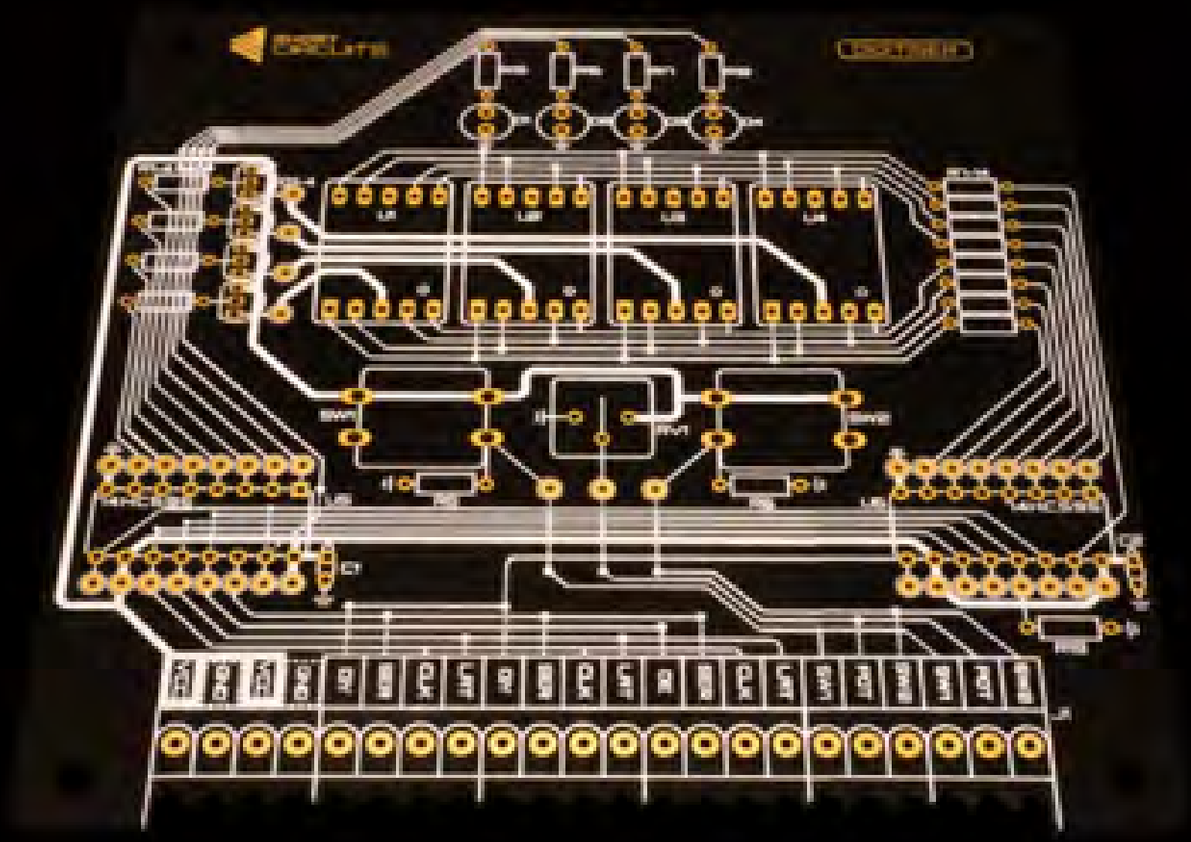
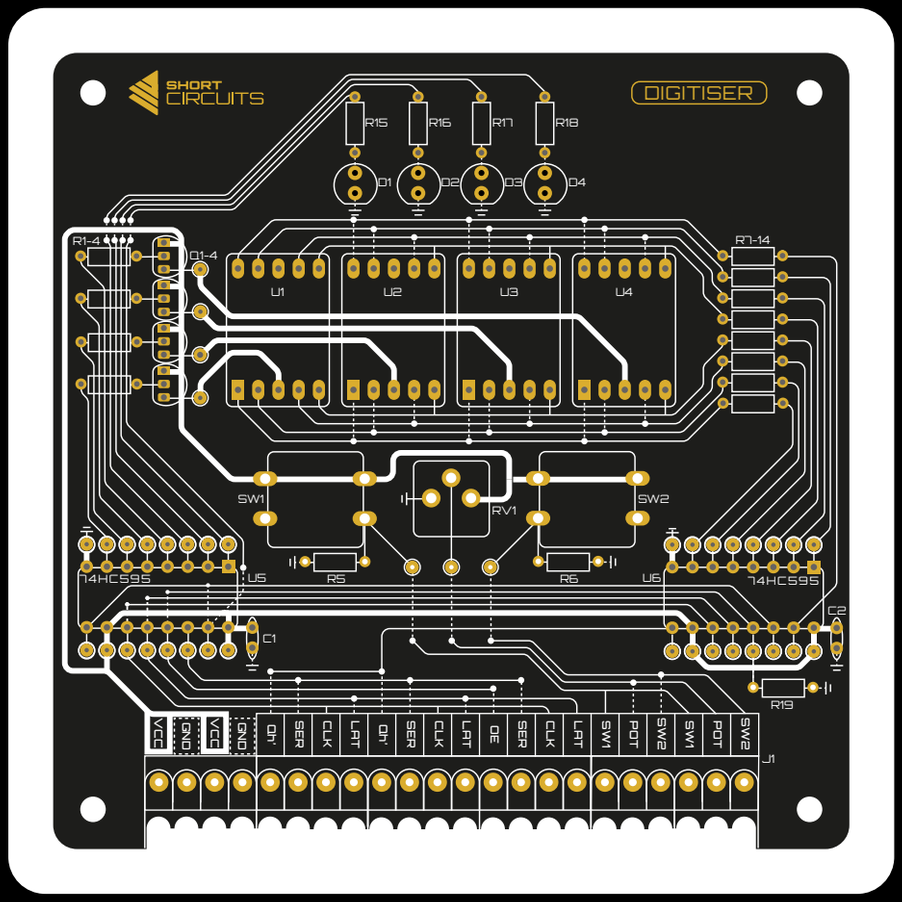
SHORT CIRCUITS

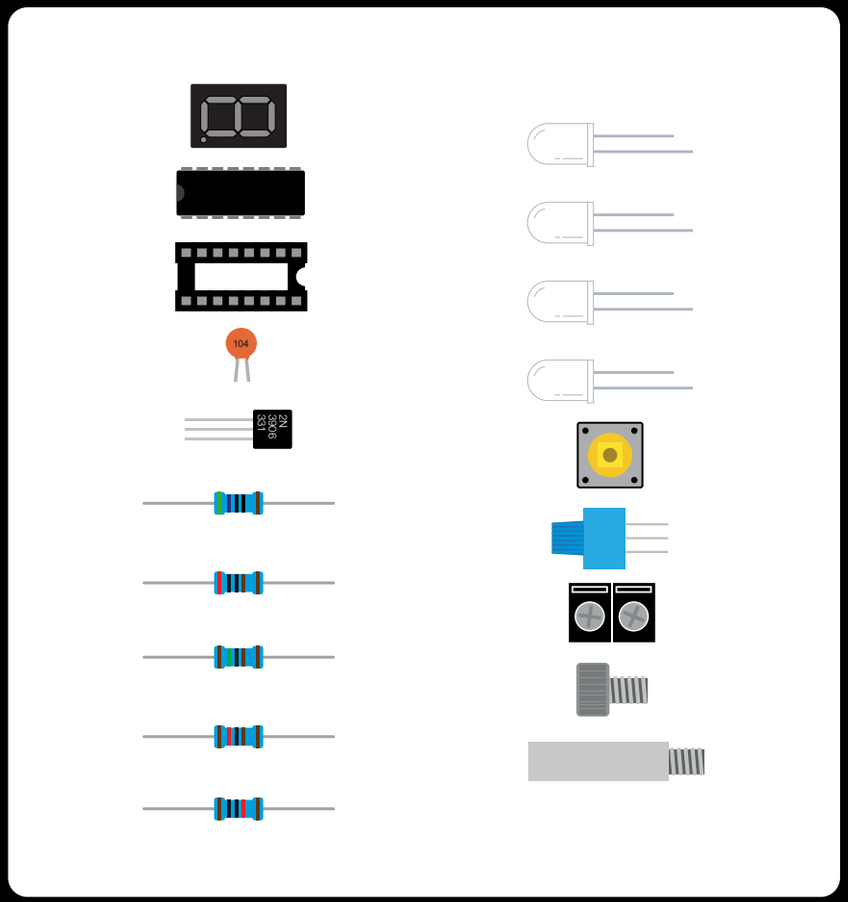


DIGITISER

INSTRUCTION MANUAL

The Kit





Printed Circuit Board

The Printed Circuit Board (PCB) is made of resin and fiberglass, with a layer of copper on the top and bottom. This is a 2 layer board, but boards with more layers are common. We can use PCB design software like the free KiCAD, or Eagle, Altium etc to lay out our circuit. The circuit is comprised of footprints that hold the various components, and traces which are copper connections between certain components.

When the finished design is sent to the manufacturer, they etch away the copper that isn't needed to form the circuit. The board is then coated in a coloured layer that protects the circuit. This is the mask. A different colour is then applied. This is the silkscreen layer, which is where the footprints, designations and other information are found. Short Circuit PCBs have the traces drawn on this layer, to help you follow the circuit.

Components

The kit comes with a veriety of components. They make the circuit function. Designing circuits can be incredibly complicated. Our kits are designed to be simple to learn. They are not always the most efficient or effective way to design devices with these functions. They are designed to teach various concepts that can be applied in your own designs.

The Manual

There is quite a lot of information in this manual, so here is a run down of each section and what you are likely to find.

The manual is a work in progress and we would appreciate your feedback. Please head to the forums if you have any ideas or constructive criticism. As the manual is digital, we will be updating it regularly.

Symbols and Designations

Use this table to identify the different components, connections and features of the PCB and schematic.

PCB Design

This is the PCB design, which features the ground plane that's on the back of the board. This isn't shown on the PCBs silkscreen, but you can see it's borders if you angle the board at a light.

Schematic

The schematic shows the circuit design in it's simplest form. Each part of the circuit is shown separately. Connections within these sections are shown with white lines. You can find tags next to pins connected to other parts of the circuit. Each tag has a corresponding tag in the part of the circuit that it is connected to.

Bill of Materials

The Bill of Materials (BOM) is a list of parts and their values. This can be used to find replacement parts or to check the datasheets for each component.

Circuit Explanation

The rest of the Circuit section of this manual explains how and why the circuit works. You can skip this bit and head straight to building your kit, then check back later to learn how it works. Or you can go head first into the how and why, then start building. It's up to you.

Assembly Instructions

This is where you can learn how to solder your components to the PCB. The tips at the start are very useful. They may prevent you from getting frustrated. The soldering iron can be hard to tame. Make sure to keep that tip tinned and shiny! The diagrams on the right pages show the components mentioned in the instructions. You can use this to match the components to the footprints on the board, and to make sure you are using the correct resistor in the correct place.

Testing for Faults

Be sure to go through this section to minimise any risk of breaking something. If you have a short and you connect the board to power, you might overheat a component, or burn out the power supply.

Coding Basics

Follow the tutorial to make your D13 LED blink. This is the famous sketch that most people start with when programming Arduinos and Arduino compatible boards.

Project Ideas

Here are some of the devices you can make when combining your Motherboard with other kits from Short Circuits. Be sure to check out the selection at www.shortcircuits.cc

Component Index

The Component Index will give you details about each component (except some connectors). We've included information about how the component works, its construction, how to find out if it has failed and much more. You can use this as a reference when designing circuits or trying to fix them.

DIGITISER

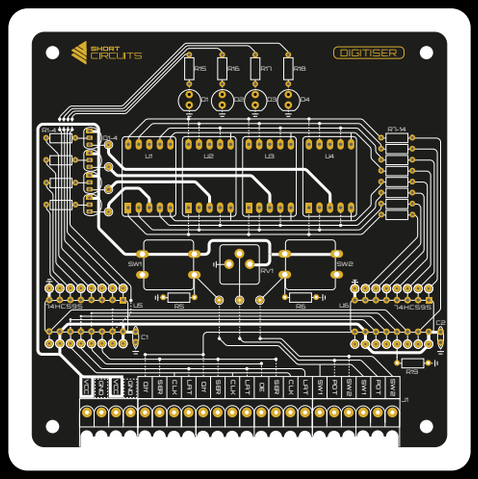
The DIGITISER is a display output board with a few inputs thrown in, to let you control your device. The DIGITISER's display consists of 4 x 7-segment displays. Each of these 7-segment displays has 7 LED's inside that light up the different segments. Turning on different combinations of these segments enables you to display numbers, and even text.

Powering all 28 segments would take more IO pins than the MOTHERBOARD has. For this reason, we have used 2 x 595 shift registers. These shift registers are able to control 8 outputs each and they only use 3-4 microcontroller pins.

For inputs, we have added 2 switches and a potentiometer. The switches can be used to select the mode, increment a counter, start a stopwatch etc. The potentiometer can be used to dim the display, or to quickly change what is being displayed.

Contents

Kit Contents



1 x Printed Circuit Board

4 x 7-Segment Displays

2 x 595 Shift Registers

2 x DIP-16 Sockets

2 x 100nF Capacitors

4 x PNP Transistors

8 x 560Ω Resistors

1 x 2K Resistors

1 x 1.5K Resistor

5 x 1.2K Resistor

4 x 10K Resistor

1 x Blue LED

1 x Green LED

1 x Orange LED

1 x Red LED

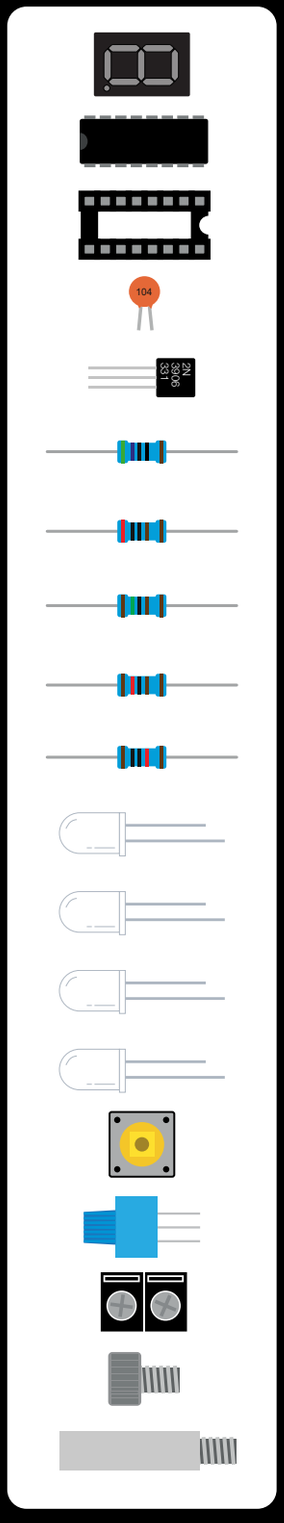
2 x Tactile Switches

1 x 10K Potentiometer

11 x Screw Terminals

4 x 6mm Hex Screws

4 x Male Female Standoffs

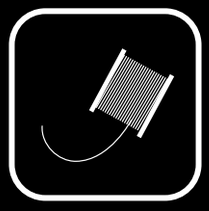


Tools Needed

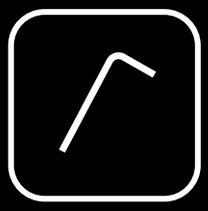
Soldering Iron

Screwdriver
2mm

Solder
0.3 - 0.5mm

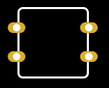
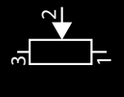
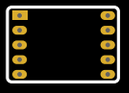
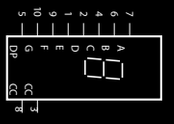
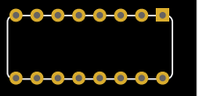
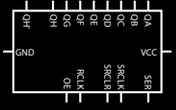


Alan Key
2mm

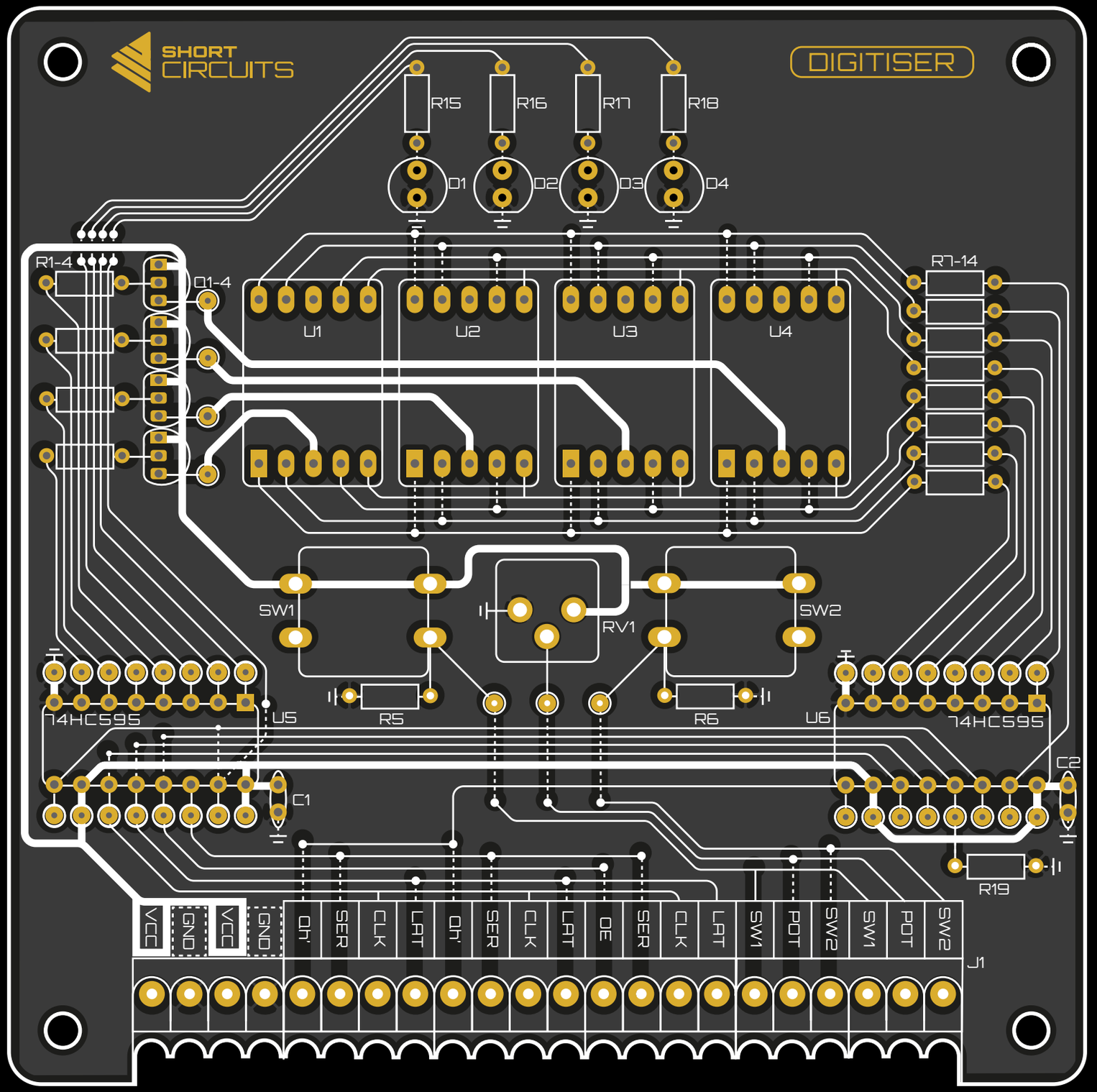


Side Cutters

Circuit - Symbols and Designations

	PCB	SCHEMATIC	DESIGNATION
Copper power trace			
Copper signal trace			
Copper trace on back of board			
Through hole solder pads			
Surface mount solder pads			
Resistor			R
Ceramic capacitor			C
LED			D
Momentary Switch			SW
Potentiometer			RV
7-Segment Display			U
74HC595 Shift Register			U
PNP Transistor			Q
Screw terminal			J
Connected to VCC			
Connected to GND plane			
Mechanical hole			

Circuit - PCB Design

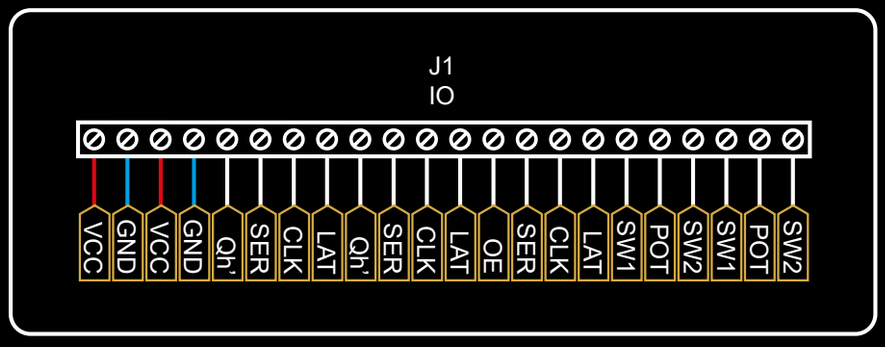


Ground (GND) Copper Area On Back of PCB

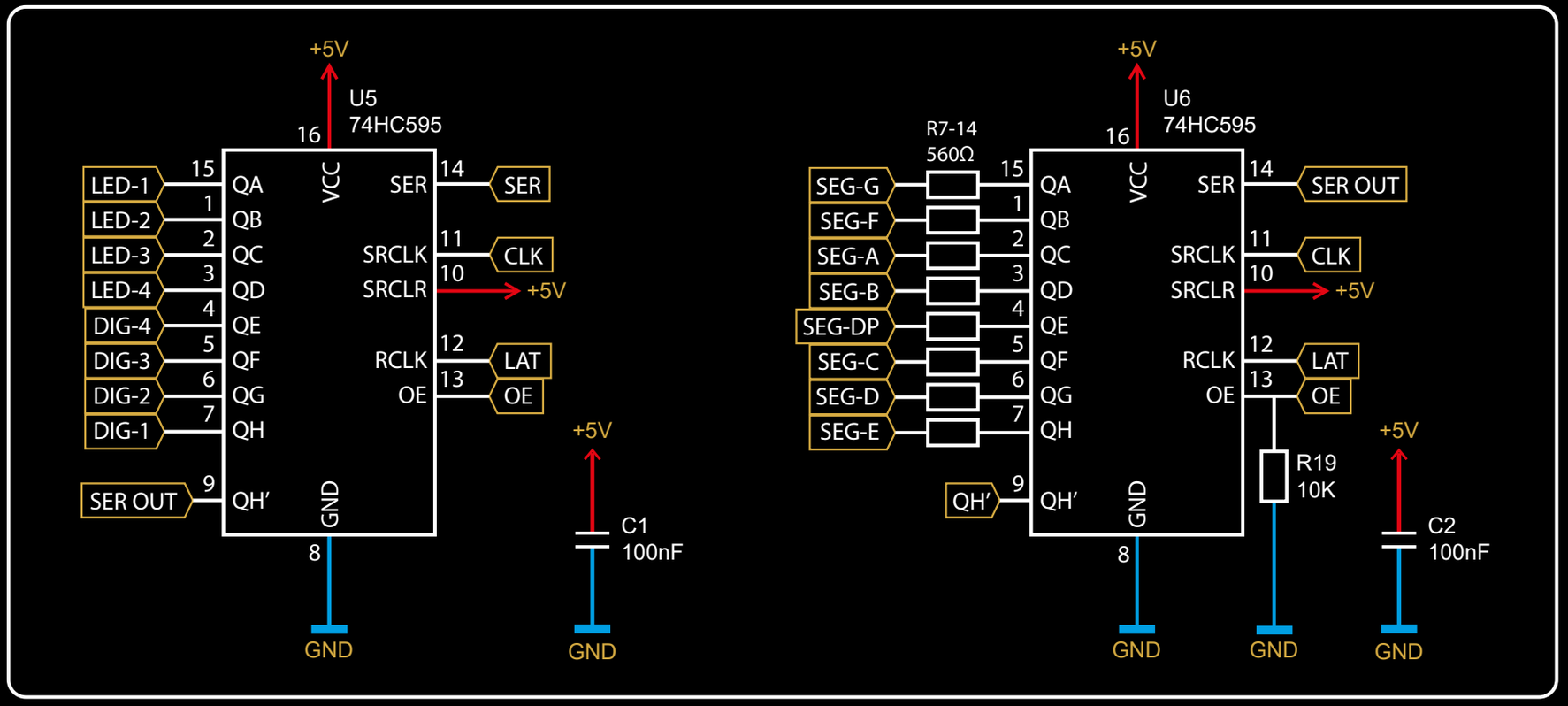
CIRCUIT

Circuit - Schematic

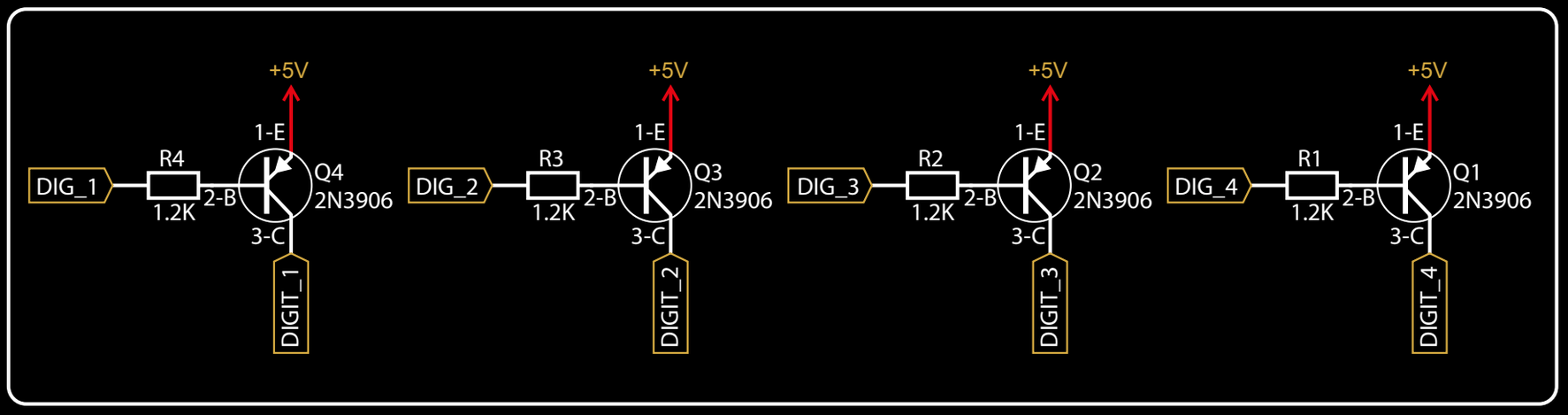
Terminal Block IO



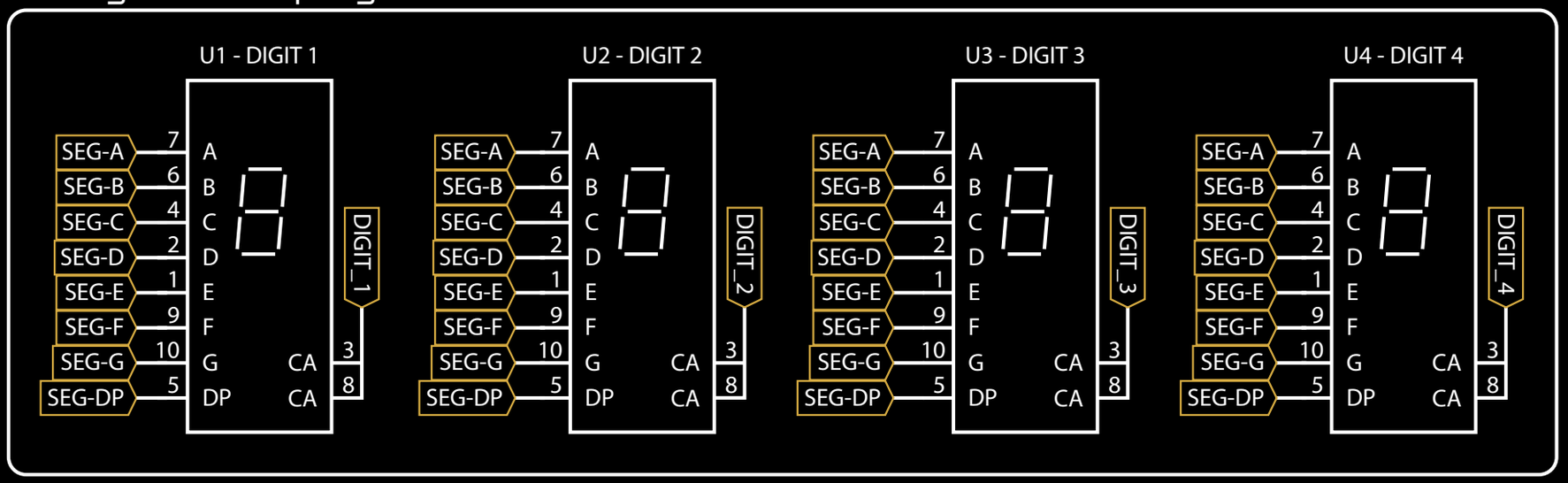
Shift Registers



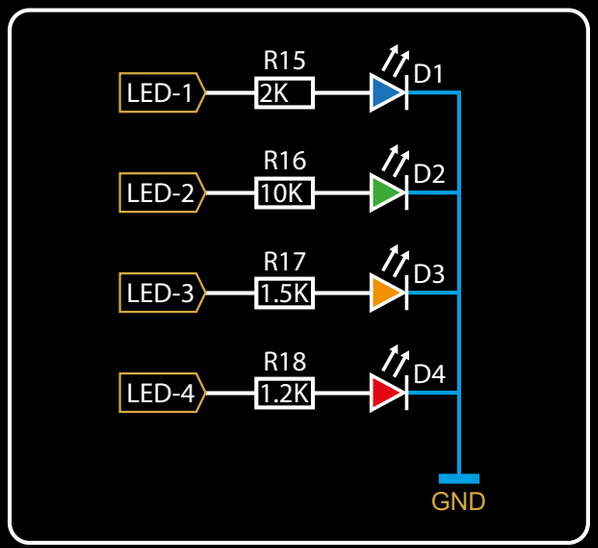
Transistor Switches



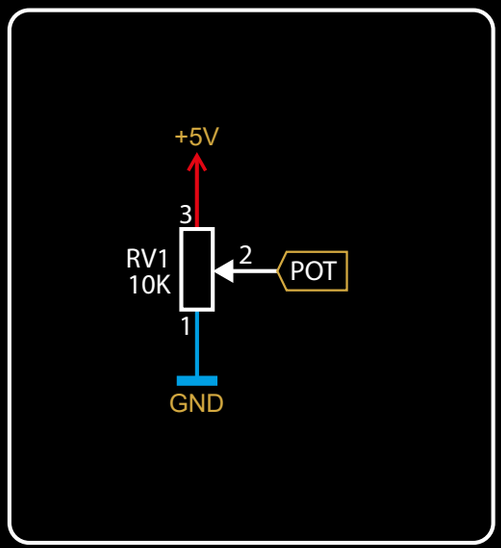
7-Segment Displays



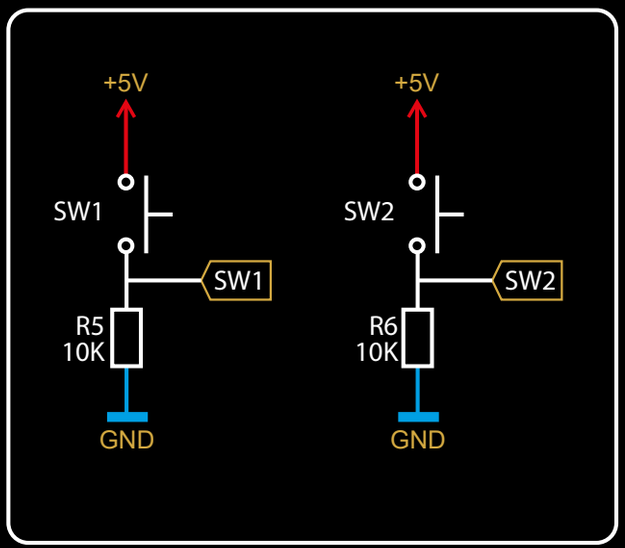
Mode LEDs



Potentiometer



Switches



CIRCUIT

Bill of Materials (BOM)

Designation	Value	Name	Footprint / Pitch	Datasheet
C1, C2	100nF	Ceramic Capacitor	2.54mm	
D1	3.2V, 20mA, Blue	LED	5mm	
D2	3.2V, 20mA, Green	LED	5mm	
D3	2.2V, 20mA, Orange	LED	5mm	
D4	2V, 20mA, Red	LED	5mm	
J1	11 x 2 pos	Screw Terminals	3.5mm	
Q1 - Q4	2N3906	PNP Transistor	TO-92	
R1 - R4	1.2K, 1/4W	Resistor THT	7mm	
R5, R6, R19	10K, 1/4W	Resistor THT	7mm	
R7 - R14	560Ω, 1/4W	Resistor THT	7mm	
R15	2K, 1/4W	Resistor THT	7mm	
R16	10K, 1/4W	Resistor THT	7mm	
R17	1.5K, 1/4W	Resistor THT	7mm	
R18	1.2K, 1/4W	Resistor THT	7mm	
RV1	10K Potentiometer	Potentiometer THT		
SW1, SW2	12mm SPST 4pin	Switch	12mm	
U1 - U4	Common Anode, Red	7-Segment Display		
U5, U6	74HC595	Shift Register	DIP16	

Circuit - Shift Registers

74HC595 Shift Register

The 74HC595 shift register is a Serial In, Parallel Out (SIPO) shift register. Data in electronics is stored as 1's and 0's which represents a switch in an on or off state. One of these switches is a bit, and 8 bits make a byte. In a SIPO Shift Register data is sent 1 bit at a time into the 8 bit register, these can then be output all at the same time in parallel.

Bits of data are sent from the microcontroller using the shift register's Serial, Clock and Latch pins. First, the Latch pin is turned on ("sent high"), this tells the shift register that data is going to be sent via the Serial pin. The bits are sent to the Serial pin one at a time. Whenever the Clock pin is sent high, the current state of the Serial pin (on or off, 1 or 0) is pushed through the shift register. After all 8 bits are shifted, the Latch pin should go low again, this sends the bits to the outputs. See the component index for a diagram.

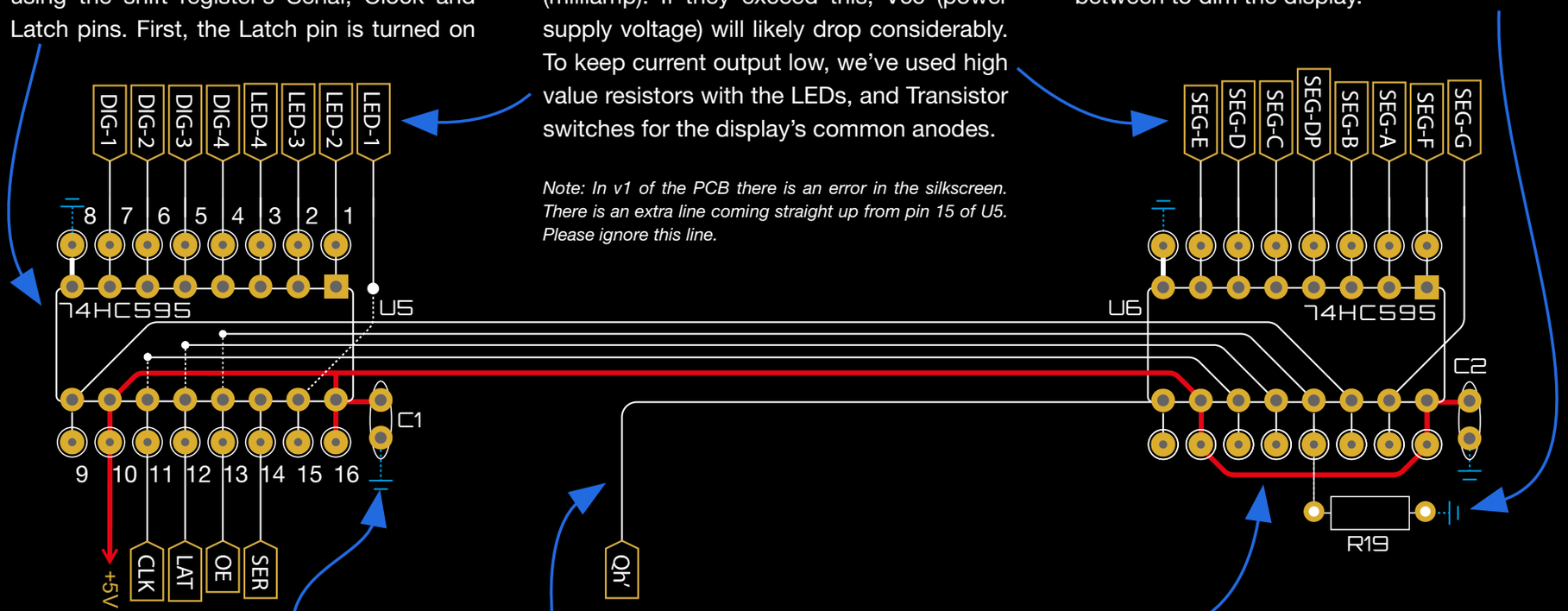
Output pins

The outputs can comfortably output 6mA (milliamp). If they exceed this, Vcc (power supply voltage) will likely drop considerably. To keep current output low, we've used high value resistors with the LEDs, and Transistor switches for the display's common anodes.

Note: In v1 of the PCB there is an error in the silkscreen. There is an extra line coming straight up from pin 15 of U5. Please ignore this line.

Output Enable

Output Enable (OE) is used to instantly turn all outputs off, or on. To enable the outputs, the OE pin must be pulled low. To turn them all off, OE must be pulled high. There is a pull-down resistor connected to OE, which means it is on, until it is pulled high by the microcontroller. If connected to a PWM enabled microcontroller pin, OE can be used to dim the display. Use analogWrite(0); to turn the display on, analogWrite(255); to turn them off. You can use any number in between to dim the display.



Bypass Capacitors C1, C2

These are decoupling capacitors. Their function is to smooth out any noise (spikes or drops in voltage) due to other components affecting the power supply to the Integrated Circuit. It 'decouples' the chip from the noise created by other parts of the circuit. It does this by holding an amount of charge and releasing it when the voltage drops, to compensate. It is placed between power and ground, as close to the IC's (chip's) VCC pin as possible.

Qh' Serial Output

Several 595 Shift Registers can be chained together using the same 3 or 4 data pins of a microcontroller. Bits are sent from the microcontroller into the board's SER input, then to the SER pin of the first shift register, out the Qh' pin of the first, into the SER pin of the second. The data can then be chained to another shift register (on the RGB Matrix for example) by connecting the output Qh' terminal (from the second register) to the SER terminal of the next board.

5V VCC

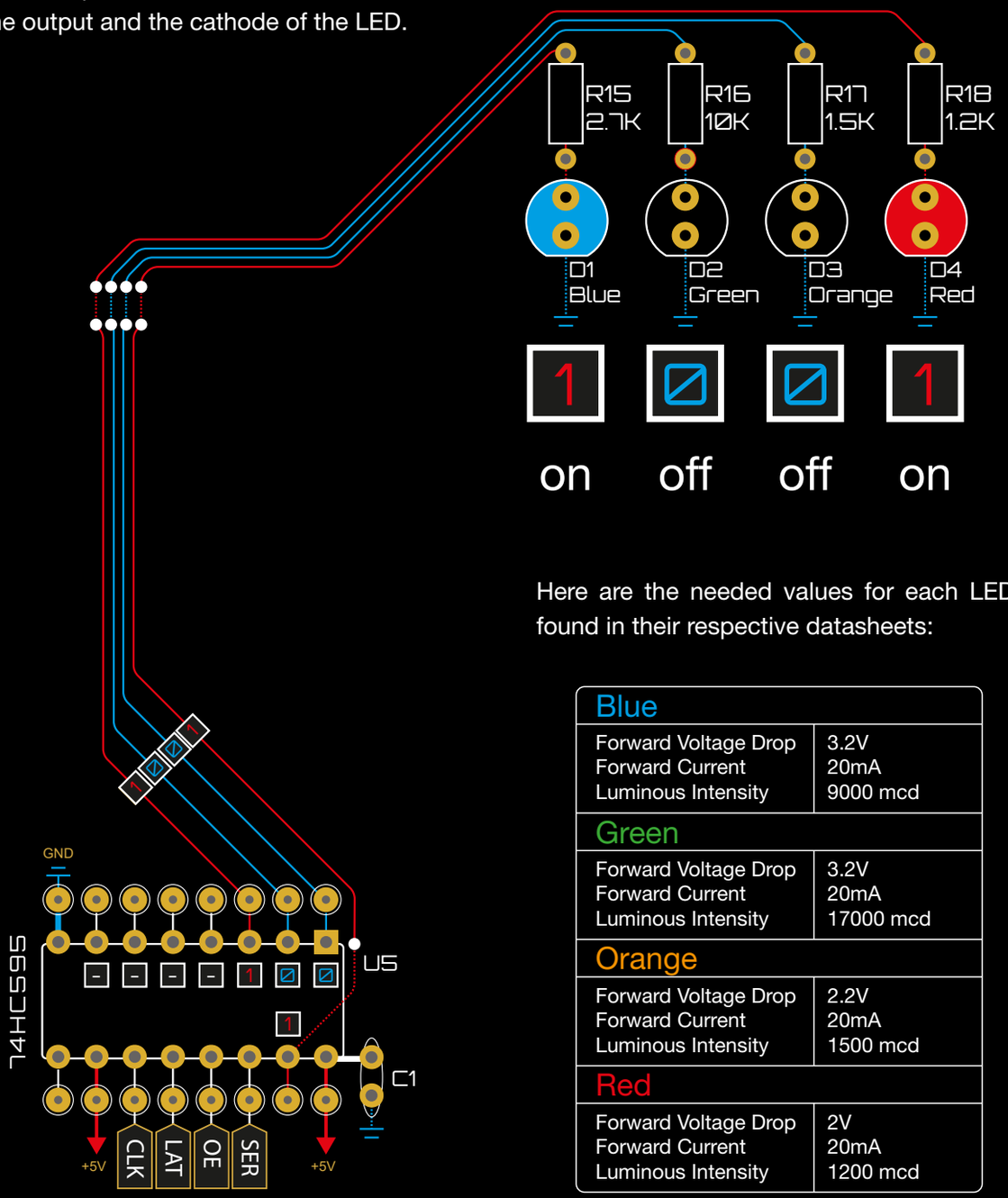
The 74HC595 can run on 2 to 6 Volts. So our 5V VCC works fine. The problem with using the 595s is their recommended output of 6mA per pin. This isn't enough to drive several LEDs, so we use the output of the 595 to activate a Transistor switch, which in turn lets more current flow safely outside the chip. There are several better shift registers out there, but these teach us some valuable lessons about the limits of components and ways to get around them.

Circuit - Mode LEDs

Shift Register Outputs

In our circuit, the LED's anodes (+) are connected to the Shift Registers outputs (through a current limiting resistor), and their cathodes (-) are connected to ground.

To turn an LED on, the corresponding Shift Register output must be set to High. This will create a potential difference of 5V between the output and the cathode of the LED.



Current Limiting Resistors

To prevent the LEDs from burning out, current limiting resistors are added in series with them. To calculate the value needed, we can use Ohm's law. Because we have 4 different colour LEDs next to each other, we will have to take into account the brightness of each.

Here are the needed values for each LED, found in their respective datasheets:

Blue	
Forward Voltage Drop	3.2V
Forward Current	20mA
Luminous Intensity	9000 mcd
Green	
Forward Voltage Drop	3.2V
Forward Current	20mA
Luminous Intensity	17000 mcd
Orange	
Forward Voltage Drop	2.2V
Forward Current	20mA
Luminous Intensity	1500 mcd
Red	
Forward Voltage Drop	2V
Forward Current	20mA
Luminous Intensity	1200 mcd

Let's first calculate the resistor value for each LED while ignoring the brightness. Let's limit the current to 0.0025A. This is well below the point where the output of the shift register would experience any significant voltage drops.

$$R = \frac{V_{source} - VF}{IF}$$

$$R = \frac{5 - 3.2}{0.0025} \qquad R = 300\Omega$$

$$R = \frac{5 - 3.1}{0.0025} \qquad R = 760\Omega$$

$$R = \frac{5 - 2.2}{0.0025} \qquad R = 1120\Omega$$

$$R = \frac{5 - 2}{0.0025} \qquad R = 1200\Omega$$

Now we can adjust for brightness. The dimmest LED can stay as it is. That's the red LED. We can adjust the others in relation to this because the relationship between Forward Current and Luminous Intensity is directly proportional.

$R = 300$ $\quad mcd = 9000$ $\quad R_{new} = 300 \times \frac{9000}{1200} = 2250\Omega$

$R = 760$ $\quad mcd = 17000$ $\quad R_{new} = 760 \times \frac{17000}{1200} = 10766\Omega$

$R = 1120$ $\quad mcd = 1500$ $\quad R_{new} = 1120 \times \frac{1500}{1200} = 1400\Omega$

$R = 1200$ $\quad mcd = 1200$ $\quad R_{new} = 1200 \times \frac{1200}{1200} = 1200\Omega$

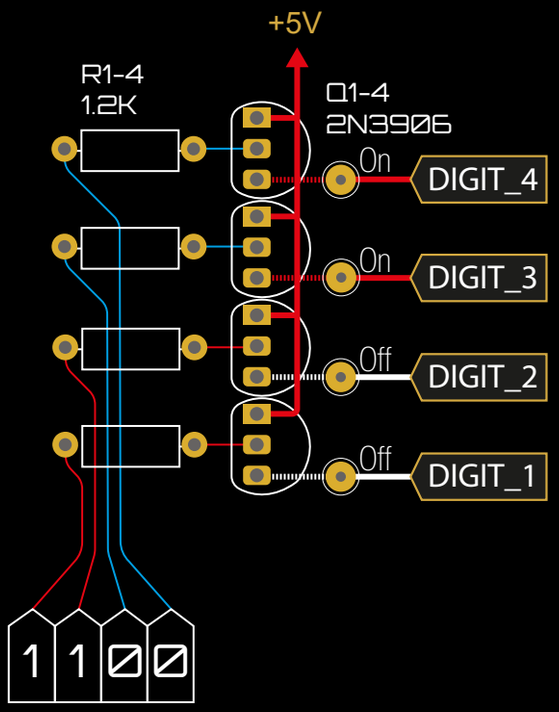
We can now test the closest common resistor values with our LEDs and check to see if their brightnesses match. We have further adjusted the resistor values so all 4 colours look roughly the same brightness.

Circuit - Transistor Switches

PNP Transistors

A Transistor can be used as a switch to apply higher currents to a different part of the circuit with only a small amount of current to trigger it. This is perfect for our scenario, as the 595 Shift Register can't source enough current to power 8 LEDs on one output pin, or 32 LEDs across 4 pins (our worst case scenario).

PNP Transistors turn on when a small amount of current flows from Emitter to Base. When enough flows from Emitter to Base, a much greater current will flow from the Emitter to the Collector. See the Component Index for more info.



How the circuit works

In circuit, when the Shift Registers output pin is pulled low (fig.1), a small amount of current flows from the Transistors Emitter to its Base. Remember, conventional current flows from positive to negative, if there is a potential difference between them. In our case, the Emitter is at 5V and the Base (after a resistor) is at 0V.

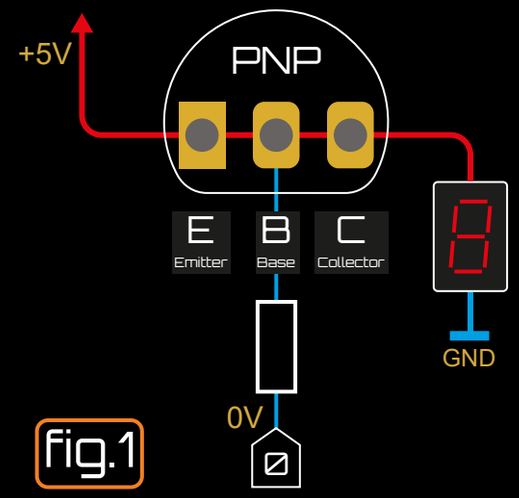


fig.1

When this Base Emitter (BE) current flows, it will open the "gate" between the Collector and Emitter (CE), allowing higher currents to flow through the transistor. This then allows current to flow through the 7-segment display's common anode, and into any segments that are currently pulled low. Thus, turning them on.

When the shift register pin is pulled high (fig.2), there will be no potential difference between the Emitter and the Base. So no current will flow, and the "gate" will not open.

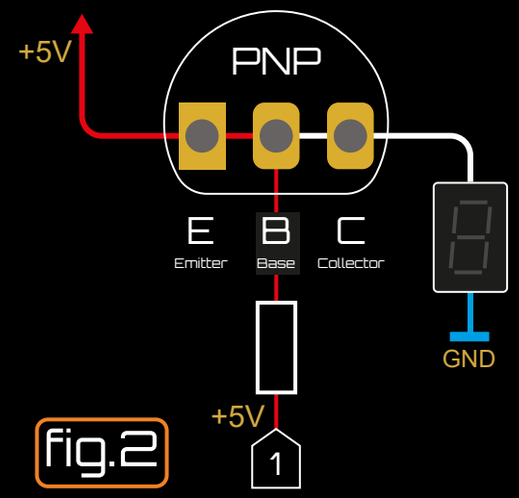


fig.2

If none of the segments in the display are pulled low, nothing will happen either, as there will be no potential difference between the 5V coming from the Transistor and the cathodes of the display.

Switching

To use a Transistor as a switch, we need to make sure the Transistor is in Saturation Mode. For this we need to apply more than enough current to the Base to ensure the Transistor switches on fully.

To calculate the Base Resistor value, we need to know a few things. Firstly, how much current will flow through the Transistor and the display. We also need to know the Current Gain (h_{FE})

We will be using 5mA of current per segment, to stay within limits. If all of the segments were on (showing an "8" and a decimal point), then we would have 8 x 5mA flowing through the Transistor. A total of 40mA. This would be -40mA from the Shift Register's point of view. The 2N3906 is rated for -200mA, and the Shift Register can take 6mA on each output, and 70mA total, so we are well within its limits.

As the Transistor is an amplifier, the current gain (h_{FE}) represents how many times the Base current is amplified (multiplied) by, to equal the Collector current. According to the datasheet, at -50mA, this would be 60 times. So if we apply a 60th of the Collector Current to the Base, it will turn the transistor on (when the Collector Current is delivering around -50mA).

This would be the minimum amount of current needed. Any fluctuations in temperature etc would make this unreliable as a switch.

Circuit - Transistor Switches

To ensure we turn the Transistor on fully we can use a lower Current Gain for our calculation. There are different methods to ensure your calculations result in hard saturation. We will use the h_{FE} of 10 method, which is commonly used.

So we know the Collector Current to be around -40mA and we will assume the Current Gain to be 10. We also know that Vcc is 5V.

With these values we can calculate the value of our Base Resistors (R1-4).

First we calculate the current needed at the Base.

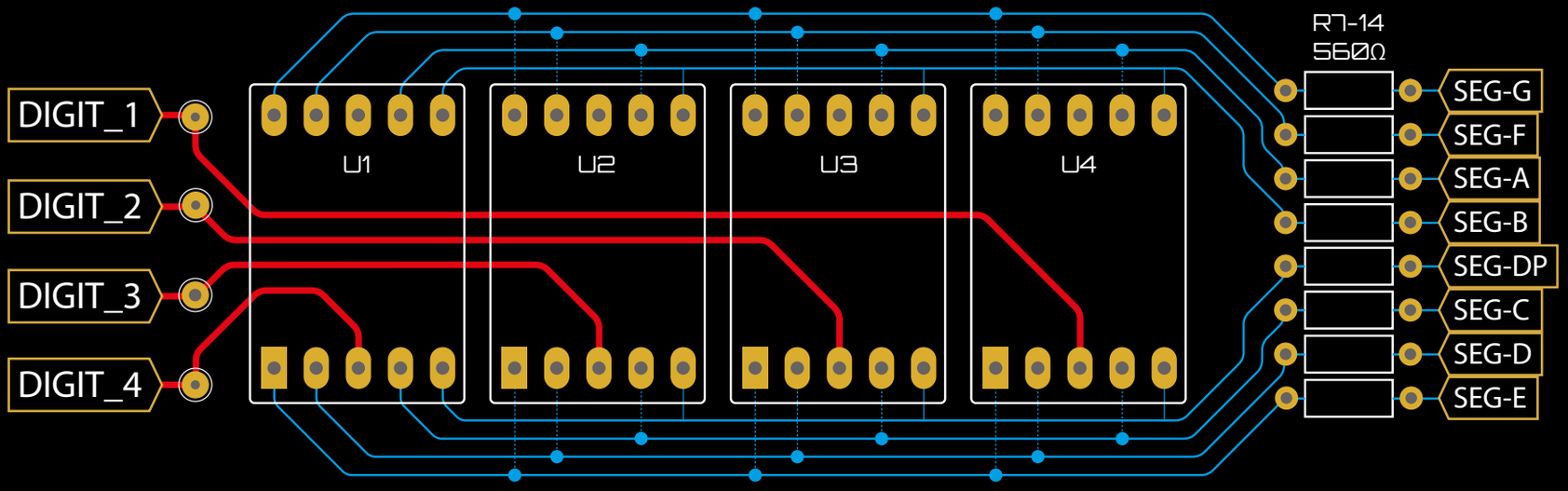
$$I_B = \frac{IC}{H_{FE}} \qquad I_B = \frac{-0.04}{10} \qquad I_B = -0.004A$$

Then we use Ohm's law to calculate the Resistance needed.

$$R = \frac{V}{I} \qquad R = \frac{-5}{-0.004} \qquad R = 1250\Omega$$

So we now know that a 1200Ω resistor would result in the Transistor operating in saturation mode, allowing enough current at the Collector for our display, with minimal resistance.

Circuit - 7-Segment Displays



The Matrix

To avoid using a Shift Register for each digit, the segment displays are wired up in a matrix. This means that cathodes with the same letter from each of the displays are connected together. For example, segment A from U1, U2, U3, and U4 are all connected to one output of the second Shift Register (U6) through a current limiting resistor.

The downside to a matrix, is that you need to cycle through the digits one at a time in order to show a different number on each digit. This makes the code more complicated, and the results in a dimmer display. However, freeing up IO pins on the microcontroller to interface with other devices is a much better payoff.

LEDs in Parallel

Because of the matrix the segment LEDs are wired in parallel. If all 4 digits were on at the same time, the distribution of voltage across the 4 LEDs could be unbalanced and cause dimming in some of the segments. Fortunately, this is avoided when cycling digits.

Current Limiting Resistors

To calculate the current limiting resistor's value, we will need to consider the voltage drop (CE) across the Transistor. This can be found by looking at the CE Voltage drop vs. Collector Current graph in the transistor's datasheet. The segments are bright enough at 5mA and only one digit will be on at a time, so we can use 5mA as our current.

$$R = \frac{V_{source} - V_{LED} - V_{CEsat}}{I} \qquad R = \frac{5 - 2 - 0.1}{0.005}$$

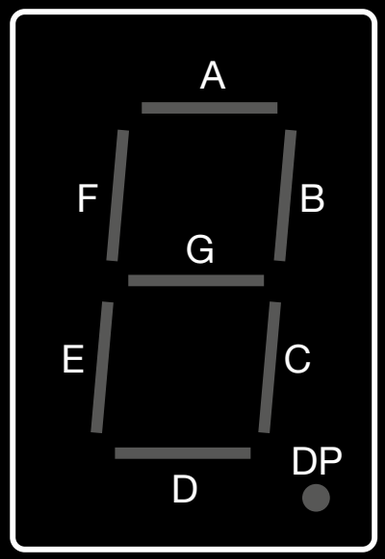
$R = 580\Omega$ (560Ω nearest common value)

Circuit - 7-Segment Displays

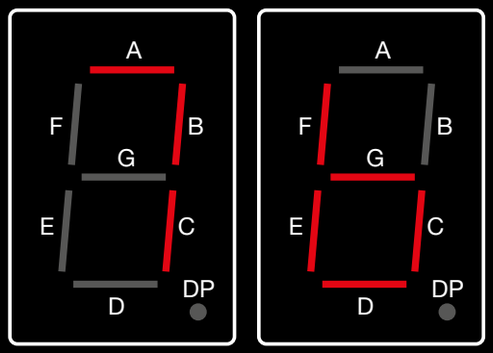
The Segments

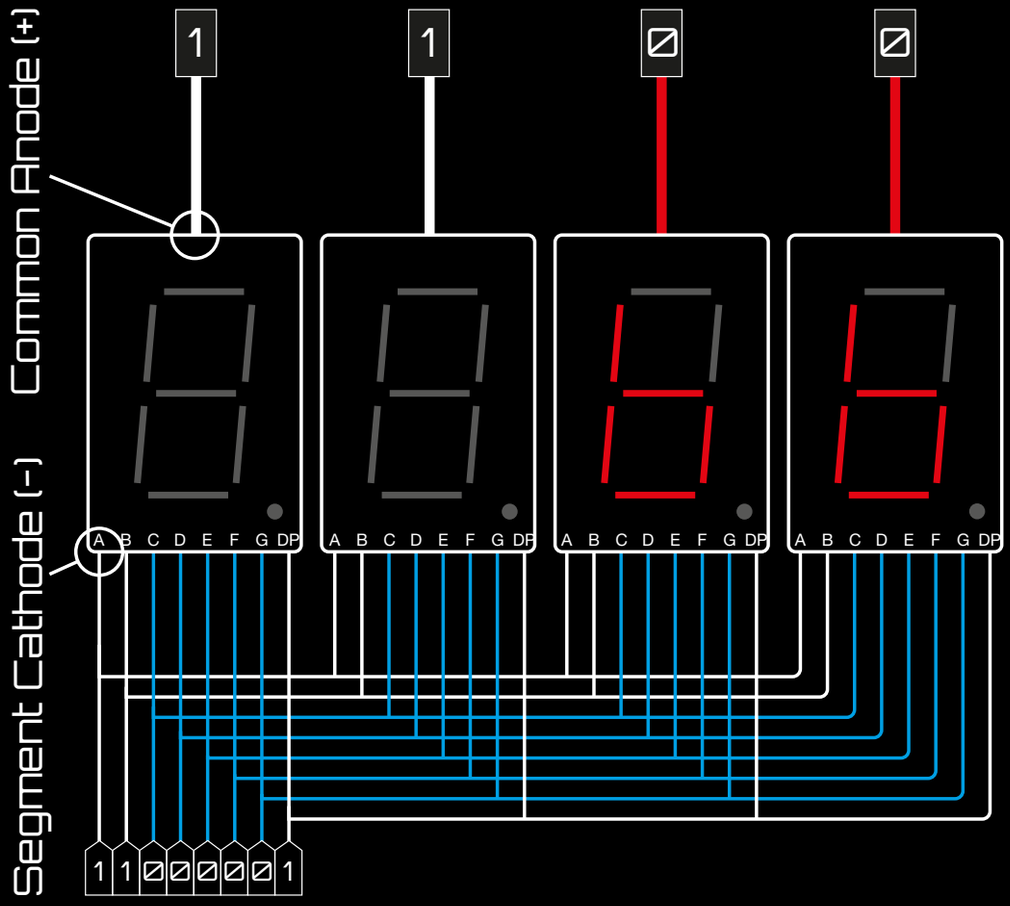
7-segment displays are named after the 7 segments that, when all are on, show the number 8. By turning off different combinations, you are able to show numbers 0-9 and even a crude but readable A-Z. Not all 7-Segment displays have seven segments. The one we are using has 8. the eighth being the decimal point in the corner.

Here is a diagram that shows the segments and their reference letters:



By turning on A, B and C, we can display the number 7. By turning on C, D, E, F and G, we can display the number 6.





Tips for Coding

The first thing to note is that the segment reference letters don't match up to the Shift Register output letters. So we need to reference the datasheet or a table to help us write the code.

There are also two Shift Registers. This means there will be two lines of code to shift bits into the registers. The first line of the code containing 8 bits, goes into the second Register (U6), and the second line of bits ends up in the first Register (U5). This is because the bits are loaded into the Shift Register one after another, with the next bit pushing the previous bit through the Register. The first bit to be sent ends up on the last output of the second Register. The last bit to be sent ends up on the first output of the first Register.

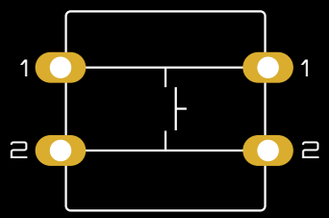
Another thing to note is that turning on a segment cathode requires a 0 and turning on a digit requires a 0 (because of the Transistor Switches). Turning on an LED requires a 1.

As previously mentioned, the displays are wired in a matrix. This means we will shift each digit separately. We can copy the shiftout code 4 times, one for each bit, or we can create a for loop to repeat the code 4 times while changing the shift bits for each.

All of this information can be found with a neat diagram in the coding section. You can easily reference it when the time comes to code.

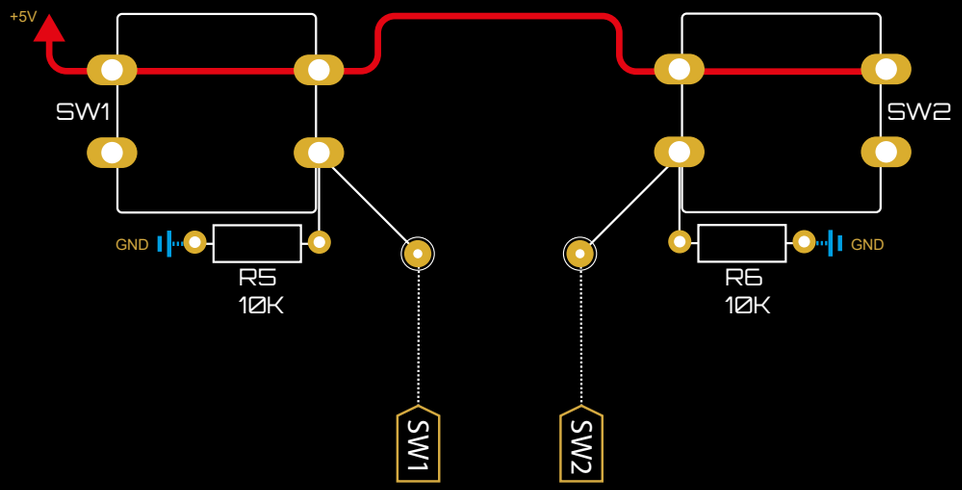
Circuit - Switches

The switches used in the Digitiser kit are Single Pole, Single Throw switches. These are a little different from the SPST switch used on the Motherboard. They are through hole components, rather than surface mount. But more importantly, they have 4 pins. The diagram below shows the internal connections. The pins labeled 1 are connected and the pins labeled 2 are connected. When the switch is pressed, all the pins will be connected.



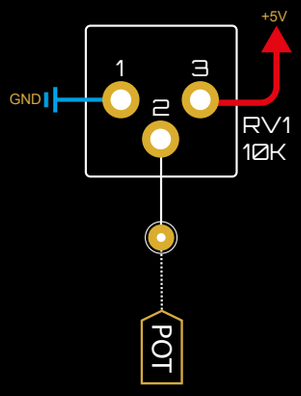
When laying traces for your own circuits, you can use either side, or pin 1 from the left and pin 2 from the right etc. It really doesn't matter. You should however, always check the datasheet, or use a multimeter to check which pins are connected when the switch is not pressed.

Other than the extra pins, the switch is wired up the same as the Motherboard's switch. One pin is connected to Vcc and the other is connected to both the data pin of the microcontroller and to GND through a 10K pull down resistor.



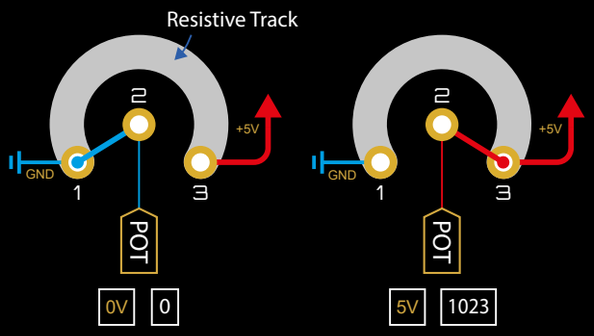
When the switch is not pressed, the data line will read 0V as the line is pulled down (although weakly) through the 10K resistor to ground. When the switch is pressed, the data line will read nearly 5V. This is because the 5V (with nearly 0 resistance) connected to pin 1 of the switch overpowers the 0V (that has 10,000 ohms of resistance between it and the data line). Without this resistor, every time you pressed the switch you would create a short circuit.

Circuit - Potentiometer



A potentiometer is essentially a resistive track with 3 contact points along it. Pin 1 and 3 are at either end of the resistor, and pin 2 moves along the track. By connecting ground (0V) at pin 1 and 5V at pin 3, pin 2's voltage sweeps from 0 - 5V as you turn the knob.

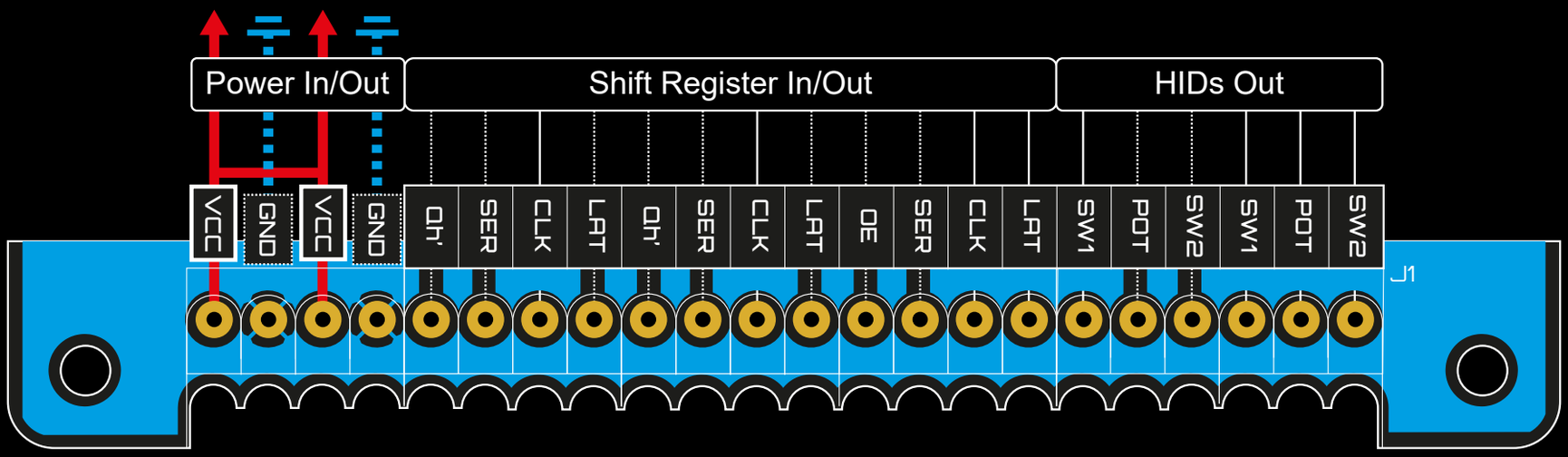
When our 10K potentiometer's knob is turned anticlockwise all the way, the resistance between pin 2 (connected to the microcontroller) and pin 1 (GND) would be close to 0Ω. The resistance between pin 2 and pin 3 (VCC) would be around 10,000Ω. This would give a reading of near 0V at pin 2, which would give an analogRead() of 0.



When the knob is turned fully clockwise, the resistance between pin 1 and 2 would be 10,000Ω and the resistance between pin 2 and 3 would be nearly 0Ω. As pin 3 is connected to Vcc, pin 2 would read 5V and give an analogRead() of 1023.

If the knob were anywhere in between, then the voltage would be between 0V and 5V depending on where pin 2 is connected to the resistive track.

This analogRead() value can be used to set the analogWrite() value required to control the Output Enable (OE) pin of the Digitiser's Shift Registers. This allows us to use the Potentiometer as a dimmer.



Power

Like all Short Circuit boards, the Digitiser has 2 Vcc and 2 GND terminals. This allows you to chain multiple boards together. Simply use one set as an input from the previous board (The Motherboard for example) and the other as an output to the next board in the chain.

Ground Plane

The Digitiser uses a ground plane to link all the parts of the circuit that need to be connected to GND. This is a large area of copper that covers the back of the board and is linked to the 2 GND terminals.

Note how the ground plane is connected to these terminals. The pads are linked via thermal relief spokes. These spokes connect the pad to the ground plane in four places while leaving a gap between the ground plane and the rest of the pad. This makes it easier to solder the component to the pad. If there were no thermal relief spokes, the heat from the soldering iron would rapidly conduct away from the pad into the large amount of copper in the ground plane. This would make it difficult to melt the solder and form a reliable connection.

The ground plane is only broken when other circuit traces need to be routed along the back to avoid other traces on the front. This is done as little as possible. Care is taken to make sure that no parts of the ground plane are completely cut off from another. Also, it is important to consider the electrical current's return path. Make sure that the paths from each GND connection back to the Power IO are as unobstructed as possible.

Shift Register Inputs and Outputs

The Shift Registers need 3 data inputs to function. These are Serial, Clock and Latch. OE or Output Enable is used to dim the display using a PWM enabled pin from the microcontroller. Qh' outputs serial data from the Digitiser's Shift Registers to Shift Registers on another board. This allows you to control both the Digitiser and the RGB Matrix with the same 3 microcontroller pins.

Human Interface Devices

Human Interface Devices (HIDs) include anything that humans use to control digital computing devices. This includes your mouse and keyboard, gamepad, and other PC peripherals. Here, we are referring to the two Switches (think mouse buttons or keyboard keys), and the Potentiometer (like a mouse scroll wheel or volume control).

Assembly Instructions

General Soldering tips

1. ALWAYS KEEP YOUR TIP CLEAN

To ensure the soldering iron can transfer enough heat from it to your solder/component leg you must keep the tip clean and shiny. A dull tip means the outside layer of metal has oxidise. This oxidised layer is a poor transferrer of heat. Because of this, you will have to hold the tip against the component for a longer period of time, which can result in the component failing. It's also very frustrating.

To keep your soldering iron tip clean, wipe it on a wet sponge or wire ball, then apply some solder to coat it, then wipe it again. Ideally, you should do this after every component. At the very least, do it after every 4 components.

2. CONTACT

When soldering, make sure the tip of your iron is making contact with both the leg of the component and the pad on the PCB. Apply heat to the area, then, within a second or two, apply the solder to the point of contact.

3. HEAT

It is better to be too hot than too cold. As mentioned earlier, when the iron tip isn't hot enough, you have to hold it on longer. This allows heat to transfer into the component and could cause a failure. It is better to set your soldering iron a little hot so the solder melts instantly and flows around the leg of the component with ease. You can start at around 350ºC and adjust from there.

4. SOLDER

Leaded solder is much easier to work with, which makes it easier to learn with. It can be hard to find in some countries, but can often be ordered from China. There are potential health risks, but these are very low. Make sure you have a fan pointing away from your work area to blow the fumes away. Work in a well ventilated area.

Thinner is better. Working with a thick wire of solder can get messy. Use 0.4-0.5mm solder for more control. You will have to feed more into the solder joint, but you have more control when there are other pins close by that you want to avoid. This is essential when soldering surface mount components and Integrated Circuits.

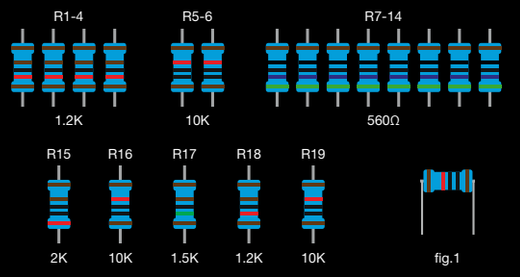
5. SAFETY

350ºC is obviously very hot. Stuff catches fire at this temperature. Skin fries. Please be careful. Remember to turn it off when you are finished. This will prevent a potential house fire and also save your iron tip from continued oxidation.

6. ANGLE OF ATTACK

When soldering, make sure you position the board so that it is easy to access the area you are working on. It is easy to make a mistake when you are trying to maneuver your iron into position around some obstacle. You may have to spin the board 180 degrees, or make sure you have snipped the legs off the previous component. It's easy to be impatient so try to plan ahead.

Resistors

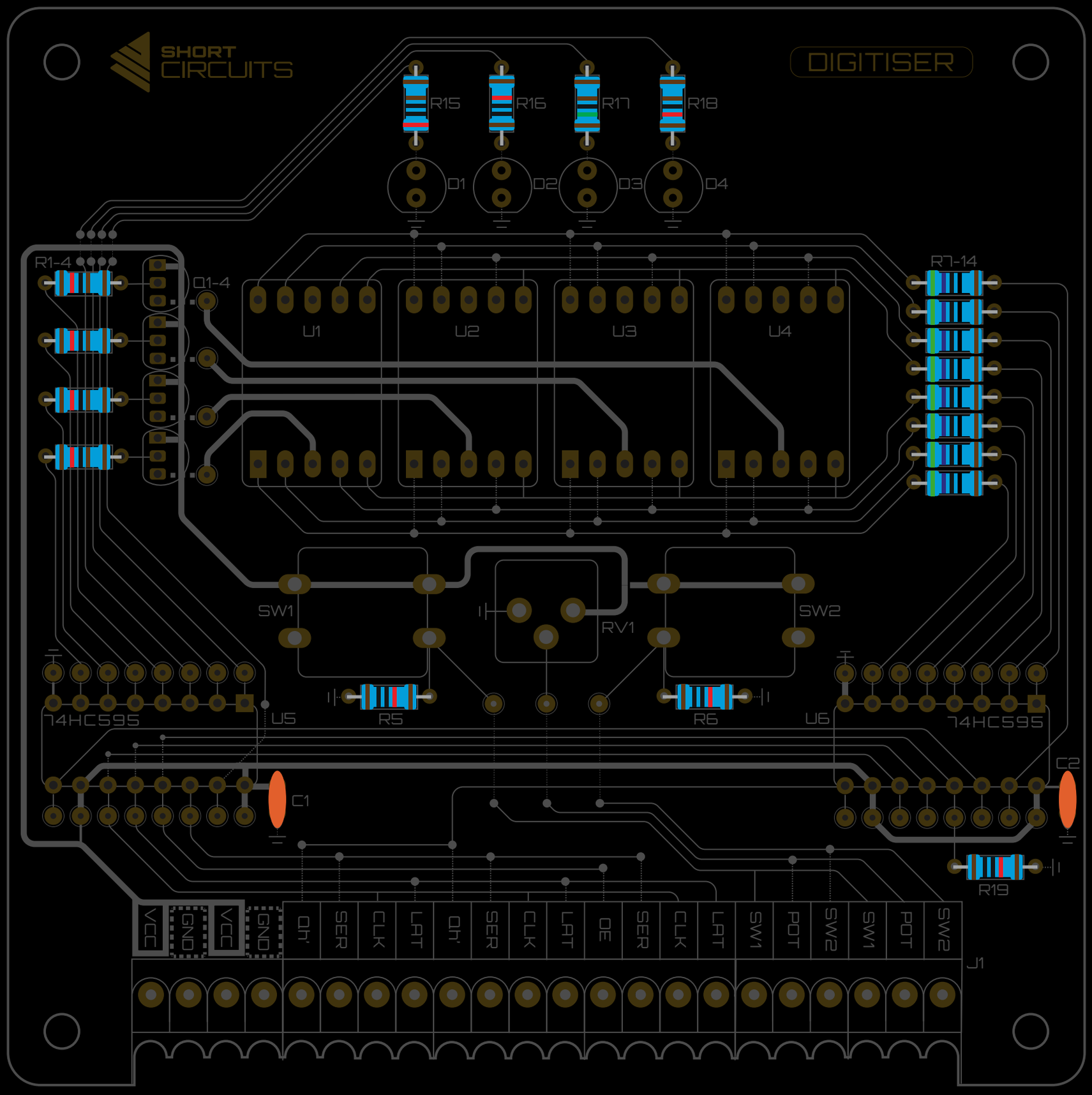


Bend each leg of the resistor by 90 degrees, as close to the resistor as possible (fig.1). The holes are fairly close together, so this step is important. Then insert the legs into the correct holes by finding the correct reference (R1, R2 etc) on the board, or by using the picture on the next page as a guide. Polarity is not important with resistors, so either way is fine.

Hold the resistor in place with sticky tack or tape, then flip the board over, ready for soldering. Solder each leg by holding the soldering iron tip to the leg and the pad, then apply the solder to the joint. Let it flow around the leg and create a shiny cone. Use flush cutters to cut the legs off, then briefly hold the soldering iron to the joint to re-flow the solder.

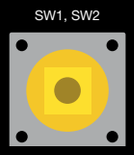
Ceramic Capacitors

Install the ceramic capacitors into the corresponding holes (C1, C2). They will easily fall out, so add some sticky tack or tape to secure them. Polarity does not matter. Flip the board and solder the pins to the pads, then trim and re-flow.


SHORT CIRCUITS
DIGITISER
R15
R16
R17
R18
D1
D2
D3
D4
R1-4
D1-4
R7-14
U1
U2
U3
U4
SW1
RV1
SW2
74HC595
U5
R5
R6
U6
74HC595
C1
C2
R19
VCC
GND
VCC
GND
Oh'
SER
CLK
LAT
Oh'
SER
CLK
LAT
OE
SER
CLK
LAT
SW1
POT
SW2
SW1
POT
SW2
J1
ASSEMBLY

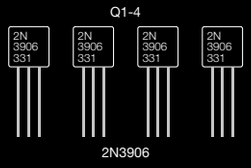
Assembly Instructions

Switches



The switches will fit into their holes quite tightly. If they don't pop in flat to the PCB, check that the legs are all aligned properly. They should hold on their own, so flip the board and solder the pins to the pads. The pins are short so they shouldn't need trimming.

PNP Transistors

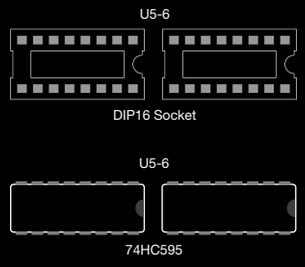


The transistors can be tricky to solder because the legs are so close together. We have spaced the pads out a little to make it easier. Notice the semi-circular shape of the transistor looking at the top. Match this with the footprint on the PCB.

The outer legs of the transistor will spread out a little as you insert it into the holes. Don't push it too far as the legs will spread too much and crack the black casing of the transistor. Stop when there's a 2-3mm gap. Add sticky tack to hold in place, flip, solder, snip and re-flow.

If you accidentally bridge two pads together, clean your iron, re-tin the tip, then drag the tip through the gap between the pads. This should separate them. If it doesn't, repeat the process. If your tip isn't clean, the solder won't stick to the tip when you drag it through, so make sure that tip is clean and re-tinned.

DIP16 Socket / 595 Shift Registers



We are going to solder a socket to the PCB rather than the chip itself. We do this to avoid heating up the IC and to enable us to swap it out if it stops working.

Insert the sockets into the areas marked U1 and U2. Pay attention to the semi-circle indicator and match the one on the socket with the one on the board.

Note: an error on v1 of the board means the semi-circles have accidentally been omitted. Take note that the pad for pin 1 of the IC is square. The semi-circle should be at the same end as the square pad.

Now, stick it in place with sticky tack and solder 1 corner pin, then the diagonally opposite corner. Now check alignment and whether its flush with the board. If one corner is too high, add pressure to it while heating the pin underneath.

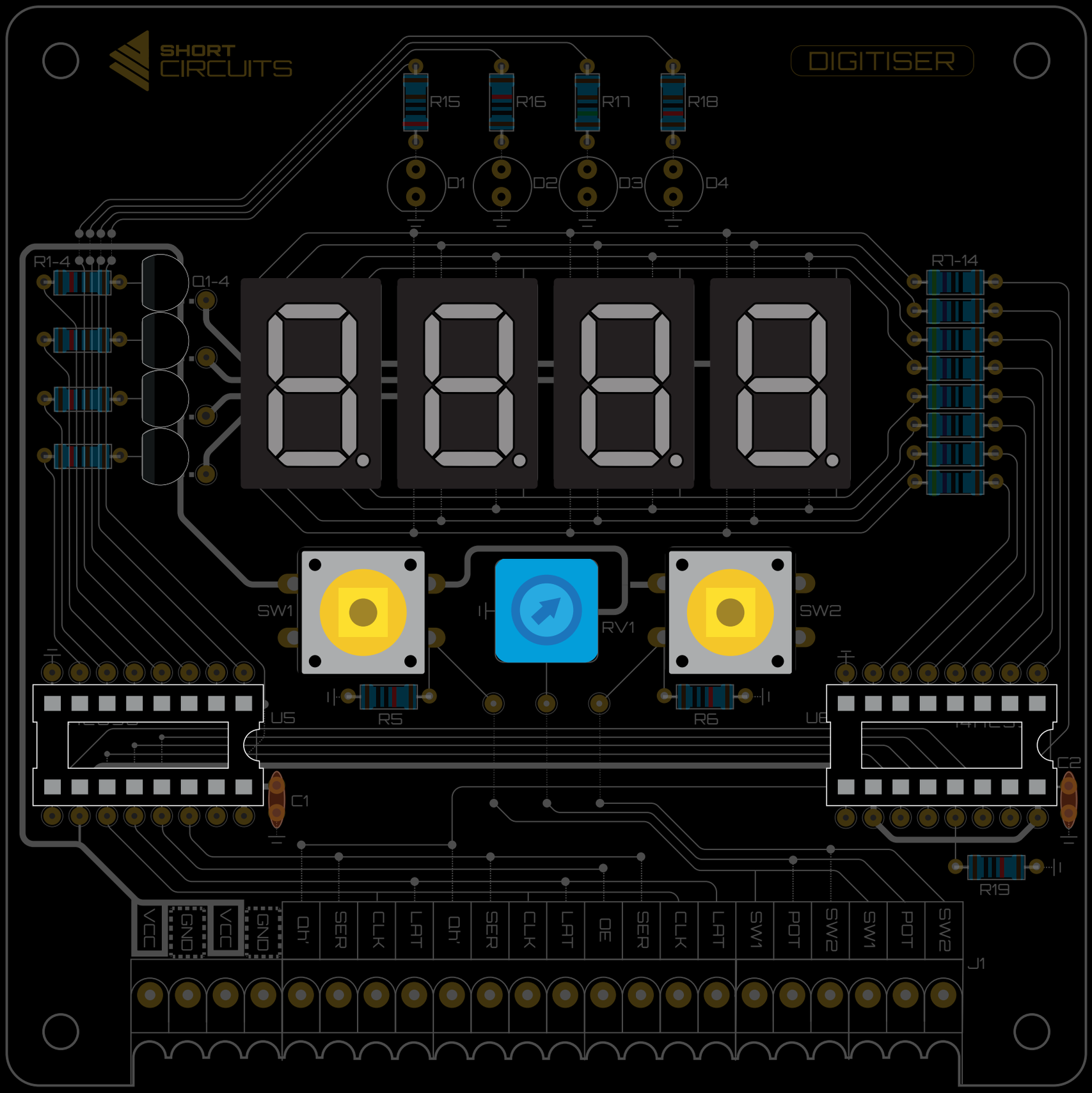
Now you can solder all the other pins knowing that the socket will stay exactly where it needs to be. Remove the sticky tack before soldering the rest of the pins. As with the Transistors, if you accidentally bridge two of the pins, wipe your clean iron tip through the center of them until they are separated.

Potentiometer

Insert the legs of the Potentiometer into RV1 on the board. Hold the potentiometer with some sticky tack and align the blue case with the footprint. Flip the board and solder the three pins. Snip and re-flow.

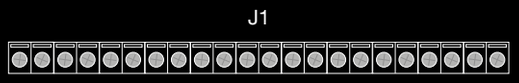
7-Segment Displays

Be aware of the orientation of the display when inserting them into the PCB. Make sure the decimal point is in the bottom right corner. You can insert all 4, secure them then solder the pins. Try not to hold the iron on too long, as the white plastic casing can melt. You will need to snip the legs off these as they are quite long and pokey. Make sure you re-flow the solder joints after you snip the legs to make sure the solder hasn't cracked.


SHORT CIRCUITS
DIGITISER
R15
R16
R17
R18
D1
D2
D3
D4
R1-4
D1-4
R7-14
SW1
RV1
SW2
R5
R6
U5
C1
C2
R19
VCC
GND
VCC
GND
Qh'
SER
CLK
LAT
Qh'
SER
CLK
LAT
OE
SER
CLK
LAT
SW1
POT
SW2
SW1
POT
SW2
J1
ASSEMBLY

Assembly Instructions

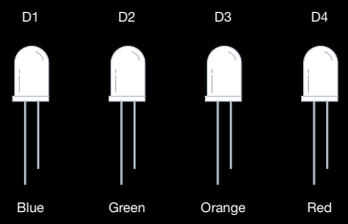
Terminal Blocks



The terminal blocks are interlocking, so slide each of them together to make a chain of 11. To make this line of terminal blocks fit in their holes, you may need to squeeze the blocks together a bit.

If you are using the Rack case to display your boards, I would suggest clipping the legs off. These can interfere with the wires and cause a short when they are sandwiched between the case and the board.

LEDs



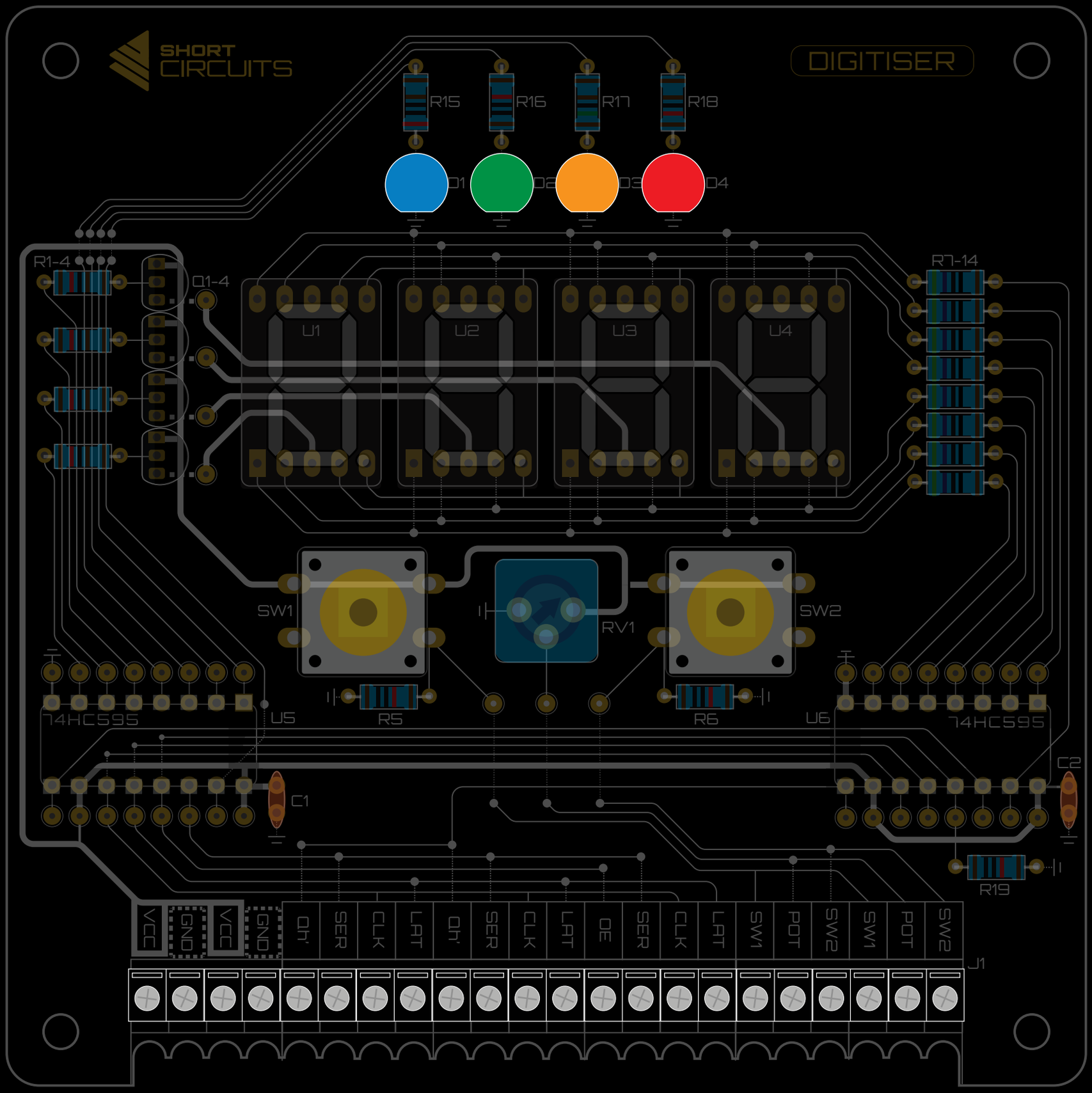
Wait, how do I know which LED is which? Well, we'll have to turn them on.

Be careful! Do not connect them directly to GND and VCC, they will likely blow!

To check the colour, follow the code guide in this manual to turn the LED circuits on. Now you can insert the LEDs into any position to find out what colour they are. LEDs are polarity sensitive. The short lead is negative and the long lead is positive. You can also tell by finding the flat edge on the rim of the LED, that's the negative side. The flat part of the symbol on the PCB is also negative, so match them when inserting the LED.

Make sure you disconnect the board before you solder the LEDs!

Solder the legs from the other side like the other components. Snip and re-flow.


SHORT CIRCUITS
DIGITISER
R15
R16
R17
R18
D1
D2
D3
D4
R1-4
D1-4
R7-14
U1
U2
U3
U4
SW1
RV1
SW2
74HC595
U5
R5
R6
U6
74HC595
C1
C2
R19
VCC
GND
VCC
GND
Dir.
SER
CLK
LAT
Dir.
SER
CLK
LAT
OE
SER
CLK
LAT
SW1
POT
SW2
SW1
POT
SW2
J1
ASSEMBLY

Testing for faults

Before you power on the board, there are a few things we need to do.

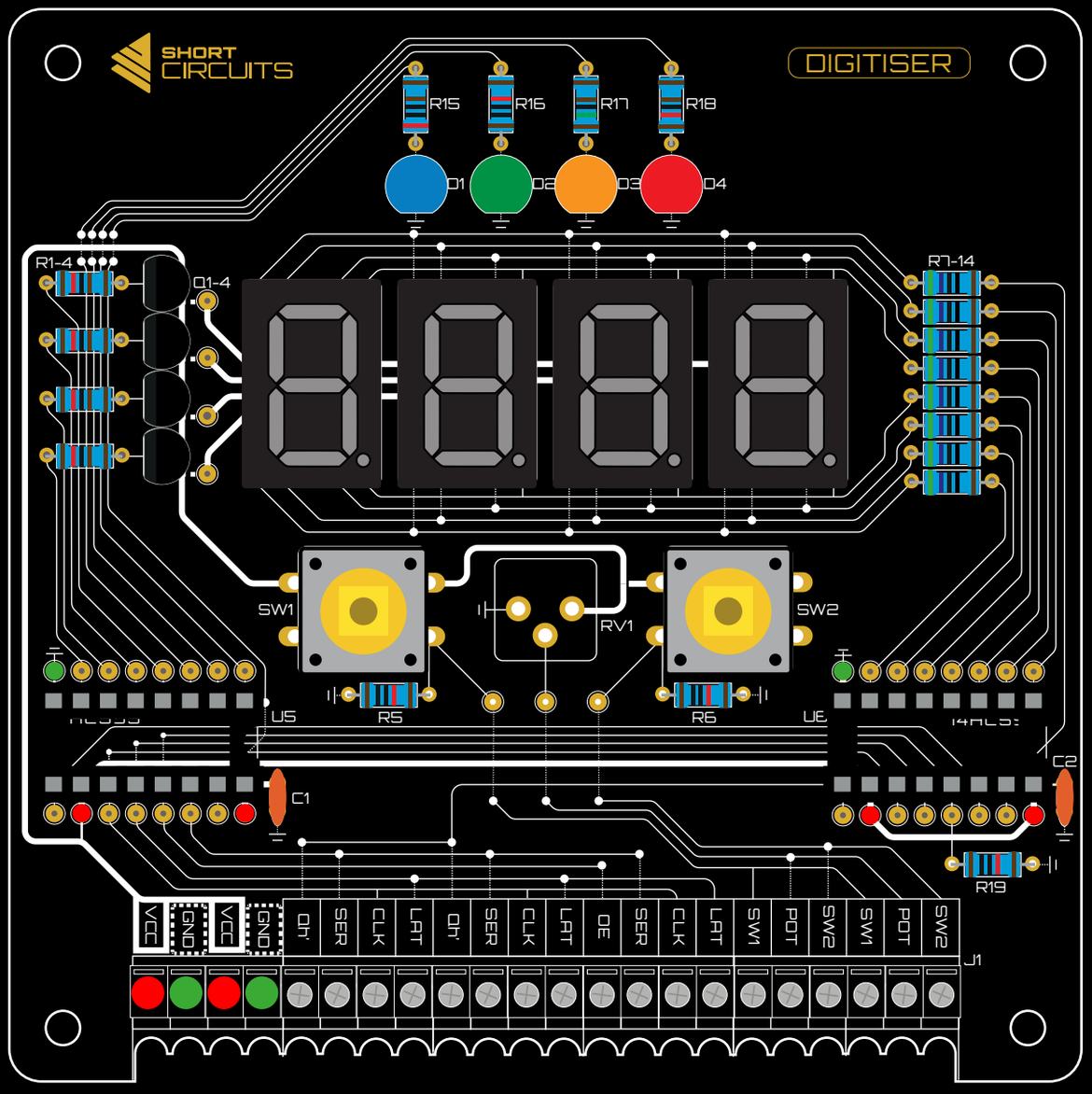
Visual Check

Firstly we need to do a visual check of the back of the board. We are looking for solder bridges that connect two pads that aren't meant to be connected. A magnifying glass is a good tool to have when doing this. The most likely areas for this are highlighted with red boxes on the diagram of the back of the PCB.

If you see any solder bridges, bring your iron up to temperature and drag it between the two pads. You may have to repeat this a few times. Make sure your iron is clean or the solder won't cling to it.

Short Circuits

Now we need to check for short circuits (the bad kind). If we have a short circuit somewhere on the board and we connect it to power, we could damage something. So to check for shorts, get your multimeter and put it in continuity mode (•)) Make sure the black cable is plugged into the common socket and the red cable into the red socket that also has the Ω symbol on it. Touch the probes together and make sure it makes a sound. Now press the black probe on one of the green circles indicated in the diagram. These are all connected to the large ground plane on the underside of the board. Keep it held there while you press the red probe onto one of the red circles indicated in the diagram. The screw of the Vout terminal and the GND terminal (the metal screw is connected to ground) are the most convenient. Making sure they are both making contact with metal, listen for a sound from your multimeter. If there is none, excellent, you don't have a short between Vcc and GND. You can jump over to the Power Test. If you here a constant sound from your meter when the probes are in contact with Vout and GND then you have a short. Check the back of the board again for solder bridges. Focus on the areas marked with boxes on the diagram on the next page. This is where Vcc and GND are closest.

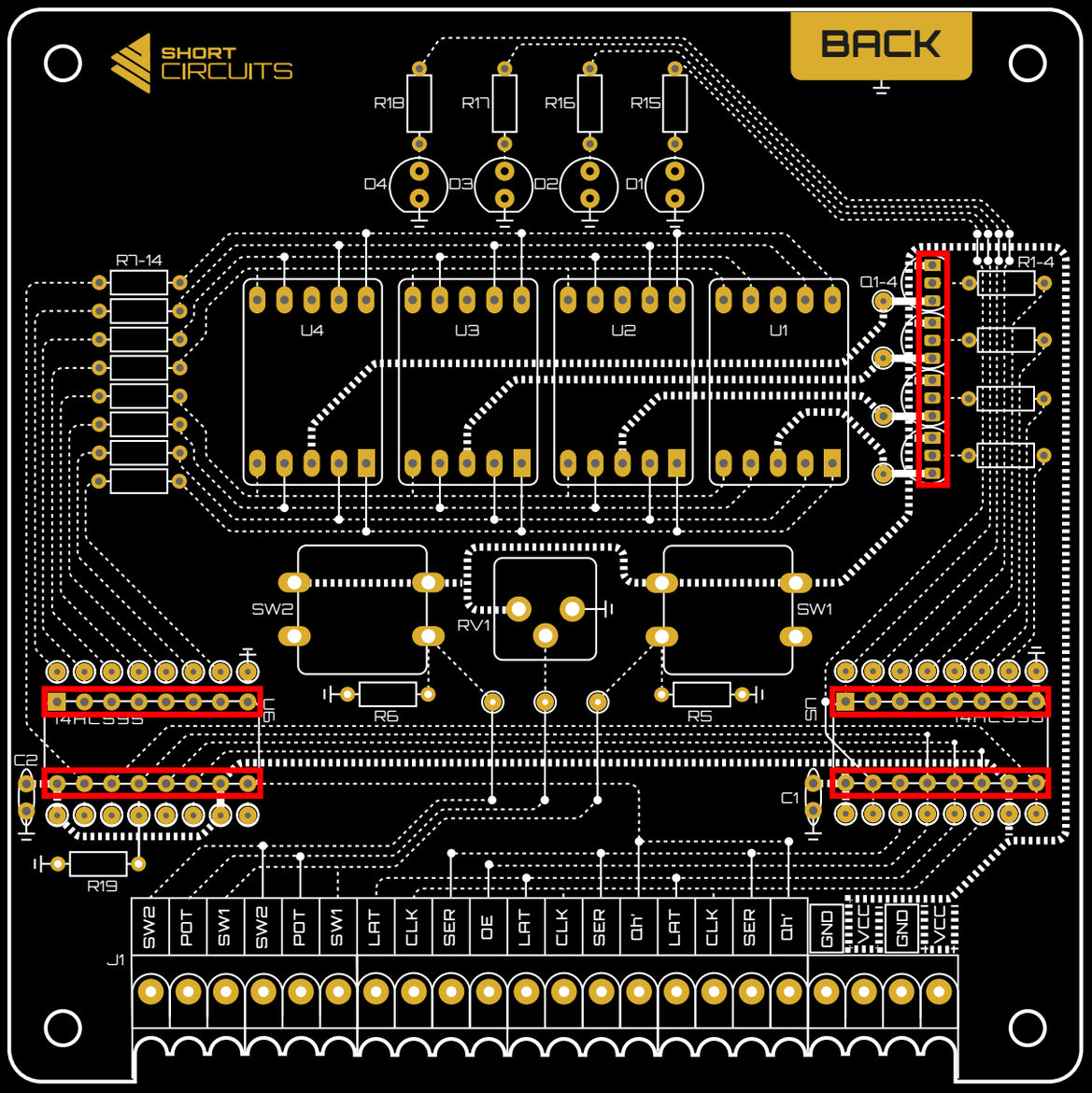


Polarity

Check all the components that are polarity dependent. In this circuit, that would be the LED's, the Transistors and the Shift Registers.

Powering the Board

We can now connect the Digitiser to the Motherboard. Connect VCC, and GND to the Motherboards Power Out terminals, and SER, CLK and LAT to any of the digital IO terminals (make note to change the code to reflect which pins you have connected the data lines to).



Power up the Motherboard via USB or with a 5V power adapter wired to its Power in Terminals. The PWR LED should light up on the Motherboard. If not, check the Motherboard's manual for tips.

Upload the code in the following chapter and check if all the LEDs and segments are lit.

If none of them are lit, check power is getting to the board. Set your multimeter to DC Voltage: (⎓). Put your red multimeter lead on VCC and the black lead on GND. The multimeter should read somewhere between 4.6V and 5V. If not, check the your VCC and GND connections. Recheck for shorts. Check polarities again. Check everything that should be connected is, and everything that shouldn't isn't.

Segments or LEDs not working

If a whole digit is not working, check the transistor it is connected to for solder bridges and follow the circuit back to the shift registers output pin.

If the same segment on each of the digits is not working, then check the resistor and the shift register output pin it is connected to.

If an LED isn't working, check its connection with its shift register output and try swapping LEDs to check the LED and the circuit at the same time.

If the potentiometer or the switches don't work, it is probably something to do with the code. Check for a bridge near the terminal blocks, then check your code.

Still having problems? Head over to the forums on our website for help and support. www.shortcircuits.cc

Coding Basics - Dimming and Switches

The following code example and explanation will test all of the Digitiser's functions and teach you the basics. We will cover the following:

- Using the potentiometer to brighten and dim the display
- Using a switch to make changes to the display
- Addressing shift register outputs to change what is displayed

The usual setup is required. This includes adding variables for pins, adding variables for other functions, assigning pins as outputs etc. Copy the code (with or without comments) to your sketch or download the full sketch HERE.

The pin variables tell the microprocessor which pins to address when the code executes. The other variables will handle the numbers we need to make everything work.

```
1  //Output Pin Variables
2
3  const int SER = 3;    // Shift Register Serial Data pin
4  const int CLK = 4;    // Shift Register Clock pin
5  const int LAT = 5;    // Shift Register Latch pin
6  const int OE = 10;    // Shift Register Output Enable pin
7
8  //Input Pin Variables
9
10 const int sw1 = A0;    // Switch 1 pin
11 const int Pot = A1;    // Potentiometer pin (must be analog pin)
12 const int sw2 = A2;    // Switch 2 pin
13
14 //Other Variables
15
16 int mode = 1;     // counter for Mode select
17 int swState = 0;     // Holds the current state of the switch
18 int lastswState = 0;     // Holds the previous state of the switch
19 int potRead = 0;     // Holds the current potentiometer value (0-1024)
20 int potDimVal = 0;     // Holds the converted dimming value for output to OE (0-255)
21
22 //Void Setup - runs once at start
23
24 void setup()
25 {
26     pinMode(OE, OUTPUT);
27     pinMode(SER, OUTPUT);
28     pinMode(CLK, OUTPUT);
29     pinMode(LAT, OUTPUT);
30
31     analogWrite(OE, 0);
32 }
```

If you need clarification on this section of the code, please refer to the Coding Basics section of the Motherboard's Manual. To help you understand the code, comments have been added. When the Arduino IDE compiles the code ready to transfer it to the microcontroller, it ignores everything after // on a single line and between /* and */ for multiple line comments.

Potentiometer Dimmer

At the beginning of our loop we will set our dimming value using the potentiometer. We first store the current value read at the pin our Potentiometer is connected to. We store this value as potRead (line 41). We then need to convert this value to something we can send to our OE pin. Analog pins return a value between 0 and 1023. Our PWM pins use a value from 0 to 255. To "map" these values we use the map function (line 42).

map(value,fromLow,fromHigh,toLow,toHigh)

The map function expects 5 values between its brackets. The first, value, is the input value you are mapping. In this case, the reading from the potentiometer, potRead.

We then need to set the range of this value. As the analog pin returns a value from 0-1023 we will use 0 as the low value and 1023 as the high value.

The function then expects the range that this will be mapped to. This will be the PWM values between 0 and 255. As you can see, we have used these in reverse. This is because the OE pin of the shift register is active low. So 0V or 0 will be fully on, and 5V or 255 will be fully off.

Now that potDimVal is equal to the mapped value of potRead, we can use this to set our

```
33
34  //Void Loop - repeats forever
35
36  void loop()
37  {
38
39  //DIM DISPLAY USING POTENTIOMETER
40
41  potRead = analogRead(Pot);
42  potDimVal = map(potRead,0,1023,255,0);
43  analogWrite(OE,potDimVal);   // Set brightness
44
45  //MODE SWITCHING
46
47  swState = digitalRead(sw1);   //read the pushbutton input pin
48
49  if (swState != lastswState)   // compare swState to its previous state, if it changed...
50     {
51     if (swState == HIGH)   // ...and the current state is HIGH then the button was pressed
52        {
53           if (mode < 4)   // if mode is less than the number of modes
54           {
55              mode++;   // add 1 to mode, thus changing to the next mode
56           }
57           else   // if mode is not less than the number of modes...
58           {
59              mode = 1;   // ...set mode to 0 (the first mode)
60           }
61        }
62        delay(5);   // Delay a little bit to avoid bouncing
63     }
64
65  lastswState = swState;   // save the current state as the last state
66
```

PWM value for the shift register's OE pin (line 43). Here we use the function analogWrite, which requires the pin, followed by a value (from 0-255). Our pin is labeled OE and the value has been saved in the potDimVal variable.

Mode Switching

To display more information on the Digitiser's display than just the time, we will use one of the switches to act as a mode select.

When the switch is pressed, the number stored in the variable "mode" will change. Later in the code we will use this number to determine what is displayed.

If we told the microcontroller to do something every time the switch read HIGH (when it is pressed down), that something would happen every time the code looped. As the code loops several hundred times per second, our code would switch modes at that speed. Obviously not ideal.

For our code to recognise that the switch has been pressed once, regardless of whether it is still pressed, we need to compare its state, with the state it was in the last time void loop lopped. If there is no change, then nothing will happen.

We don't want the mode to change when we let go of the switch either, so we should only change the mode when the state has changed and reads HIGH. Both things need to be true for the mode to change.

Let's look at the code. On line 47 we simply save the current state of our switch's input pin to the swState variable.

We then use an if statement to compare the current state swState to the last state saved to lastswState. Obviously we haven't got to the part of the code that saves the last state, but bear in mind that once you've switched the device on, it's gone through this code lots and lots of times. The condition of our if statement is that these two values are not the same. We use the Not Equal To operator (!=). So, if they are not equal, the code contained within the subsequent { } will be executed.

As we have another condition to check, we need to use another if statement within the last (line 51). This if statement needs to make sure the current state is high. To check if something is equal to something else we

use the equal to operator (==). If this is true, the code withing the if statement's brackets will execute.

We have 4 modes to cycle through. So we need the value to increment 3 times, from 1 to 2 to 3 to 4, then reset to its original value of 1.

At the beginning of our sketch we set mode to equal 1. We will now add 1 to mode but only if it is less than 4. We will use an if statement (line 53) to do this. If mode is less than (<) 4 the statement will be true. If it is true it will add 1 to mode. We add 1 to a variable by writing the name of the variable followed by the increment operator (++). If we wanted it to decrement then we would use the decrement operator (--).

If it's false (it has already reached 4) we need to reset it to 1. To do this we follow our last if statement with an else statement (line 57). If mode is less than 4 add 1 to mode, else, mode equals 1 (line 59). We only use one = when we are replacing a value with another.

After we have closed the if statement and the else statement's brackets (lines 60-61), we can add a short delay (line 62) to stop any "bouncing". As the microcontroller is looping through void loop() so quickly, the code could sense minute disconnects and reconnects when you push the button. A short delay will avoid these being read as a press of the switch.

We can now close the first if statement (line 63) and set the lastswState variable to equal the current one (line 65), ready for the next time around the loop.

We can now send data to our shift registers depending on which number is stored in the mode variable. We can simply use an if statement to hold the code for each mode (lines 69, 76, 83, 90). If mode equals 1 then do this. If mode equals 2 then do this other thing etc.

```
67  //Shift Out Digits
68
69  if (mode == 1)
70    {
71      digitalWrite(LAT,LOW);
72      shiftOut(SER,CLK,LSBFIRST,B11101011);
73      shiftOut(SER,CLK,LSBFIRST,B10001110);
74      digitalWrite(LAT,HIGH);
75    }
76
77  if (mode == 2)
78    {
79      digitalWrite(LAT,LOW);
80      shiftOut(SER,CLK,LSBFIRST,B01001100);
81      shiftOut(SER,CLK,LSBFIRST,B01001101);
82      digitalWrite(LAT,HIGH);
83    }
84
85  if (mode == 3)
86    {
87      digitalWrite(LAT,LOW);
88      shiftOut(SER,CLK,LSBFIRST,B01001001);
89      shiftOut(SER,CLK,LSBFIRST,B00101011);
90      digitalWrite(LAT,HIGH);
91    }
92
93  if (mode == 4)
94    {
95      digitalWrite(LAT,LOW);
96      shiftOut(SER,CLK,LSBFIRST,B00101011);
97      shiftOut(SER,CLK,LSBFIRST,B00010111);
98      digitalWrite(LAT,HIGH);
99    }
100 }
```

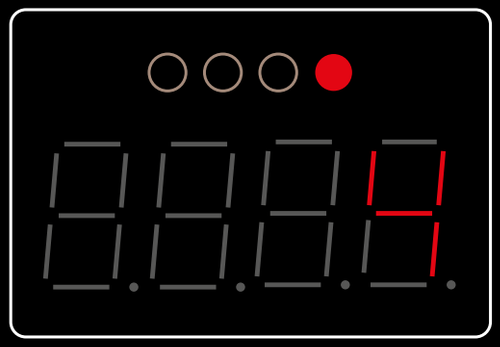
Shifting Out

Shifting data to the shift registers is pretty simple. First we need to set the Latch pin low so the register can accept data (line 71). We learned how to set a pin high or low in the Motherboard manual, so this should be easy.

We then use the function ShiftOut which is a function written into the Arduino IDE to help us interface with shift registers. The code is actually pretty simple. It shifts out a byte of data, one bit at a time. Each bit is shifted into the register through SER after each clock

```
shiftOut(SER,CLK,LSBFIRST,B00101011);
shiftOut(SER,CLK,LSBFIRST,B10000111);
```

Actual Output



0 0 0 1 0 1 1 1

pulse is sent through CLK.

ShiftOut requires 4 things to function. It needs the Serial data pin, the clock pin, either MSBFIRST (Most Significant Bit First) or LSBFIRST (Least significant Bit First) followed by a byte of data (8 bits).

The first two are easy, we can just add the names we assigned for those pins.

MSBFIRST and its counterpart determine which order the bits are shifted into the register. The most significant bit is the one with the most value, which is the one on the left. The least significant bit is on the right. The first bit in will end up on the last output of the shift register. You can use either, but the bit order does matter. Check out the diagram above to help visualise this.

We have 2 shift registers chained together, that's why we are using ShiftOut twice. This further complicates things, but the diagram above should straighten that out.

Using the diagram we can determine the byte needed to show each number from 0-9. Remember, in our circuit, to turn a segment on we use a 0. To turn a digit on we use a

U5

Shift Register Output ON	Shift Register Output	Function	Diagram
1	A	LED1	
1	B	LED2	
1	C	LED3	
1	D	LED4	
0	E	Digit 4	
0	F	Digit 3	
0	G	Digit 2	
0	H	Digit 1	

U6

Shift Register Output ON	Shift Register Output	Segment Reference	Diagram
0	A	G	
0	B	F	
1	C	A	
0	D	B	
1	E	DP	
0	F	C	
1	G	D	
1	H	E	

0. To turn an LED on we use a 1. Sending the following bytes in the second ShifOut will result in the display showing the corresponding numbers:

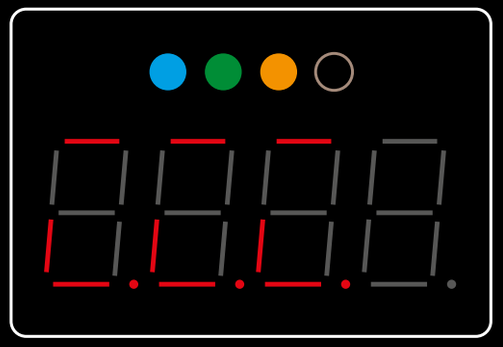
0 = B10001000	5 = B00011001
1 = B11101011	6 = B00111000
2 = B01001100	7 = B11001011
3 = B01001001	8 = B00001000
4 = B00101011	9 = B00001011

To keep things simple in our example, we will show numbers 1 to 4 on digits 1 to 4 when cycling through modes 1 to 4. We will also turn on the corresponding mode LED.

```
shiftOut(SER,CLK,MSBFIRST,B00101011);
shiftOut(SER,CLK,MSBFIRST,B10000111);
```

1 1 1 0 1 0 0 0

Actual Output



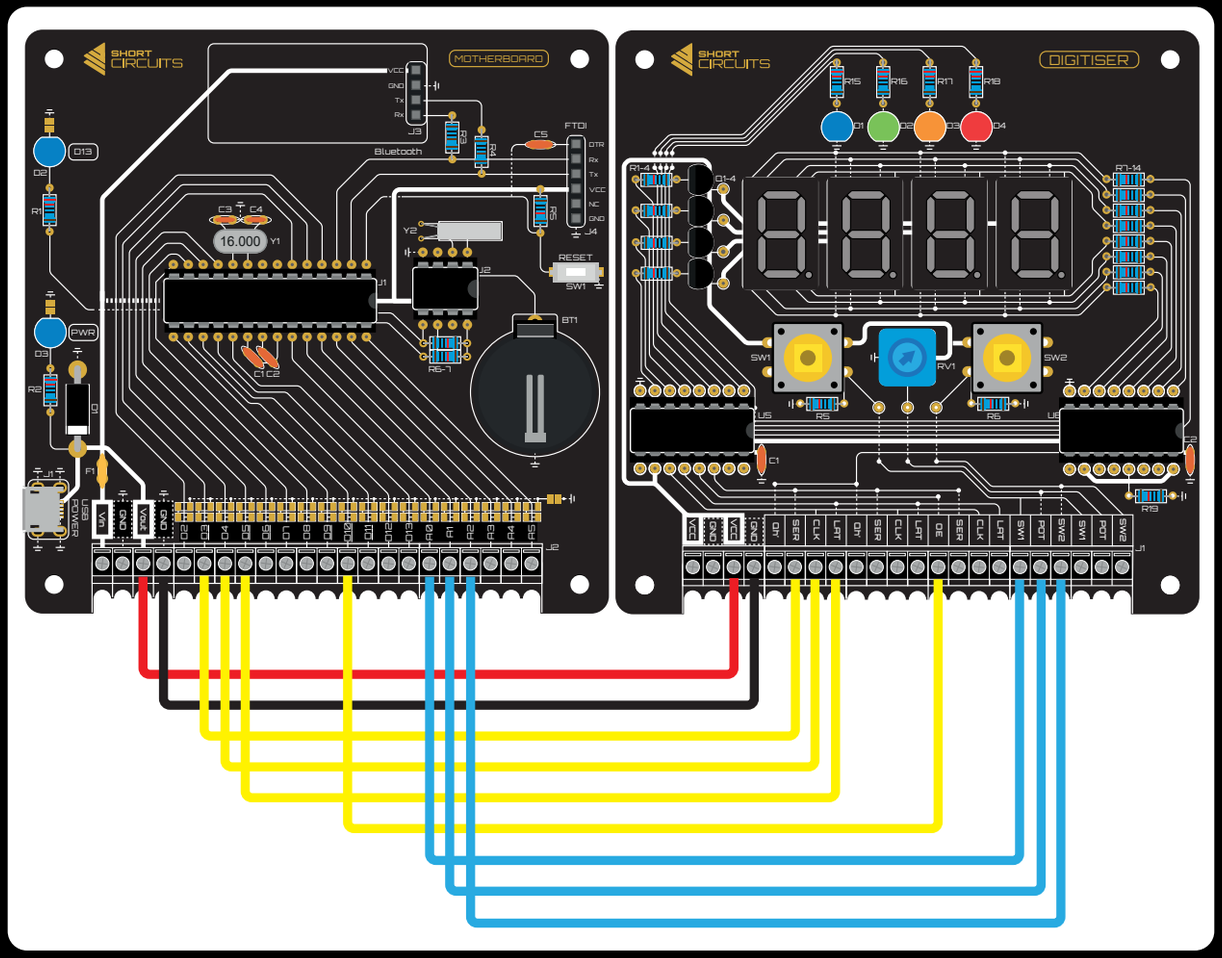
We already have the second ShiftOut line of code sorted (just pick a number and write the byte!). We need to make sure the right number is showing on the right digit. Using the table above, we can work out which bits to send in the first ShiftOut for each mode. if mode equals 1 we want digit 1 and LED 1 to be on. If mode equals 2 we want digit 2 and LED 2 on etc. Also, remember that digits turn on with a 0 and LEDs turn on with a 1.

Mode 1:
```
shiftOut(SER,CLK,LSBFIRST,B10001110);
```
Mode 2:
```
shiftOut(SER,CLK,LSBFIRST,B01001101);
```
Mode 3:
```
shiftOut(SER,CLK,LSBFIRST,B00101011);
```
Mode 4:
```
shiftOut(SER,CLK,LSBFIRST,B00010111);
```

Once the data has been shifted, we send the Latch high to prevent any further data going through. Close each of the if statements and the void loop and you have a completed sketch. Upload it to your motherboard with the Digitiser connected to the correct Motherboard pins and play around with it.

Check out the other sketches in our forums. More coding guides coming soon!

Project Ideas - Clock



To build a digital clock capable of displaying the time, date and year, you will need the MOTHERBOARD and the DIGITISER kits.

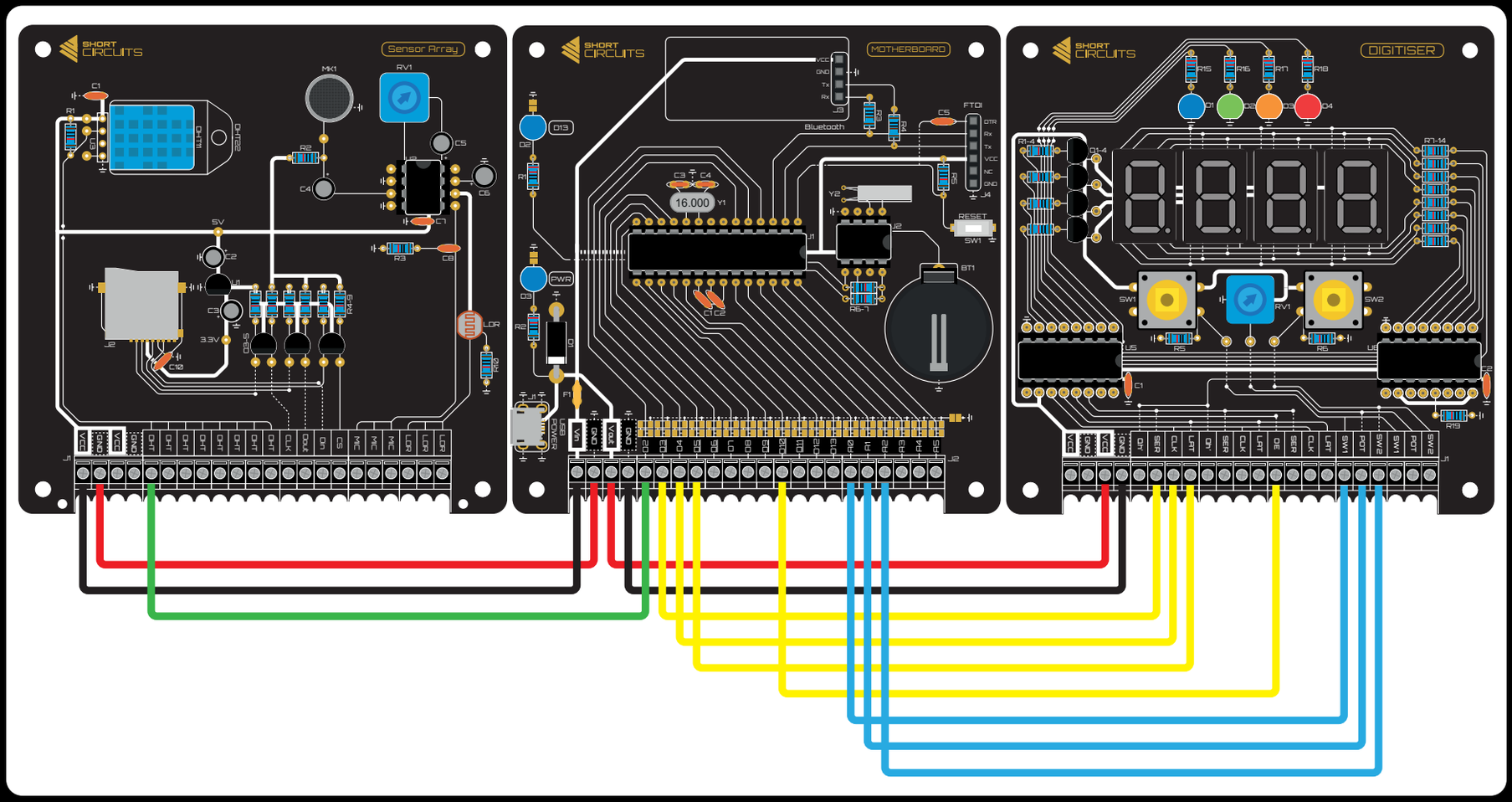
Connect the DIGITISER up to the MOTHERBOARD as shown in the diagram. If you are going to stack the boards without a case, then the wires need to be as short as possible. A good way to manage this is to insert a wire into one of the screw terminals on the lower board, then stack the boards using the provided standoffs. Cut the wire to length while offering it up to the screw terminal directly above it. Now remove the sheathing from the tip of the wire and push it firmly into the screw terminal. Make sure none of the strands of wire are hanging out the edges. Remove any extra slack if the wire is too long. If you are happy with the length, you can use it as a guide when cutting all the other wires.

If you are going to panel the boards, or stack them in a case, then leave a decent amount of slack. Forward planning is key here. Assemble it without wires, and take some measurements to make sure.

Once connected, head over to the forums and download the DigiClock sketch and upload it using the Arduino IDE and the instructions in the Programming section of the Motherboard's manual.

Remember to put a battery in the holder (BT1) of the Motherboard so the DS1307 can remember the time if you unplug the power.

Project Ideas - Environment Display



Adding the SENSOR ARRAY to the previous build enables you to add temperature and relative humidity to the digital display. You can add sound levels and light levels too, but we're sticking with a simple clock and environment sensor display in this project.

If you are stacking these, you will get better results if the Sensor Array faces outwards. This will expose the DHT11 sensor to the outside world, and minimise the effect of heat from the other boards on the sensor. If the board is flipped, it may make sense to move the connections around a bit, for neatness and to use less wire. Make sure you change the pin references at the beginning of the code if you do this.

Wire the boards as shown in the diagram above. You can switch outputs on the SENSOR ARRAY and DIGITISER as long as you switch to an IO port that has the same label. Switching outputs on the MOTHERBOARD is fine, just make sure the Potentiometer is connected to an analog pin, and the Output Enable pin is connected to a PWM pin. Change the code to match these changes.

Either write your own code from the hints in each kit's manual, or head over to the forums and grab the EnvironmentDisplay sketch. Upload the sketch using the Arduino IDE and make sure all the functions work. If they don't, check the pin references at the beginning on the code.

Play around with the variables and code to better understand it. Then you can make changes to suit your requirements.

Component Index

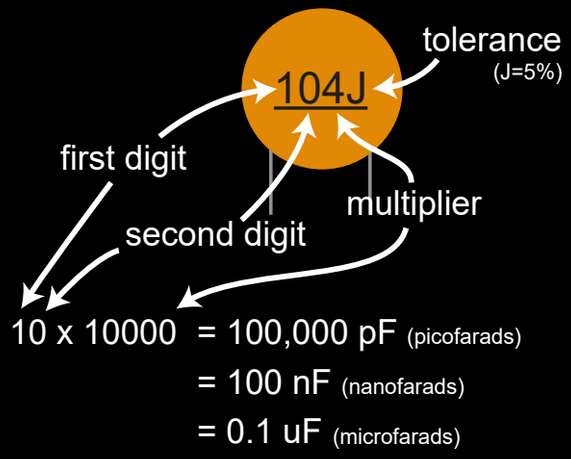
Here you will find information about each component that this kit includes. We have included some of the different sizes and shapes you may find out in the wilderness, different uses for each component, and important data that can be found in datasheets to help you design circuits around them. We have also outlined what happens when these components go pop and how to diagnose and replace them. This information can be used to fix your household appliances, rather than throwing them away. Make do and mend, as they used to say!

Capacitor - Ceramic

Overview

Capacitors are so named for their ability to store a certain capacity of electrical energy (Capacitance), measured in farads (F). A certain amount of capacitance exists between any two conductors that are in close proximity (this can sometimes be seen in an LED matrix in the form of ghosting, and can also cause problems in other sensitive circuits). Capacitors use this in a controlled manner, for various purposes. They usually consist of two conductors separated by a dielectric substance. A dielectric is an insulator (does not conduct electricity) that can be polarised (negative on one side and positive on the other) by an electric field. In this case, the dielectric material is ceramic. A dielectric substance increases the amount of electrical energy that can be stored compared to non-dielectric substances like air.

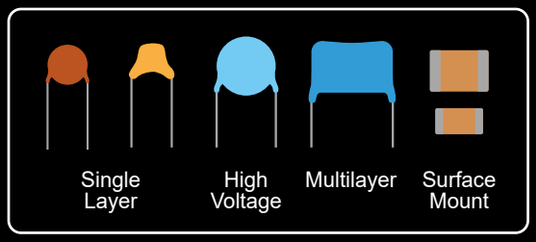
Identification



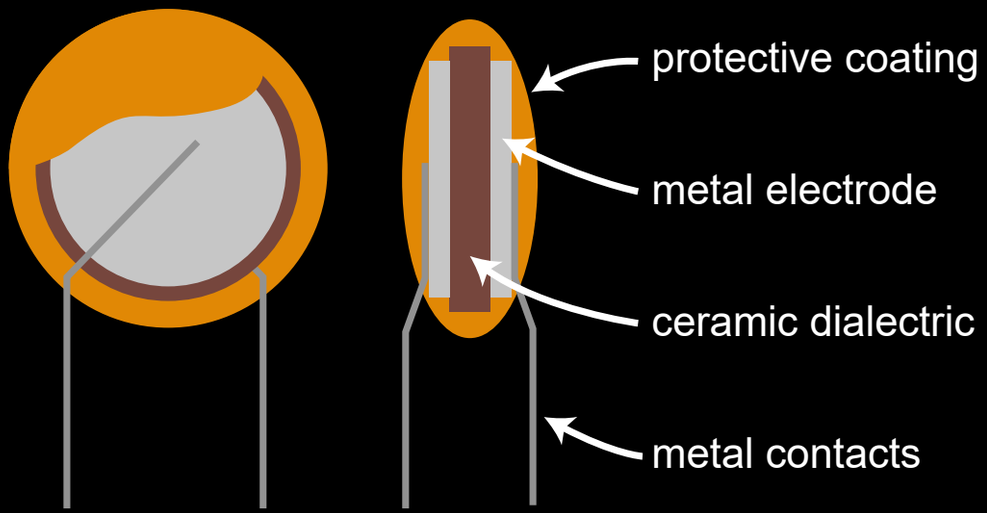
Quick Reference

Check Polarity	✗
Positions	2
Type	passive
Schematic Symbol	⊣⊢
PCB Symbol	
Designator	C

Common Varieties



Physical Construction



Important Ratings

Parameter	Typical Values
Capacitance	0.5pF - 1uF (SL) 1pF - 470uF (ML)
Capacitance Tolerance	± 5 - 20%
DC Rated Voltage	10V - 20kV
Pitch	2.54 - 12.7mm

SL = Single Layer
ML = Multilayer

COMPONENTS

Capacitor - Ceramic

Troubleshooting

Unlike electrolytic capacitors, ceramic capacitors rarely fail in normal use. However, if the voltage exceeds the datasheet's recommended maximum values (breakdown voltage), they can and will produce magic smoke. If they look burnt, or smell burnt, then the charge may have arced across the dielectric layer and will have overheated considerably.

A simple test to see if the capacitor is functioning as it should is to measure it's resistance. As the capacitor charges, the resistance will increase. Before all tests make sure you discharge the capacitor by bridging the leads with a screwdriver. Set your multimeter to Ohms and attach your multimeter probes to each lead. If the resistance climbs to infinity, then the capacitor is functioning as it should. If the resistance reads 0 then the dielectric layer has been compromised and the capacitor will need replacing. This method unfortunately doesn't check its capacitance.

The only reliable way to test a capacitors capacitance is to remove it from the circuit and test it with a multimeter capable of reading capacitance. Set your multimeter to capacitance (Look for the ⊣⊢ symbol) and connect the multimeter to the legs of the capacitor. If the value is no longer within the stated tolerance, replace it.

Common Uses

Decoupling

Capacitors are often used to protect certain components from interference from other parts of the circuit. Most common applications are right next to any integrated circuit (IC). The capacitor acts as a storage reserve. If the voltage drops below the required voltage, the capacitor will use it's stored energy to make up the difference. If the voltage increases above the required voltage, the capacitor absorbs the excess voltage. The microcontroller, or other sensitive device will see a much more even voltage because of this. Decoupling capacitors are placed as close to the sensitive parts of the circuit as possible, for maximum effectiveness. To smooth higher frequencies, you would use a low value capacitor (like a 10nF - 0.1uF ceramic capacitor). To smooth the lower frequency noise, a higher value would be used (often a 1-10uF electrolytic capacitor). Decoupling capacitors feature in most of our kits.

ATMega328P's Decoupling Capacitors

Filtering

Unlike resistors, who's resistance stays constant no mater what frequency of signal that's passing through it, Capacitors are reactive devices that resist higher frequencies less and lower frequencies more. Because of this, they will block DC signals (very low frequencies) and allow AC signals (alternating at higher frequencies) through. This is useful when you need AC and DC in a circuit, but only want AC signals in a certain part. A common example of this is a microphone circuit. The microphone needs a DC signal to power the device, but records AC signals (sound) from the environment. When we pass the AC noise through to an amplifier circuit, we want the AC signal, but not the DC. Adding a capacitor in series will remove the unwanted DC signal. (See the Sensor Array for a working example of this)

AC Filter on the Sensor Array (electrolytic Capacitor)

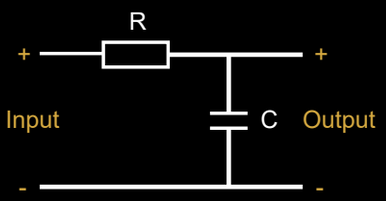
Capacitor - Ceramic

Another form of filter that uses capacitors is an RC filter. The most common RC filters are low-pass and high pass filters. As the name suggests, the low-pass filter lets low frequency signals pass but not high frequencies. High pass filters simply achieve the opposite. This is a passive filter as it uses passive components (resistors and capacitors). The following are the simplest of low-pass and high-pass filters, the equation that governs their properties and a graph to show the typical relationship between frequency (Hz) and amplitude (given in volts).

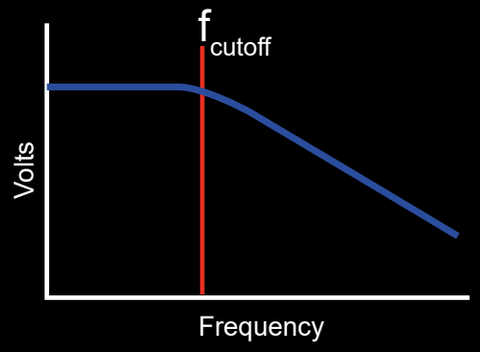
Time Delay

In an RC circuit, capacitors take time to reach their maximum store of electrical energy when exposed to a voltage source, and to deplete that store when the voltage source is removed. This can be used to create a time delay between turning on a voltage supply and a component receiving the voltage it needs to turn on, or read a logic level high for on a microcontroller for example. We can use a simple equation to work out how much time it will take the capacitor to reach approximately 63% of the supply voltage. This is referred to as the RC time constant and uses the symbol tau (T). In the example circuit the LED will gradually get brighter until the capacitor reaches capacity, following the curve of the graph.

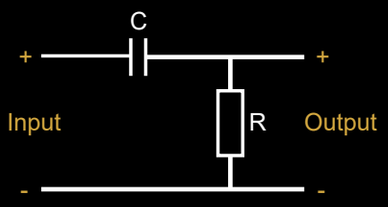
Low-Pass Filter



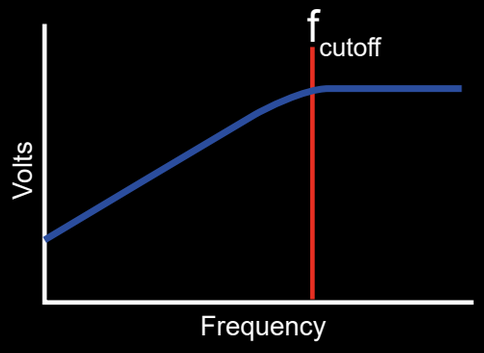
$$f_{cutoff} = \frac{1}{2\pi RC}$$



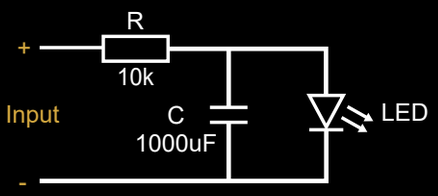
High-Pass Filter



$$f_{cutoff} = \frac{1}{2\pi RC}$$



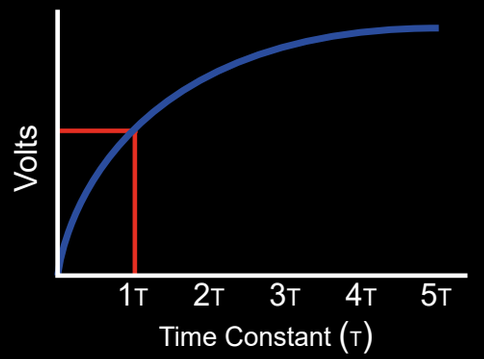
Time Delay Circuit



T(seconds) = R(Ohms) x C(farads)

T = 10,000 x 0.001

T = 10 seconds



COMPONENTS

Diode - LED

Overview

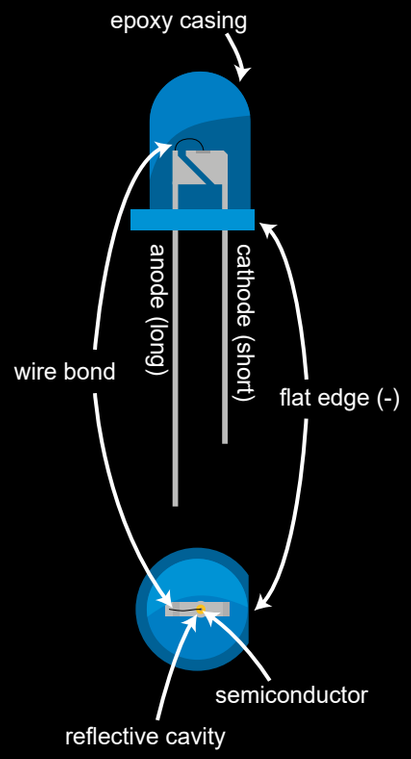
An LED is a Light Emitting Diode. Like all diodes, they are polarity sensitive. They are semiconductor light sources that emit light when current flows through them. Different semiconductor materials produce photons with different amounts of energy and so different colours. The structure of the LED is designed to emit light efficiently using a reflective cavity and shaping the case to act as a lens.

Troubleshooting

Fortunately it's easy to know when an LED isn't working to spec. If the polarity is correct, the voltage is sufficient, the series resistor is the correct value, and it doesn't light up, it's most probably toast. Here's how to calculate the value for the series resistor, with an example:

$$R = \frac{V_{source} - VF}{IF} \qquad R = \frac{5 - 3}{0.02} \qquad R = 100\Omega$$

Physical Construction



Quick Reference

Check Polarity	✓
Positions	2
Type	active
Schematic Symbol	
PCB Symbol	
Designator	□

Common Varieties


Through Hole


Surface Mount

Important Ratings

Parameter	Typical Values
Forward Voltage (VF)	1.6 - 36V+ (typ)
Forward Current (IF)	30mA (max)
Reverse Voltage	5V
Power Dissipation	100mW
Luminous Intensity	0.4mcd - 700cd
Wavelength	280 - 800nm

Resistor

Overview

Resistors do exactly that, they resist the flow of electrons through them. This resistance, measured in Ohms (Ω), is fixed and can be relied on to compliment integrated circuits, divide voltages and protect other components from too much current. But due to the law of conservation of energy, this energy needs to go somewhere. In this case, it is converted into heat. This is why it is important to take note of the power rating of resistors, measured in Watts (W).

Resistors can be made out of many materials, but most commonly metal or carbon film. the film is wrapped around a ceramic core and the whole thing is protected by an insulated layer, usually cream, or blue in colour.

Troubleshooting

If a resistor is bad you can usually tell. It likely went up in a puff of grey smoke. as soon as you exceed the resistors power rating, it's going to get hot. Fortunately they are cheap and easy to replace. and easy to troubleshoot. Keep in mind that resistance cannot be measured in an operating circuit. Voltage and Current can however. So you could use Ohms law to calculate the resistance.

1. Remove one lead from the circuit
2. Turn your multimeter to its resistance setting
3. Set your multimeter to the lowest range that exceeds the value of the resistor
4. Place the multimeter's probes on each of the resistors leads (polarity doesn't matter) and note the reading.
5. If the value is outside the range of tolerance, then the resistor is bad and needs replacing.

Quick Reference

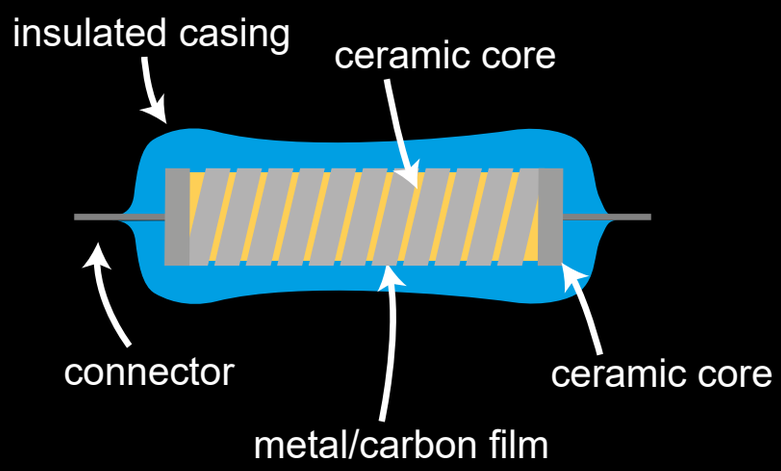
Check Polarity	✗
Positions	2
Type	passive
Schematic Symbol	
PCB Symbol	
Designator	R

Common Varieties

Through Hole

Surface Mount

Physical Construction



Important Ratings

Parameter	Typical Values
Resistance (R)	0Ω - 10GΩ
Power Rating (W)	0.1W - 250W
Tolerance (%)	±0.01% - ±10%
Temp Coefficient (ppm/°C)	±15 - 50ppm/°C
Max Voltage (V)	200V - 500V
Min Operating Temp (°C)	-70 - -25°C
Max Operating Temp (°C)	70 - 450°C

Resistor

Reading Resistor Values

Surface Mount

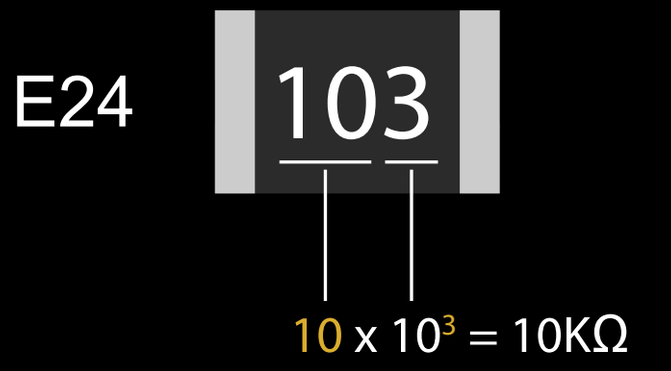
Surface mount resistors use a few coding systems. If you see just numbers, the resistor is probably using the E24 system. If it has a letter at the end then it is probably E96.

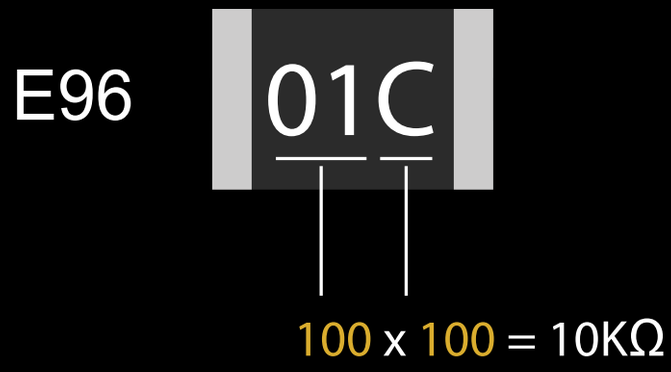
E24

The first two numbers of an E24 resistor represent the 2 most significant digits. The third number represents the magnitude (10 to the power of the third number).

E96

An E96 resistor starts with two numbers that can be looked up in the table, followed by a letter that can be looked up in the other table.



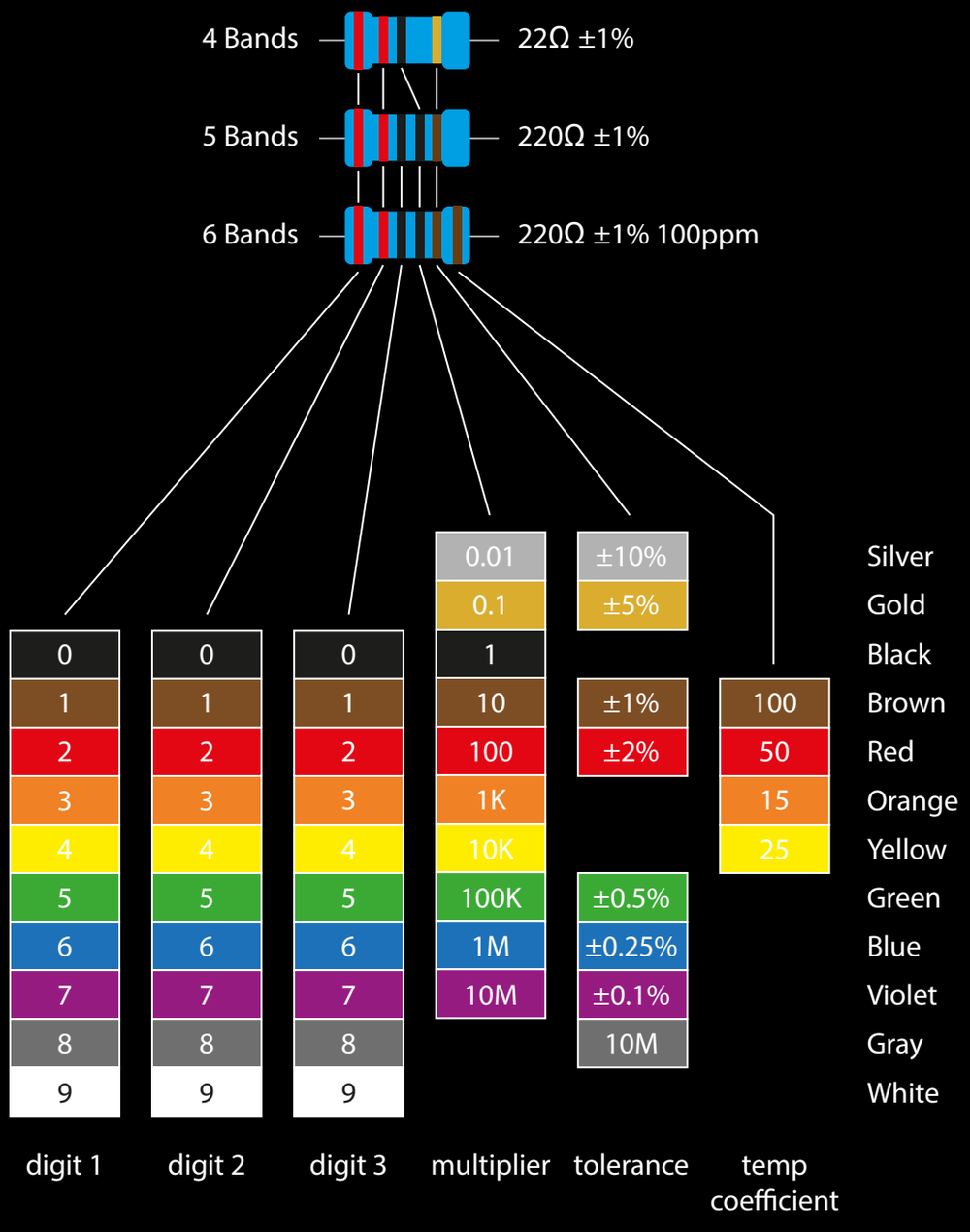


Code	Value	Code	Value	Code	Value	Code	Value	Code	Value	Code	Value
01	100	17	147	33	215	49	316	65	464	81	681
02	102	18	150	34	221	50	324	66	475	82	698
03	105	19	154	35	226	51	332	67	487	83	715
04	107	20	158	36	232	52	340	68	499	84	732
05	110	21	162	37	237	53	348	69	511	85	750
06	113	22	165	38	243	54	357	70	523	86	768
07	115	23	169	39	249	55	365	71	536	87	787
08	118	24	174	40	255	56	374	72	549	88	806
09	121	25	178	41	261	57	383	73	562	89	825
10	124	26	182	42	267	58	392	74	576	90	845
11	127	27	187	43	274	59	402	75	590	91	866
12	130	28	191	44	280	60	412	76	604	92	887
13	133	29	196	45	287	61	422	77	619	93	909
14	137	30	200	46	294	62	432	78	634	94	931
15	140	31	205	47	301	63	442	79	649	95	953
16	143	32	210	48	309	64	453	80	665	96	976

Code	Value
Z	0.001
Y or R	0.01
X or S	0.1
A	1
B or H	10
C	100
D	1000
E	10000
F	100000

Through Hole

Through hole resistors have 4 to 6 coloured bands which represent digits, a multiplier, tolerance and temperature coefficient. Use the following diagram to work out the resistors value.



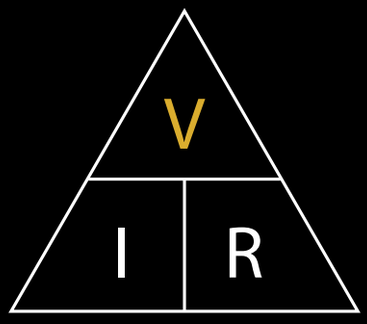
Resistor

Ohm's Law

Ohms law is the most basic and useful piece of maths used in electronics. We try to keep things maths free, but this one is unavoidable. It describes the relationship between Resistance (R), Voltage (V), and Current (I).

Ohm's law states that the current through a conductor between two points is directly proportional to the voltage across the two points. So, if the resistance stays the same, then as voltage increases, current decreases, and vise versa. If 2 of the three values (V, I and R) are known, you can easily work out the other using the following triangle.

Ohm's law can be used to calculate voltage drops across components, the current flowing through a circuit, the supply voltage, and the resistance across a component (although some components like an LED do not have a fixed resistance value). This can be useful when diagnosing problems in circuits. If the current is too high, maybe the resistance has dropped across a component for example.

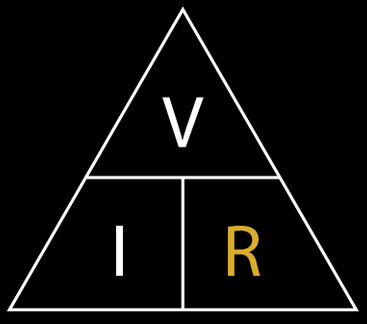


$$V = I \times R$$

Finding the voltage drop across an LED:
(supply voltage = sum of all voltage drops in a series circuit)

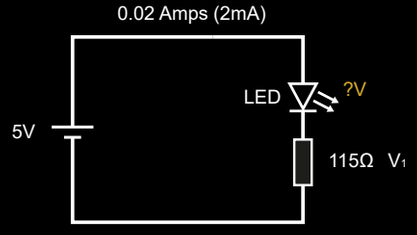
$$I = \frac{V}{R}$$

Finding the value of a current limiting resistor:



$$R = \frac{V}{I}$$

Finding the current draw in a series circuit:
(the current that flows through the resistor is the same as the current that flows through the LED
because they are in series)

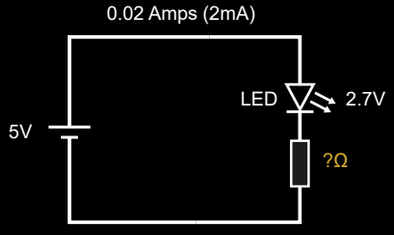


$$V_1 = 0.02 \times 115$$

$$V_1 = 2.3V$$

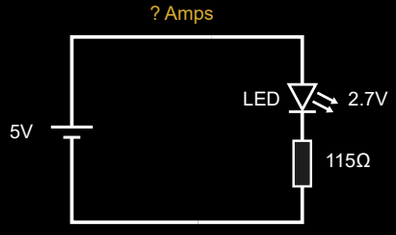
$$V = 5 - 2.3$$

$$V = 2.7V$$



$$R = \frac{5 - 2.7}{0.02}$$

$$R = 115$$



$$I = \frac{5 - 2.7}{115}$$

$$I = 115$$

Resistor - Potentiometer

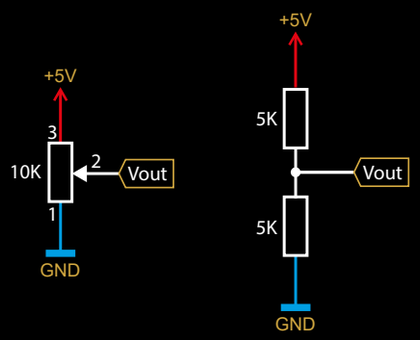
Overview

A potentiometer is a component that offers variable resistance, determined by the position of the wiper on a resistive track. Some potentiometers have twisting knobs, some are long and straight (think of a volume slide or cross fader on audio equipment).

They can be used to give an analog reading between 0V and Vcc. This function can be used to gradually change volume, speed or a multitude of other variables.

A potentiometer acts as an adjustable voltage divider. If you imagine a 10K potentiometer with the knob turned half way, you would in fact have a 5K resistor either side of the center pin. This would give you a value that is half of the supply voltage. In a 5V circuit, this would be 2.5V.

Potentiometer Voltage Divider



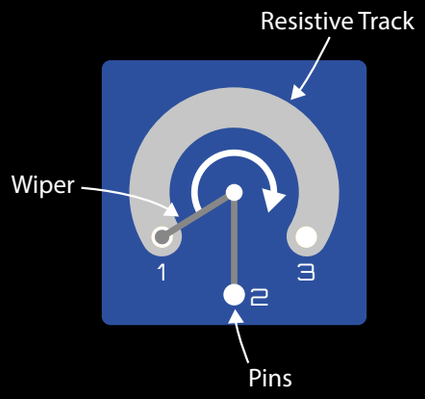
We can calculate Vout by using the voltage divider equation. This asks for three values, the input voltage and 2 resistor values.

$$Vout = Vin \times \frac{R2}{R1 + R2}$$

$$Vout = 5 \times \frac{5000}{5000 + 5000} \qquad Vout = 2.5$$

Physical Construction

As stated previously, a potentiometer consists of a resistive track with a wiper that moves along this track.



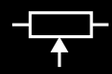
Troubleshooting

To test whether a potentiometer is working correctly, we can measure the resistance between the pins. First measure the resistance between pin 1 and 3, this should have a value close to the components stated value and within its stated tolerance. In our case, this would be 10K ohms. The potentiometer used in the Digitiser and Sensor Array kits have a tolerance of ±10%, so a value anywhere from 9K to 11K would be in tolerance.

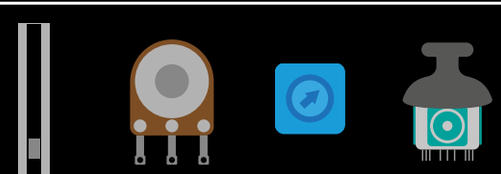
Now we can test the wiper. Turn or slide the pot all the way in any direction. Measure the resistance from pin 1 to 2. This should be nearly 0 ohms or nearly 10K ohms, depending which way you adjusted it. When you turn the pot all the way the other way, this should read nearly the opposite value.

How would a potentiometer fail? probably the same way any resistor would fail, if too much wattage passed through it, causing it to burn up. If this happened you would expect an open circuit between some of the pins. If there are any issues, swap the potentiometer out for another.

Quick Reference

Check Polarity	✗
Positions	3
Type	passive
Schematic Symbol	
PCB Symbol	
Designator	RV

Common Varieties



Slide Pot Classic Pot Trimmer Pot Joystick

Important Ratings

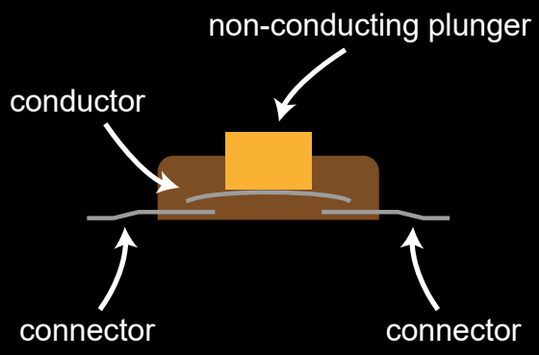
Parameter	Typical Values
Max Resistance	10-10mil Ohms
Minimum Resistance	2 Ohms Max
Resistance Tolerance	±1-50%
Power Rating	0.01-25W
Temp Range	-55 to 125˚C

Switch - Momentary (SPST)

Overview

A momentary switch, or tactile switch, is a switch that will only be closed when pushed. Removing your finger from the switch will open the circuit. As opposed to a switch that is flipped and stays in that position, like a light switch in your home.

Physical Construction



Troubleshooting

Most tactile switches have a dome shaped contact that will spring back after you let go of the switch. This can wear out over time and lose its click.

To test a switch, use a multimeter in continuity mode. Place a probe on each contact and press the switch. If the multimeter beeps, your switch is good.

Quick Reference

Check Polarity	X
Positions	2
Type	passive
Schematic Symbol	
PCB Symbol	
Designator	R

Important Ratings

Parameter	Typical Values
Contact Current Rating	5mA - 1A
Operating Force	0.5N - 6.5N
Min Operating Temp (°C)	-55 - -20°C
Max Operating Temp (°C)	70 - 160°C

Shift Register - 74HC595

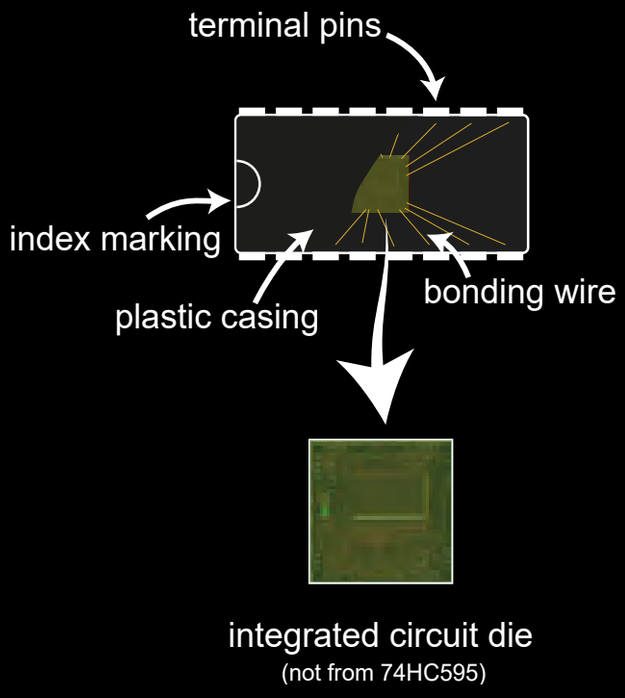
Overview

The 595 Shift Register is an 8-bit serial-in, parallel-out shift register. This means it takes 8 bits, one bit at a time as in input and shift those out at the same time as outputs. They are referred to as SIPO (Serial In, Parallel Out), rather than PISO, SISO, or PIPO.

Shift Registers are used to store data. They store data using flip flops / latches, which have two stable states, either on or off, which is of course represented by a logic level 1 or a 0. Flip flops are made from lots of transistors.

SIPO shift registers are great for reducing the number of microcontroller pins needed to control lots of outputs. A PISO shift register would be used to read multiple inputs on just a few microcontroller pins.

Physical Construction



integrated circuit die
(not from 74HC595)

Troubleshooting

The best way to troubleshoot an IC is to swap it out and see if the circuit works with a known working IC. If the circuit still doesn't work, then the fault is in a different component. If it does start working with the new IC, then the problem is fixed.

Check the pinout diagram and check each pin. The GND pin should obviously read 0V with a multimeter. The Vcc pin should read the same as your circuits voltage source. You would need an oscilloscope to read the serial, clock, serial out and latch pins, as these change voltage from 0 to Vcc many times a second. If the output pins are reading well below Vcc when turned on, they are probably supplying too much current. Reduce the load on the output.

If you don't have a spare IC to replace the potentially broken one, try desoldering the IC, or removing it from its socket. You can then build a simple circuit to test its functionality on a breadboard. Hook it up to a microcontroller programmed to send all 1's to the shift register, then test each output's voltage in respect to GND. If they all read around Vcc then they are OK. Now set the outputs to all 0's. Test them again. They should now all read 0V. If nothing changes, then there may be something wrong with the input lines, or the code. If they all change accept one, then there might be something wrong with the output. Either way, you will need to replace the IC.

How do these things break? Well, they will most likely break from using it outside of its recommended ratings. This could be from asking for too much current from a single output, or the total current from all outputs. It could also be from applying more voltage than the specifications allow.

Always check the max ratings and calculate the amount of current going through each output, and the total across the whole IC and compare these numbers with the datasheet.

If your circuit is asking too much of the shift register, look for one with more max current. There are many flavours of shift register to choose from. You could always use transistor switches to reduce the current draw, or lower the current for each output with a higher value current limiting resistor.

Important Ratings

Parameter	Min.	Typ.	Max.
Operating temperature			150°C
Supply Voltage (V_{cc})	-0.5	5V	7V
Input Clamp Current			±20mA
Output Clamp Current			±20mA
Continuous Output Current (Output)			±35mA
Continuous Current (Vcc or GND)			±70mA
High Level Input Voltage (4.5V Vcc)	3.15V		
Low Level Input Voltage (4.5V Vcc)			1.35V

Quick Reference

Check Polarity ✓

Positions 16

Type IC

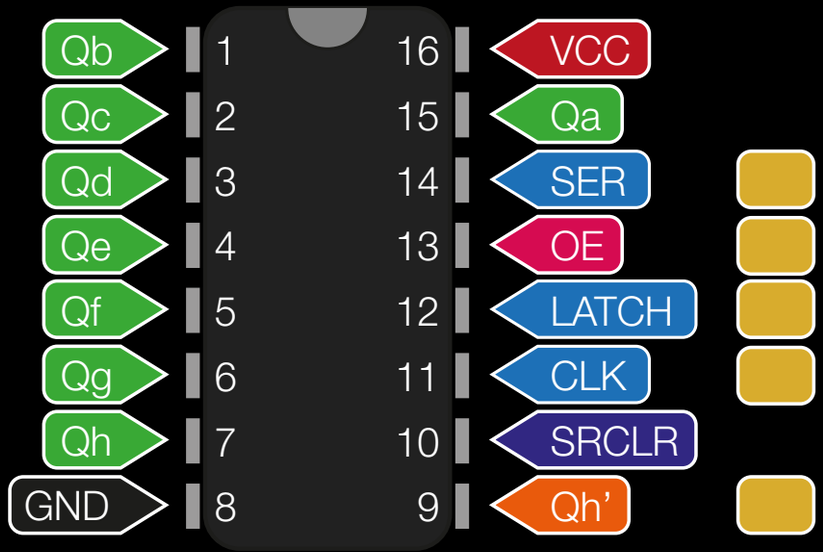
Schematic Symbol

PCB Symbol

Designator U

COMPONENTS

Shift Register - 74HC595

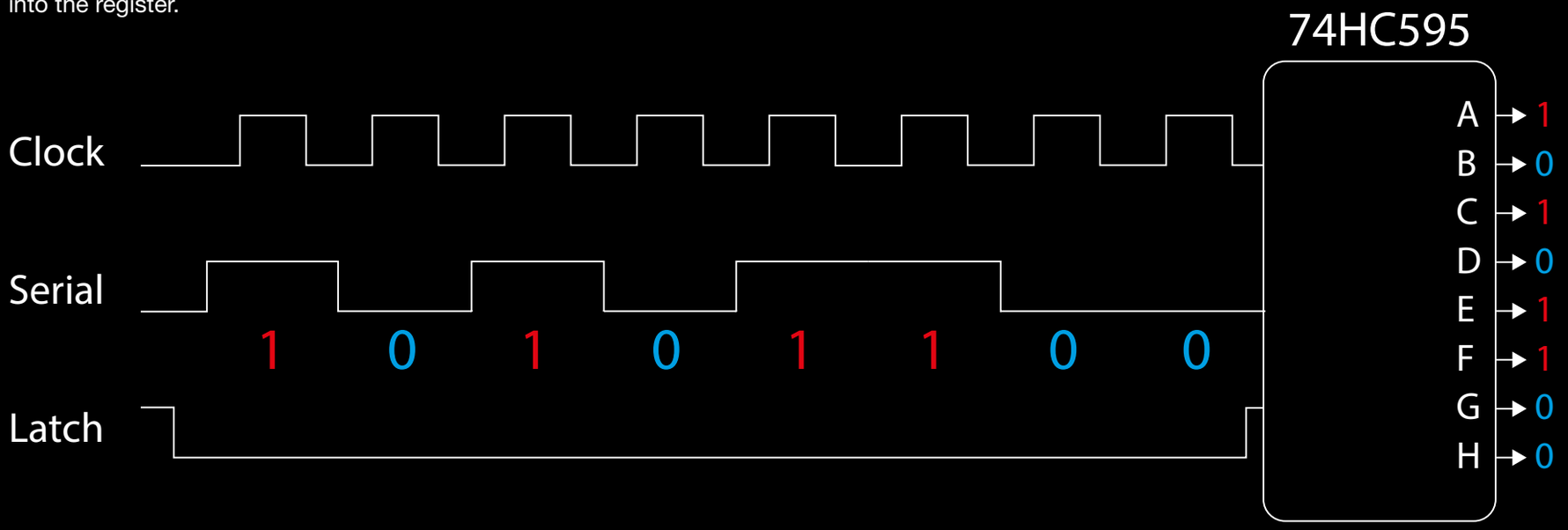


- Power
- Ground
- Serial Communication
- Output Pin
- Serial Output
- Shift Register Clear
- Output Enable
- Connected to Terminal Block (DIGITISER, RGB MATRIX)

Shift Register - 74HC595

Controlling the Shift Register

The 595 shift register can hold 8 bits of data, which are received through its Serial data line. When the Clock pin pulses, the current value of the Serial pin is shifted into the register. For the shift register to accept data, the Latch pin must be Low. Here is a diagram of the process of shifting 8 bits into the register.



ShiftOut

To achieve this shifting of bits in an Arduino sketch, we use the function ShiftOut(). This is already built in to the Arduino IDE, so you don't have to write it in your sketch. You can just call the function. For reference, the code it uses is something like the code opposite.

A for loop runs 8 times (once for each bit). It checks whether you are using LSBFIRST or MSBFIRST, then sets the Serial pin to the first bit of the byte depending on the order. It then sets the clock pin high then low to shift the bit in.

```
void shiftOut(uint8_t dataPin, uint8_t clockPin, uint8_t bitOrder, uint8_t val)
{

   uint8_t i;

   for (i = 0; i < 8; i++)
      {
         if (bitOrder == LSBFIRST)
            digitalWrite(dataPin, !!(val & (1 << i)));
         else
            digitalWrite(dataPin, !!(val & (1 << (7 - i))));

         digitalWrite(clockPin, HIGH);
         digitalWrite(clockPin, LOW);
      }
}
```

COMPONENTS

Seven Segment Display

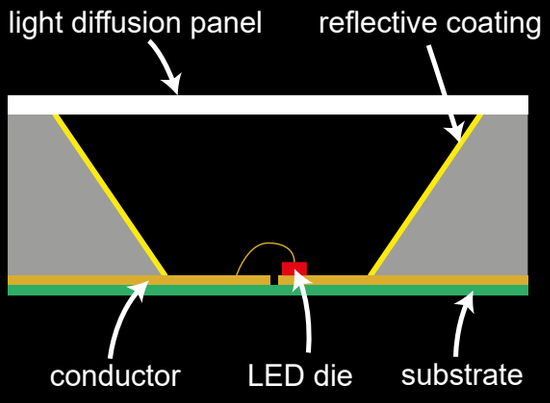
Overview

A 7 Segment display module is an array of LEDs in the shape of an eight. They are usually found in the single digit or 4 digit variety. Some have decimal points on each digit, and some have a colon between the second and third digit. They also come in a range of colours.

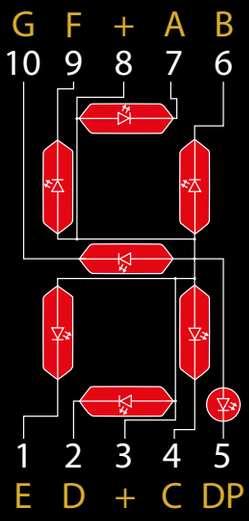
Segment Displays are available in common anode and common cathode varieties. Under each segment and decimal point lies an LED. If the display is common anode, each of the LED's anodes are connected together. Their cathodes are broken out to individual pins. If the display is common cathode, all the cathodes are linked and the anodes are broken out to pins separately.

Physical Construction

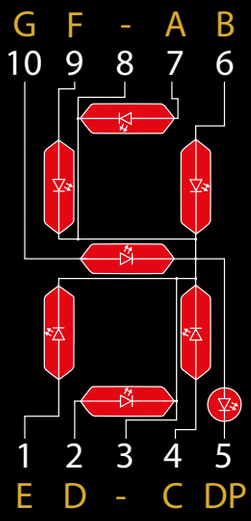
Each segment has a surface mount sized LED at the bottom the channel. Reflectors on the side of the channel reflect the light up towards the diffusion panel, which diffuses the light, giving it an even distribution. The LEDs are connected to conductors which are then connected to either the common anode or cathode, or a segments individual cathode or anode respectively.



Common Anode



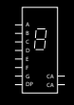
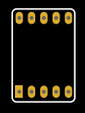
Common Cathode



Troubleshooting

7 segment displays are like any other LED when it comes the troubleshooting. Make sure you use a current limiting resistor (see the LED section for the calculations involved). Attach this to the common pin so all segments are protected. Then just apply the correct voltage to the common pin and the pin of the segment you are testing. Go through all the segments testing them. If any don't work, they have burnt out and you need another display.

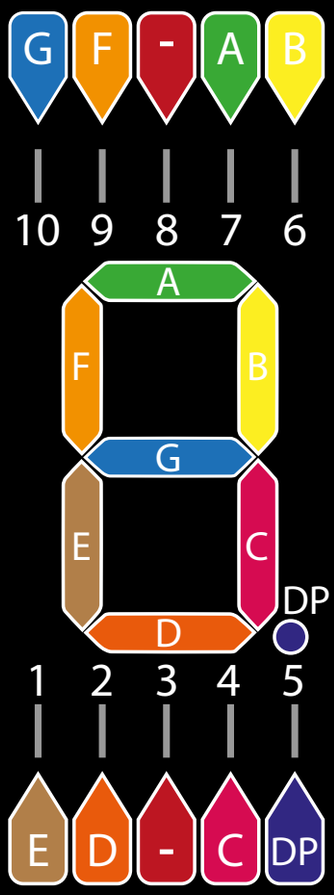
Quick Reference

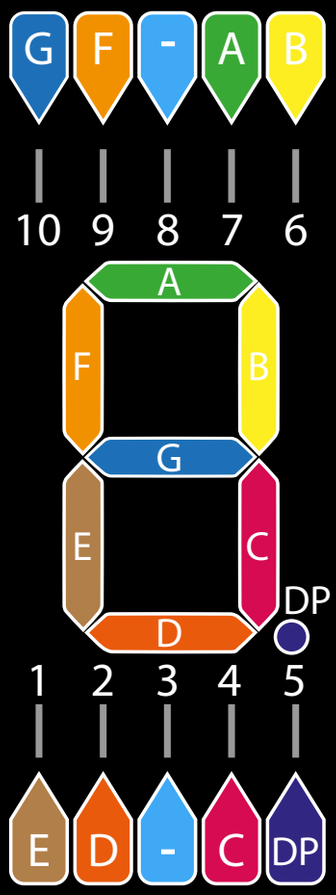
Check Polarity	✓
Positions	10
Type	active
Schematic Symbol	
PCB Symbol	
Designator	U

Important Ratings

Parameter	Typical Values
Forward Voltage (VF)	1.5 - 25V (typ)
Forward Current (IF)	4-200mA (max)
Power Dissipation	50mW - 1W
Luminous Intensity	0.4 - 6000mcd
Wavelength	280 - 800nm

Seven Segment Display





Common Anode
Segment Cathode

Common Cathode
Segment Cathode

COMPONENTS

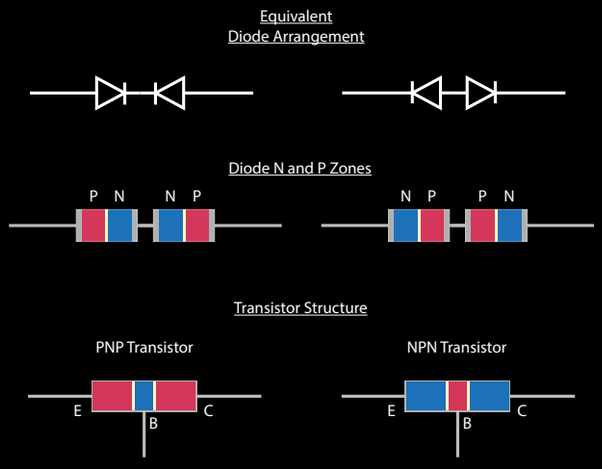
Transistor - BJT

Overview

Transistors are a very important part of the electronics we use every day. There are billions of them in modern CPUs. They can be used to store data, amplify voltage and as switches. The most common transistor is the bi-polar junction transistor (BJT). They come in two flavours, Negative Positive Negative (NPN) and Positive Negative Positive (PNP).

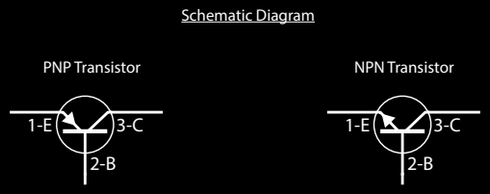
A transistor is essentially 2 diodes connected in reverse series. Meaning they are connected one after the other, but in opposite directions. This wouldn't actually work as a transistor in real life, this is just a model to help you understand how a transistor works.

The NPN transistor is like 2 diodes connected at their anodes. The PNP transistor is the same as 2 diode connected at their cathodes.



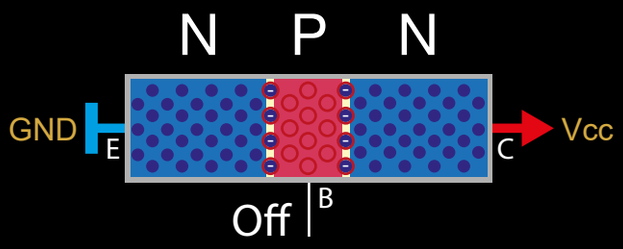
This gives the PNP transistor a cathode center with anodes either side. the NPN transistor has an anode center with cathodes on either side. In reality, the P zones have free spaces for electrons, so is described as positive. The N zones have free electrons, so is described as being positive.

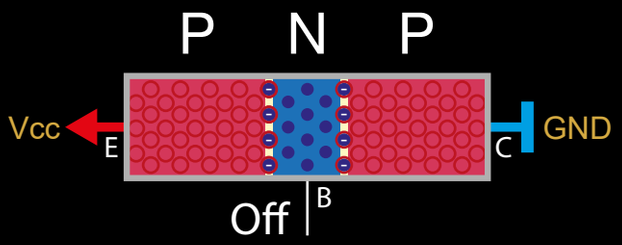
A Transistor has 3 terminals: Base, Collector and Emitter. For electricity to conduct through the Collector and Emitter, a small amount of current needs to be applied to the Base. This creates an amplifying effect. This is great for when you need to switch a higher powered circuit with a low power device, or amplify a signal.



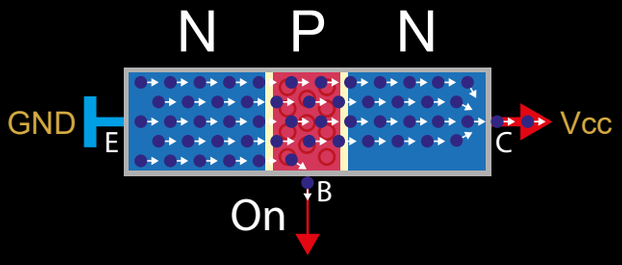
To understand how a transistor works we will refer back to the "electron holes" and "free electrons" from the Zener Diode model, in the Motherboard's manual. The N-type doped material has plenty of free electrons, while the P-type material has plenty of electron holes.

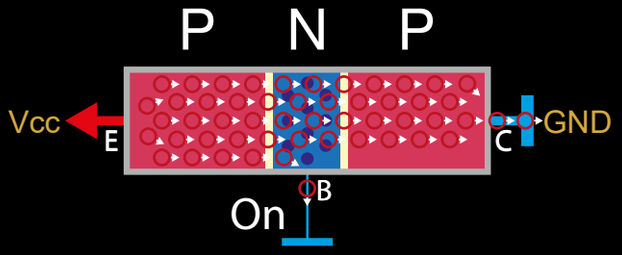
Between these layers are areas called depletion zones. This is where free electrons are occupying electron holes. An atom with a free electron is still considered neutrally charged as the number of electrons and the number of protons in the atom are equal. When a free electron occupies an electron hole, a negatively charged atom is formed. This repels the free electrons in the N zones, preventing conductivity. In this state, the transistor is off.





When a small amount of current is applied to the Base (by creating a potential difference between E and B), the free electrons (NPN) or the electron holes (PNP) start to flow to the Base. If there is enough current flowing through E and B, the electrons, or holes will start to flow from E to C at a higher rate.





When the Base current is kept within a certain range, the transistor can be used as an amplifier. If there is too much current at the Base, there would be less amplification, but the transistor acts as a reliable switch, being either fully open or fully closed. When the transistor is used as a switch it is said to be functioning in its saturation zone.

The amount of amplification, referred to as gain, is determined by the transistors hFE

value. When in saturation mode we can assume this is 10. 10 is overcompensating, but we are working in the saturation zone anyway, so we can safely over estimate to ensure we stay in this region. If we were using the transistor as an amplifier, we would be a lot more careful and accurate when it comes to gain.

Troubleshooting

To test a transistor, first remove it from the circuit. Use a multimeter's diode function (⇥) to test each of the connections between the Base, Emitter and Collector. The red bars on the table represent where you should put the red lead. The black bars are where you should connect the black lead. Obviously you need to know which lead is which, so check the Internet for a datasheet by typing the number that's printed on the transistor.

The table will tell you what value to expect depending on what your leads are connected to. For example: If you put your black lead on the Emitter and the red lead on the Base of an NPN transistor, a value between 0.4 and 0.9V will indicate that that junction hasn't been destroyed. OL means open loop and indicates that not electricity can flow. If you get an open loop where you should have a value, then the junction has been fried.

NPN

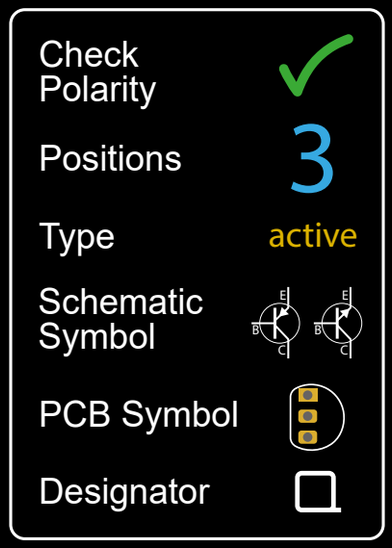
	B	E	C
B		0.4-0.9V	0.4-0.9V
E	OL		OL
C	OL	OL	

PNP

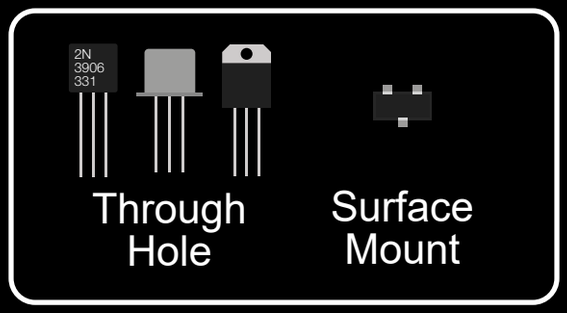
	B	E	C
B		OL	OL
E	0.4-0.9V		OL
C	0.4-0.9V	OL	

This won't tell you whether the transistor is working within the datasheets specifications. If in doubt, swap it out. They are cheaper than chips.

Quick Reference



Common Varieties



Important Ratings

	Typical Values	
Parameter	NPN	PNP
Max Voltage CE	1500V	-500V
Max Voltage CB	1500V	-500V
Max Voltage EB	150V	-150V
Collector Current (Ic)	1mA to 800A	-1mA to -15A
Max Power Dissipation	20mW-500W	20mW-500W

This manual was written and designed by Martyn Evans.
The circuit designs are inspired by many different sources with hands on testing and experimentation.
If you recognise anything as your own, and think you deserve a mention,
please feel free to contact admin@shortcircuits.cc and let Martyn know.