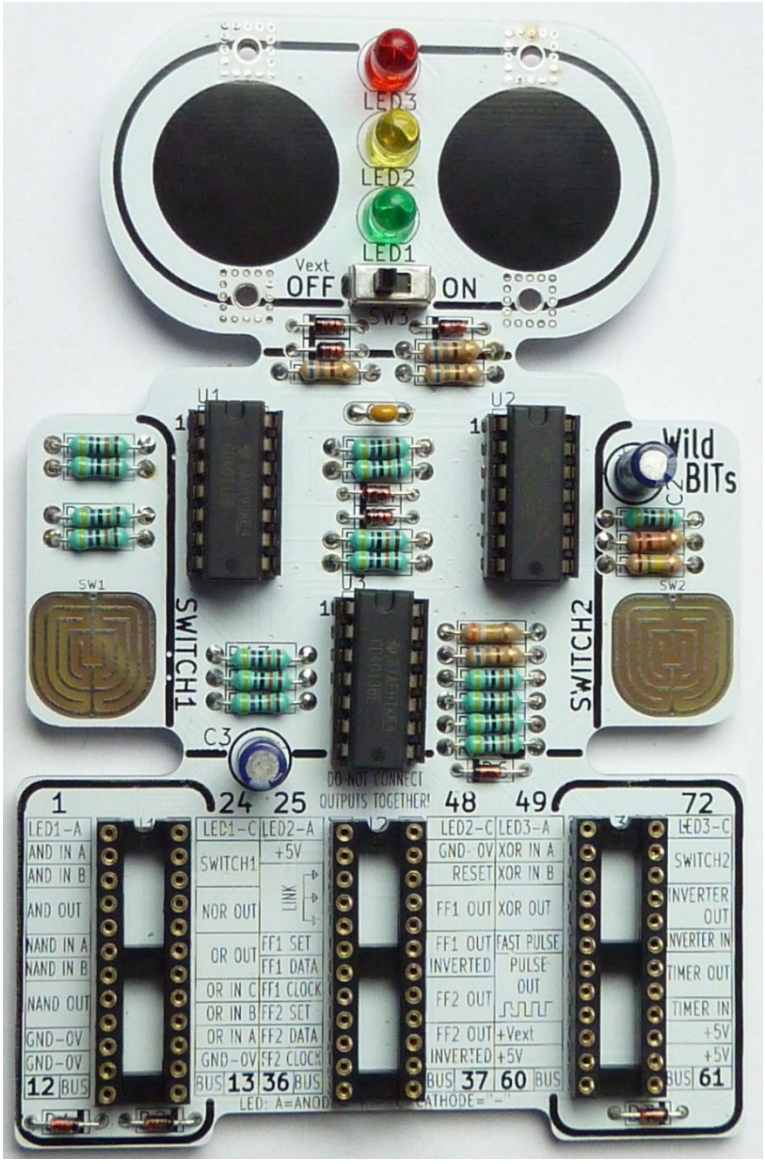
Wild-BITS

Wild-BITS manual.



A Service Kring JOTA-JOTI project.

Satisfied with your Wild-BITS kit, do you have inspiring ideas?
Send us a message, see the last page for the details.

Wild-BITS

Introduction:

The Wild-BITs is an electronics building kit that makes it easy to get acquainted with digital electronics. Dgital electronics are the basis for the computers that we use nowadays. When we say computers you have to think about the PC, laptop, game computer but also your smartphone or the control of a car or an airplane. All these computers are built from the same type of building blocks. The Wild-BITs kit introduces you to a number of these digital building blocks. In this manual we would like to tell you something about these digital building blocks, some details about how computers work and finally we want you to “play” with digital circuits yourself. Although we are not going to build a computer, you can do a lot of fun experiments and useful things with a limited number of digital building blocks. If you understand how these digital building blocks work, you can also design and build new circuits yourself with the Wild-BITs! If you have come up with something nice yourself, we would love to hear from you! Send a photo or drawing of your own circuit to info@kitbuilding.org.

Wild-BITS

What is the meaning of digital:

Nowadays we use the word digital quite often, but what does digital mean? Digital is derived from the Latin word "digitus" which means finger. You can best imagine it as counting on your fingers, a finger does or does not participate in the count, “just count a little bit" is actually not possible. This is exactly what a computer does with its digital signals. A signal is there (there is an electrical voltage) or a signal is not there (there is no electrical voltage).

Bits:

A computer uses "bits". A bit is a signal (present on a wire) that has a high or a low level. A bit is either valid or not, there is no middle way. Whether or not to participate is expressed in different ways:

Bit participates, the bit is:	high, true, "1", active, +5 Volt
Bit does not participate, the bit is:	low, false, "0", not active, 0 Volt

Digital or analog:

As you can see, there are only two states for a bit. There are also systems where a signal can have any value between a lower limit and an upper limit, these systems we call analog systems.

For example, think of a bag filled with apples. The number of apples in the bag is an integer, you can count it on your fingers (digital). The weight of the apples in the bag is a number that can have any value (analogue). This number is of course dependent on the number of apples, but a bag with the same number of large apples has a higher weight than a bag with the same number of smaller apples.

Bytes:

This word is used very often these days, but what does it actually mean? A byte is a group of 8 bits together. You could imagine these 8 bits as eight fingers. The special thing is that a computer, with its eight bits, can count to 255 .... on our ten fingers we can only count up to 10. How is that possible? A computer uses binary numbers. Binary means bivalent. We humans are used to working with decimal numbers, this means based on ten. Chances are that this has to do with the fact that we have ten fingers....

Binary numbers:

A binary number is represented as a series of ones and zeros. For example, the binary number 01101010 is equal to our decimal number 106. To convert a number from binary to decimal, it is useful to know how a binary number is constructed.

A number only consists of a combination of "1" and "0".

- The rightmost bit is the least significant bit, or the lowest value bit (smallest value).

In short: LSB - Least Significant Bit

- The leftmost bit is the highest significant bit, or the highest value bit (so value).

In short: MSB - Most Significant Bit

If we read the number from right to left, the numbers always increase by a factor of two.

Binary name	**MSB**							**LSB**
Decimal value	128	64	32	16	8	4	2	1
Power of 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

We can get the decimal value of a binary number by multiplying the decimal values and the binary values. This sounds complicated, but it's actually not. Let us take the following binary number as an example: 11010011.

If we want to convert this number, we enter the following formula:
Decimal number = 1 x 128 + 1 x 64 + 0 x 32 + 1 x 16 + 0 x 8 + 0 x 4 + 1 x 2 + 1 x 1 = 211

Only the numbers that are multiplied by a "1" participate. The numbers that are multiplied by zero result in zero, so they do not count.

Binary name	**MSB**							**LSB**
Binary number	0	0	1	0	1	0	1	1
Decimal weight	128	64	32	16	8	4	2	1
Binary x decimal	0	0	32	0	8	0	2	1

The decimal number is: 32 + 8 + 2 + 1 = 43

If you now consider that you have ten fingers, you could count up to 1023 when you apply binary counting! A raised finger is a "1", a bent finger is a "0". Give it a try, with a little practice you get the hang of it quickly.

Wild-BITS

Digital circuits:

Numbers are important for computers, because they are super-fast calculators. But they can control devices as well. They also utilize bits, ones and zeros for doing this. Only now do the ones and zeros not have the meaning of a number, but are they used for example to switch an LED on or off or to determine whether a switch is pressed or not.

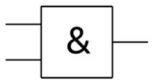
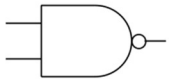
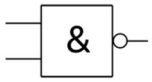
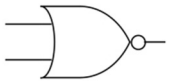
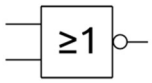
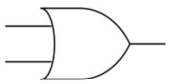
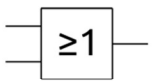
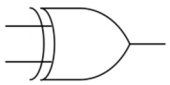
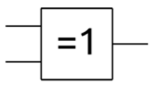
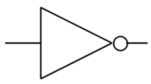
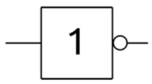
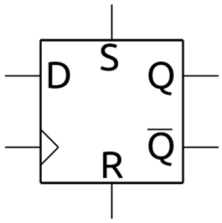
These digital signals can easily be processed with digital ports. A digital port has a number (usually two) of digital inputs and a digital output.

There are a number of different digital ports on the Wild-BITs circuit board, also known as logic ports or logic.

Digital ports on the Wild-BITs:

- Inverter
- AND-port
- NAND-port
- OR-port
- NOR-port
- XOR-port

Symbols are used to draw a circuit. Each port type has its own symbol. The different symbols are shown in the table on the right, in this manual we use the American symbols, these are the easiest to distinguish from each other.

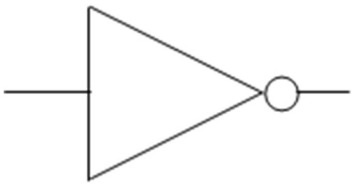
American	British	Logic function
		AND
		NAND
		NOR
		OR
		EX-OR
		INVERTER
		D-FLIP-FLOP

Inverter:

The simplest port is the "inverter", it has one input and one output. The output is the reverse of the input. We can indicate this in a truth table.

Truth table inverter	
Input	**Output**
1	0
0	1

An electrical diagram is drawn to indicate how the logic gates are connected to each other. Symbols are used for the various components in such a diagram. The symbol of an inverter looks like this:



The entrance, or input, is on the left. The exit, or output, is at the circle on the right. If we apply a voltage of 5 Volts on the input, we call that a logical "1", then we find in the truth table that the output gives a logical "0", it gives 0 Volts. If we apply a voltage of 0V (GND = ground) at the input, a logical "0", then we read in the truth table that the output will give a "1", the output will give a voltage of 5 volts.

We can do the following experiment: We connect a LED to the output, with the Anode (LED .... -A) at the output of the inverter (inverter out). The other connection of the LED, the Cathode (LED ... C), we connect to the GND - 0V. If there is now 5 Volts on the output of the inverter, the LED will light up. 5 Volts will be present at the output of the inverter, when the input is at 0 Volts.

Now we connect a switch (SW1 or SW2) to the input of the gate, for example switch 1. If we now touch switch 1 (with a wet finger), it outputs 5 Volts. The LED will then di . Let go of the switch, the voltage at the inverter input becomes 0 Volt, then the LED will light up.

Ground (GND):

In electronics we often use the word earth or "ground". This ground, abbreviated as GND, is usually connected to the negative pole of the battery. The voltage there is 0 volts, all other voltages in the circuit are measured with respect to this reference. Compare it with the earth's surface. For example, we measure our own length, the height of a building in relation to the surface of the earth. This is similar to earth or GND in electronics.

The inputs of all logic gates are provided with pull-down resistors. These resistors have a very high resistance value, so little energy is lost in them, but they do ensure stability in electronic circuits. The purpose of these resistors is to define the level (low) on the inputs of the logic gates. This prevents the inputs from 'floating' and hence electronic circuits from showing strange behavior. To prevent this strange behavior you could also connect the input to the ground, but this connection would have to be disconnected in order to give the input a high level (otherwise you will get a short circuit!). The pull-down resistors keep the level at the input low, but make it possible to set the input at a high level with little energy. You can best imagine the function of a pull-down resistance as gravity. Compare it to a marble court. It needs gravity to ensure that the marbles always roll down and stay in the gutters. If you were to use a marble path in the weightless space, the marbles would behave very differently, they would not even follow the marble path and would simply find their own way in space, as if there was no marble path. Just close your eyes and imagine…..

Different types of logic ports:

There are many different logical gates. A selection of these ports are available on the Wild-BITs, the ports present are:

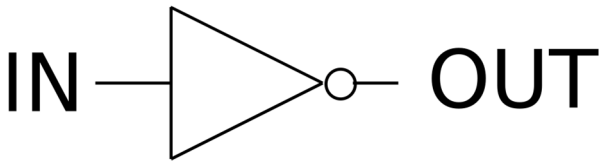
- Inverter
- AND port
- NAND port (non-AND port)
- OR port (OR port)
- NOR port (non-OR port)
- XOR port (exclusive OR port)

Each port has its own applications in a circuit. But before we look at complete circuits, we must first get to know the individual ports.

Inverter:

The inverter was described before, but for the sake of completeness we describe it again:

Symbol:



Truth table:

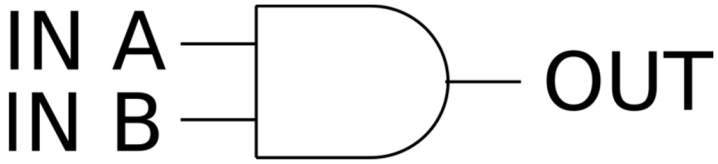
Inverter	
IN	**OUT**
1	0
0	1

Description:

The inverter inverts its input signal. A low signal (0) will be a high signal (1) and vice versa.

AND-port:

Symbol:



Truth table:

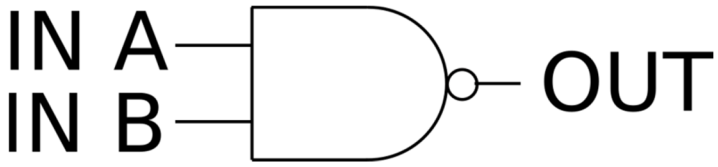
AND-port		
In A	**In B**	**Out**
0	0	0
0	1	0
1	0	0
1	1	1

Description:

The AND-port will output a “1” when both inputs are high (“1”). In all other cases (input-combinations) the output will be “0”.

NAND-port:

Symbol:



Truth table:

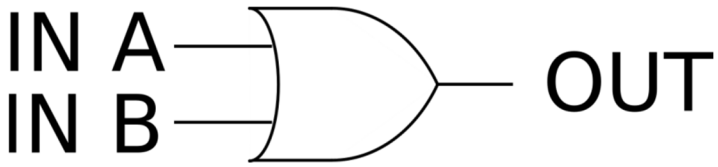
NAND-port		
In A	**In B**	**Out**
0	0	1
0	1	1
1	0	1
1	1	0

Description:

The NAND-port, or not-AND-port operates in a similar way as an AND-port with an inverter in series. The output of a NAND-port will be high (“1”) when one or both inputs are low (“0”).

OR-port:

Symbol:



Truth table:

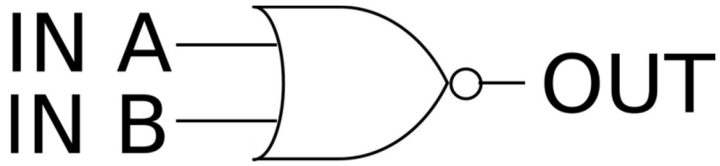
OR-port		
In A	**In B**	**Out**
0	0	0
0	1	1
1	0	1
1	1	1

Description:

The OR-port checks if one or both of its inputs are equal to “1”. As soon as at least one input is “1” the output will become “1”.

NOR-port:

Symbol:



Truth table:

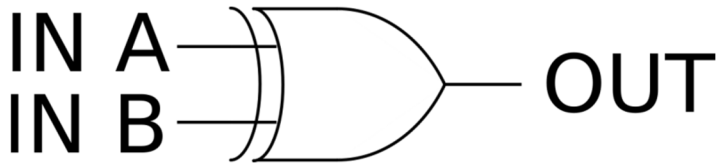
NOR-port		
In A	**In B**	**Out**
0	0	1
0	1	0
1	0	0
1	1	0

Description:

The NOR-port, or not-OR-port, operates in a similar way as an OR-port with an inverter in series. In short, the output of this port will only be “1” when both inputs are “0”.

XOR-port:

Symbol:



Truth table:

XOR-port		
In A	**In B**	**Out**
0	0	0
0	1	1
1	0	1
1	1	0

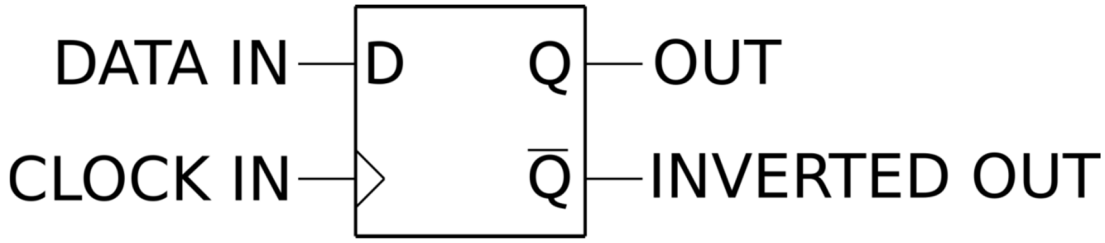
Description:

The XOR-port, or exclusive-OR-port, will only give a “1” at its output when just one input is at a high level. When both inputs are at a low or high level, the output will be “0”. With other words, the output will only be one when the inputs are not equal.

The FLIP-FLOP – the DATA-FLIP-FLOP:

All ports discussed above respond directly (instantaneous) to a change at their inputs. So they cannot remember anything. But computers derive much of their power from their memory. FLIP-FLOPs can be used for these memory elements.
A FLIP-FLOP is an element that can remember the level (a "1" or a "0") at its input. There are a number of different types of FLIP-FLOPs. There are two D-FLIP-FLOPS on the Wild-BITs (DATA FLIP-FLOPs). How these works will be discussed below.

Symbol:

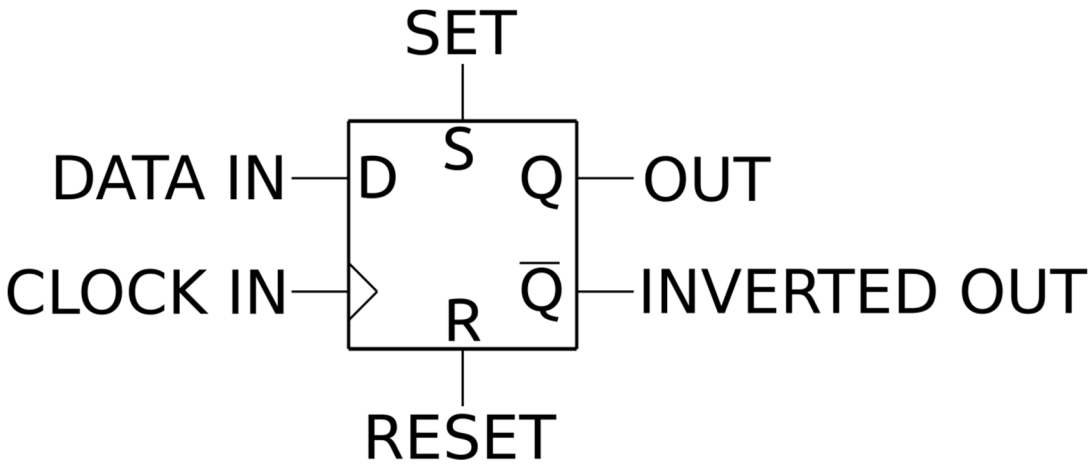


Description:
The DATA-FLIP-FLOP, in short D-FF, has two inputs and two outputs. The outputs are a normal output and an inverted output (this is indicated by the dash above the "Q").
There are two inputs, shown on the left. A DATA input and a CLOCK input.

The command to hold the level on DATA-IN is given via the clock input. The moment the clock changes its level from a "0" to a "1" (this is called the rising clock edge), the FLIP-FLOP will transfer the level from DATA IN to the "Q" output. This level will hold the output regardless, whether DATA IN is going to change to another level or not. Only with the next rising clock, from "0" to "1", the output "Q" will take over the level of DATA IN at that moment.

The data can therefore only be processed with a rising clock. In a computer all clock inputs are connected together, this means that all FLIP-FLOPs take over data at the same moment. Such a system is called a synchronous system. A system that does not use a clock is called asynchronous.

The D-FF on the Wild-BITs has two additional inputs, namely a SET and a RESET input. So the symbol is actually as follows:



Function of the SET input:
The output (Q) can be set to a high level regardless of the state of the clock and DATA IN (/ Q then becomes low).
Function of the RESET input:
The output (Q) be set to a low level regardless of the state of the clock and DATA IN (/ Q then becomes high).
Note: If both SET and RESET are high at the same time, both OUT and INVERTED OUT become high (!). In this case, INVERTED OUT is not the opposite of OUT.

Truth table:

Inputs				**Outputs**	
CLOCK	**SET**	**RESET**	**Data In**	**Output Q**	**Inverted Output /Q**
↑	0	0	0	0	1
↑	0	0	1	1	0
↓	0	0	X	Qo	/Q
X	0	1	X	0	1
X	1	0	X	1	0
X	1	1	X	1	1

The truth table looks more complex compared to the truth tables of the other ports. We also see a number of symbols in that we have not seen before. What do these symbols mean?
The "X" means "don't care", so it doesn't matter if this is a "0" or a "1".

- The up arrow "↑" means that the clock changes from a low to a high level.
  (a rising clock).
- The down arrow "↓" means that the clock changes from a high to a low level.
  (a falling clock).
- Qo means the old state of the output Q. In this table this indicates that the output does not change.
- /Q means that the /Q output remains at the same level.

Bits, nibbles en bytes:

In combination with computers we often use the terms "bits" and "bytes". A bit can be "1" or "0".
We call a group of 4 bits a nibble. Two nibbles together, or 8 bits, we call a byte.
Of course we can display a binary number as a series of ones and zeros. But we can also write a number in a hexadecimal way.

Decimal	Binary	Hexadecimal		Decimal	Binary	Hexadecimal
0	0000	0		8	1000	8
1	0001	1		9	1001	9
2	0010	2		10	1010	A
3	0011	3		11	1011	B
4	0100	4		12	1100	C
5	0101	5		13	1101	D
6	0110	6		14	1110	E
7	0111	7		15	1111	F

To display eight bits, a byte, two hexadecimal "numbers" are combined.
For example, hexadecimal "00" is equal to decimal 0. But hexadecimal "11" is equal to 17. It is 1x16, this is the meaning of the first 1, the second 1 simply has the value 1.
The total is 16 + 1 = 17.
Another example; hexadecimal 3A is equal to, 3 x 16 + 10 = 58 decimal.
The highest is FF, this corresponds to a value of 255.

Wild-BITS

ASCII-characters:

We have already talked about ones, zeros, bits and bytes and that numbers can be presented by this. But what about letters now? How can a computer, a digital system, deal with letters?
A computer uses coding for this, because a computer can only handle numbers. Every letter, punctuation mark or other special character has been given a number. So we no longer just talk about letters, but about characters. These number-character combinations are recorded in a special table, the ASCII table. ASCII is the abbreviation for "American Standard Code for Information Interchange".

ASCII-table

In this table we see all the letters, and the numbers as a symbol (not as a value!), Furthermore, we see the space and a command to go to the next line (LF = Line Feed).
The Wild BITs is a fairly simple system, so ASCII characters are a bit too complicated, but of course this is a nice fact!

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Wild-BITS

Time to do experiments with your Wild-BITs:

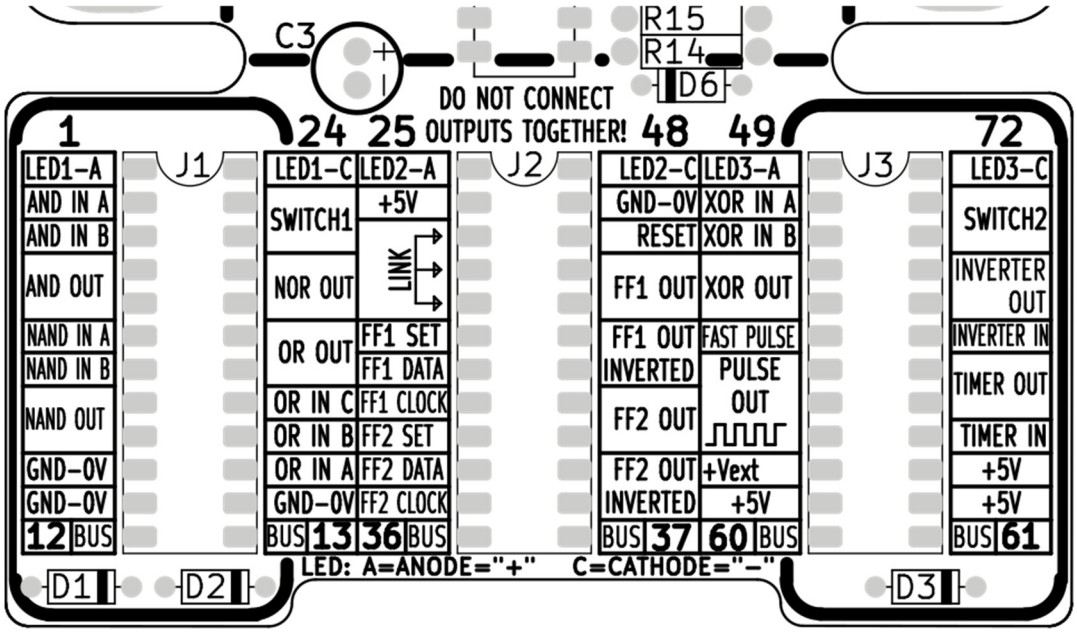
We assume that you have assembled your Wild-BITs kit. How to do this is described in the separate building description

On the printed circuit board of the Wild-BITs there are a number of different digital ports, two FLIP-FLOPs and a few other useful circuits for experimenting with digital logic. To be complete, an overview:

- Inverter
- AND port
- NAND port
- OR port combined with a NOR port
- XOR port
- Two D-FLIP-FLOPs
- Timer, duration 20 seconds
- Clock generator, switchable between 1 Hz and approx. 10 Hz
- Three LEDs
- Two touch switches

The Wild-BITs gets its energy from two CR2032 button cell batteries or from an external power source. (For example, from an Arduino board)

In the experiments it is important to keep the following in mind:
Do not connect any outputs to each other, this can damage the logic gates.



At the bottom of the Wild-BITs PCB is the connection field (J1 to J3). With the contacts there, connections can be made between the various components of the Wild-BITs.

The numbers in bold can be used as a reference. The count is the same as the count of the connections of an integrated circuit or IC. With the notch at the top, pin 1 is always at the top left. The count runs from top to bottom on the left and continues from bottom to top on the right. Every IC-socket on the connection field has 24 pins.

De diverse aansluitingen op de Wild-BITs print:

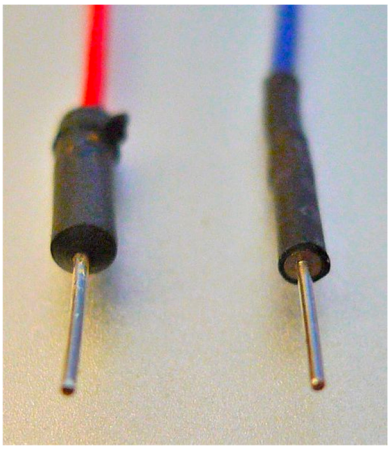
- GND-0V: these connections are at ground level. All these connections are connected to each other and are connected to the negative side of the battery. From a logic point of view, there is a logical "0" present on these pins.
- + 5V: all these connections are connected together, a voltage of 5 Volts is present. From logic point of view, this is a logical "1".
- Vext: an external supply-voltage can be connected here. For example, when using an Arduino.
- LED1A and LED1C: allows LED1 to be connected to the circuit. The "A" means Anode, this is the positive pole of the LED. At the other connection there is a "C" which means Cathode, this is the negative pole of the LED. The same also applies to LED2 and LED3.
- SWITCH1: this connection is connected to touch switch SWITCH1. By touching the contacts, the output of the switch will become "1". This also applies to SWITCH2. If the switches are not touched, the level is a logical "0".
- BUS: all BUS contacts are connected together. These contacts can be used to distribute a signal that must be available at multiple spots. When connecting multiple Wild-BITs, the BUS can be used to distribute signals between across the various Wild-BITs.
- LINK: these are three contacts that are connected to each other. You can use LINK to split one connection into two extra connections.
- PULSE OUT: the clock signal is on pulse-out. The pulses come every second, this means a frequency of 1Hz.
- FAST PULSE: if this connection is connected to PULSE OUT, the clock frequency will increase to approximately 10 pulses per second (10Hz).
- TIMER: with this circuit a short pulse can be extended to 20 seconds.
- AND IN A & B: these are the inputs of the AND port, the output is connected AND OUT.
- NAND IN A & B: these are the inputs of the NAND port, the output is connected to NAND OUT.
- OR IN A, B, C: these are the inputs of the combined OR / NOR gate. These inputs are linked to the OR and to the NOR port. The OR and NOR gate outputs are connected with OR and NOR out.
- XOR IN A&B: are the inputs of the XOR port, the output is connected to XOR-out.
- INVERTER IN: this is the input of the inverter, the output is connected to INVERTER-OUT.
- FF connections: the connections with FF in the name are connected to the FLIP-FLOPs.

- The connections belonging to FLIP-FLOP1 or FLIP-FLOP2 are:
  - FF1 DATA: the DATA input for FLIP-FLOP1.
  - FF1 CLOCK: the CLOCK input of FLIP-FLOP1.
  - FF1 SET: the SET input of FLIP-FLOP1.
  - RESET: this input is connected to the RESET of FLIP-FLOP 1 and FLIP-FLOP 2.
  - FF1 OUT: this is the output of FLIP-FLOP 1.
  - FF1 OUT INVERTED: this is the inverted output of FLIP-FLOP 1.

Building circuits:

To be able to make a circuit with the Wild-BITs, the various components must be connected to each other. This can be done by inserting wires in the contacts. For this you can use:

- Wire supplied with the kit, this is insulated wire with a solid core of 0.5 mm
- Excess pieces of wire from components (for example, from the resistors)
- Purchased breadboard wires, use the round wires with a diameter of 0.5 mm (there are also square pins, they do not fit in the Wild-BITs)

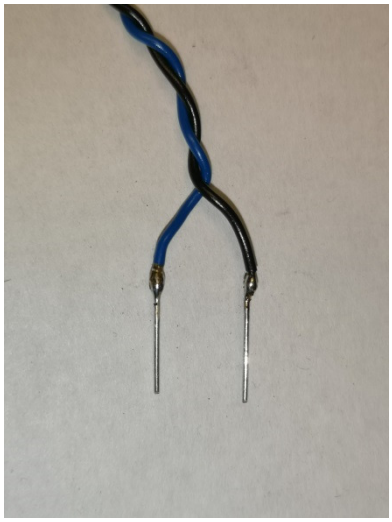


Connecting multiple Wild-BITS:

It is possible to build larger circuits with multiple Wild-BITs. In that situation, a connection must be made between the GND-0V between all Wild-BITs used. It is possible to use each board separately with batteries, but it is possible to connect the + 5V between the various boards as well. It is then possible to supply multiple Wild-BITs with one board.

A useful tip:

If you want to connect insulated wires to the Wild-BITs, use a cut-off wire from (for example) a resistor. Solder the insulated wire to this cut-off wire. See the photo below.



Experiments with the Wild-BITs:

In this chapter a number of experiments / circuits with the Wild-BITs are described. These are meant to discover and learn about digital electronics. You can make many more circuits, you can think of these yourself or modify existing circuits.

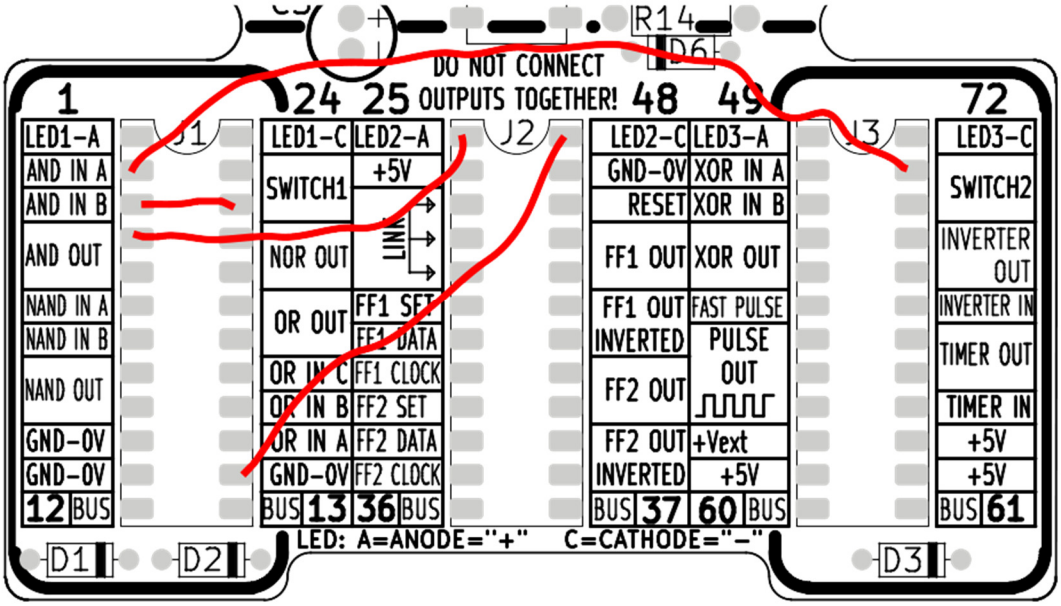
The following circuits are described:

- Discover the logical gates
- How do LEDs work? - Application in a Logic Probe
- Flashing light
- Oracle or decision-maker
- Quiz-master
- Crack the logic
- Water-alarm, game for the steady hand, nerve spiral
- Shift-register
- Traffic light
- 20 second timer
- Binary counter
- Wild-BITs in combination with the Arduino
- Your own circuits

Discover logic ports:

The Wild-BITs offers many possibilities to experiment with logical gates (ports). The first step is to get to know the different building blocks, how can this better be done than by doing some experiments.

You supply a logical "0" or a "1" on the inputs of the gates, the output of the logical gate can be connected to an LED. Try the different combinations of ones and zeros on the input and see how the output responds, does the LED come on or does it go out?
The connection diagram for the AND port is shown below as an example.



The inputs of the AND-gate are normally “0” (via the pull-down resistors, see the explanation elsewhere in this manual). By touching SWITCH1 or SWITCH2 with a damp finger, you can set the IN A and IN B inputs to a high level, a logical "1". When the LED lights up, the output is high, a logical "1". If the LED is off, the output of the gate is low, a logical "0".

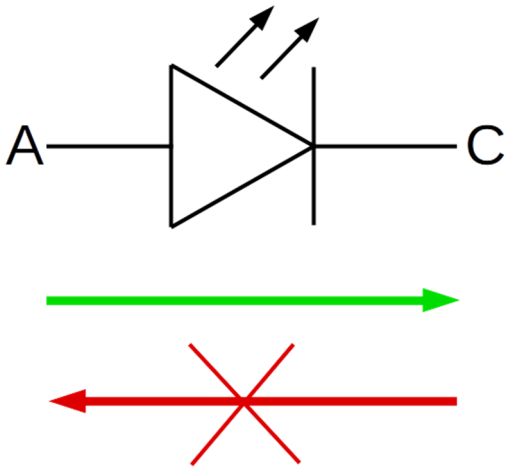
As guidance you can fill in the table below:

In A	In B	Out	Remarks
0	0		No switch activated
0	1		Switch 1 activated
1	0		Switch 2 activated
1	1		Both switches activated

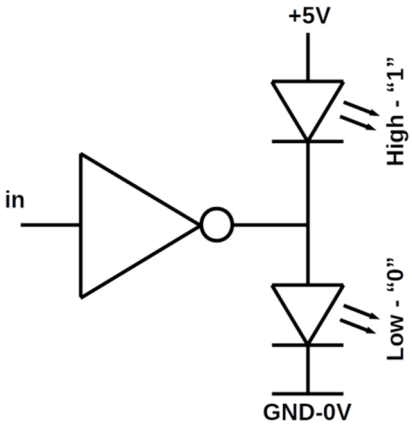
You can do this for all different ports. Note, you can only connect one switch to the inverter. At the OR gate, which has three inputs, you can only connect two inputs. The third input can be unconnected.

How do LEDs work? Application in a logic probe:

The LEDs on the Wild-BITs are connected in such a way that they can be used as flexibly as possible. An LED has two connections, an Anode (A) and a Cathode (C). A LED will only give light (and allow current to flow) if the current goes from the Anode to the Cathode, the green arrow. The current cannot go in the opposite direction, so current flow from C to A through the LED is not possible.

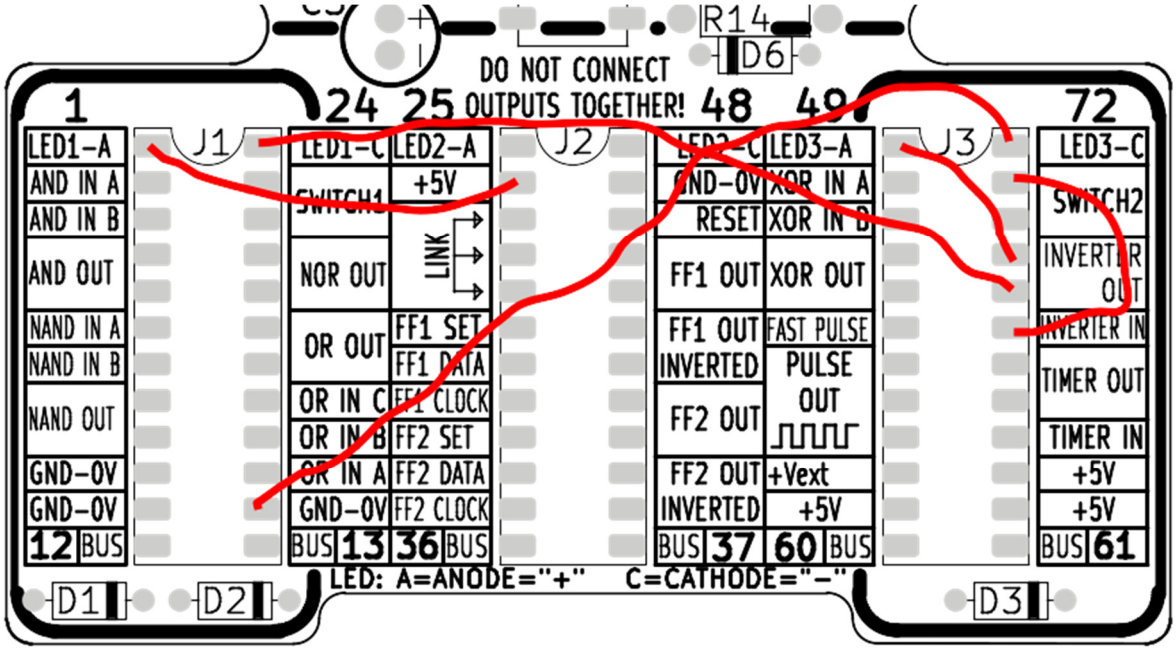


The most logical way to use an LED is with the Cathode connected to the ground (GND-0V). If a voltage (logical "1") is now applied to the Anode, the LED will light up. Another way to connect the LED is to connect the Anode with the +5V. If the Cathode is now placed at a low level (logical "0"), it will light up as well. If we now connect the cathode of the LED to the output of a logic gate, then this will light up if the output of the logic gate is a "0".
We can use this if we make a so-called "logic probe". A logic probe is a measuring instrument with which you can see in a logic circuit what the level is on an input or output of a logical gate.



Here you can see the electrical diagram. One LED High- "1" will illuminate if the logic level at "in" equals "1". If the logic level is "0", the LED Low - "0" will illuminate. We can choose a green LED for the High level LED and a red LED for the Low level LED.

We can build this circuit on the Wild-BITs as follows:



By touching SWITCH 2 you change the logic level on the inverter input from low to high. The green LED will light up instead of the red LED.

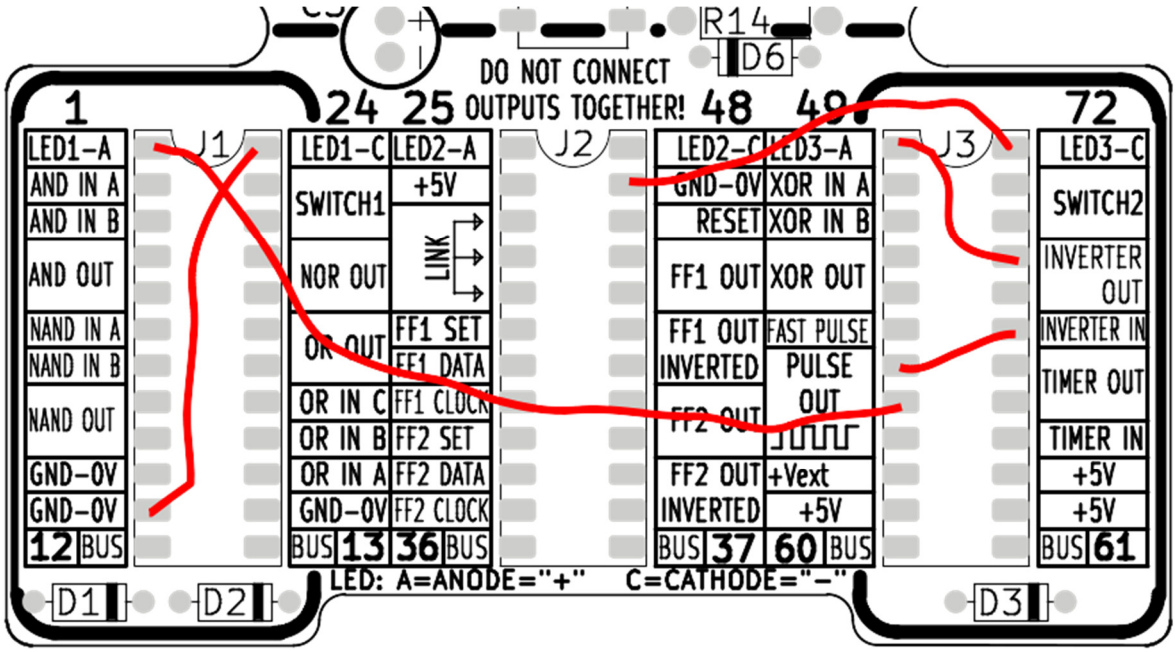
Instead of connecting the inverter input with SWITCH 2, you can also connect the input of the inverter with Pulse Out. The Pulse Out output automatically changes from low to high and vice versa. A complete cycle takes approximately 1 second.

Important note, with a LED, a resistor must always be connected in series to limit the current through the LED. With the LEDs on the Wild-BITs, this resistor is already in series with the LED (although the resistor is not always shown in the circuits). If you want to connect even more LEDs, put a resistor of approximately 470 Ohm in series with the LED (colors: yellow-purple-brown-gold).

Blinking light:

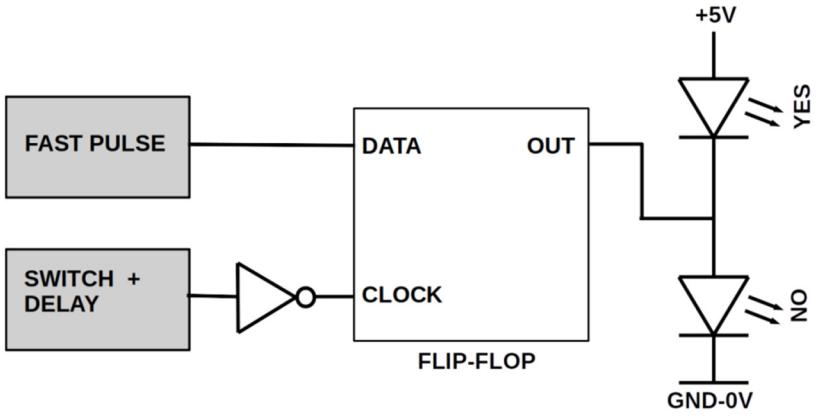
You can make excellent flashing lights with the Wild-BITs. These circuits are very well suited for experiments. The “PULSE OUT” becomes high once per second (half a second) and once low (half a second).

Below is the configuration to make a blinking light:



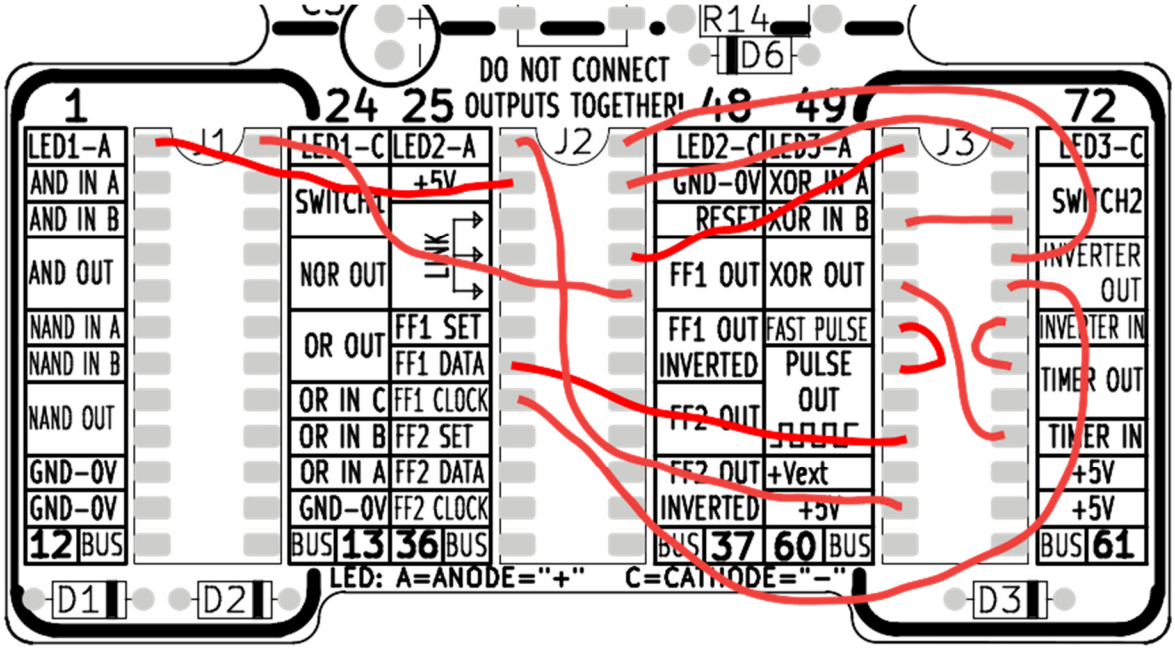
Oracle, decision maker:

Sometimes it is difficult to make a choice. Using digital logic you can create a circuit that can help you make a decision.

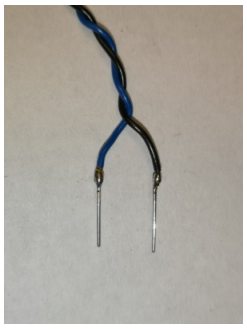


The timer starts as soon as you press the SWITCH. The yellow LED then lights up. Once the timer has finished, its output goes to "0". The inverter converts this from a low-to-high transition (positive edge). At this time, depending on the level of FAST PULSE, connected to the data-, the output of the FLIP-FLOP will go to a "high" or a "low" level. The red or green LED will light up and stay on after releasing the SWITCH.

The wiring for this circuit looks like this:



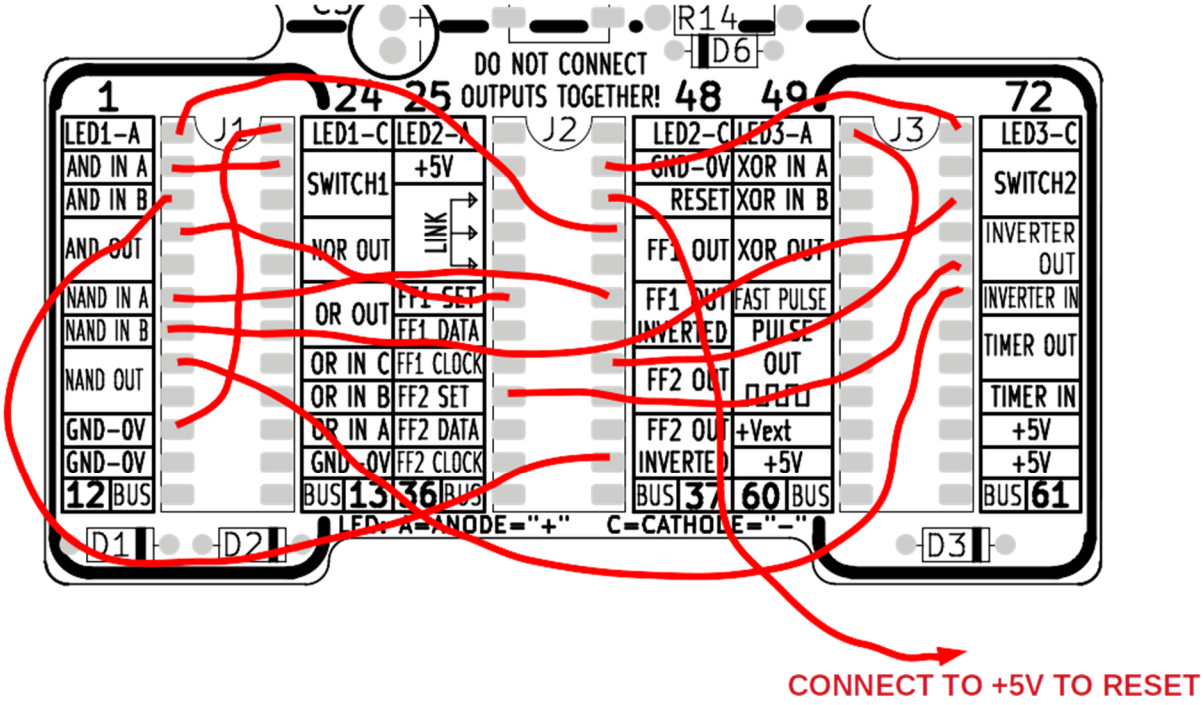
Quiz-master:



With the Wild-BITs you can build, and be, a Quiz master. Just like in the famous game shows, you can now see who first pressed the button. Of course, you can use the switches of the Wild-BITs, but you can also use a few “real Quiz buttons” or make your own switches from, for example, paper clips or a clothes peg. To connect these buttons to the Wild-BITs you can use a few cut pieces of wire from components. You can solder these cut-off pieces of wire to the insulated wires. These pieces of wire fit perfectly in J1 ... J3. You can connect the other end of the insulated wires with the button.
You connect one wire to the "+ 5V" and the other wire to the inputs of the ports that are connected in the diagram to Switch 1 and Switch 2.

On the Wild-BITs you build the circuit below. Switch 1 and Switch 2 are the participant buttons. The reset is a wire that you can connect to "+ 5V" to reset the Quiz master. If you use external buttons, you can also use Switch 1 or Switch 2 for the reset instead of the wire.

On the Wild-BITs you build the circuit below. Switch 1 and Switch 2 are the participant buttons. The reset is a wire that you can connect to "+ 5V" to reset the Quiz master. If you use external buttons, you can also use Switch 1 or Switch 2 for the reset instead of the wire.

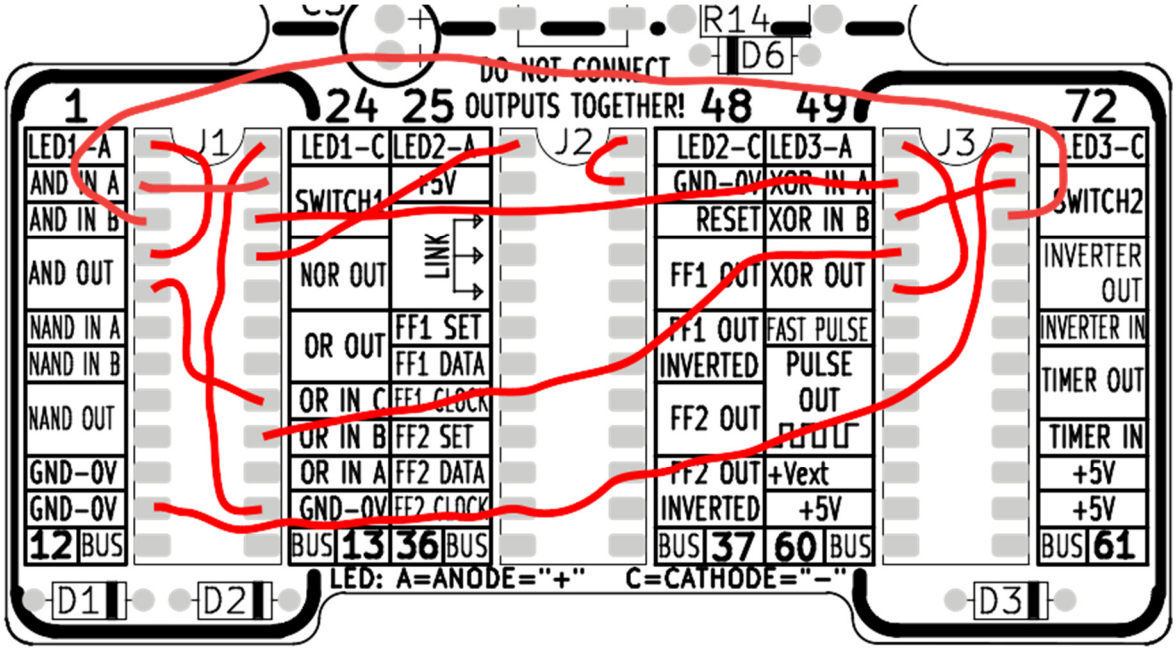


NOTE: Two wires cross each other: AND-OUT is connected to FF1-SET. The other wire is between FF1-OUT INVERTED and NAND IN A.

Crack the logic:

This circuit controls all three LEDs via the two switches. Try the different combinations of the switches, see how the LEDs react. Use the table to record the results.

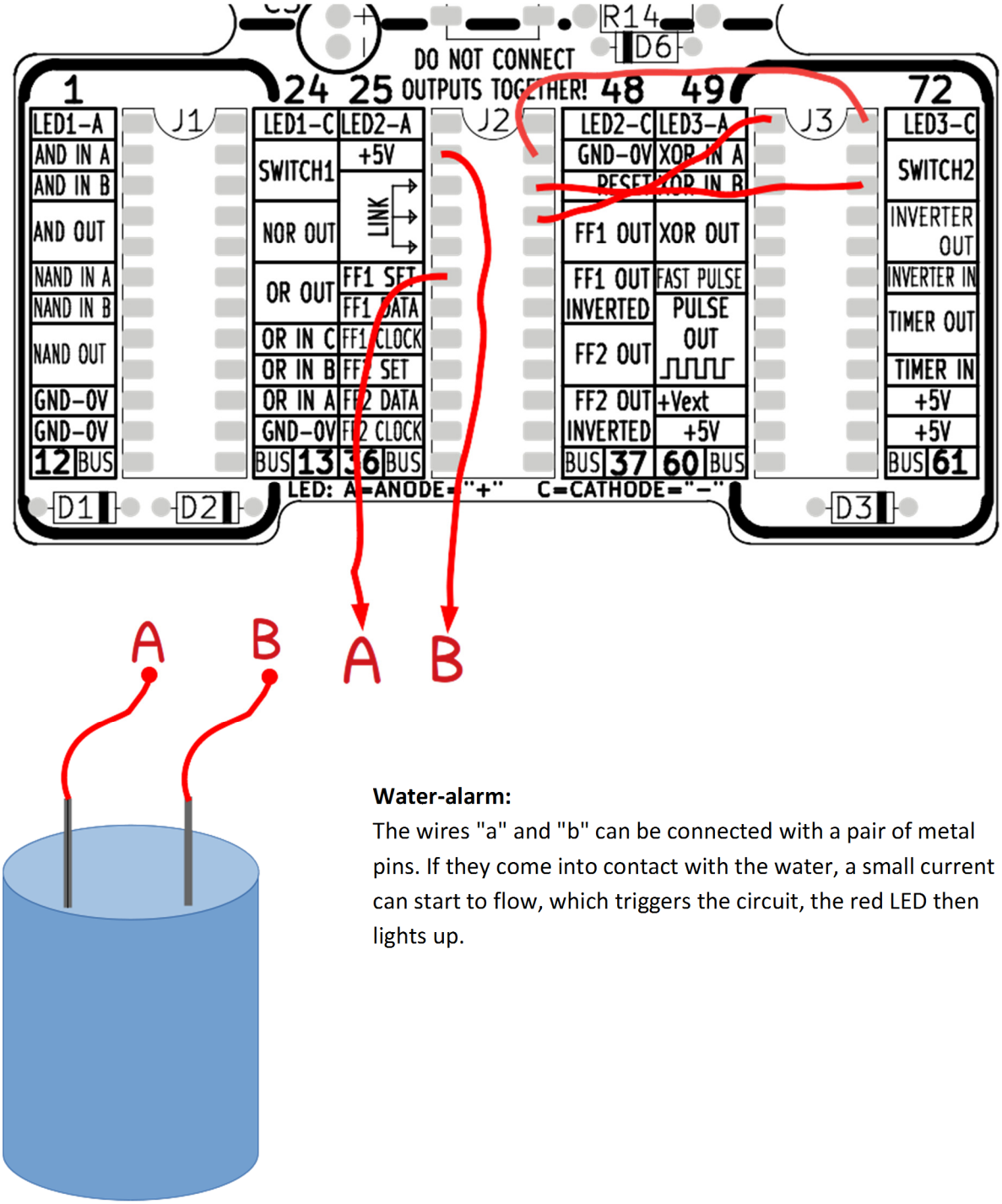
Can you imagine how the circuit works? If you understand this, you are on the right track to come up with new logic circuits yourself!



Switch 1	Switch 2	LED 1	LED 2	LED 3

Water-alarm, buzz wire, game for a steady hand:

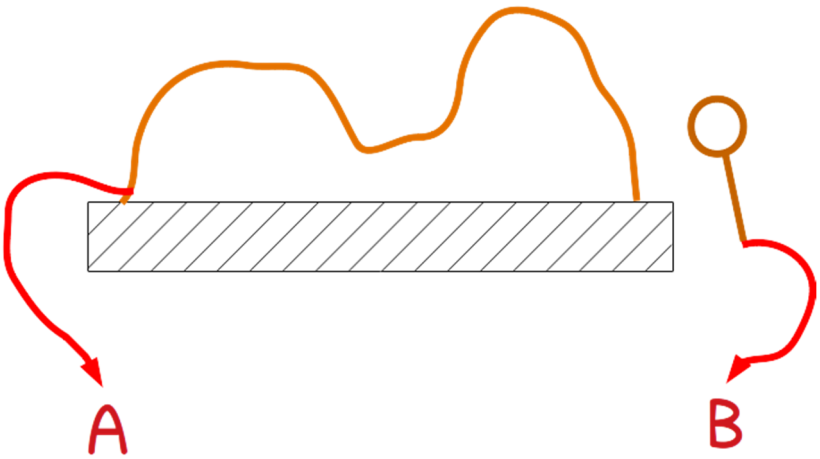
The following circuit can be used for a number of applications. The contacts / wires A and B do not normally make contact, if they do make contact then the red LED lights up. By pressing SWITCH 2 the red LED goes out again. The wires A and B can be connected to a buzz wire, a steady hand game or a pair of metal pins that can detect water.



Water-alarm:

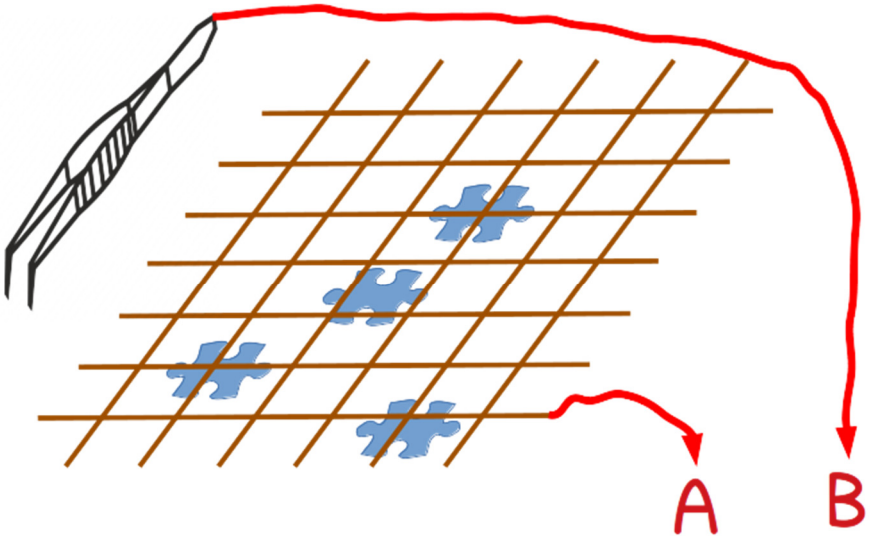
The wires "a" and "b" can be connected with a pair of metal pins. If they come into contact with the water, a small current can start to flow, which triggers the circuit, the red LED then lights up.

Wild-BITS



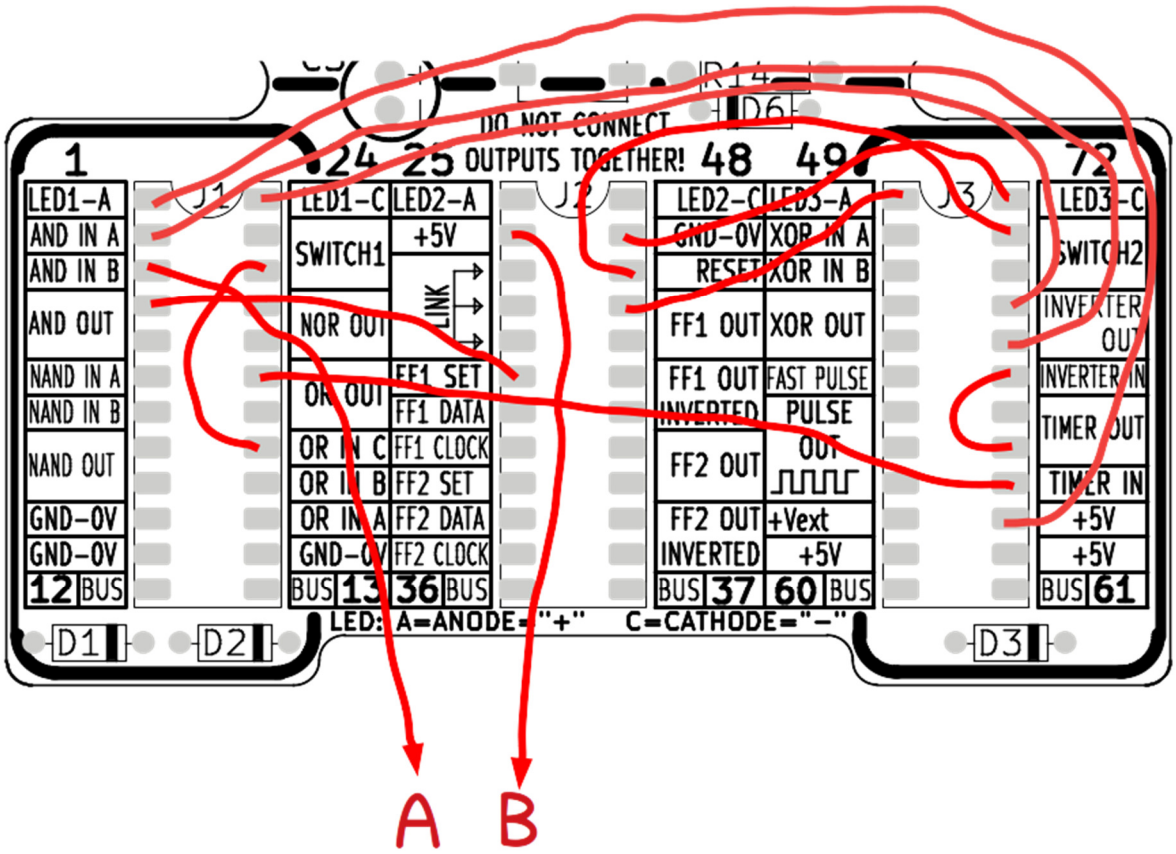
Buzz wire:
You can make the spiral from copper wire or iron wire. Bend a piece of wire in a "challenging" shape and mount it on a board. Connect wire A with the wire. Make an eye in another piece of wire and hook it around the wire on the board. Now try to move the eye from one side to the other of the spiral without making contact. The challenge becomes greater if you put some time pressure on the game. The fastest wins!

Steady hand game:
The game for the steady hand is inspired by Dr. Bibber. With Dr. Bibber the intention is to help the patient by removing "diseases" or diseased parts of the body. These are placed in small compartments in the patient. If you touch the edges of the boxes, then Dr. Bibber will make a sound. With this circuit and some other materials, you make your own variant. Use some metal mesh or use a number of metal wires to create a grid (all wires must be connected!). Connect the wires with connection A. Take a pair of metal tweezers and connect it with an insulated flexible wire with point B. Now try to remove small objects that are under the mesh by removing them with the tweezer through the mesh. The mesh may not be touched by the tweezers! Who can get the most items from under the mesh the quickest?



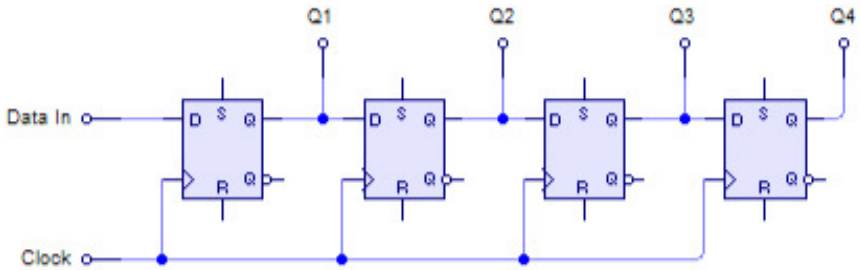
Pimp your circuit with a delay:
You can expand the circuit with a delay. After touching Switch 1, the trigger of the circuit is deactivated for approx. 20 seconds. If a contact is made between A and B during this period, this will not lead to an alarm. If the circuit is again sensitive to contact between A and B, the yellow LED will light up.

The modified circuit is shown below:



Shift-register:

A commonly used circuit in digital technology is the so-called "shift register". The circuit consists of a number of flip-flips that are connected one after the other (connected in series). With each clock pulse, the logic level at the output of one flip-flop is transferred to the next flip-flop.



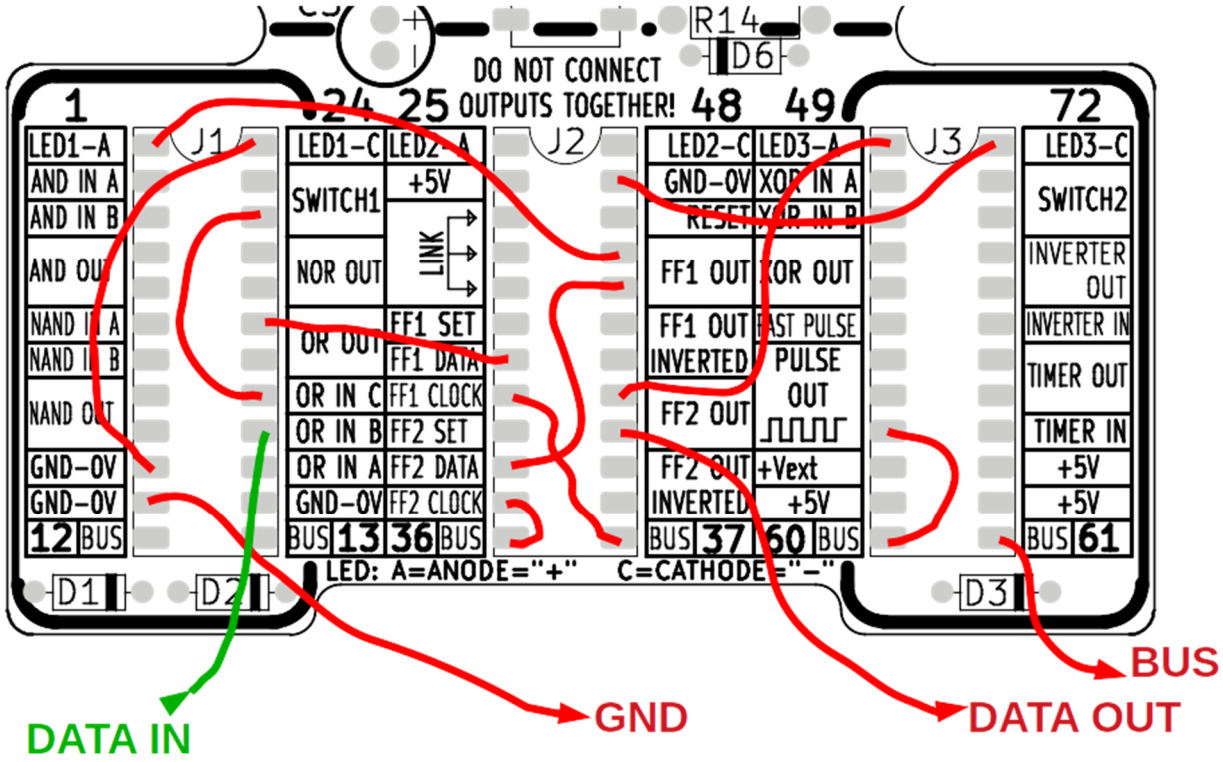
Shift-register with flip-flops (source: wikipedia.org)

On the left side the data (a "1" or a "0") comes in, at "DATA IN". If the signal goes from low to high on the clock line (rising edge), the data on the input of the flip-flop “D” is taken over by the flip-flop on its output “Q”.
So if there is a "1" on "Data in", it will appear on Q1 after a rising clock edge. At the next clock edge this "1" will appear on Q2. For example, with each clock pulse the "1" scrolls one step further through the shift register. A bit pattern, consisting of ones and zeros can be moved through the shift register step-by-step.

It is possible to connect the output of the shift register (output of the rightmost flip-flop) to the data input. A digital port must then be used to be able to enter data (for example, an OR port). Once a pattern has been entered, it will continue to scroll continuously. You can make a shift register as large as you want, this circuit can easily be made with multiple Wild-BITs. You can use this to create very nice light effects! For example, a running light.

Wild-BITS

The circuit is built as follows:



It is possible to put multiple Wild-BITs in series. Then connect the "GND" and "BUS" of the boards to each other. **PLEASE NOTE: ON ONLY 1 Wild-BITs, YOU CONNECT “PULSE OUT” TO THE BUS.** You connect the "DATA OUT" connection with "DATA IN".

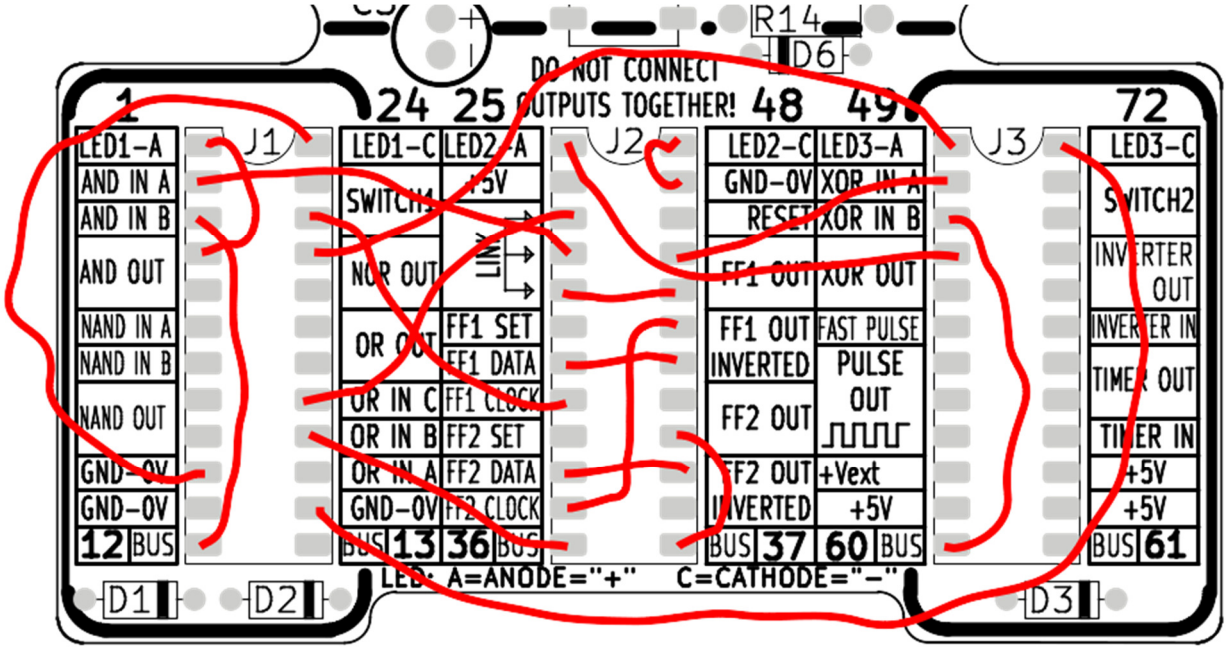
You now have two options:

- Reconnect DATA OUT from the last Wild-BITs to the first DATA IN input, now the DATA goes into a circle.
- Leave the last DATA OUT unconnected. A pattern will now end with the last Wild-BITs.

Traffic light:

You can also create a traffic light with the Wild-BITs. The LEDs 1 to 3 form the traffic light, by pressing Switch 1 you can change the color of the traffic light. Place the traffic light at the door of your room, so others know if they can come in or not.

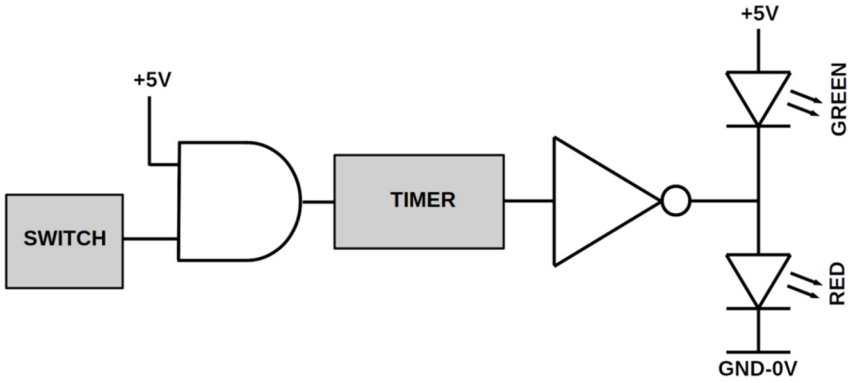
When the light is green, pressing Switch 1 will cause the yellow LED to light up, when pressed a second time the yellow LED will stay on, when the third time is pressed, the red LED will light up. If you press Switch 1 again, the green LED lights up again.
The connection diagram is shown below:



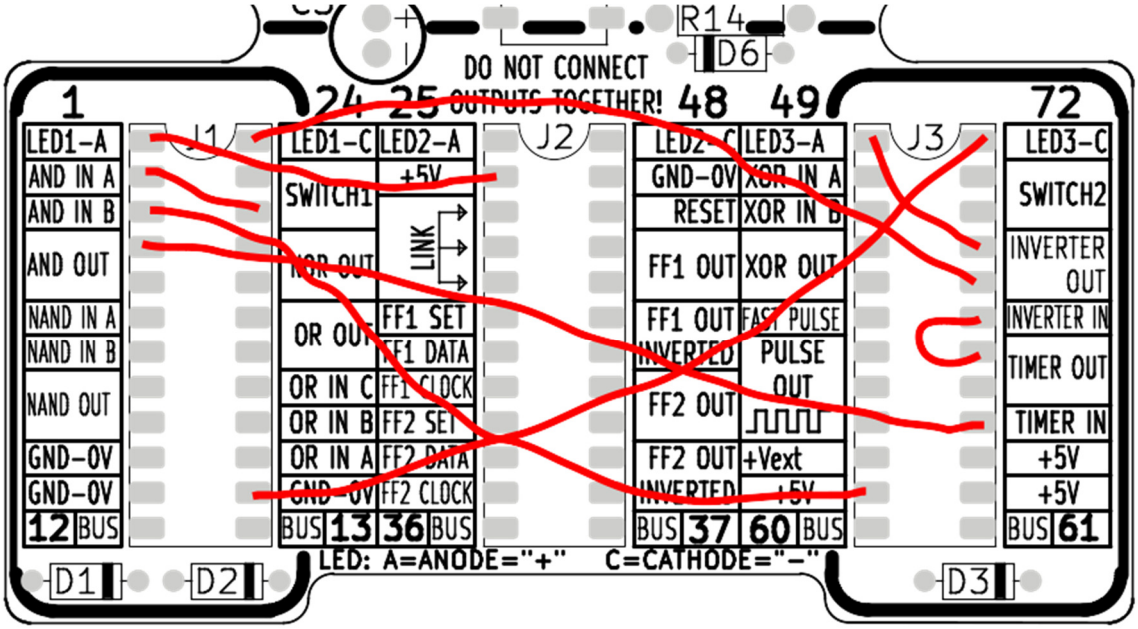
If you start using this circuit, it is possible that the LEDs will jump very quickly. This is due to a phenomenon that we call “bouncing”. We feel that we only press the switch once. In reality, the switch bounces a bit before it makes good contact. A lot of small pulses are generated first before there is real contact. The circuit will also respond to the small pulses.

20 second timer:

For fast games you sometimes need a timer for a short time. With the Wild-BITs you can make a timer for 20 seconds.



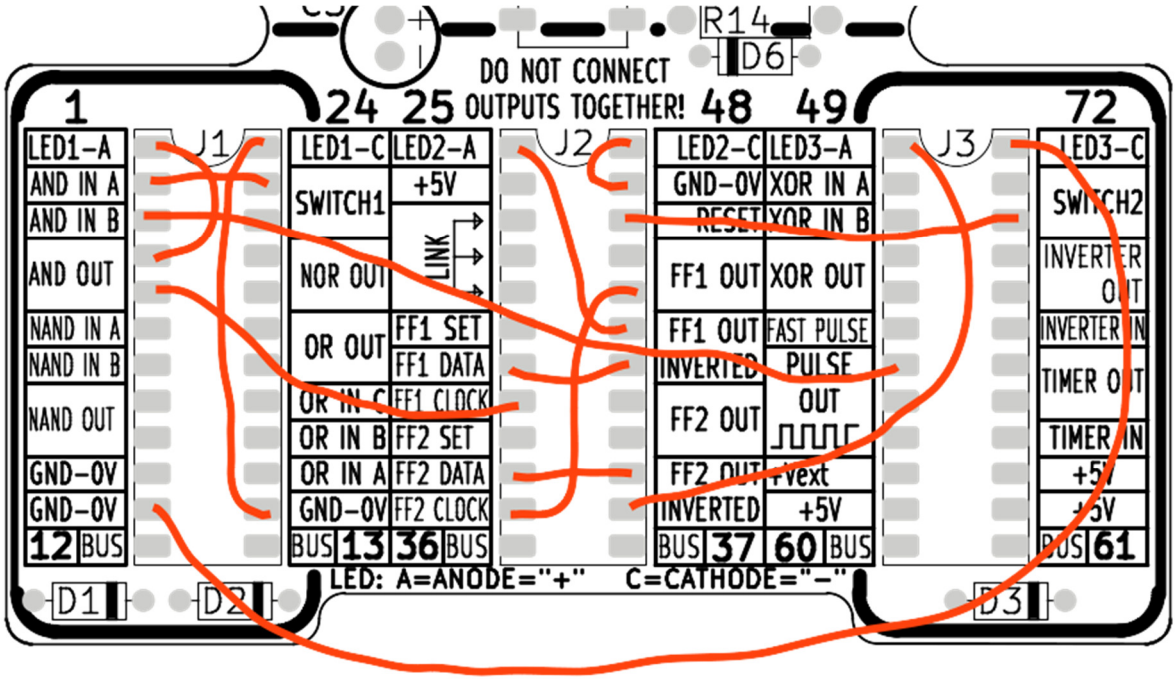
The timer is activated by a high pulse from Switch 2, as long as the pulse (high level) at the input remains high, the timer keeps waiting (the output is high). The capacitor is charged in the timer, after the disappearance of the high signal at the input, the capacitor is discharged in approximately 20 seconds (to a low level). The inverter controls the LEDs. To be able to activate the timer, sufficient power must be available to charge the capacitor in the timer. For this reason, the timer can best be controlled from a logical gate. An AND-port has been chosen here. One input remains constantly high (+ 5V), the other input is activated by the SWITCH.
Thus, after releasing SWITCH1, the timer will remain active for approximately 20 seconds.
The necessary connections are shown below:



Binary counter:

The binary counter is the circuit that is also printed in text on the back of the Wild-BITs PCB. Many ports and the FLIP-FLOPs on the PCB are used with this circuit. Ideal for testing the operation of your Wild-BITs kit.

The test circuit, for the two-bit binary counter, looks like this:

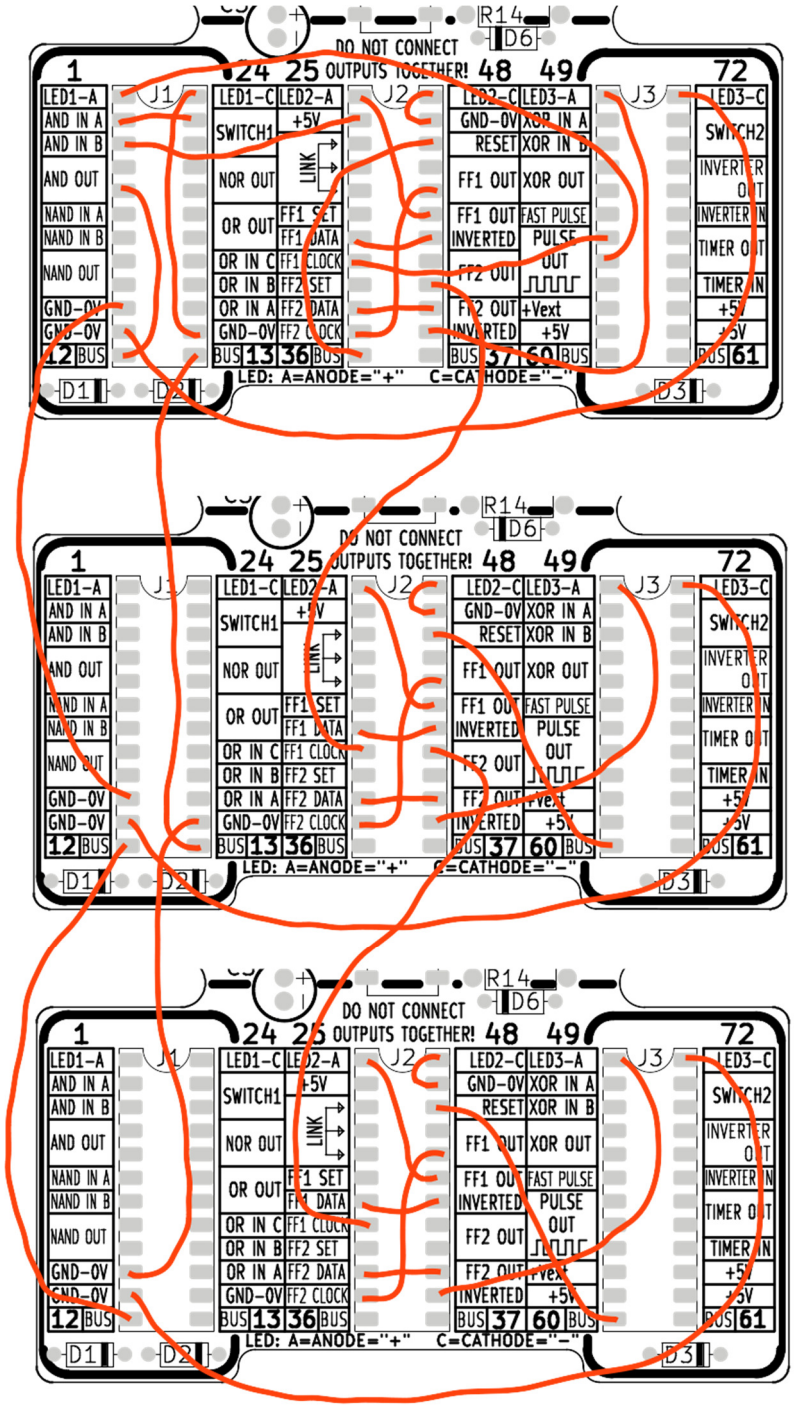


The LEDs on the PCB will count binary from 0 to 3 continuously. This circuit is perfectly suited to connect multiple Wild-BITs with each other. Two bits are added with every Wild-BIT. So with two Wild-BITs you make a four-bit counter, which shows 16 different binary combinations, this counter counts decimal from 0 (the first combination) to 15 (the last combination).

Number of Wild-BITs	Number of bits in the counter	Number of binary combinations	Highest decimal value
1	2	4	3
2	4	16	15
3	6	64	63
4	8	256	266
5	10	1024	1023

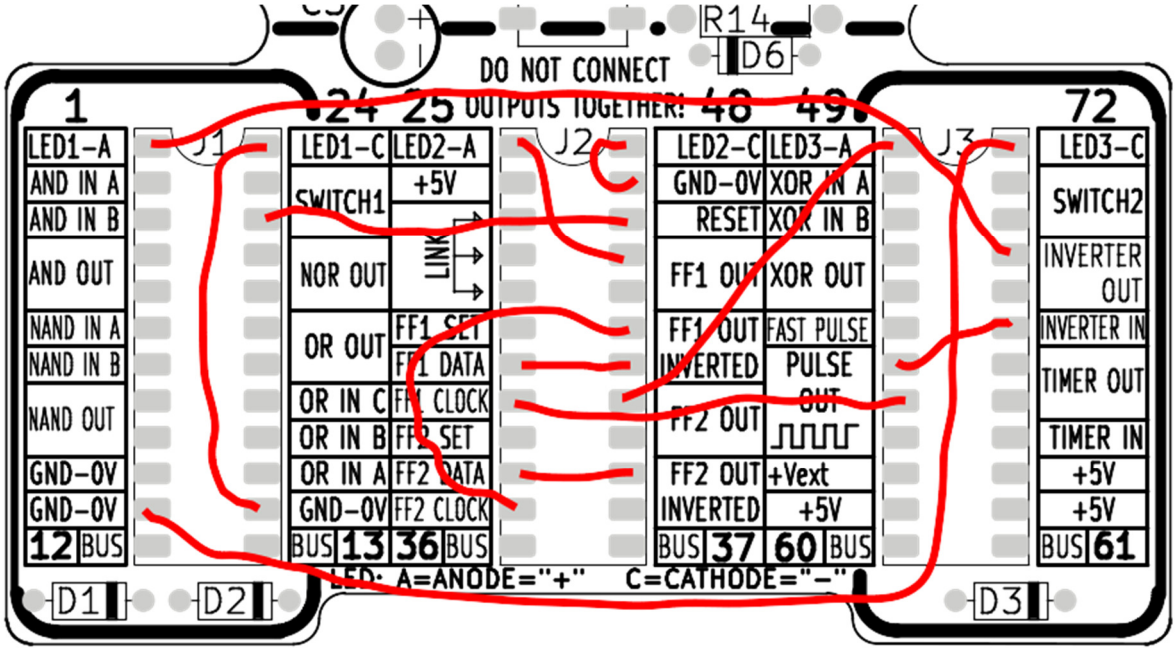
Wild-BITS

Here you see an example of how you can connect three Wild-BITs kits together. With this circuit the function of SWITCH 1 (activate counting) has changed, SWITCH 1 is now the reset switch. SWITCH2 no longer has a function. Wires are laid between the Wild BITs between the GND-0V, the reset signal (this is on the “BUS”) and the clock signal. The clock signal runs from FLIP-FLOP 2 of the first Wild-BITs to FLIP-FLOP1 on the second Wild-BITs, and so on. In this way it is therefore possible to link a number of Wild-BITs kits together.



3 bits counter:

A two-bit counter is described on the back of the Wild-BITs. You can easily convert this to a 3-bit counter, see the diagram below.



Wild-BITs in combination with Arduino:

If you have an Arduino, you can connect it to the Wild-BITs.
NOTE: some Arduinos (there are many types!) Work with a voltage of 3V. The Wild-BITs can also work with this voltage, connect it to Vext. If the Wild-BITs switch is in the “on-position”, the 5V voltage of the Wild-BITs will come to certain connections, which could damage a connected Arduino. To be safe, remove the batteries from the Wild-BITs.

A simple example consists of the use of the LEDs on the Wild-BITs by the Arduino. For this you can very easily use the program "Blink", this is stated in the examples in the Arduino software. Below you can see the altered program to use external LEDs.

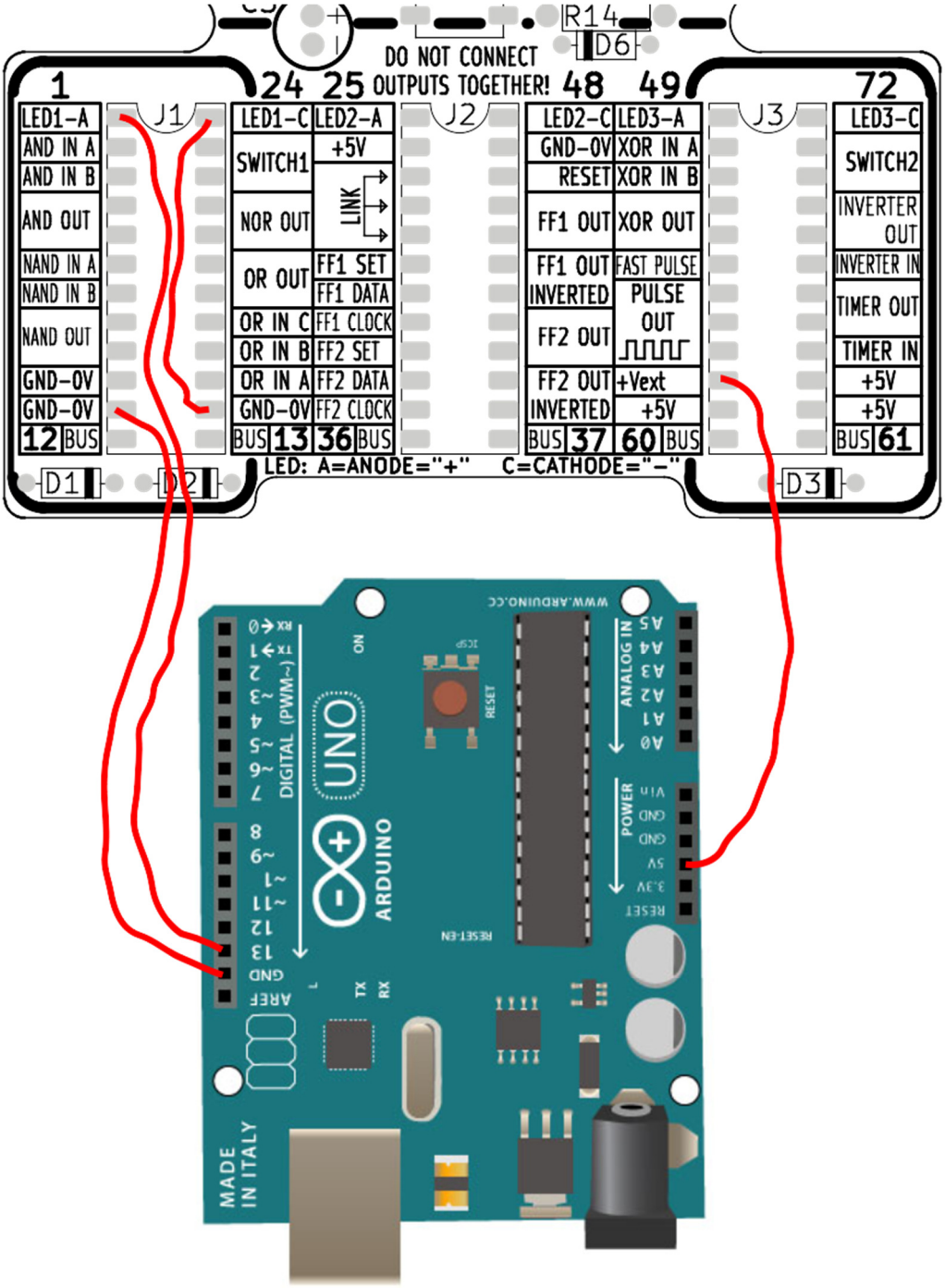
```
/*
Blink  Turns on an LED on for one second, then off for one second,
repeatedly.   This example code is in the public domain.
*/

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:

void setup() {

// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
digitalWrite(led, HIGH);      // turn the LED on (HIGH is the voltage level)
delay(1000);                  // wait for a second
digitalWrite(led, LOW);       // turn the LED off by making the voltage LOW
delay(1000);                  // wait for a second
}
```

When connecting the Arduino to the Wild-BITs you always have to make the following two connections:
1. GND of the Arduino - GND of the Wild-BITs
2. + 5V from the Arduino - Vext from the Wild-BITs

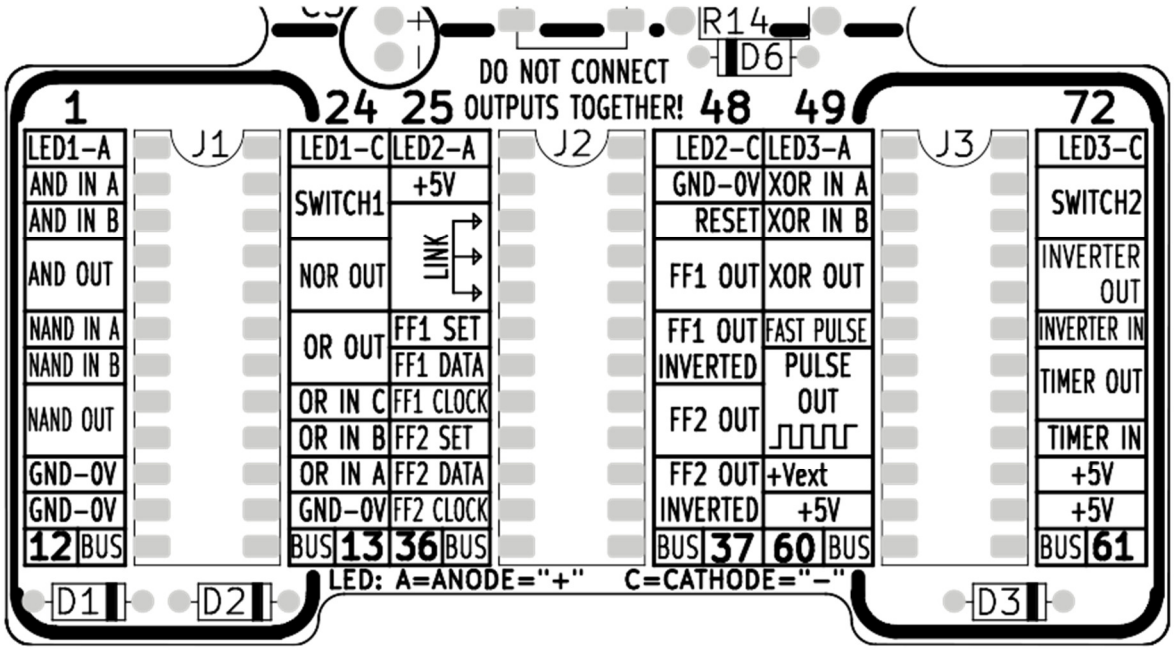
The other connections depend on the interaction between the Arduino and the Wild-BITs. The example here shows an Arduino UNO, but you can also use other Arduinos, such as an Arduino NANO.

If you want to use Switch 1 or Switch 2, it is best to “polish” the signal with a digital port. Connect Switch 1 (or 2) to, for example, the XOR or the OR port. You then connect the output of this port to the digital input of the Arduino.

The schedule for the above program is given below.

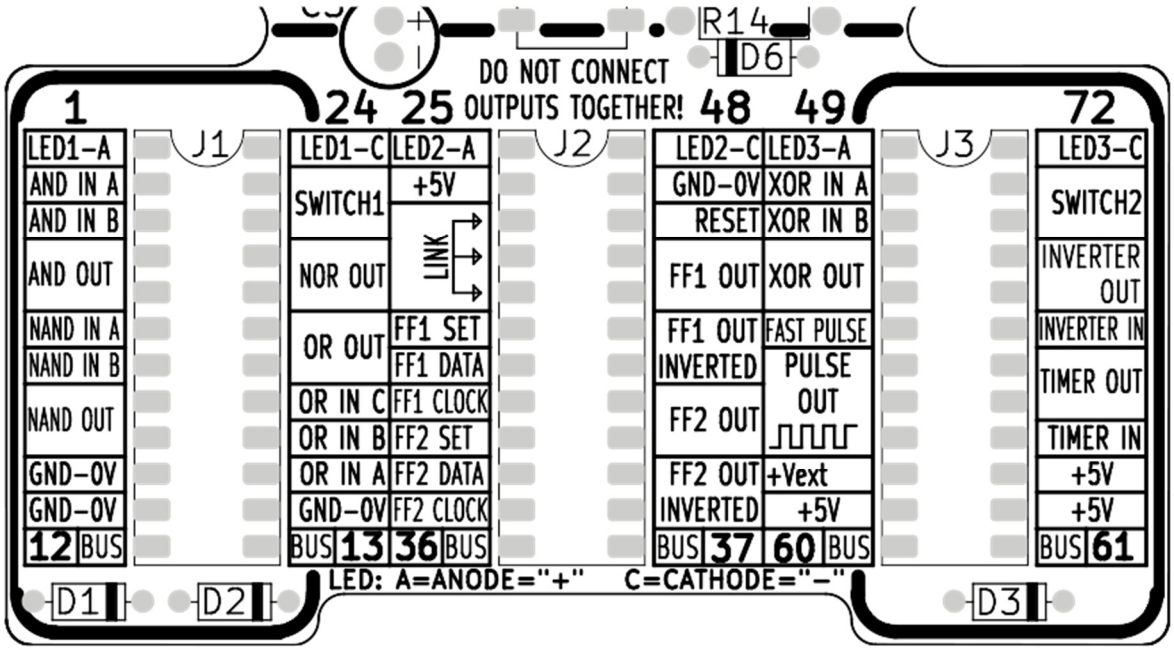


This is my circuit (1):



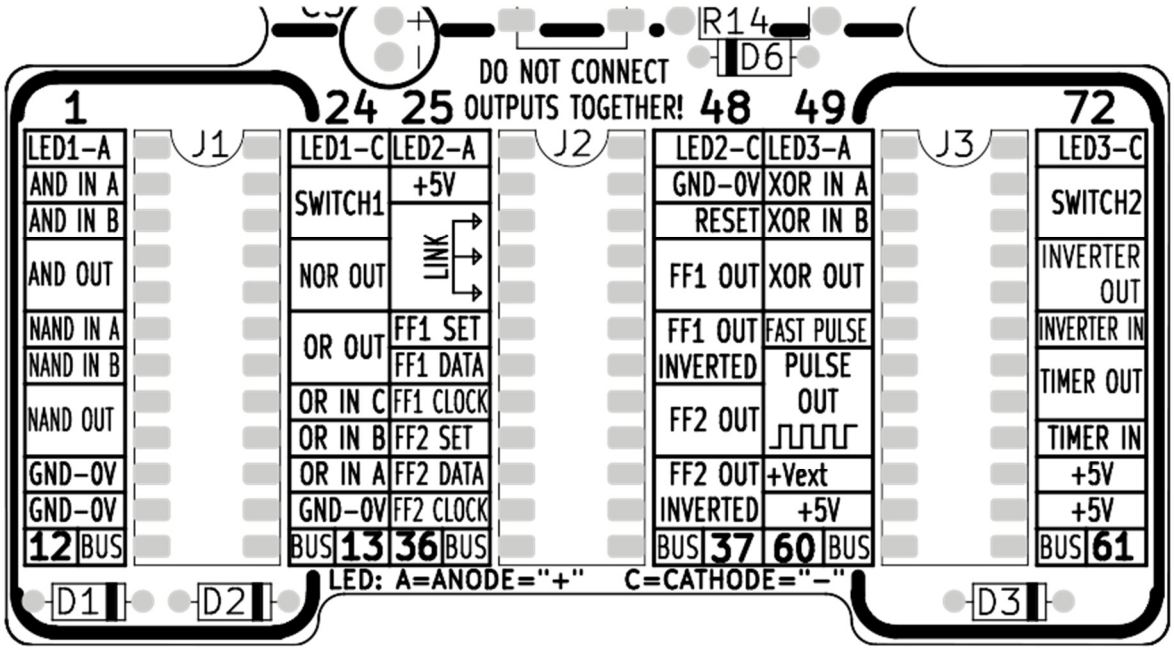
Function/description:

This is my circuit (2):



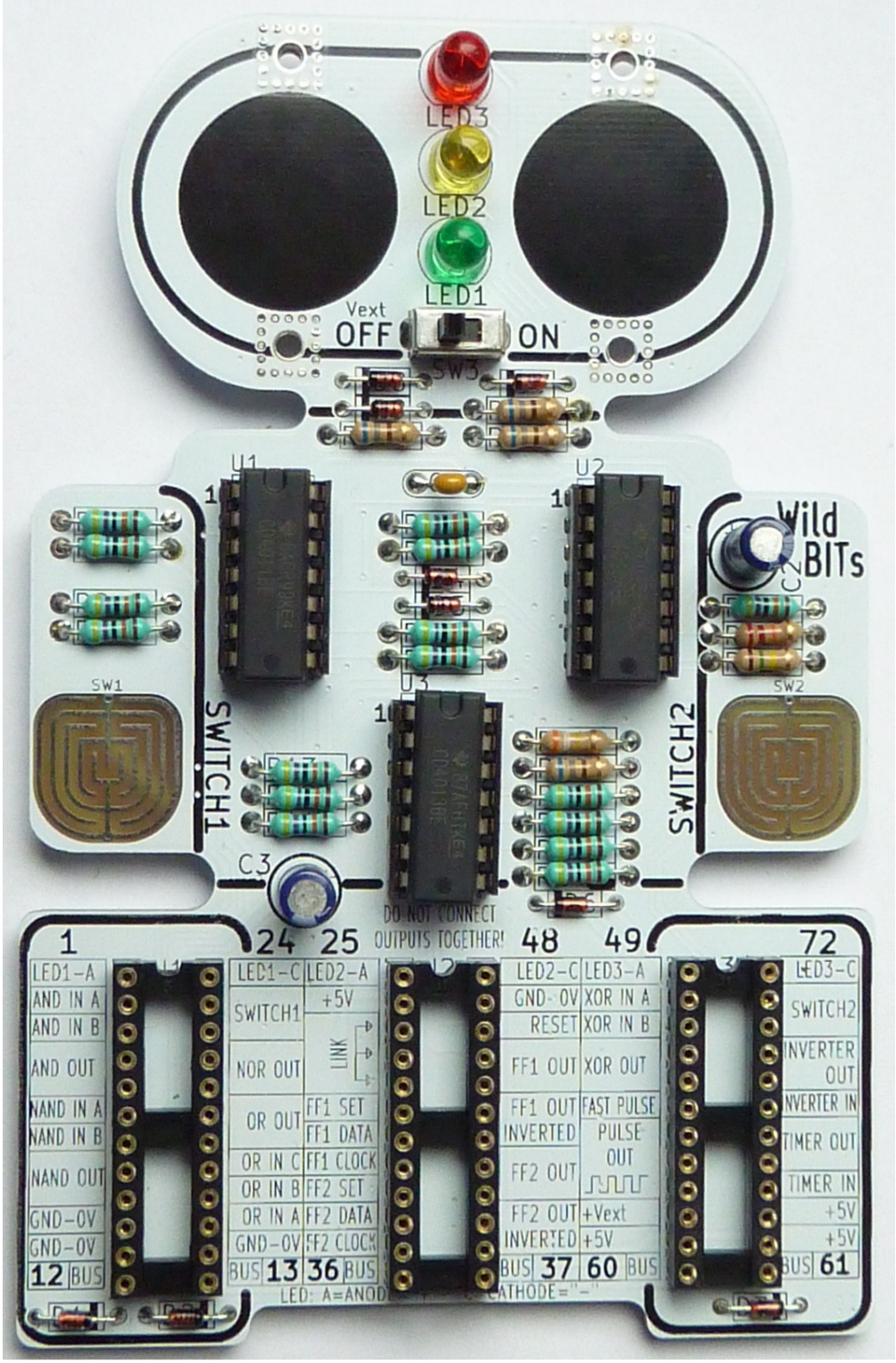
Function/description:

This is my circuit (3):

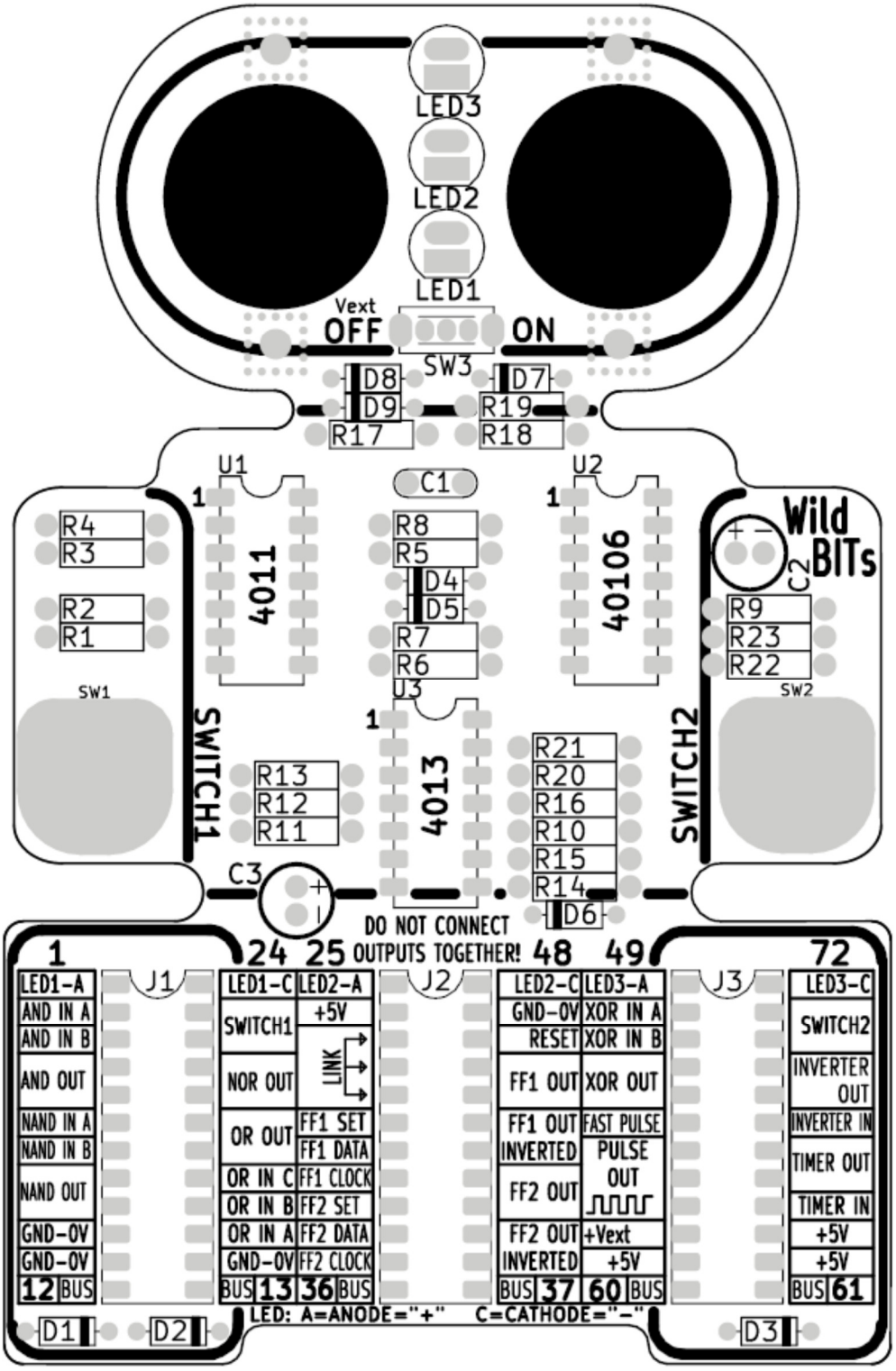


Function/description:

Built Wild-BITs PCB:



Component setup:



Feedback:

Do you have any comments or would you like to give feedback about the Wild-BITS?
Do you have comments or questions about the Service Kring JOTA-JOTI?
Then contact us via the contact form on the site www.kitbuilding.org.

On behalf of the Service Kring JOTA-JOTI we wish you a lot of fun with the Wild-BITS!