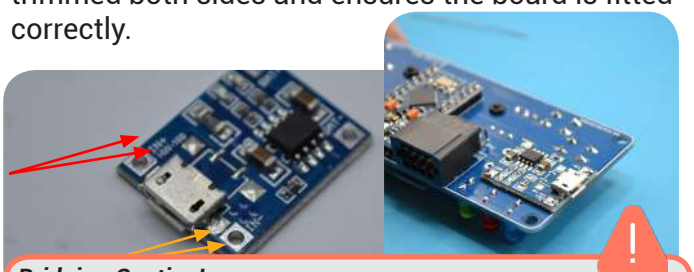


Solder & Trim the Charge Protection Board

Once fitted and aligned, solder the corner pin to hold the board in place. If you need to realign the board, apply heat to the soldered pin and move the board alignment at the same time. Be very careful if you do this as your fingers will be near the soldering iron.

Once aligned, solder the other 3 pins in place. Once all pins are soldered on one side, trim the legs.

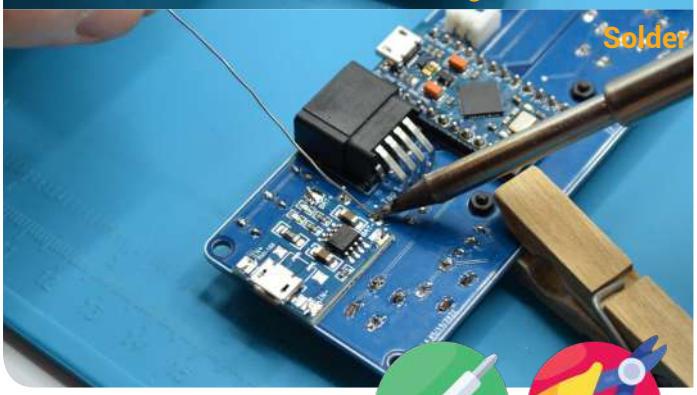
Then, flip the PCB over and solder & trim the 4 pins on the other side of the board (see photos for reference). This ensures the pins are soldered and trimmed both sides and ensures the board is fitted correctly.



Bridging Caution!
The left-front pin cannot be bridged to the USB port (hence the gap) - (Red Arrow).
The right-front pin doesn't matter if it is bridged – this has no effect to the board, console or charging (Yellow Arrow).

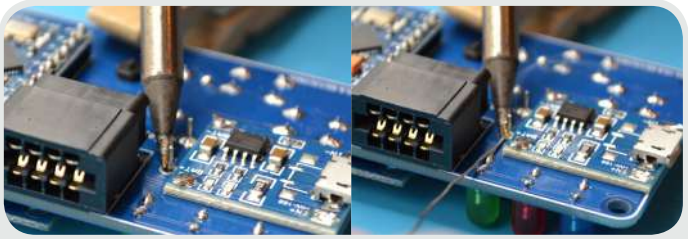
Parts Needed

10



Solder & Trim the Buzzer

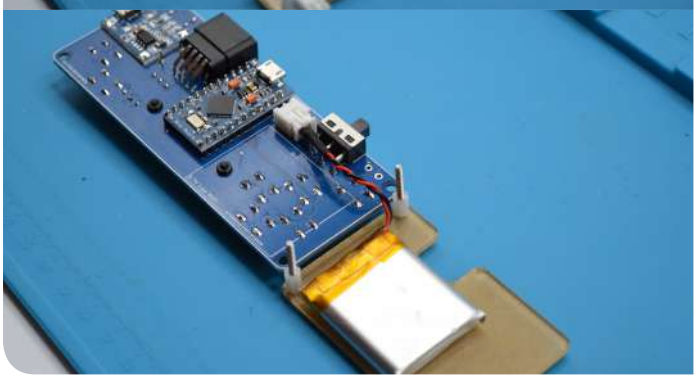
Fit the Piezo Speaker in its outlined place on the PCB, place the positive pin of the component into the positive hole on the 8BitCADE XL board (as noted on the buzzer with a plus, and on the PCB with a smaller +). Once fitted, bend the legs to secure the buzzer in place. Then, solder the component in place and trim the protruding legs to the bottom of the solder joint.



When soldering in the buzzers left top pin, its important to angle the soldering iron away from the Cartridge Connector. See the above pictures on how to correctly solder in the last connector!

Parts Needed

11



Plugging the Battery in

Before plugging in the battery, ensure the switch (from the rear side) is turned all the way to the right (closest to the JST battery connector). This ensures the unit is off while we plug in the battery.

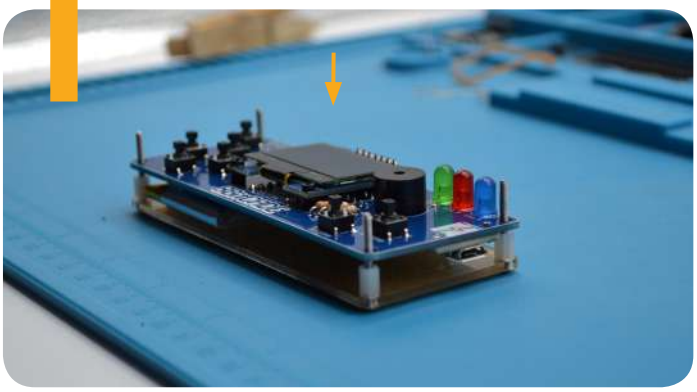
Plug the battery in the JST connector by holding the male wire connector and pushing it inside the female connector mounted on the PCB. There is only one way the connector can go in so if it does not fit, turn the cable around and try again. Do not force it in.

Parts Needed

Dont Forget The Case

Lets add the last touch - the acrylic case! In this section, we will go through how to make it

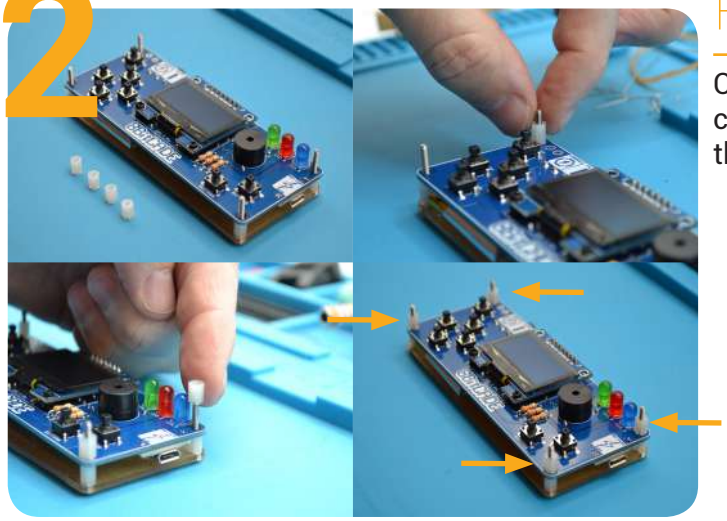
1



Place on the back cover

Place the back panel + battery assembly onto the rear side of the PCB. So the 20mm bolts are poking through!

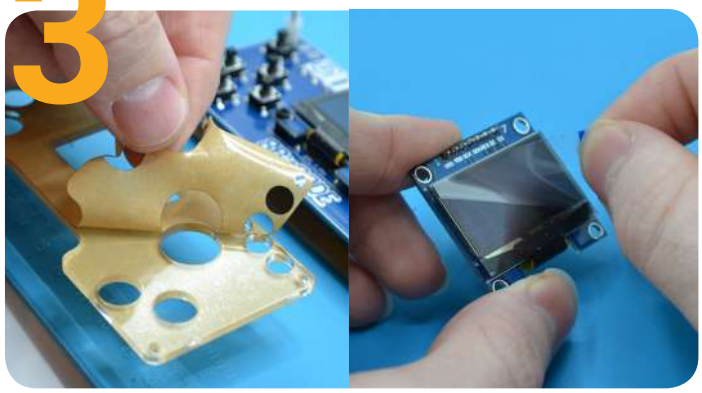
2



Place Button Caps

Click on all the buttons. The reset button will not click on but will just slide and sit on the button. See the photos for reference.

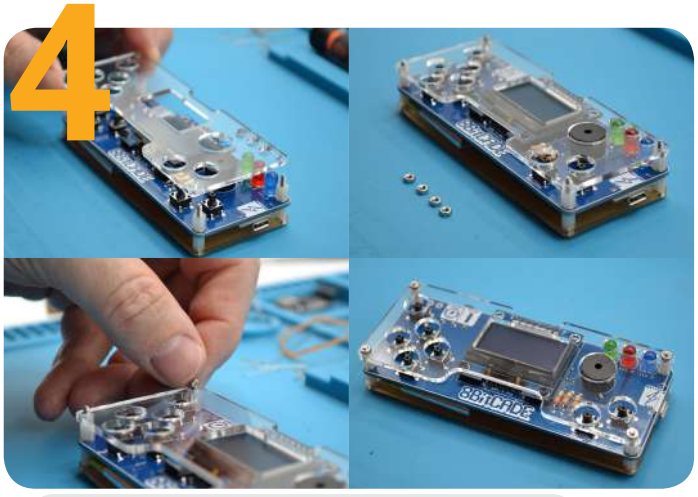
3



Peel off the paper

Peel off the brown protective layer and clear cover from the front panel (both sides) and the screen.

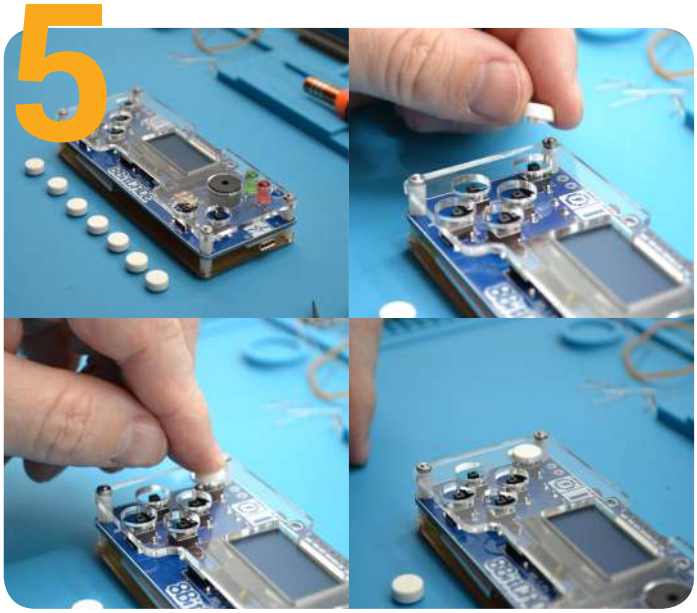
4



Secure the top panel

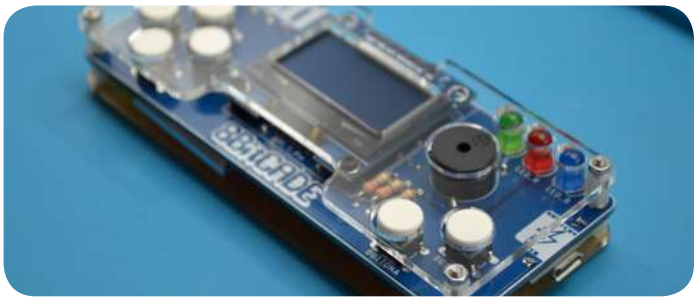
Now that we have peeled off the top panel, place it on top of the console, so the button holes line up, and tighten it in place with the metal bolts.

5

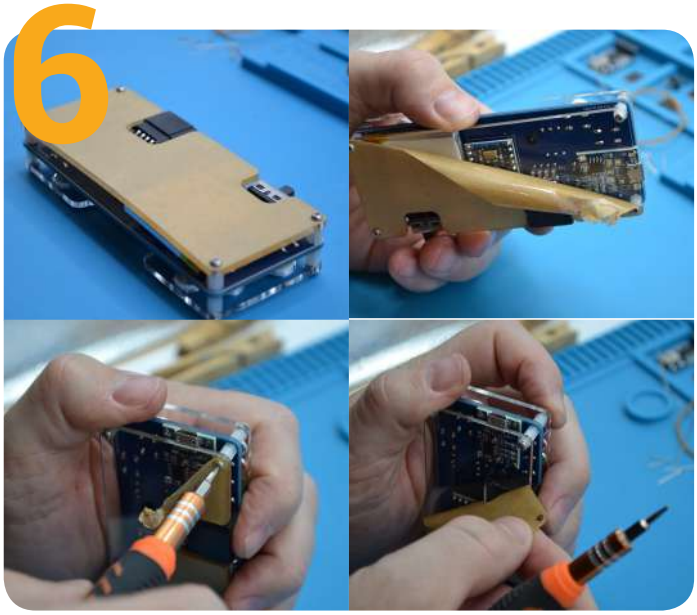


Push in the button caps on

Take the 7 white button caps and push them in place with your thumb. If, at first, it doesn't click in place, slightly rotate the button to make sure its properly aligned - then try again!



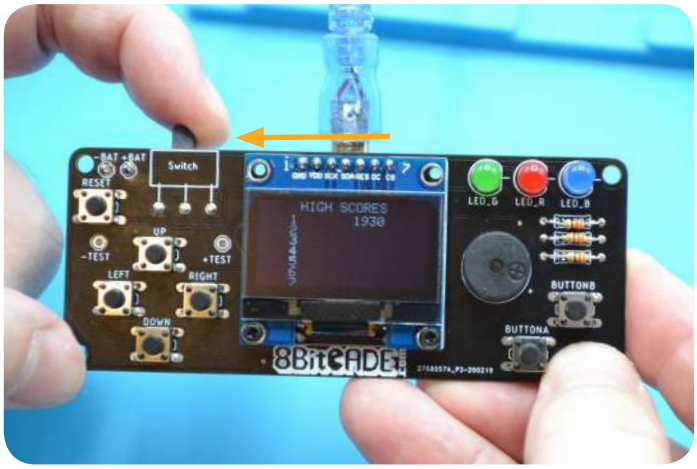
6



Removing the paper!

The last bit we need to do is to remove the paper from the back. To do this, peel it off like you did with the front panel.

However, you may find that some parts get stuck under the screw heads, to remove these parts, simply: loosen off the screw; peel off the paper; re-tighten the screw. Check out the photos for more info.

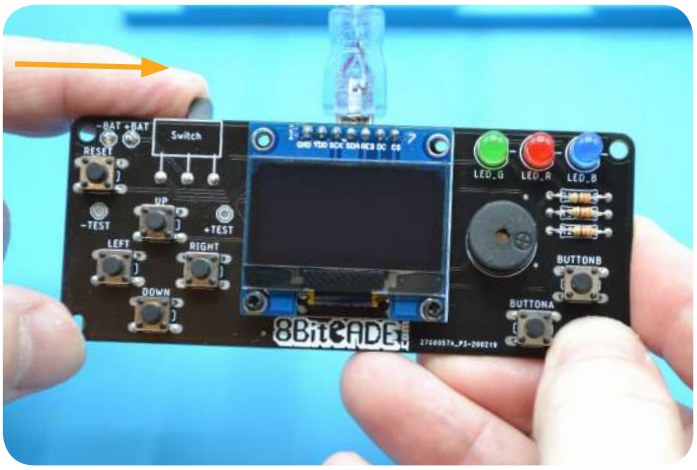


Switch to the LEFT

When the power switch on your 8BitCADE is pushed to the left, and is plugged in to your PC via your USB cable, the unit is on and is CHARGING.

Power Switch to the left
=
Console turned ON

USB plugged in + Power Switch to the left
=
ON & Charging



Switch to the RIGHT

When the console is plugged in to your PC and the power switch is to the right, it is ON and NOT charging.

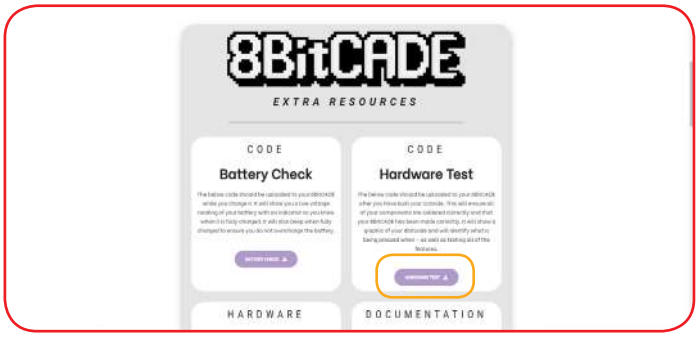
Power Switch to the right
=
Console turned OFF

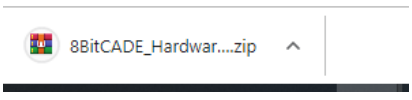
USB plugged in + Power Switch to the right
=
On & NOT Charging

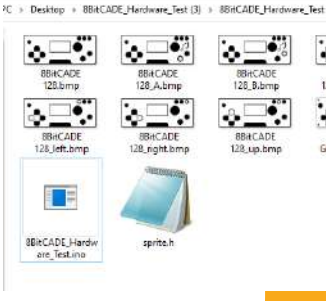
Charging Caution! When the console battery is low, it will turn itself off. From its low state, it would only need about an hour to charge to full capacity. The battery has both over and under charge protection circuitry - but it is not recommended leaving your battery on charge, unattended! When charging your battery you should use the Battery Check sketch at www.8bitcade.com/resources

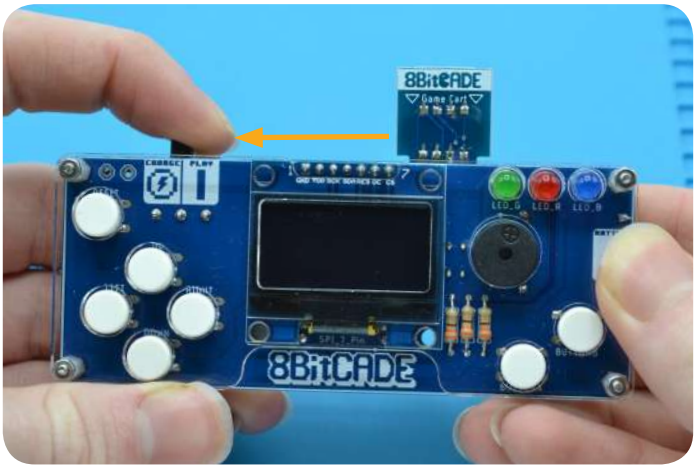
Peel & Play!

The first thing we need to do is plug in the console and complete the hardware test. This will make sure everything is assembled correctly. To do this, download the hardware test over at:
https://8bitcade.com/extra-resources/original
To test if your console is working as it should!







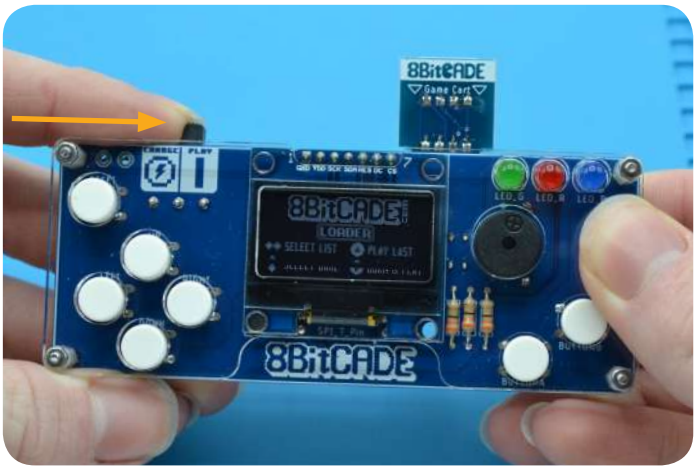


Switch to the LEFT

When the power switch on your 8BitCADE is pushed to the left, and is plugged in to your PC via your USB cable, the unit is on and is CHARGING.

Power Switch to the left
=
Console turned ON

USB plugged in + Power Switch to the left
=
ON & Charging



Switch to the RIGHT

When the console is plugged in to your PC and the power switch is to the right, it is ON and NOT charging.

Power Switch to the right
=
Console turned OFF

USB plugged in + Power Switch to the right
=
On & NOT Charging

Charging Caution! When the console battery is low, it will turn itself off. From its low state, it would only need about an hour to charge to full capacity. The battery has both over and under charge protection circuitry - but it is not recommended leaving your battery on charge, unattended! When charging your battery you should use the Battery Check sketch at ***www.8bitcade.com/resources***

Peel, program & Play!

If you have the flash cart, all you need to do is plug it into the Cartidge connector and turn your console off/on! Then the 8BitCADE XL loader will pop up (if it doesn't, plug the connector in the other way round). You then need to navigate to the "8BitCADE Maintenance" page (by clicking left once) and then, by clicking down twice, find the Hardware test. Go ahead and click button A to open the program! Once open, press the various buttons to test that everything is working as it should!

However, if you dont then the first thing we need to do is plug in the console to a computer and complete the hardware test. Turn to the next page to learn how to do this!

Install your first program!

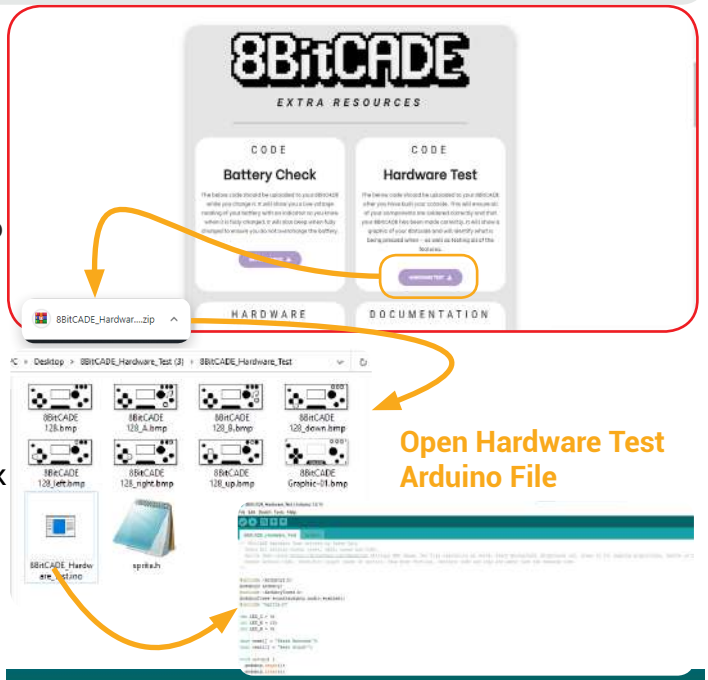
Lets do a 2 in 1, setup your console, test the hardware and show you how to install programs!

1. Install Arduino!

Watch this video here to install Arduino!:
https://www.youtube.com/watch?v=hFNmWMufm2I

2. Download the Hardware Test

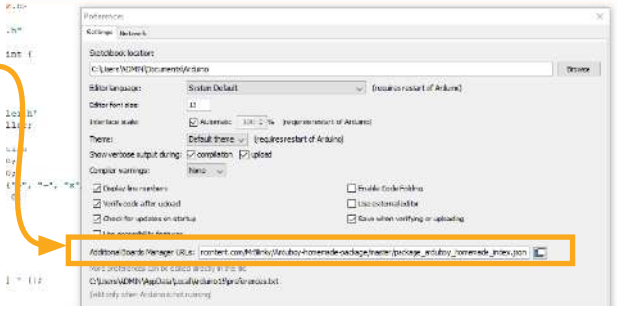
Go to **8bitcade.com/extra-resources/#original** to download and open the **8BitCADE_Hardware_test.ino**



3. Board/Library Install

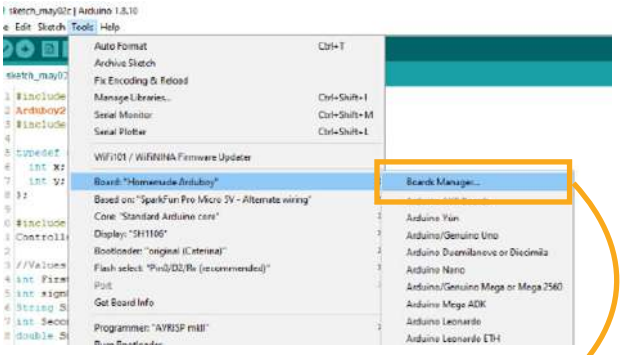
The first thing we need to do is open the Arduino file. We then need to include a specific library that will make programming your 8BitCADE much easier. The Arduboy2 Library. To begin this setup, we must first head on over to preferences in Arduino and add a link to allow us to access important board and library files.

Firstly, click “File” on the top left taskbar of your screen, then click “preferences”. A window like the one shown should appear. Where it says “Additional Boards Manager URLs” Click the icon
And type in, on a new line:
https://raw.githubusercontent.com/MrBlinky/Arduboy-homemade-package/master/package_arduboy_homemade_index.json

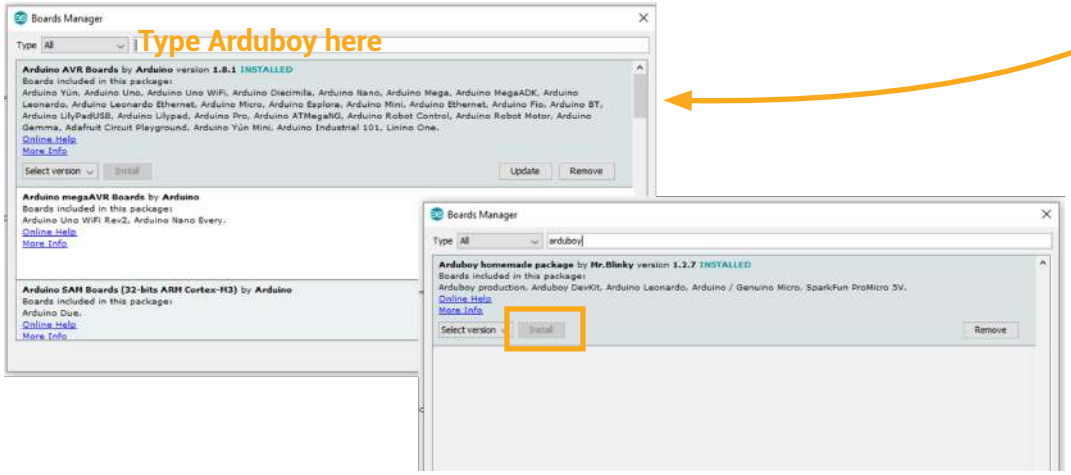


This will allow us to access the board and library information. Next, we need to install the board and all of its libraries. To do this, exit the current window and click “Tools” and select “Board: […]” > “Boards Manager”

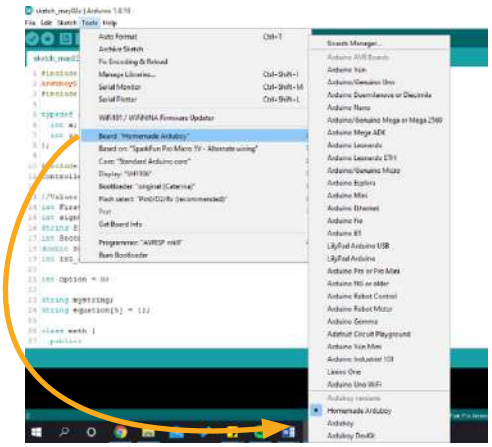
The below window should pop up (it will take some time as all your libraries are being checked/updated if need be).



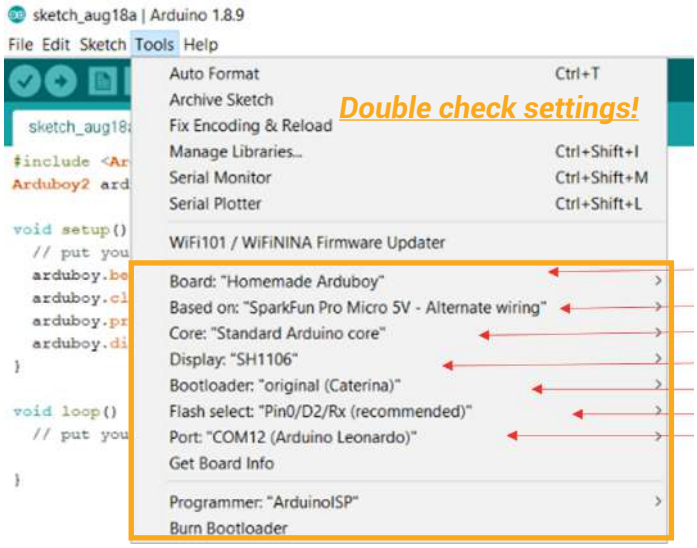
Next type in “Arduboy” and install the “Arduboy Homemade Package” By “Mr Blinky”.
Next, we need to head on over to select the board we just downloaded. To do this head over to the toolbar and click “Tools” and select “Board: […]” > “Home Made Arduboy”



The next step is important, and you should double-check your settings. Check and change your board values to be exactly like the photo below:

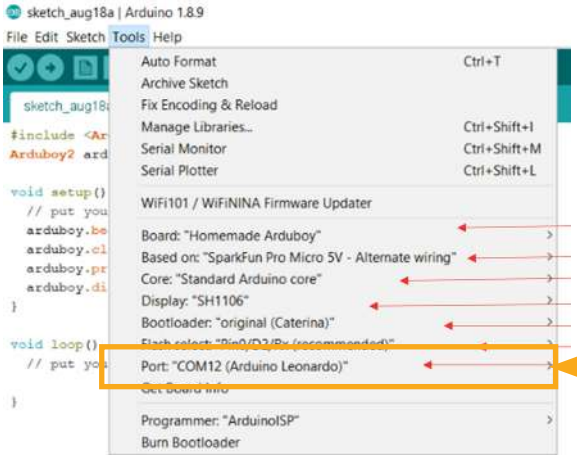


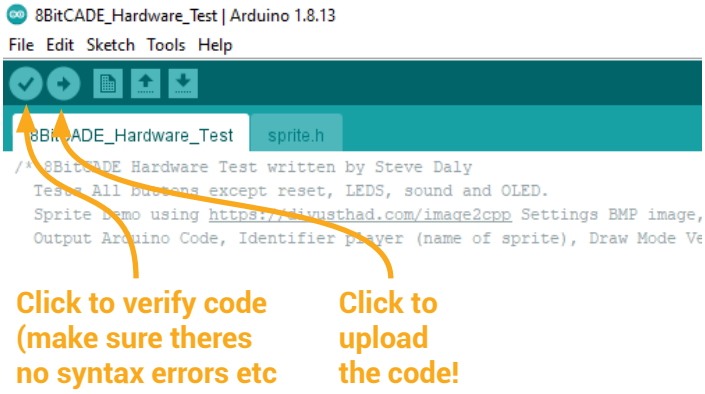
Select Homemade Arduboy Board



5. Click Upload

Go ahead and click the verify button to make sure everything is installed correctly etc and then click upload (while your 8bitcade original is installed) to load the program!





If you just plugged in your console, go back to the tools tab, click Port and click the COM port that says (Arduino Leonardo). Note it may not be the same COM12 (number) as shown here!

Where to find more games?

Two ways to install games, we showed you how to do it via the Arduino IDE but what about .HEX?

The first thing you should check out is the 8BitCADE Original Game section! This has everything you need to know about gaming on the console!

https://8bitcade.com/game/8bitcade/

Here you can find both the .ino files and .hex files. Theres also a cool video that explains the difference between these files! Same games, different files!

Video tutorial: How to use the 8BitCADE Activator to load a single .hex file game!
https://youtu.be/LUaoe1LzaGo?t=248

Troubleshoot:

A Lot of things can go wrong with a DIY product. Check out the 8bitcade.com/make section to find the TROUBLESHOOT section that will guide you through common issues you might have. If your problem doesn't seem to be covered/you tried the fix but it did not work, then do the following:

1. Firstly check your overall assembly to ensure that there are no short circuits, bridging soldering joints or damages from assembly.
2. Email support@8bitcade.com with your order number, issue and photos to help us understand what your problem is.
3. Sit tight and wait for us to respond ASAP!

Thank you kindly for your support and patience, we hope nothing goes wrong but if it does we are here to help you out!

8BitCADE
Make Learn Game

Rare achievement unlocked
8100 - Make your own 8BitCADE

Congratulations! Your First Console!

Phew - you've made it to the end. Take a moment to appreciate what ***YOU*** have built! You've brought out the ***MAKER*** in you, are you ready to bring out the ***GAMER*** in you?

Next Steps

Have you ever wanted to learn how to program your own game? Head on over to *8bitcade.com/learn*! Here you will find two sections:

1. Is the Foundation Tutorials. **Start here** as these cover how to set-up the Arduino IDE, your console and the basics of coding in Arduino - including some more advanced theories such as classes!
2. 8BitCADE Original Learn Page - Once your all setup, you can move onto the console specific learn page that includes a variety of project based coding tutorials such as:
   - Code your very own Tetris Game!
   - Code your first game "Pong"!
   - Code "Space invaders"!
   - Code a top down platform game called "Dino Smash"!

And so much more! so what are you waiting for, head on over to *8bitcade.com/learn* to take the next steps in learning to code!

Create, share & Join the community!

We, at 8BitCADE, love seeing what all you makers have achieved! Send in a photo of you and your new console and we will share it on our social media! Created a game? Send us a short video demo of you playing the game! Join the 8BitCADE community today!

Find us at *@8BitCADE* on instagram or *fb.com/8BitCADE* on Facebook!

You can also join the 8BitCADE Facebook group at: *www.facebook.com/groups/8bitcade*
Or pop onto the Community here if you have any questions: *community.8bitcade.com/*

How to use the 8BitCADE Memory Card

Before we jump into the Part Dictionary, lets go over how to use the optional Memory Card!

(The memory card can be purchased at www.8bitcade.com/shop/MemoryCard)

Introduction

8BitCADE memory expansion games cartridge has an onboard 8 pin serial flash chip with the capacity of 16mb and operating voltage of around 2.7 to 5V. It is solid state (doesn't use any mechanical parts like a disk/card) is non-volatile (when turned off, the data is still stored and not lost) memory storage device. It is used to store games and other programs which can be loaded using the 8BitCADE Activator App (kindly developed by Mr Blinky, a member of the Arduboy Community). The 8BitCADE Activator can load single HEX files (single games) or a Games Image (collection of games with a loader screen or 'splash screens') with a menu system to help you organise your games into selectable categories. Software is also available to allow users to create their own games image with customised menu graphics!

Because the hex files are so small, the memory expansion games cartridge can hold about 500 games on a single chip. The chip has already been programmed with 300+ games, but if you want to update it with additional games or programs then use the link below to access the 8BitCADE Games Image Creator.

Video Links to Important Apps:

- **8BitCADE Games Image Creator Video Link** (to create a bin file with multiple games and customised splash screens).
  Link: ***https://www.youtube.com/watch?v=SuQGrGcxehU&t***
- **8BitCADE Activator Video Link** (Used to load single hex files onto your 8BitCADE or a games image).
  Link: ***https://www.youtube.com/watch?v=LUaoe1LzaGo&t***

8 Pin Serial Flash Chip (16Mb) Part Code W25Q128FVIQ

The round circle on the 8 pin DIP (dual inline package) indicates pin 1 of the chip. Part of the schematic for the 8BitCADE Memory Games Cartridge is below showing the memory chip, capacitors and contact pads (to make contact with the expansion port).

CS: Chip Select,
MISO: Master In Slave Out,
WP: Write Protection Pin,
GND: Ground connection,
VCC: 3.7 – 5V,
HOLD: Hold Pin or Reset,
SCK: Serial Clock,
MOSI: Master Out Slave In

Lets go over the pins!

CS > Chip Select

The chip select pin enables and disables the memory chip operation. When the CS pin is held high the chip cannot send data via any of the data output pins (MOSI) and the devices power consumption will be at standby levels unless an internal erase, program or write status register cycle is in progress. When CS pin is held low the device will be enabled, power consumption will increase to active levels and instructions can be written to and data read from the device. A pull-up resister on the CS pin is used to pull high when not in use.

WP > Write Protect Pin
A hardware protection pin which can be used when held low to protect about 4kb worth of memory of the chip. This pin is permanently held high on the 8BitCADE as it is powered by VCC as seen in the schematic.

GND > Ground Connection.

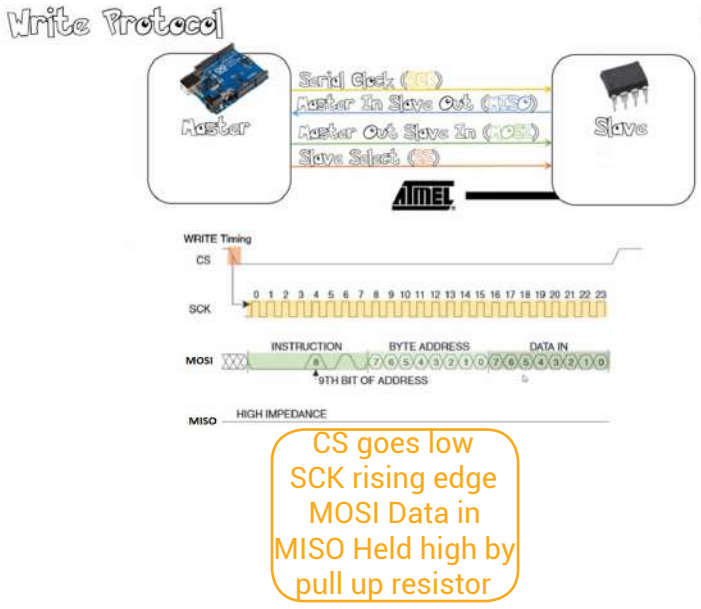
VCC > 2.7 – 5.0V supply voltage to power the chip.

HOLD - Hold pin or reset
The HOLD pin on the 8BitCADE is held permanently high by the VCC supply as seen in the schematic. The HOLD pin allows the device to be paused while it is actively selected. When HOLD is brought low, while CS is low, the MISO pin will be off and signals on the DI and CLK pins will be ignored. When HOLD is brought high, device operation can resume. The HOLD function can be useful when multiple devices are sharing the same SPI signals.

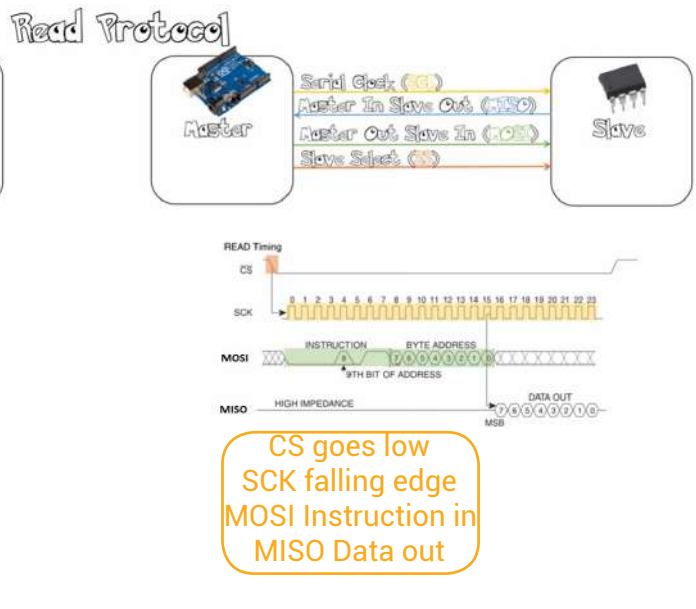
SCK – Serial Clock (with 10K pullup resistor)
Serial Clock pin receives a timed input signal from the Pro Micro (Master device) to provide timing for data read/write functions.

MOSI – Master Out Slave In (Data Input with Pullup Resistor) & MISO > Master In Slave Out (Data Output)
The W25Q128FV supports standard SPI (as used on the 8BitCADE), Dual SPI and Quad SPI operation. The 8BitCADE uses standard SPI instructions, using unidirectional DI (input via MOSI) pin to serially write instructions, addresses or data to the device on the rising edge of the Serial Clock (SCK) input pin. Standard SPI also uses the unidirectional DO (output via MISO) to read data or status from the device on the falling edge of (SCK).

Operation
The W25Q128FV is accessed through an SPI compatible bus consisting of four signals: Serial Clock (CLK), Chip Select (/CS), Serial Data Input (MOSI) and Serial Data Output (MISO). Once the chip select pin is held low, standard SPI instructions use the DI input pin to serially write instructions, addresses or data to the device on the rising edge of CLK signal. The DO output pin is used to read data or status from the device on the falling edge of CLK signal.



CS goes low
SCK rising edge
MOSI Data in
MISO Held high by pull up resistor



CS goes low
SCK falling edge
MOSI Instruction in
MISO Data out

Figure 1 and 2 are good examples of SPI communication but are NOT specific to W25Q128FV chip Further resources on SPI: **https://www.youtube.com/watch?v=AuhFr88mjt0**
Video on general Serial Communication: **https://www.youtube.com/watch?v=IyGwvGzrqp8**

WP > Write Protect Pin
A hardware protection pin which can be used when held low to protect about 4kb worth of memory of the chip. This pin is permanently held high on the 8BitCADE as it is powered by VCC as seen in the schematic.

GND > Ground Connection.

VCC > 2.7 – 5.0V supply voltage to power the chip.

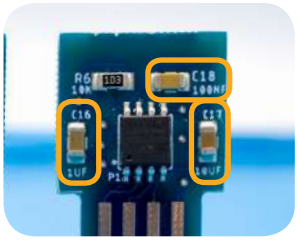
HOLD - Hold pin or reset
The HOLD pin on the 8BitCADE is held permanently high by the VCC supply as seen in the schematic. The HOLD pin allows the device to be paused while it is actively selected. When HOLD is brought low, while CS is low, the MISO pin will be off and signals on the DI and CLK pins will be ignored. When HOLD is brought high, device operation can resume. The HOLD function can be useful when multiple devices are sharing the same SPI signals.

SCK – Serial Clock (with 10K pullup resistor)
Serial Clock pin receives a timed input signal from the Pro Micro (Master device) to provide timing for data read/write functions.

MOSI – Master Out Slave In (Data Input with Pullup Resistor) & MISO > Master In Slave Out (Data Output)
The W25Q128FV supports standard SPI (as used on the 8BitCADE), Dual SPI and Quad SPI operation. The 8BitCADE uses standard SPI instructions, using unidirectional DI (input via MOSI) pin to serially write instructions, addresses or data to the device on the rising edge of the Serial Clock (SCK) input pin. Standard SPI also uses the unidirectional DO (output via MISO) to read data or status from the device on the falling edge of (SCK).

Operation
The W25Q128FV is accessed through an SPI compatible bus consisting of four signals: Serial Clock (CLK), Chip Select (/CS), Serial Data Input (MOSI) and Serial Data Output (MISO). Once the chip select pin is held low, standard SPI instructions use the DI input pin to serially write instructions, addresses or data to the device on the rising edge of CLK signal. The DO output pin is used to read data or status from the device on the falling edge of CLK signal.

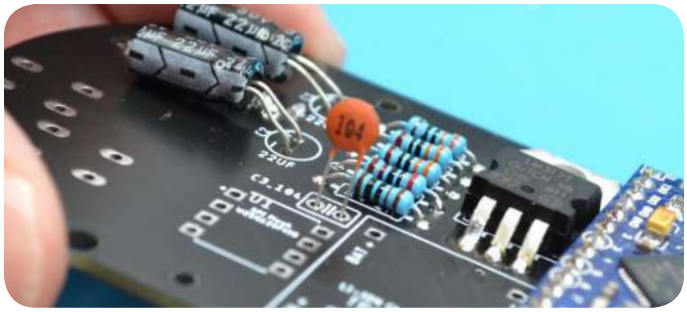
Electrolytic Capacitors for Decoupling
The 8BitCADE Memory Expansion Games Cartridge has three capacitors fitted across the VCC input (voltage supply) and ground. Electrolytic Capacitors are used for decoupling, reducing electrical interference which would otherwise stop the memory chip from reading or writing data.



What is a capacitor?
A capacitor is a two-terminal, electrical component which essentially stores electrical energy and releases it at a given time. Apart from energy storage, capacitors are also used for voltage spike suppression (often referred to as decoupling), and complex signal filtering (very good at filtering signals at certain frequencies). Decoupling capacitors suppress high-frequency noise in power supply signals. They take tiny voltage ripples, which could otherwise be harmful to delicate integrated circuit, out of the voltage supply. Not all capacitors are the same material or have the same use. Each capacitor is built to have a specific amount of capacitance. The capacitance of a capacitor tells you how much electrical energy it can store, more capacitance means more capacity to store this electrical energy. The standard unit of capacitance is called the farad, which is abbreviated F.

Electrolytic Capacitors (Excellent Signal Filtering)
The capacitors for the 8BitCADE are electrolytic. Electrolytic capacitors are usually polarized. They have a positive pin (anode) and a negative pin (cathode). When voltage is applied to an electrolytic capacitor, the anode must be at a higher voltage than the cathode. The cathode of an electrolytic capacitor is usually identified with a '-' marking, and a coloured strip on the case. The leg of the anode is often longer to help identify it.

Ceramic Capacitors (Small Capacitance & Fast Charge, Discharge Frequencies)



A popular capacitor is the ceramic capacitor, which is often rather small in size and value, its maximum is only about 10µF. Ceramic capacitors are very stable, but their small capacitance value can be very limiting. They are well-suited for high-frequency (fast) applications and are often seen on Arduino boards fitted to the resonator (16MHZ Clock)..

WARNING - If voltage is applied in reverse on an electrolytic cap, they'll break by popping open. After popping an electrolytic will behave like a short circuit and would damage the 8BitCADE circuit.

Without capacitors or with the incorrect value and type of capacitor the electrical interference created in the circuit would stop the microprocessor or Serial Flash chip from working correctly. Decoupling capacitors are often connected between the power source (5V, 3.3V, etc.) and ground. It's not unusual to use two or more of different values, as shown in the schematic, because some capacitor values will be better than others at filtering out certain frequencies of electrical noise. When physically placing capacitors in a circuit, they should always be located as close as possible to the integrated circuit, the further away they are, the less effective they will be.

Writing Games to the Memory Expansion Games Cartridge (Serial Flash Chip)

Games can be loaded on the 8BitCADE directly from the Memory Expansion Games Cartridge to the Pro Micro using the main 8BitCADE menu system to browse games categories, find games and then loading them by pressing the A Button. To return to the main menu you need to press the RESET button on the console. To load the last game loaded, press the B Button.

What are Hex Files (.hex)?

Hex files are a text file containing the hexadecimal codes of the binary images of the compiled Arduboy compatible games. These are created by opening the original code of the game in the Arduino IDE (a programming environment) and then exporting the game as a hex file. This is done to make the game very compact and fast loading, taking up little space on the games cartridge and reducing loading times.

What is a Games Image (.bin)?

A games image is a collection of games with a menu screen, categories and splash screens to help users organise, find and load games quickly. Like an in-game menu system!

How do you load single games?

To load new games that are not stored on the Memory Expansion Games Cartridge (hex files) on to your 8BitCADE you must use the 8BitCADE Activator. Here you will be able to select single Hex files to load onto your games console from a PC Computer.

Video tutorial link: How to use the 8BitCADE Activator to load a game in hex.
https://www.youtube.com/watch?v=LUaoe1LzaGo&t

How do you load a new games image?

To load a new games image (multiple hex files with the menu system) on to your 8BitCADE you must use the 8BitCADE Activator. Here you will be able to select single games image to load onto your games console from a PC Computer.

Video tutorial link: How to use the 8BitCADE Activator to load a games image.
https://youtu.be/LUaoe1LzaGo?t=248

How do you create a new games image?

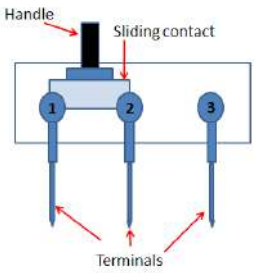
To create a games image you can use the 8BitCADE Image Creator. This will allow you to create your very own games image, menu screen, games categories, games and splash screens (title screens for the games). Basically, you can customise your own games image! The software uses python to create a .CSV file (games menu), .png files (game category and game splash screens) and allows you to add hex files. Once done, all the information is stored as a .bin file and loaded using the 8BiutCADE Activator. A video shows you exactly how to use the App to create a customised games image and load it onto your 8BitCADE.
Video tutorial link: How to create your own custom games image to load onto your 8BitCADE.
https://www.youtube.com/watch?v=SuQGrGcxehU&t=1s

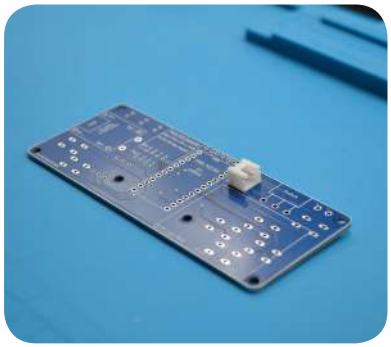
Part Dictionary

Below is an explanation of each part and why we use it on the 8BitCADE. This is a good reference sheet for all the hardware on your 8BitCADE. If you don't understand fully understand, be sure to re-read over the explanation, search online or check out the ***Arduino Community here.***

Right Angle Slide Switch

The slide switch has 3 pins, one will be powered by the lipo battery, the other will be open circuit, isolating power so the unit remains off and the last pin connects to the rest of the 8BitCADE circuit to provide power. The switch essentially acts as a break in the circuit. When you slide the switch, the circuit is closed and allows electricity to flow



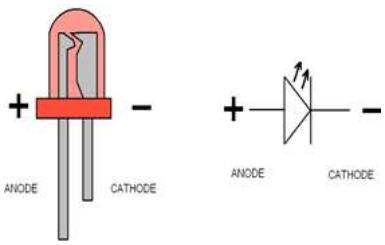


PCB Board

The PCB Board will hold all of the components together and link them up with copper tracks. These tracks are etched into a layer of copper and laminated between sheets of non conductive substrate. Through the use of copper pads and through hole joints, we solder the components in place. On your PCB, you can see white outlines, this is known as silk-screen. This is printed on the PCB last and is simply used to display text/outline on the PCB board.

Arduino Programming lead - USB Micro B Cable

Arduino use a special USB lead to connect the microprocessor board to the Arduino IDE Software for powering or programming. The Arduino Pro Micro Board uses a USB Micro B Cable. This cable will provide a steady 5V supply to your Pro Micro from your laptop/PC USB port as well as enabling you to transfer your program.



LED: Red, Green & Blue

LED stands for Light-Emitting Diode. A diode only allows electrical current to flow in one direction. An LED is a type of diode that emits light when it has current flowing through itself. It is commonly used for showing a user the state of the electronic circuit. For example to show that an electronic circuit is powered on, receiving data or sending it. Usually an LED needs about 2 volts and about 15-20 mA to operate it. But this varies among different types of LEDS. We have a 3.7V (lipo battery) or 5V (USB) power supply so we must use a resistor (330Ω) in series (in line) to reduce the current flowing through the LED, otherwise it will get damaged. Because the diode part in the name means it only allows current one way, the LED has different length legs to differentiate each terminal (positive and negative).

Part Dictionary

1.3 Inch OLED SPI Screen (SH1106 Driver)

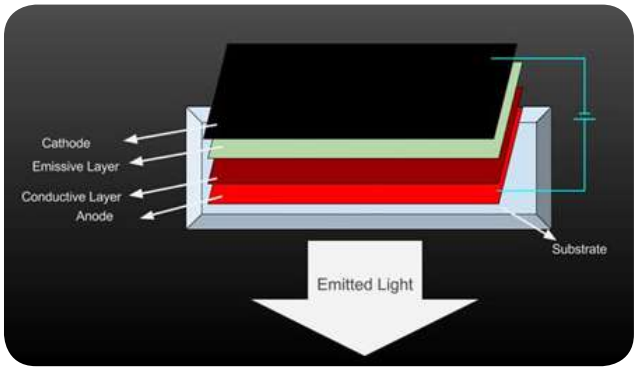
OLED stands for Organic Light emitting Diode, as it uses a thin carbon based material sandwiched between a positive and negative layer that passes electricity through it. This emits light in dots or pixels, in fact the OLED screen is 128 pixels wide by 64 pixels high. These pixels are turned on and off, individually to make images on the screen.
OLED displays are very small but have high resolution (lots of pixels per inch). These displays have no back light and makes their own light, that's why they are low power devices and ideal for portable game consoles.

At the heart of the OLED module is a powerful single-chip called an OLED driver controller and is named SH1106. It can communicate in serial (called a protocol) with the main 8BitCADE microcontroller in multiple ways, including I2C and SPI (communicate over a single wire for convenience or multiple wires for speed). SPI is generally faster than I2C but requires more input and output pins, so more wiring. While I2C requires only two pins (wires) and can be shared with other I2C components on a sort of network. It's a trade-off between pins and speed. The OLED SH1106 controller of the display has complex drivers. Vast knowledge on programming (memory addressing) is required in order to use the OLED SH1106 controller. Fortunately, Arduino uses library files to help us send basic instructions to operate such OLEDs. They were written to hide away the complexities of the SH1106 controller so that we can issue simple commands like draw graphics and display images. You will use these later on in the tutorials!

Operation of an OLED

When the voltage is applied to the OLED. The current flows from the cathode to anode , through the organic layers of the OLED. The cathode gives the electrons to the emissive layer of organic molecules and the anode removes electrons from the conductive layer of organic molecules.
At the boundary between the conductive and emissive layer, electron holes are created. These holes are filled by the electrons and the OLED emits light. The colour of the OLED depends upon the organic molecules used.

The OLED on the 8BitCADE has individual 128X64 white OLED pixels. It is 1.3 inch in size. The OLED's of other sizes are also available. The OLED used in this tutorial is monochrome (Only one colour) but you can also get OLED's having several colours. This OLED uses the SPI communication to communicate with Arduino. The SPI communication is very fast (much faster than I2C communication) so this will make our display faster, great for gaming where animations like characters can move quickly.

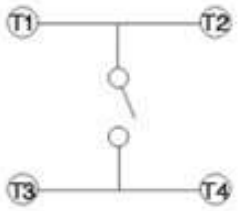


Overview

The maximum size of picture to be drawn on the OLED should be 128X64 pixels and the graphics should have a black background and white foreground because our OLED is monochrome which means that it has only one ability to draw or print in the colour white.

Part Dictionary

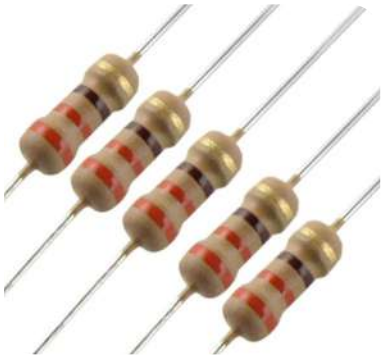
Tactile Switch - Button



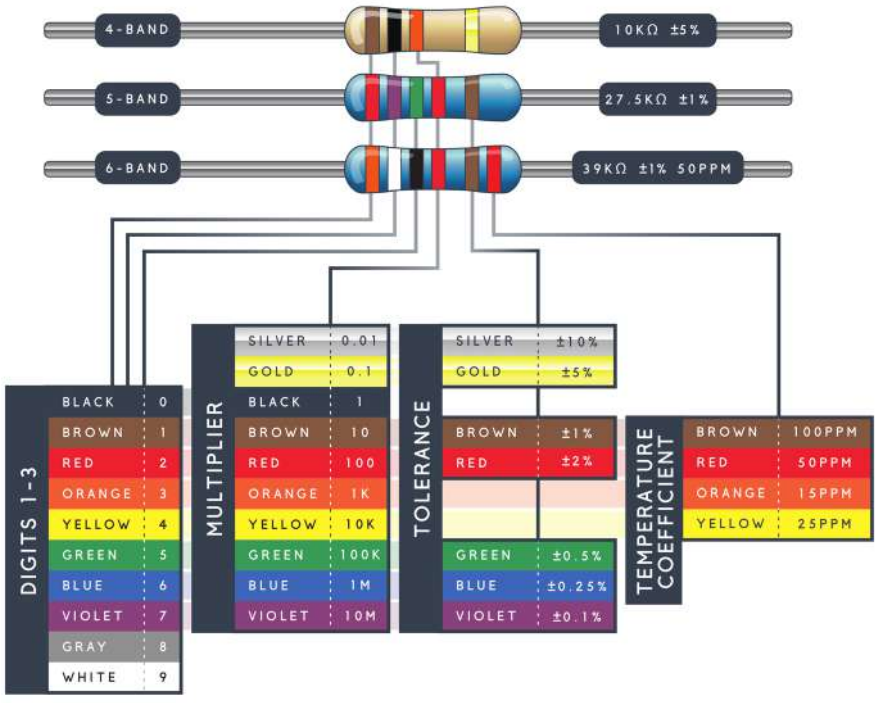
A button is a simple on-off switch. There are many kinds of buttons, distinguished by the mechanism used to close or open a circuit, but essentially all buttons belong to one of two families: those that keep the connection in either an open or a closed state, and those that return to their original (default) state once pressure is removed. A momentary button, as shown below, remains closed while pressure is applied to it, then returns to the open position once pressure is removed. This is an ideal button for a hand held game console. 6 buttons are used, Up, Down, Left Right, A and B.

This is an electrical symbol of the switch. As you can see that by pressing the button you are connecting two connections one side of the button to the other. T1 and T2 will be attached to the Pro Micro pins and T3 and T4 will be attached to GND (Ground).

330 Ohm Resistors



A resistor is an electrical component that limits or regulates the flow of electrical current in an electronic circuit. This is especially useful when you are limited by one voltage input, like 5V but you have components which require less! Resistors are usually added to circuits where they complement other components like leds! The electrical resistance of a resistor is measured in ohms. The symbol for an ohm is the greek capital-omega: Ω. Resistors come in a variety of shapes, sizes and resistance values. Resistors coloured to show what their resistance is, this is achieved using colour bands.



Resistor Chart

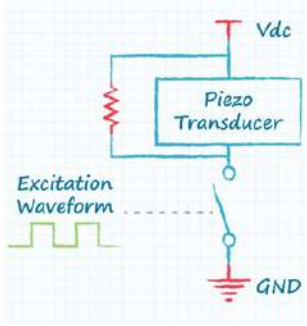
Using a standard four band resistor, the first two bands indicate the two most-significant digits of the resistor's value. The third band is a weight value, which multiplies the two significant digits by a power of ten. The final band indicates the tolerance of the resistor.
We use resistor tables to understand resistor codes, just like the one below.

Part Dictionary

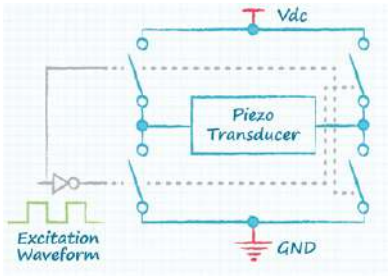
Piezo Buzzer

A piezoelectric buzzer is a loudspeaker that uses the piezoelectric effect for generating sound. The initial mechanical motion is created by applying a voltage to a piezoelectric material using a frequency modulated signal (fixed 50% duty cycle but with varying frequency). The motion is typically converted into audible sound by fixing the piezoelectric material to a thin disc and then applying electricity, we can bend the metal back and forth, which in turn creates the noise. The faster you bend the material (by applying a higher frequency signal), the higher the pitch of the noise that's produced. Using Arduino, you can make sounds by selecting a frequency to get a tone. When programming, you have to set which pins the piezoelectric buzzer is on, what frequency (in Hertz) you want, and how long (in milliseconds) you want it to keep making the tone.

The range of frequencies that can be produced depends on the microcontroller's clock frequency (Pro Micro is 16Mhz) and the timer which is being used to apply the frequency to the pin of the buzzer. The frequency to create notes varies from around 31 to 65535 Hz, but human hearing is constrained to sound frequencies below 20kHz.



Normally you connect one pin of the buzzer to the ground and the other pin to a square wave, frequency modulated signal output from (with timer function) the microcontroller.



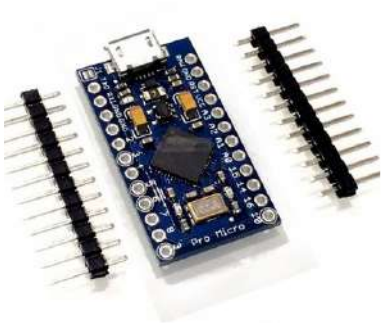
For increased volume on the 8BitCADE, we connect both pins to signal wires on the microcontroller (both require timed functions) and swap which pin is set high or low. This is called a differential drive, producing double the amplitude, gaining a higher volume.

Lithium Polymer Battery

LiPo batteries use a polymer electrolyte (electrically conducting material), instead of a liquid solution found in lithium-ion batteries. This is used to store the electrical power for us to use in our console.

The battery provides 3.7 volts and is rated at 500 mAh. This essentially means that if the battery was fully charged and the current draw of the 8BitCADE was about 50mA, perhaps during game play, it would take about 10 hours before it was fully drained. The reality of course, is that the current draw is much less and the game console would stop working due to low voltage regulation at about 2.7V.

Part Dictionary



The Brains of our 8BitCADE- Arduino Pro Micro with ATmega32U4 5V 16MHz

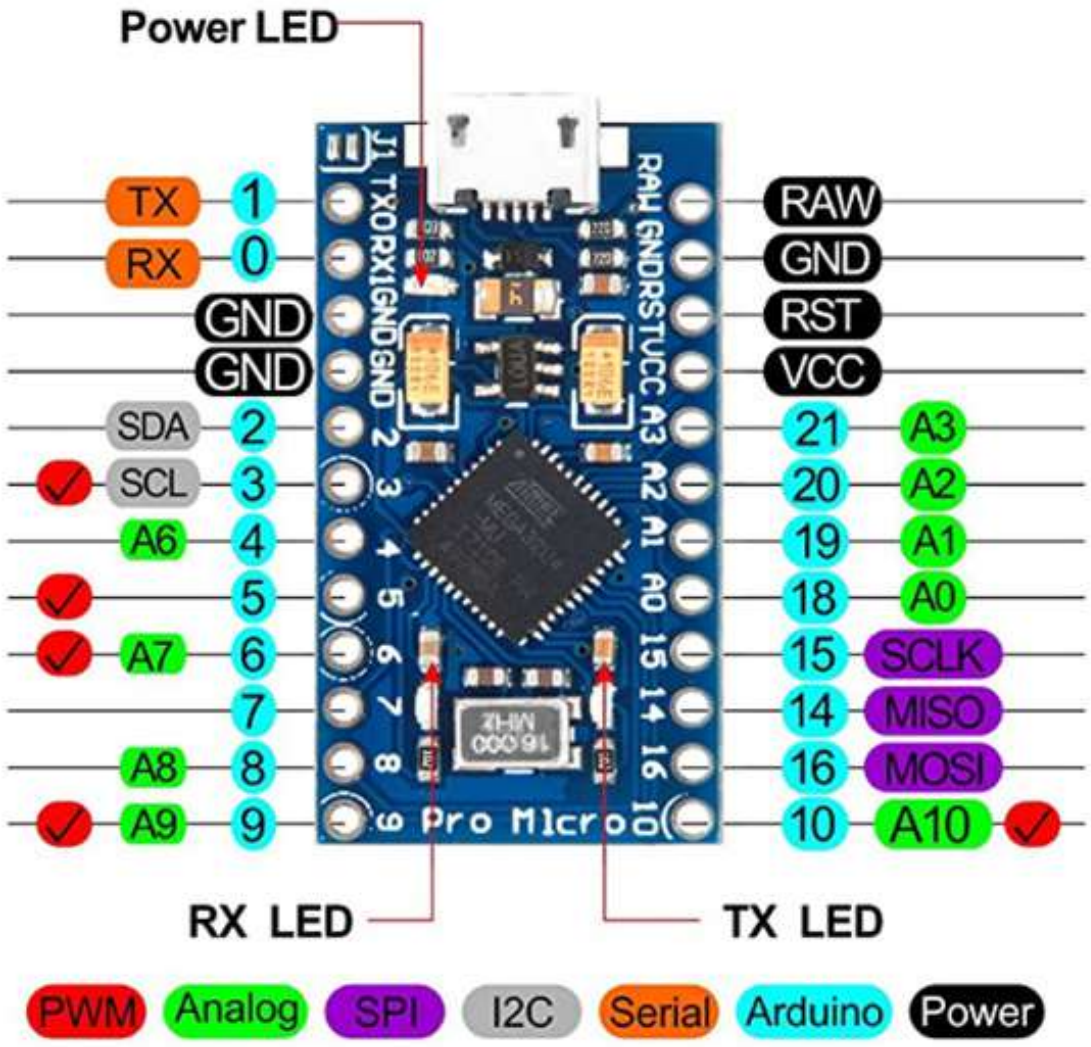
The Arduino Pro Micro is a development board which contains a microchip called an ATmega 32U4, seen below as a large black square, with small pins and the name Atmel, the original manufacturers of the chip before the company was bought by Microchip Technology.

The development board holds the chip and supports it by giving it power, breaking out the pins of the chip into numbered connections, programming the chip via usb connection and adding LEDS so you know when power is connected or when the chip is being programmed by flashing leds labeled TX (transmit) & RX (receive). The Arduino Pro Micro Development Board has 18 input/outputs, often called Input/output pins. Every pin can be used as a digital input or output. Nine of these pins have analog to digital converters (ADCs) and can be used as analog inputs (A0, A1, A2, A3, A4, A5, A6, A7, A10). These pins are useful for reading analog devices.

ATmega32u4 Microchip main features

- The ATmega32u4 has built-in USB communication, eliminating the need for a secondary microchip to be added to the board to assist in programming it (converting USB communication into serial).
- The 32U4 chip is low-power.
- The 32U4 chip is an 8-bit micro-controller featuring 32KB of flash program memory, 2.5KB SRAM, 1KB EEPROM.
- Provides 18 input/output pins which 9 can be used for connecting with analogue devices and 5 are pulse width modulated.
- Provides I2C & SPI communication
- Provides voltage regulation to power and protect the board
- Has a power led
- Has programming leds which flash when the 32u4 chip is being programmed
- Can be powered using a 3.7v lipo battery.

Part Dictionary



What do these pins do on the 8BitCADE

RAW Pin:

This is the pin which can be normally used to supply power from 5V to 12V input voltage. This voltage will then be regulated by the on board voltage regulator and applied to the ATmega32U4 microchip.

USB Connection:

If the board is powered via USB connection, the voltage at this pin will be about 4.8V (USB's 5V minus a small voltage drop). VCC is the voltage supplied to the on-board ATmega32U4 microchip.

VCC:

If used as an input voltage then no more than 5V must be supplied. This pin can be used as an output pin if voltage is supplied to the RAW pin and can be used to power other components as long as the maximum of 200ma is not exceeded. 8BitCADE uses this pin to power the development board and ATmega32U4 microchip using power from the lipo battery.

GND

GND is the common, ground voltage (0V reference) for the system.

Part Dictionary

RST

RST (Reset) can be used to restart the Pro Micro (restart existing programme). This pin is pulled high by a 10kΩ resistor which is on the board and must be connected to ground to initiate a reset. The Pro Micro will remain “off” until the reset line is pulled back to high.

Input/Output Pins:

The Pro Micro has 18 input/output or I/O pins. Every pin can be used as a digital input or output. Nine of these pins have analog to digital converters (ADCs) and can be used as analog inputs (A0, A1, A2, A3, A4, A5, A6, A7, A10). These pins are useful for reading analog devices.

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 Pro Micro, and a green LED indicates USB data going out (TX).

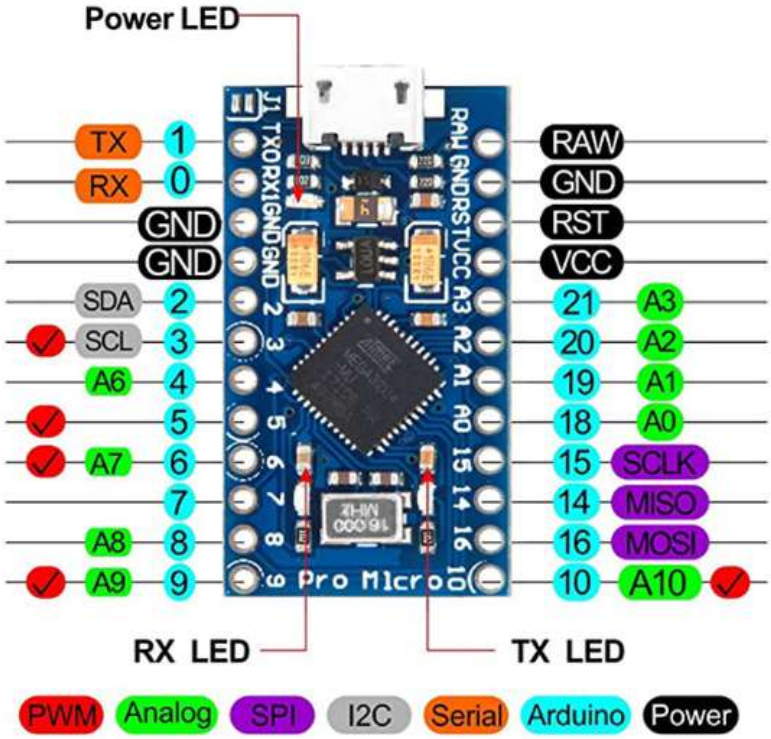
Pro Micro Serial Communication

How does the Pro Micro controller communicate internally and externally with hardware (other microchips, sensors) and your computer?
We need to understand the following:

1. Analog communication (read information from analogue sensors to measure something)
2. Digital communication
    - PWM Communication (Pulse Width Modulation Pins 3,5,6,9,10)
    - USB communication (Universal Serial Bus connected via USB Connector)
    - Serial Communication (Asynchronous Serial Pins TX & RX)
    - SPI Communication (4 wire Synchronous Serial Pins SCLK,MISO,MOSI,CS)
    - I2C Communication (2 wire Synchronous Serial Pins SDA, SCL)

Part Dictionary

Analog (green) and Digital Pins (all the rest)



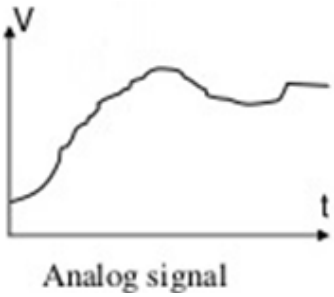
The Pro Micro has many individual pins (coloured blue 1-21) but these pins can be divided into 2 categories, digital and analog. Analog is the green pins and digital are all the other pins on the board. The Pro Micro can receive input information in both digital and analog signals but can only output digital signals.

When a microcontroller like the Pro Micro is powered and receives information from a digital pin (0,1,2,3,5,7,18,15,16) the information will be generally in one or two forms, either 5V or 0V, it will understand zero volts (0V) as a binary 0 and a five volts (5V) as a binary 1.

An analog pin (A0, - A9) might receive a signal value of 2.50V. Is that a zero or a one? We often need to measure signals that vary in voltage; these signals are called analog signals and come from analog sensors. A 5V analog sensor may output a voltage between 0.01V to 4.99V.

Analog Communication Signals

An analog input signal being received by the microcontroller can take the form of any number of values from just below the supply voltage, which could be 4.9 (from a 5V supply) to 0.1 volts (just above 0 volts level) and is generally measured over time. They are generally used to measure some sort of quantity like temperature, pressure or level of light and return a voltage level which represents the value being measured.

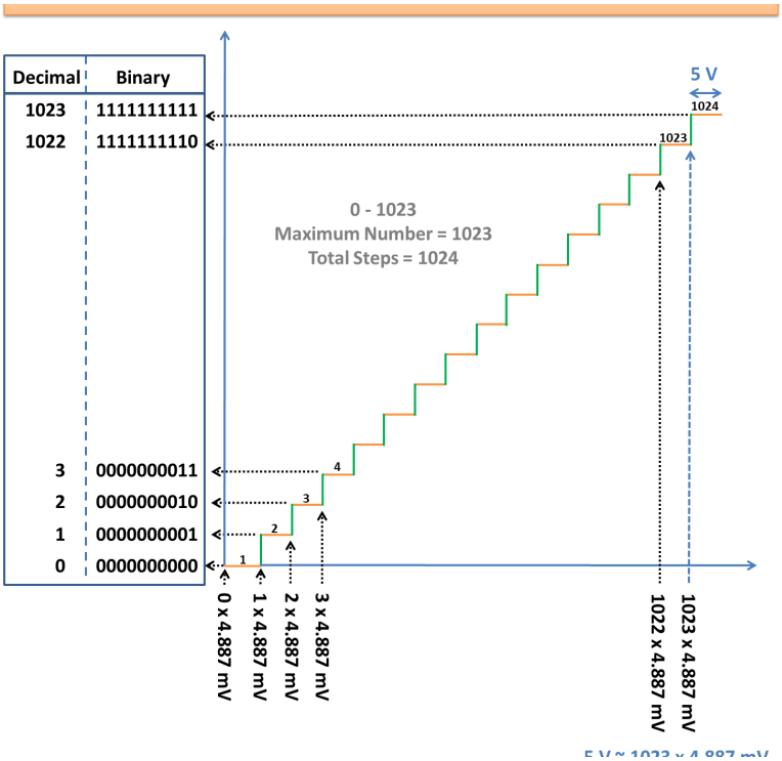


An analog waveform (V= voltage t= Time)

To enable your Pro Micro to understand this information it needs to be in a digital format. Arduino's have many built-in analog-to-digital converters which are called ADC's. These measure the value of analog signal and convert it into a binary number. The analog-to-digital converter changes the analog voltage into a digital value to enable the microprocessor to read it, as microcontrollers communicate using digital forms like binary and hexadecimal and not voltages.

This function converts the value of an analog input pins voltage and returns a digital value from 0 to 1023 which is later changed to a binary number. So when an analogue value is read by your microcontroller it returns a number, not a voltage value but essentially this value represents the voltage which has been converted.

Part Dictionary



The voltage in the graph varies between 0V and 5V. The built in ADC will convert this into a value from 0 -1023. This gives us 5000 (mV) / 1023 = 4.887 mV/per step which is equal to one increment of 1023 and is represented as 0000000001 in binary.
So if each step is worth 4.887 (mV) then what would the voltage be if 511 steps were returned as a digital value?

4.887 mV/step x 511 steps = 2,497 (mV) or 2.5V.

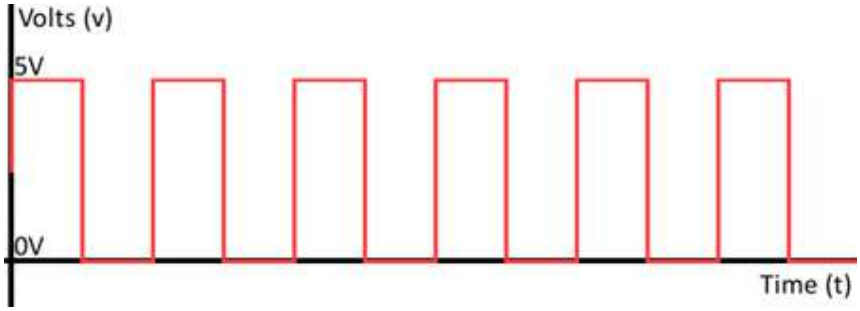
Binary 0111111111

https://www.binary-code.org/binary/10bit/0000011111/

The graph and table show the relationship between voltage, the converted ADC (10bit) value 0f 0-1023 and the binary equivalent.

Simple Digital Signal

A digital signal, on the other hand, has only two values: HIGH which is maximum voltage (5V) and LOW (0V). This will return a value of either binary 0 (low or 0V) or binary 1 (high or 5V).



These signals are typically used for input sensor like a switch. High being 'switched on' and low being 'switched off' or an LED (light-emitting diode), high being on and low being off.

Review:

From the signal we have seen so far, we know that the microcontroller can:

- receive analog signals of varying voltages (0.1-to 4.9V) and convert them into a digital format using an Analog-to-Digital Converter (ADC) which converts the supply voltage into 1023 steps (because it has 10-bit resolution, 1111111111 = 1023).
- receive digital signals which are either full voltage or zero voltage (on or off).
- send digital signals which are either on or off.

Key question: How do you send a varying voltage? Perhaps to dim a light or change the speed of a motor?

The answer is by using a pulse width modulated signal (PWM) via pins 3,5,6,9,10. This method of digital communication will provide an average voltage which can be varied from a low voltage to the maximum supply voltage (0-5V)

Part Dictionary

PWM Communication

(Pulse Width Modulation Pins 3,5,6,9,10) We know that we can turn a signal on a microcontrollers output pin high sending the maximum supply voltage out of the pin, for example, 5V. We can also turn it low which is off. But if we switch the signal on and off very quickly, about 500 times per second, and we vary the amount of time the signal is high, compared to how long the signal is low we can vary the voltage output.

Key point: Essentially we can change the proportion of time the signal is high compared to when it is low over a consistent time interval. This is referred to as a Duty Cycle Ratio.

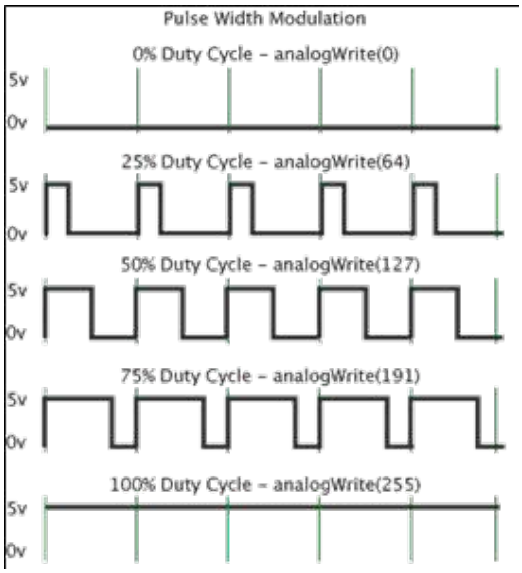
If a digital signal spends half of the time on and the other half off, we would say the digital signal has a duty cycle of 50% and resembles an ideal square wave. If the percentage is higher than 50%, the digital signal spends more time in the high state than the low state.

PWM has several uses:

- Providing an analog output between 0-5V.
- Dimming an LED
- Moving servo motors
- Generating audio signals.
- Providing variable speed control for motors.

The frequency of the PWM signal is approximately 490 Hz on pins 5,6,9,10 and 980 HZ on pins 3.

100% duty cycle would be the same as setting the voltage to 5 Volts (high). 0% duty cycle would be the same as grounding the signal.

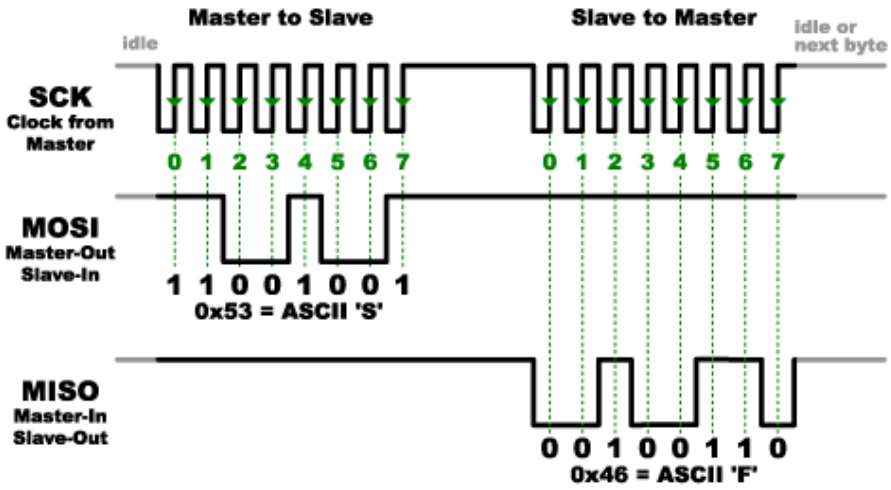


Complex Digital Communication Signals

Most communication between integrated circuits is digital. The main interfaces used within an Arduino are:

1. Parallel (lots of wires sending data)
2. Serial (Few wires and are categorised as Synchronous & Asynchronous)

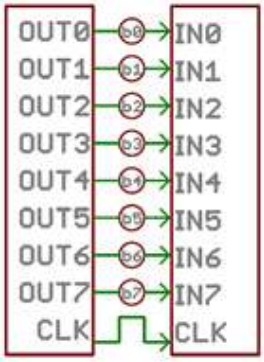
All of the types transmit data via a **coded sequence of square waves** to transfer information, lots



Some systems have many wires and transmit information very quickly but if speed is not as important, some use only 2 wires.

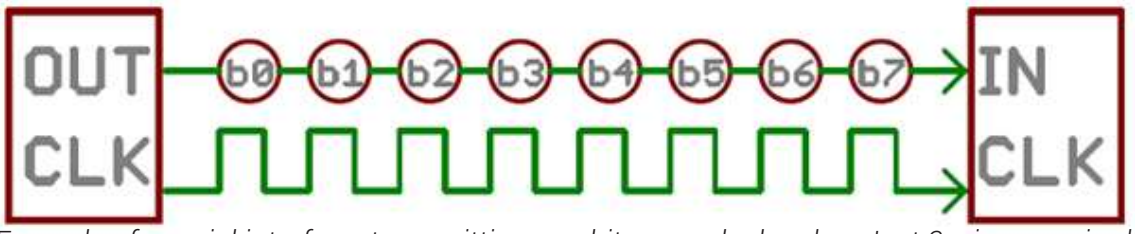
Part Dictionary

Parallel Vs Serial Communication



Parallel interfaces transfer multiple bits of data, through multiple wires at the same time. They usually require 'buses' of data – information transmitting across grouped wires of eight, sixteen, or more. As stated, data is transferred in 1's and 0's.

The example shows an 8-bit data bus, controlled by a clock, transmitting a byte every clock pulse. 9 wires are used. Very fast data can be transferred but more expensive and time-consuming to wire-up.



Example of a serial interface, transmitting one bit every clock pulse. Just 2 wires required!

Serial interfaces send their data, one single bit at a time. These interfaces can operate on as little as one wire, usually never more than four. Parallel communication certainly has its benefits. It is fast and easy to implement but requires much more input/output (I/O) lines.

Synchronous & Asynchronous Serial

The synchronous serial interface always pairs its data line(s) with a clock signal, so all devices on a synchronous serial bus share a common clock signal (for signal timing). This helps for faster serial data transfer but requires at least one extra wire between communicating devices, due to the addition of the clock signal.

Where does the clock signal come from for Synchronous data signals?

The clock signal comes from the on board 16.000 MHZ crystal oscillator or ceramic resonator. Both have the same function but work slightly differently.

The Pro Micro has an on board external 16Mhz ceramic resonator which is used to provide a time input to the main Pro Micro microprocessor, it essentially acts like a metronome, ticking at a constant interval and is used as a reference to keep communication signals in time.

Asynchronous means that data is transferred without support from an external clock signal. This means that both hardware devices which want to communicate must be set to the same data communication speed. This is called baud rate. The baud rate specifies how fast data is sent over a serial line. This value determines how long the hardware which is transmitting and receiving data for. The only requirement is that both devices to operate at the same speed. One of the more common baud rates is 9600 bps. When using Arduino IDE software on your PC and the 8ButCADE is plugged in via the USB connector, you can send data directly to it (Pro Micro) via the Serial Monitor. Because the Serial monitor is communicating via the USB port, you must set the baud rate to 9600 on your computer (via the Arduino IDE) and set the baud rate in your sketch (programme) which you load on to your 8BitCADE game console. This will synchronise data exchange without the need of a clock signal.

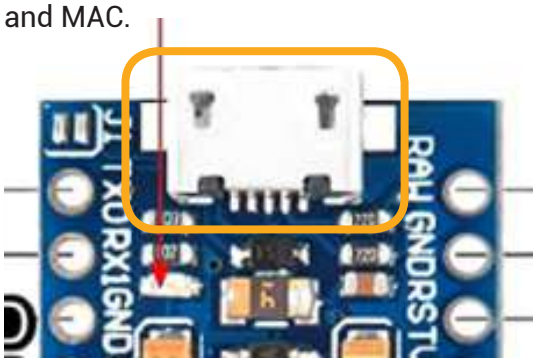
Part Dictionary

Types of Serial Protocols used on the Pro Micro

1. USB communication (Universal Serial Bus connected via USB Connector)
2. PWM Communication (Pulse Width Modulation Pins 3,5,6,9,10)
3. Serial Communication (Asynchronous Serial Pins TX & RX)
4. SPI Communication (4 wire Synchronous Serial Pins SCLK,MISO,MOSI,CS)
5. I2C Communication (2 wire Synchronous Serial Pins SDA, SCL)

USB communication (Universal Serial Bus connected via USB Connector)

The USB port your Pro Micro can be connected to the USB port of your computer, provided that the USB driver is installed (included in Arduino IDE installation). The driver causes the USB device to appear as an additional COM port on your computer. This allows the Arduino IDE software to access the Pro Micro's main microprocessor to load sketches and run programmes in the same way as it would access a standard COM port. The USB port is, therefore, more suitable for user applications running on Microsoft Windows operating systems and MAC.



Essentially this is a bridge between two different types of serial communication. In the old days' computers would have a serial connector and would have been able to connect directly to you Pro Micro via a serial port. Because technology has moved on quickly and we now have USB connectivity, so we now use a USB serial interface, essentially allowing one type of serial (on your computer) to communicate with another type (on the 8BitCADE).

Bridging the gap - USB Serial > Arduino Serial

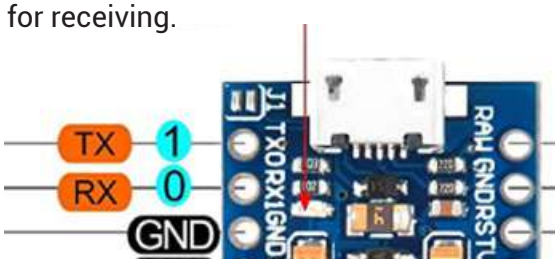
When microcontrollers and other low-level ICs communicate serially, they usually do so at a TTL (transistor-transistor logic) level. TTL serial signals must operate between a microcontroller's voltage supply range - usually 0V to 3.3V or 5V. A computers USB connection has different voltage levels as well as other connection differences which create 'a gap in serial communication'. To bridge this gap Arduino generally use one of two options, add microchip to interface with the USB serial (FTDI Chip) and TTL serial for the main microcontroller or to use a UART processor or circuitry (can be inside the main microprocessor) to interface with the USB connection.

IMPORTANT NOTE: When connecting two serial devices, it's important to make sure their signal voltages match up.

The Pro Micro main microprocessor (ATmega32u4) includes a USB-to-UART bridge, eliminating the need for a secondary microprocessor. This allows the Pro Micro to appear to a connected computer as a mouse and keyboard (UART1), in addition to a virtual (CDC) serial / COM port (UART2).

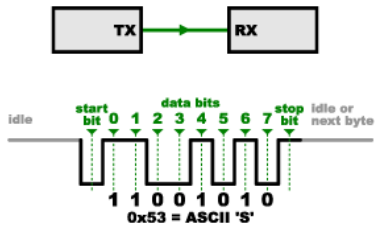
Serial Communication (Asynchronous Serial Pins TX & RX)

An asynchronous serial bus consists of only two wires - one for sending data and another for receiving.

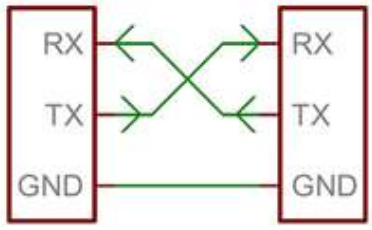


These wires are used to exchange data but because it does not make use of the clock signal from the on board ceramic resonator, there is no control over when data is sent or any guarantee that both microcontrollers are running at precisely the same speed (baud rate).

Part Dictionary



To work around this problem, asynchronous serial connections add extra information in their data to help the receiver synchronise the data as it arrives. Both microcontrollers must also agree on the transmission speed (such as 9600 bits per second) in advance.



Both serial devices should have two serial pins, one is the receiver and named, RX, and the other is the transmitter, TX. On the Pro Micro, both TX & RX pins are also connected to LEDs so that when serial communication is occurring, for example, when loading a programme (sketch) onto the main microprocessor via a computer with the USB connection, the LED's will flash to show that data is being received or transmitted.

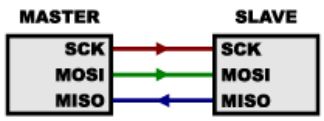
It's important to note that those RX and TX labels are concerning the device itself. So the RX from one device should go to the TX of the other,

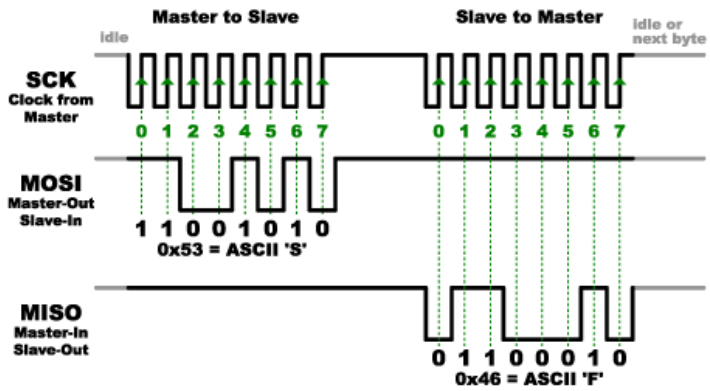
Overview:

1. Simple layout, easy to connect to.
2. Lines must have the same transmission speed set-up (baud rate) otherwise it will not work.
3. Data bits are lost as it has additional information to help synchronise it between the transmitter and receiver (10 bits of transmission time are required for every 8 bits of data sent).

SPI Communication (4 wire Synchronous Serial Pins SCLK,MISO,MOSI,CS)

The synchronous serial bus consists of at least 4 wires - one for sending data, one for receiving it, a clock signal to synchronise the data transmission and a select pin to indicate when the data should be read. Serial Peripheral Interface (SPI) is an interface bus commonly used to send data between microcontrollers and smaller devices such as memory integrated chips and OLED's (8BitCADE screen). It is very reliable as it is highly synchronised using the clock for signal timing, separate data lines and the select line to control when signals are read by each device. Because the timing signal comes from one device, this device is referred to as the master and the other devices are slaves. SPI is fast, it can send and receive data at the same time so is very good for controlling hardware used in gaming with fast-moving graphics.





SCLK- Serial Clock, comes from the ceramic resonator and is used for data transmission timing.

MISO- Master In Slave Out, receives data on this line

MOSI- Master Out Slave In, sends data on this line

CS- Control Select or Slave Select, this controls when data is read which is normally activated by sending a low signal to the line (as it is normally held high).

Part Dictionary

Overview:

1. Fast transmission rate
2. Highly synchronised data exchange.
3. Lots of wiring or connections, especially if you have multiple slaves!
4. Easy to programme using commands in libraries in Arduino.
5. Slaves can't communicate directly with each other, all information must go through the Master.

I2C Communication (2 wire Synchronous Serial Pins SDA, SCL)

The Inter-Integrated Circuit (I2C) protocol was created to enable multiple "slave" digital integrated circuits to communicate with one or more other chips irrespective of whether they are master or slave. The main problem with SPI is that you have multiple slave IC's having to communicate with a master (the hardware with the clock signal) instead of communicating with each other. You also have increased electrical connections with the addition of CS/SS pins to select the hardware to read the data. But of course, the data is processed very quickly.

Important: When the data transmission rate is not important and needs to be easily shared within a network of many integrated circuits, I2C become highly effective.

I2C bus consists of two signals: SCL which is the clock signal, this is generated by the master (Pro Micro in this case) and SDA which is the data line. All information is shared on the 2-wire network. Resistors are used to hold the signal lines high (5V). To start communication on the line the master will pull the SDA line low, advising all devices that transmission of data is now possible. Data is sent using an address which corresponds to either one of the slaves or the master device. The device with the address will read the data while all other devices will ignore it. Devices on the 2-wire network send, receive and ignore data, making it very easy to add hardware and remove it with little addition of wiring or programming.

Important: Voltages on the network should ideally be the same – either 5V or 3.3V, if they are not, the signal voltage level will need shifting (an IC to help match the voltage levels) must be used to bring them in line, this is called level shifting.