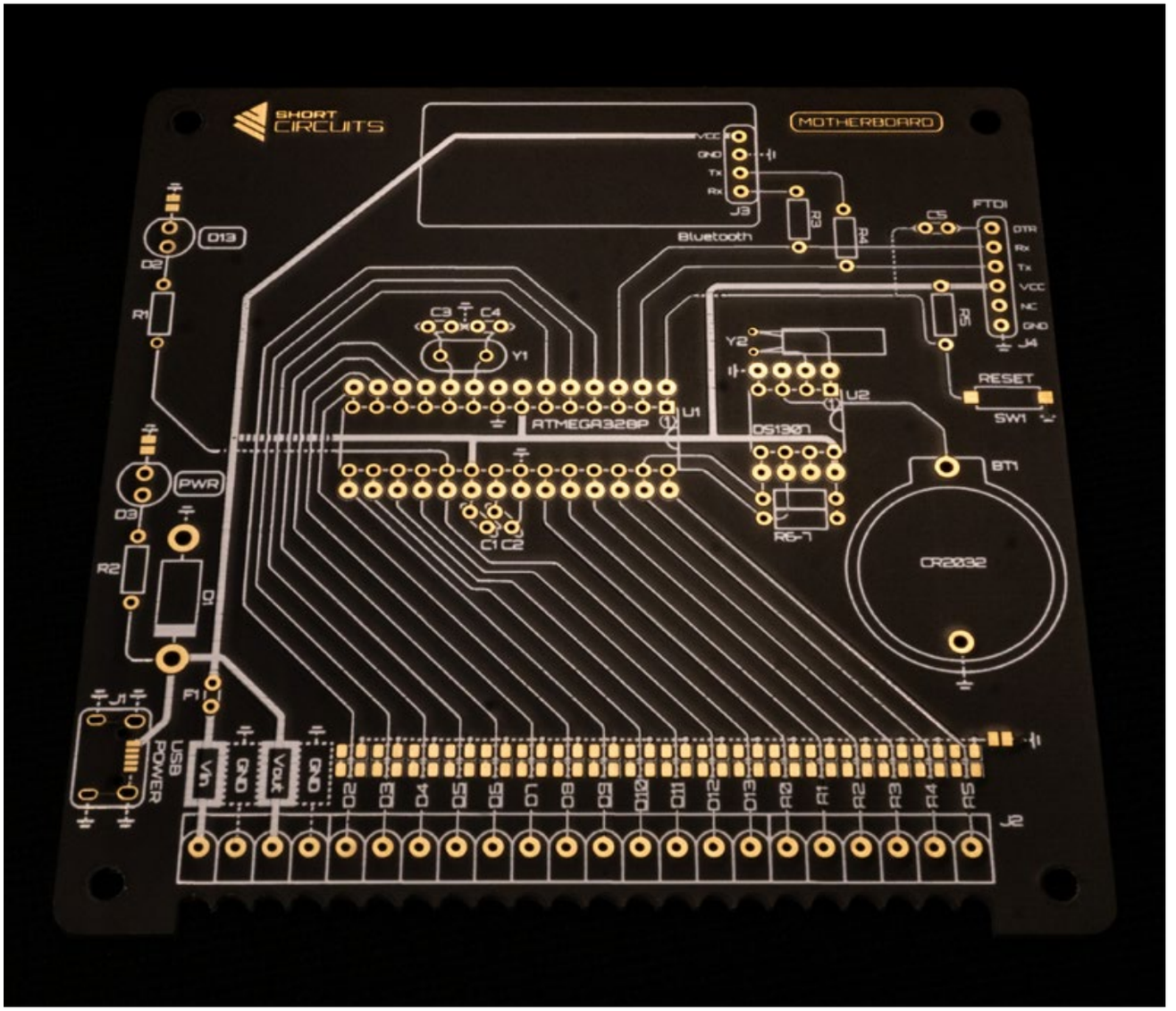
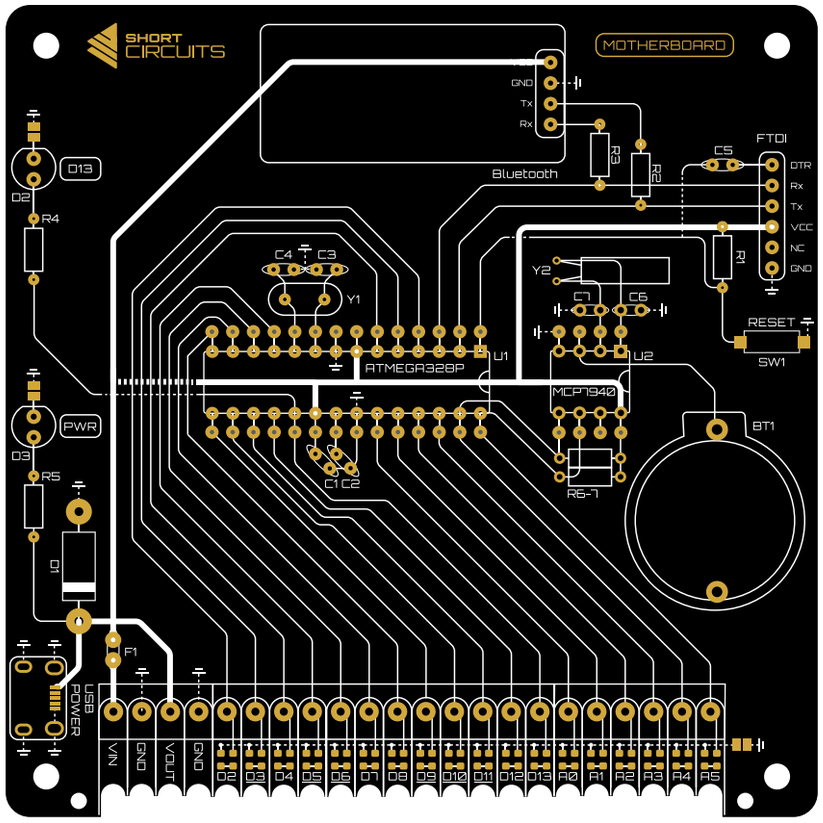
SHORT CIRCUITS

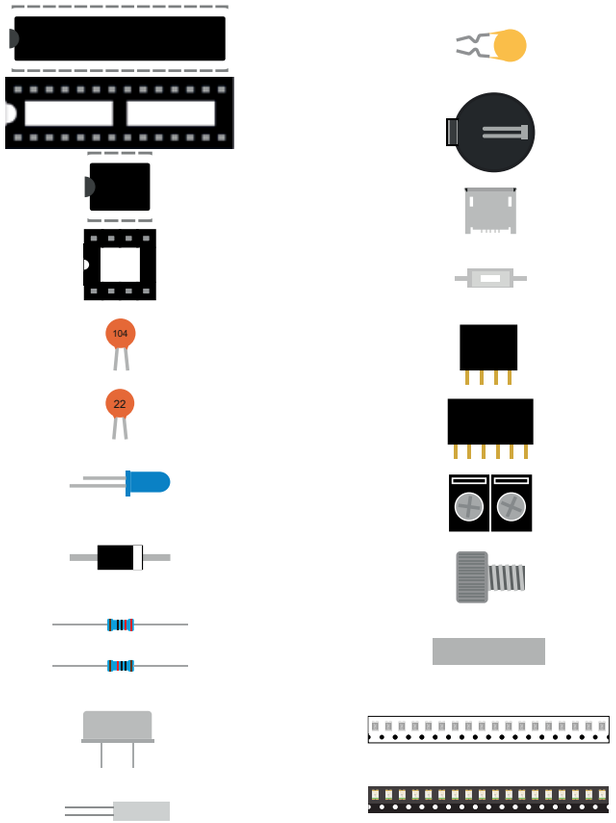


MOTHERBOARD

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 variety 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.

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.

Programming

This section will show you how to upload code to the Motherboard.

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

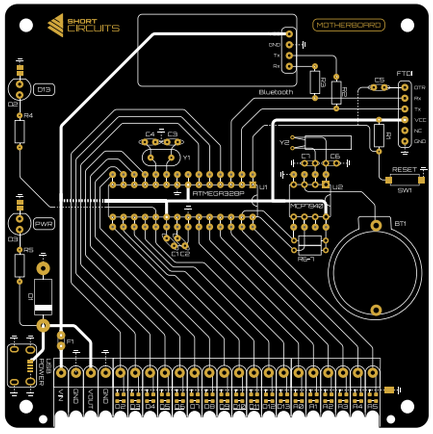
The Component Index will give you details about each component (except some connectors). We've included information about how the component works, its construction, how to find out if it has failed and much more. You can use this as a reference when designing circuits or trying to fix them.

MOTHERBOARD

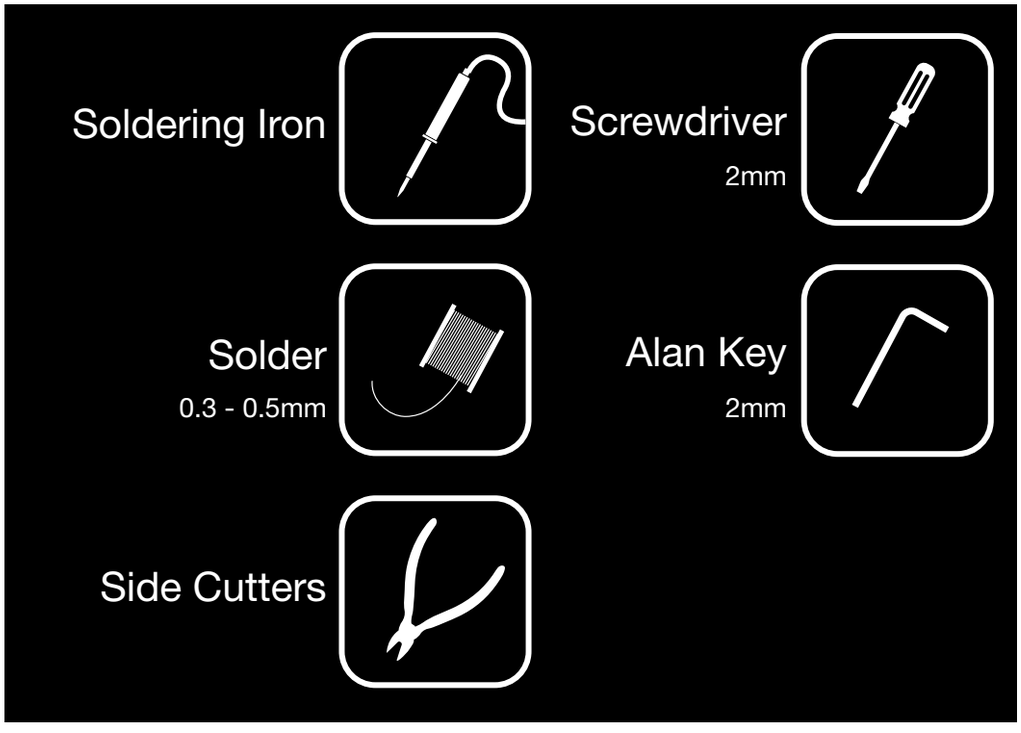
The MOTHERBOARD kit lets you build a circuit with a microcontroller at its heart. A microcontroller controls inputs and outputs depending on how you program it and what is connected to it. Inputs could be in the form of sensors, switches or other microcontrollers. Outputs could be as simple as an LED (Light Emitting Diode), number display or speakers, or as complicated as a 3D printer.

The MOTHERBOARD includes screw terminals for digital and analog Inputs and Outputs, a voltage protection circuit, headers to connect a USB FTDI module (used for programming), and a Bluetooth module. We've also included a Real Time Clock (RTC) circuit on board, so making a clock with the DIGITISER display is a breeze. This kit will be the "brain" of your project, so lets get familiar with its functionality!

Contents



1 x Printed Circuit Board



1 x Atmega328P-PU

1 x DIP-28 Socket

1 x DS1307 RTC

1 x DIP-8 Socket

3 x 0.1uF Capacitors

2 x 22pF Capacitors

2 x Blue LEDs

1 x Zener Diode

1 x Fuse

2 x 2K Ohm Resistors

5 x 10K Ohm Resistors

1 x 16MHz Crystal

1 x 32.768KHz Crystal

1 x 2032 Battery Holder

1 x USB Micro-B Socket

1 x Momentary Switch

1 x 4 Position Header

1 x 6 Position Header

11 x Screw Terminals

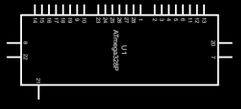
4 x 5mm M3 Hex Screws

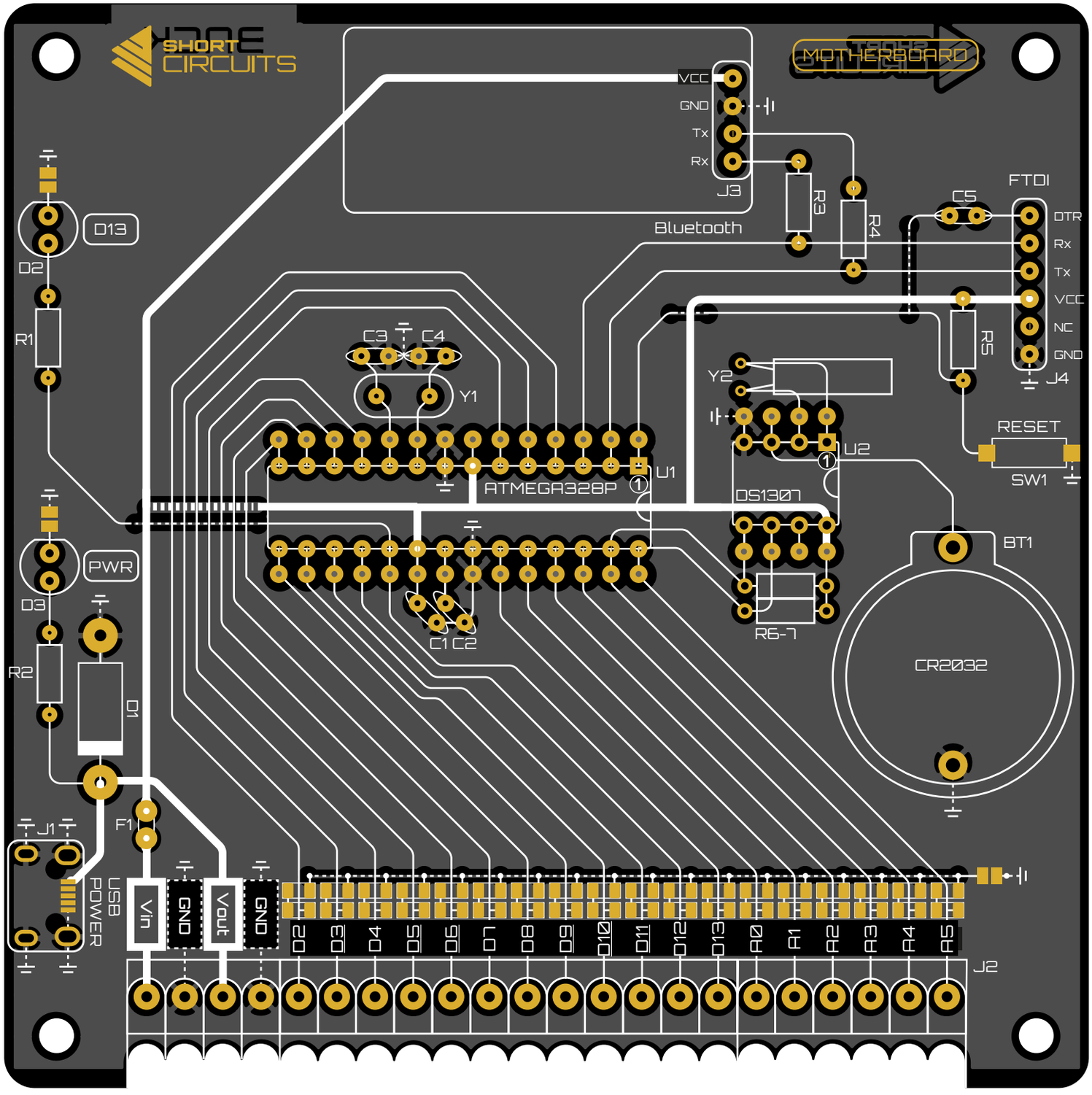
4 x Female/Female Standoffs

18 x 2K SMD Resistors

18 x Blue SMD LEDs

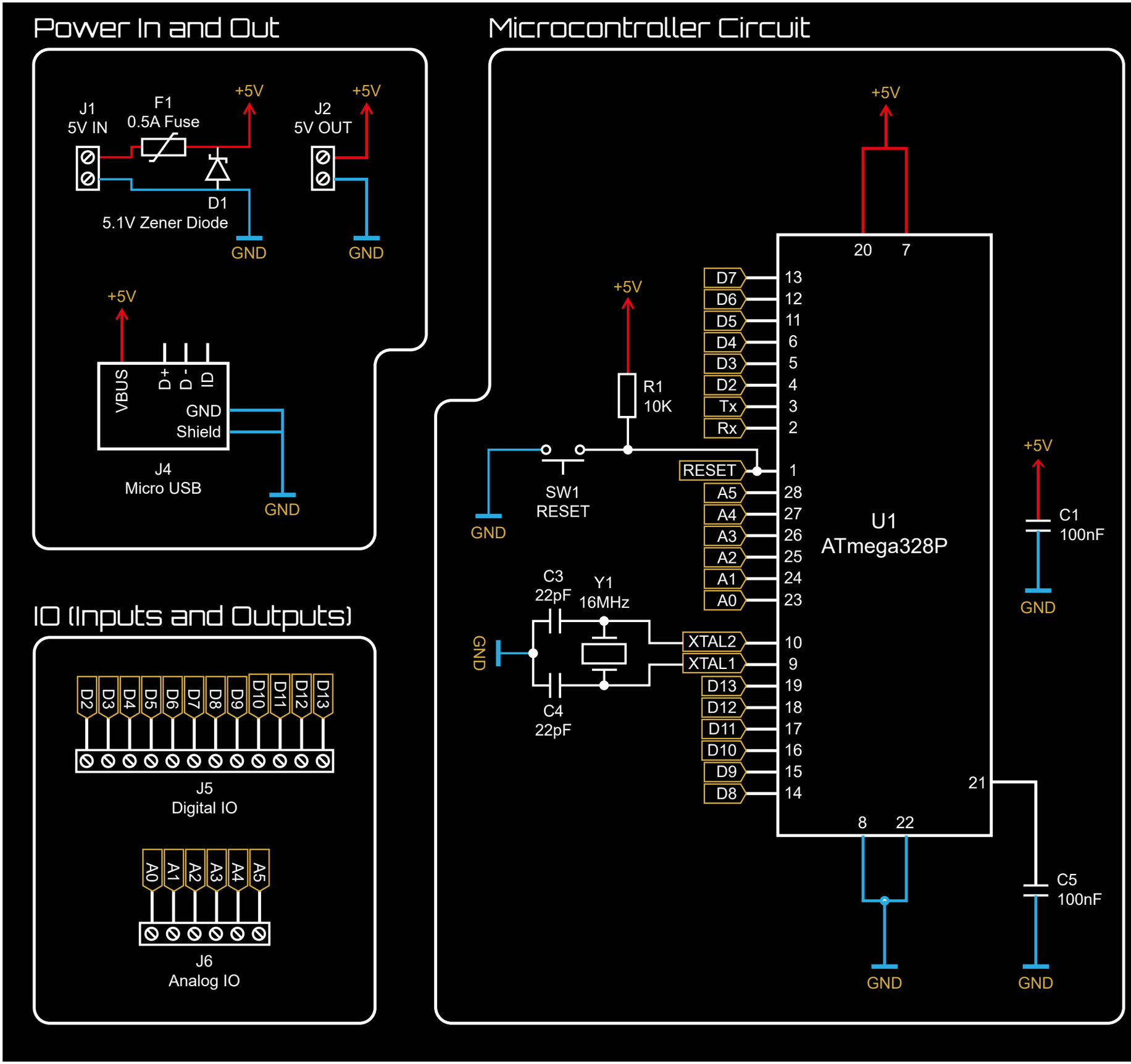
Circuit - Symbols and Designations

	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
Crystal			Y
Momentary Switch			SW
Header			J
Microcontroller			U
Connected to VCC			
Connected to GND plane			
Fuse			F
Micro USB			J
Zener diode			D
Screw terminal			J
Mechanical hole			
Jumper			JP

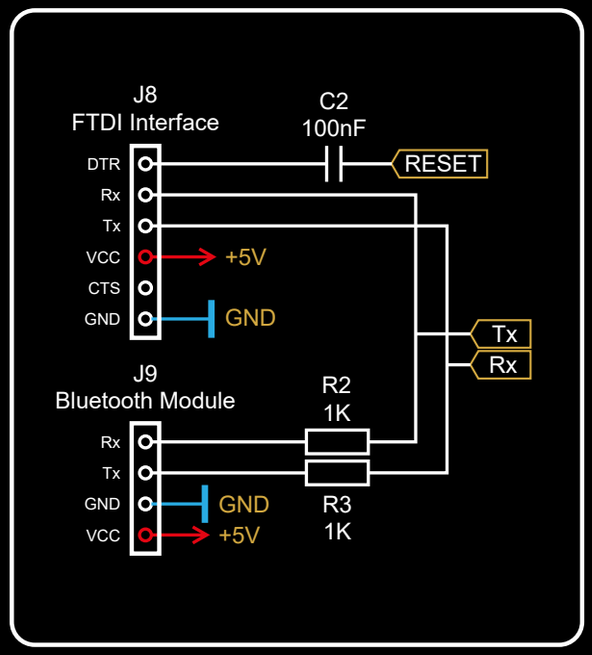


Ground (GND) Copper Area On Back of PCB

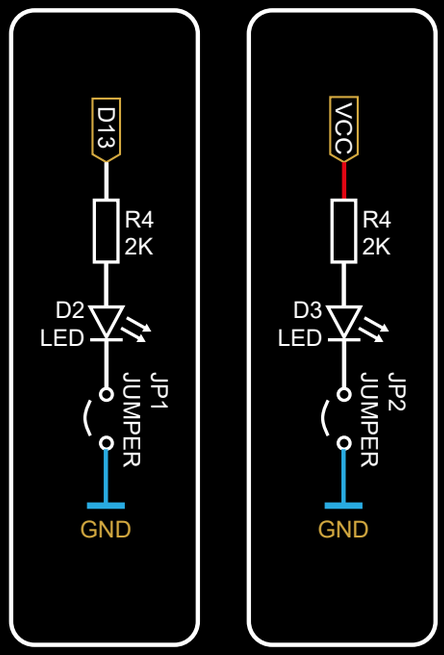
Circuit - Schematic



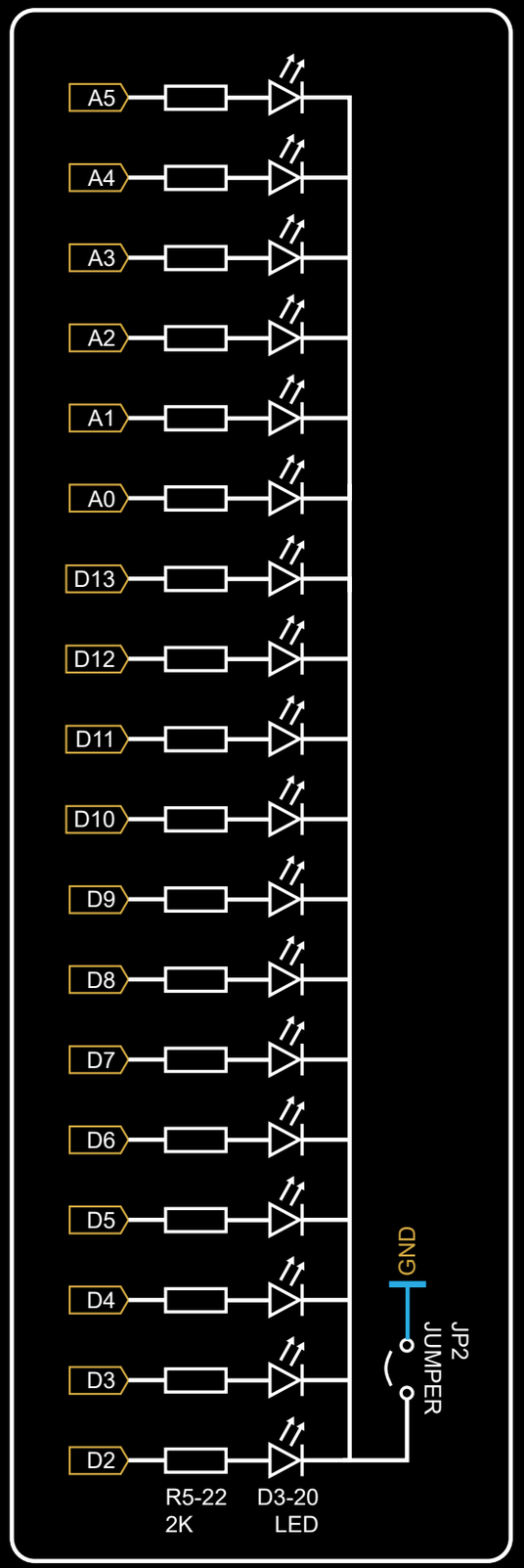
Module Headers



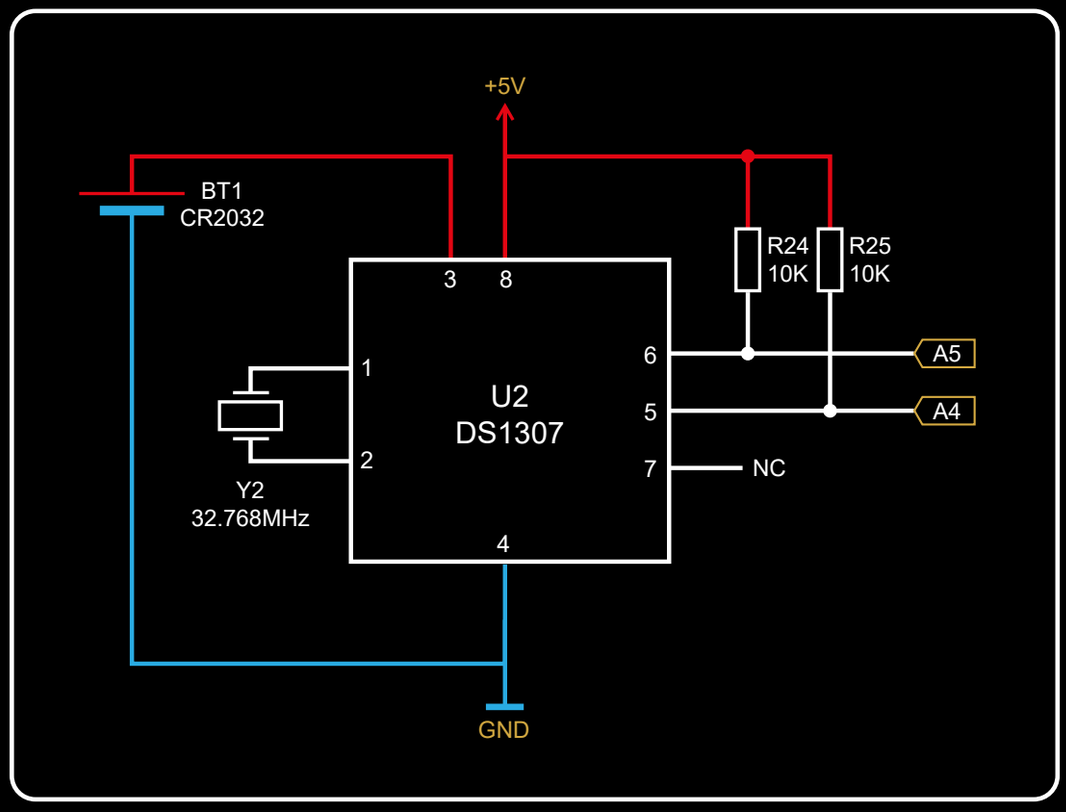
Pin 13 & Power LED



IO LED Array (Optional)



RTC Circuit



Bill of Materials (BOM)

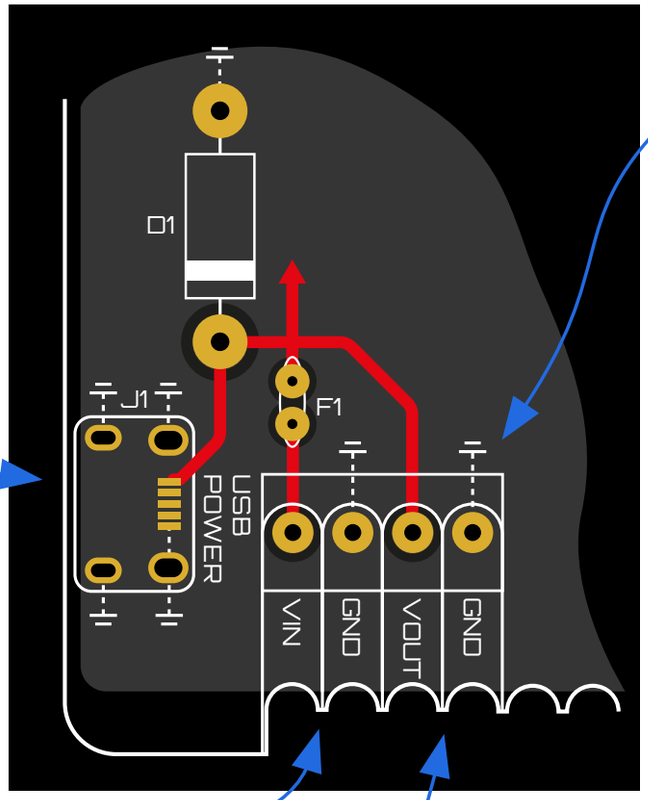
Designation	Value	Name	Footprint / Pitch	Datasheet
C1, C2, C5	100nF	Ceramic Capacitor	2.54mm	
C3, C4	22pF	Ceramic Capacitor	2.54mm	
D1	5.1V	Zener Diode	12mm	
D2, D3	25mA, 3.3V, Blue	LED	Ø5mm, 2.54mm	
D4 - D21 (opt)	20mA, 2.5V, Blue	LED SMD	0805	
F1	0.5A	Polyfuse	5.1mm	
J1	Micro B	USB		
J2	11 x 2 pos	Screw Terminals	3.5mm	
J3	4 pos	Pin Header	2.54mm	
J8	6 pos	Pin Header	2.54mm	
R1, R2	2K, 1/4W	Resistor THT	7mm	
R3 - R7	10K, 1/4W	Resistor THT	7mm	
R8 - R25 (opt)	2K, 1/4W	Resistor SMD	0805	
SW1	SPST	SPST Switch		
U1	DIP28	IC Socket	DIP28	
U1	ATmega328P	Microprocessor	DIP28	
U2	DIP8	IC Socket	DIP8	
U2	DS1307	RTC Chip	DIP8	
Y1	16MHz	Crystal		
Y2	32.768KHz	Crystal		
BT1	2032	2032 Battery Holder		

USB Power

You can power the board through the Vcc and GND terminal blocks. But USB is more convenient. It is a difficult part to solder, but don't worry, if you mess it up, you can snip an old USB cable and connect the red and black wires to the terminal block's VCC In and GND respectively. This USB socket is not used to communicate with a PC. The data connections aren't connected. This is because the ATMega328P can't communicate directly through USB. You need to use an programming board or a more feature packed microcontroller to translate for the 328P.

5v Input

This is where a regulated 5V can be connected to The MOTHERBOARD (and any other Boards connected to the 5V Output). This could come from your favourite bench power supply, or our upcoming POWER BOARD, which provides 5v (for all the 5v components in our circuit), and a variable voltage from 1.2v up to 20v+ (for most other needs). As mistakes do happen, we've added a Zener Diode and a Fuse to protect the rest of the circuit from over voltage. If the voltage exceeds the breakdown voltage of the diode (5.1v), current will start flowing through the Zener to ground, rather than the rest of the circuit where it can damage components. This will in turn increase the current draw to a point that would trip the fuse and save your components. (See Component Index for more info)



5v Output

Here we have a convenient 5v output, to power all the other boards that you want to use with your microcontroller. Power the DIGITISER, the RGB MATRIX, the SENSOR ARRAY, or any other board we produce. Boards powered from these outputs also benefit from the voltage protection circuit we added to the 5v input.

Ground Planes

The back of the board is covered with a layer of copper that is connected to GND. This is called the ground plane. We add this for a few reasons. One reason is that its just easier to design that way. When you need to connect something to ground, just connect it directly to the ground plane underneath the component. That way you don't have to worry about adding any ground traces all the way back to the power input. If this was a 4 layer board, then a 5V plane would also be a good idea. Another reason is to add some electromagnetic shielding to the circuit, this reduces the chance of erroneous results due to the effect of electromagnetic interference on our circuit.

Design Considerations

When laying out power traces, and the components themselves while designing PCBs on the computer, it is important to consider current loops. When electric current moves around your circuit, it can create electromagnetic interference (EMI). This messes with some sensitive components. The larger the loop, the more EMI is caused. This is one of the reasons why using the bottom copper layer as one big ground plane is preferable. This way, the current can take the shortest and most direct route back to the power supply. Try and make the 5V traces as direct as possible. When we designed this circuit, we had to balance this with readability, so our boards aren't optimally designed for EMI.

Circuit - Microcontroller

ATmega328P Microcontroller

The ATmega328P is the brain of our circuit. This chip can read changes in the voltage at its input/output pins and also set them to a chosen value. Using the Arduino programming software, we are able to program it to respond to inputs from other modules and send responses to other modules as outputs. This gives us the ability to create unlimited ways to manipulate other devices to build and program useful gadgets. You could read data from a Real Time Clock module then process the data to send to a 7-Segment display... A Digital Clock! Or you could read the data from a thermometer and process that data to send to an RGB LED array... A Graphic Thermometer! The possibilities are near endless!

Crystal Oscillator

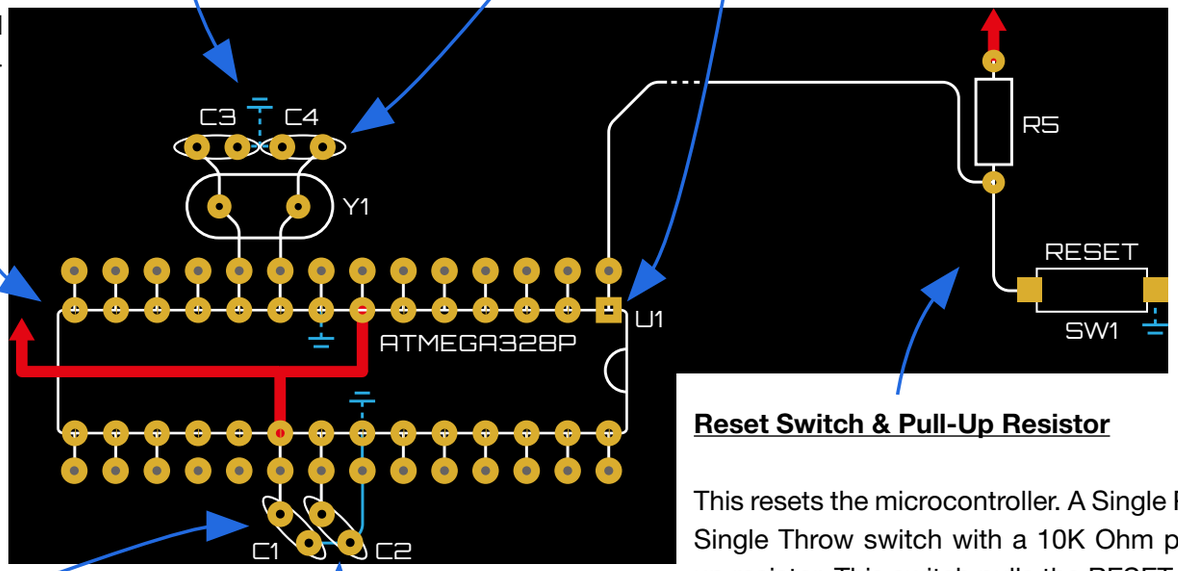
The crystal oscillator mechanically oscillates to provide a square wave signal at a stable reference frequency. This is used by the microcontroller as a clock signal, or its heart beat if you like. It determines how many times a second the microcontroller can do the actions needed to execute instructions.

Load Capacitors C3, C4

These are the load capacitors for the crystal oscillator. They are reactive components that help create the feedback loop that enables the crystal to start oscillating at the intended frequency.

Pin 1

The first pin of any IC is usually indicated by a square pad. Pins are numbered anti-clockwise.



Reset Switch & Pull-Up Resistor

This resets the microcontroller. A Single Pull Single Throw switch with a 10K Ohm pull-up resistor. This switch pulls the RESET pin on the microcontroller to ground (0V) when pressed. This resets the microcontroller. As electricity takes the path of least resistance, when the switch is not pressed, the only path is to VCC (5V) through the 10K resistor. When the switch is pressed, the easiest path is to ground. If the resistor wasn't there and the switch was pressed, there would be an uninterrupted path from VCC to GND. This is a short circuit, and will probably burn out the microcontroller.

Bypass Capacitor C1

A decoupling capacitor. It's function is to smooth out any noise (spikes or drops in voltage) due to other components affecting the power supply to the microcontroller. It 'decouples' the chip from the 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 (Integrated Circuit) pins as possible.

Bypass Capacitor C2

This is another decoupling capacitor that makes sure the analog reference voltage (AREF) measured from pin 21 of the microcontroller is at a steady value. In our case, that value is the same as our reference ground, or approximately 0v.

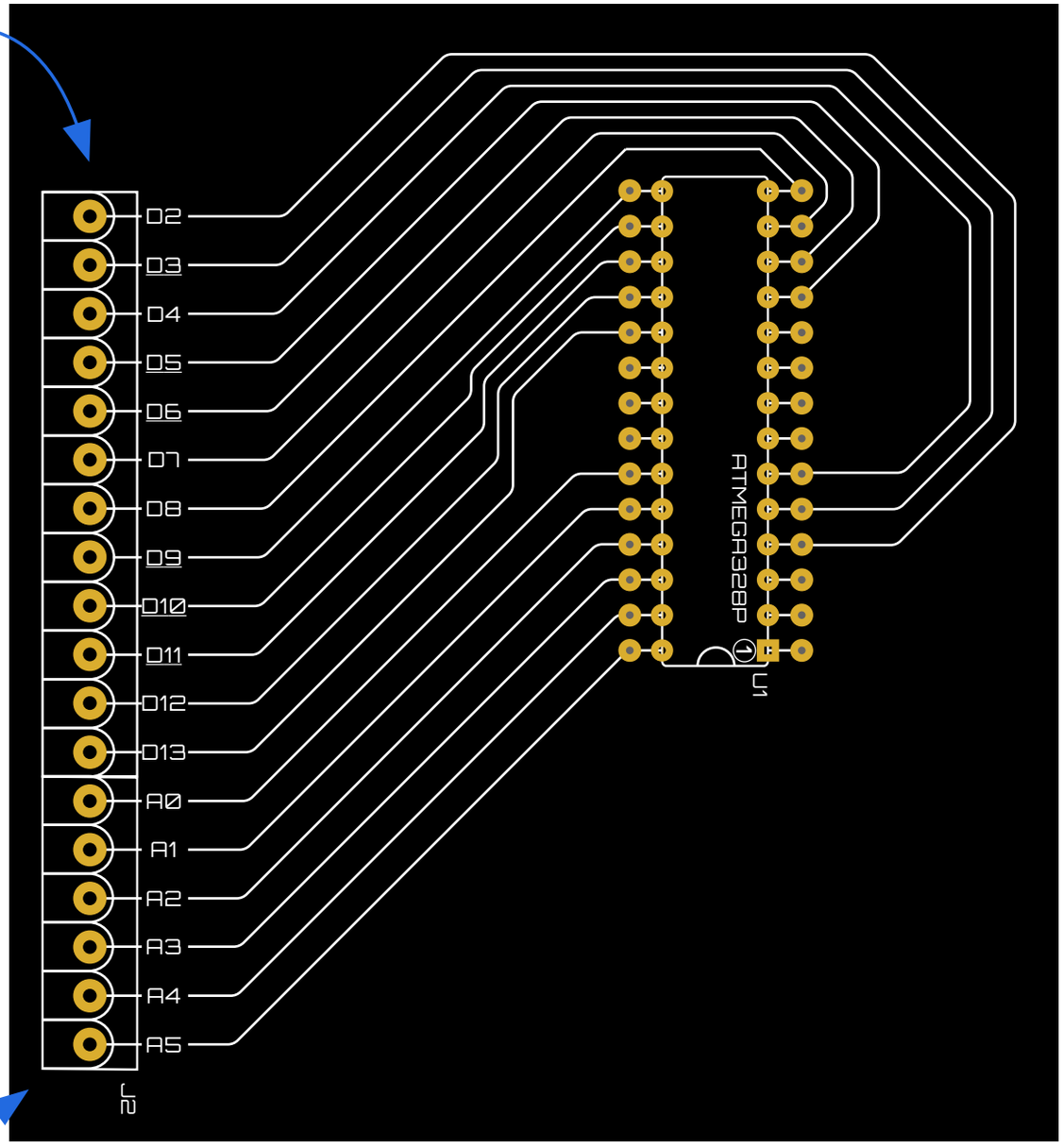
Digital IO

D2 to D13 are digital Inputs and Outputs. Digital pins can only read, or write a state of on (1, 5V) or off (0, 0V). We can however, still control the brightness of an LED through PWM (pulse width modulation), which uses the speed at which the microcontroller can turn on and off some of its digital pins. If the pulse is on 50% of the time, then the brightness will be 50%. This is called a duty cycle. D3, D5, D6, D9, D10 and D11 are PWM capable. Check for the line underneath its label on the board, or on the pinout diagram of the ATMega328P.

Analog IO

Analog pins can read voltage between 0V and the operating voltage of the microcontroller (5V) using the internal Analog to Digital Converter (ADC). The ATmega328P has a 10-bit ADC. The largest number that can be stored in 10-bits is 1024, and as such the analog pins have a resolution of 1/1024. Which means at 0V it will read a value of 1, at 5V it will read 1024 and at 2.5V it will read a value of 512. This is useful when reading things like photoresistors, or an analog audio input.

The ADC cannot output analog signals or even mimic them with PWM. Also, the programming syntax AnalogWrite() has nothing to do with the analog pins. To output sound you would use a PWM pin to create a square wave (alternating from on to off at a certain frequency).



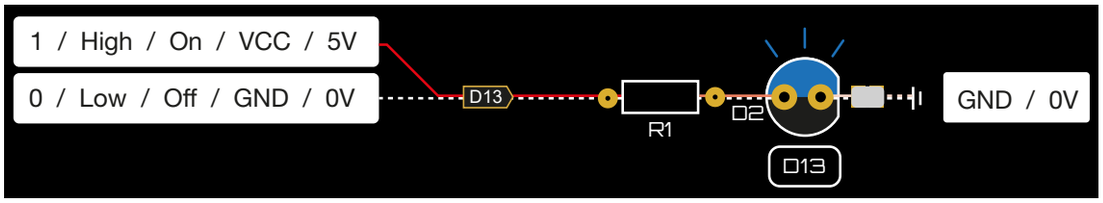
Circuit - D13 LED

D13 LED

The D13 LED is named after the reference Arduino gives to the digital Input Output (IO) pin of the ATmega328P it is connected to (see the pin-out diagram for all the pin numbers). It is connected to this IO pin because it is also the system clock pin of the SPI (Serial Peripheral Interface). When you are communicating with the microcontroller through serial, either with the FTDI module for programming, or any other device connected to Rx, Tx, MISO, MOSI, etc. The LED will turn on when SCK (same pin as D13) is pulled high, and off when it is pulled low. That's why you see it rapidly turning on and off when these things are communicating. It's great to check if your circuit is working without programming the chip or having another form of output connected.

You may ask why they aren't on all the pins as standard. Well, the current consumption of the LED, although small, may negatively affect a device connected to the same pin, or damage the microcontroller. The ATmega328P, according to its datasheet, has an absolute maximum current draw of 40mA per pin and a maximum total current draw of 200mA. Fortunately, the other kits in our series that source current from data pins (RGB MATRIX, DIGITISER etc.) all use shift registers, which source very little current from the ATMega's pins and instead get their power from the main 5V input. Shift registers also have a maximum current draw per output pin and per chip that need to be taken into account when designing with them, but there are ways around this. Check out the kits in the series that use them to find out more.

For your convenience, and for flexibility, we have added jumper pads between the LED and ground. If you want to use the LED, then solder these two pads together to allow a complete circuit through the LED. If you don't like the LED, or you want to connect a high current draw device to D13, then unsolder the jumpers.



How the circuit works

To turn the LED on, you will have to tell the microcontroller to output 5V from the relevant pin. In this case you could use the following code:

```
digitalWrite(13,HIGH)
```

(See the programming section for more info)

When the microcontroller sets its pin to 5V, the difference in voltage (potential difference) across the resistor and LED will be 5V. This is because the other end of the LED is connected to GND or 0V.
If the resistor wasn't added, 5V would be applied to the LED which would exceed its recommended limit of 3.3V (check the datasheet).

To reduce the current going through the LED we can use a resistor in series with it. To calculate the value of this resister, we use Ohm's Law. (See the section on resistors in the component index for more) Ohm's law describes the relationship between Voltage (V), Current (I) and Resistance (R). If we know the LED will cause a drop of 3.3V and the source voltage is 5V, then the Resistor will take care of the rest. So if we take the source voltage (5V) from the LED voltage (3.3V), we get the voltage across the resistor (1.7V).

$$V = 5 - 3.3$$
$$V = 1.7$$

From there we can use the typical current draw of the LED in amps (0.02), together with Ohm's law, to find the minimum resistance needed.

$$R = \frac{1.7}{0.02}$$
$$R = 85$$

If we used this value, the LED would be at full brightness and consume 20mA. That would be half the pins limit! We can reduce this by adding a much higher value resistor. We have used a 2000Ω resistor which limits the current to around 0.85mA. You could use an even higher value to reduce the current draw and the brightness of the LED. Using a lower value resistor than our calculated minimum would result in the LED having a shorter life.

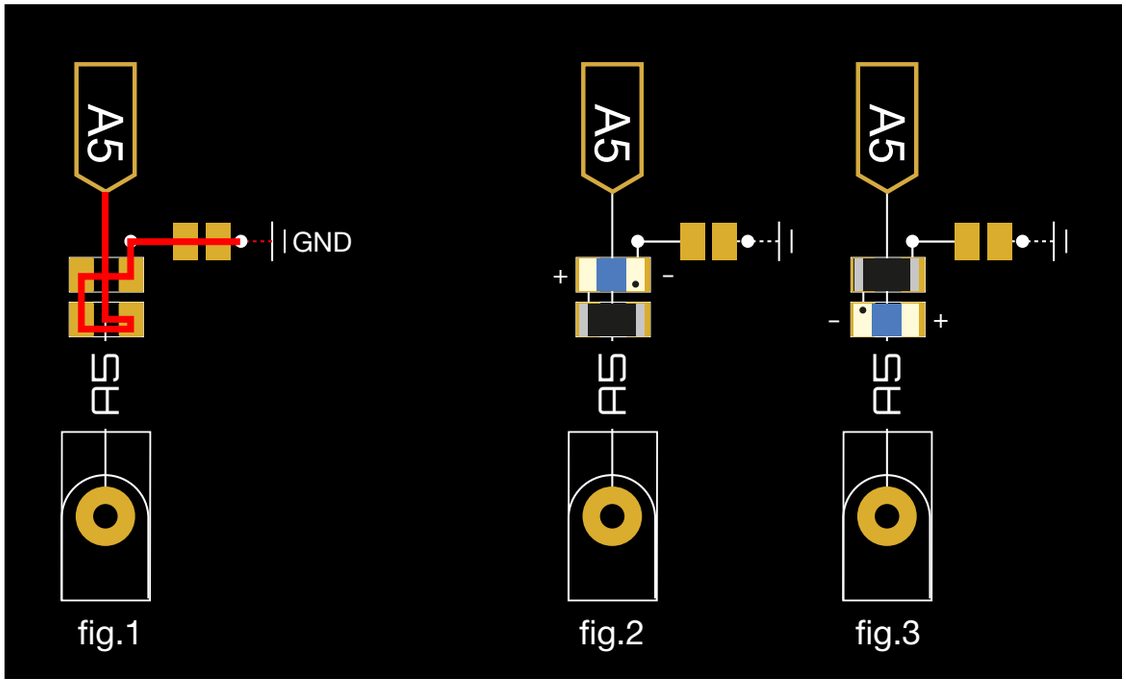
Circuit - 10 LEDs (Optional)

Data Indicator LEDs

As stated in the previous section, there are advantages and disadvantages to adding indicator LEDs to the data pins of the microcontroller. These are optional because they are very difficult to solder.

We can use the LEDs to help us understand the transfer of “data”, or 1’s and 0’s, off’s and on’s, between the microcontroller and what it is connected to. A great project to help with this is to slow down the data transfer rate between the microcontroller and a shift register. When the pins are communicating at a normal rate they will be turning on and off so fast that they will show a consistent brightness. A clock pin LED will look brighter than a latch pin LED as it will be turning on more often. By adding a delay between each step in the code, you can see exactly what is happening with the data, clock and latch pins when serial data is being sent. It’s a great way to visualise the process (Learn more about this from our other kits that use shift registers).

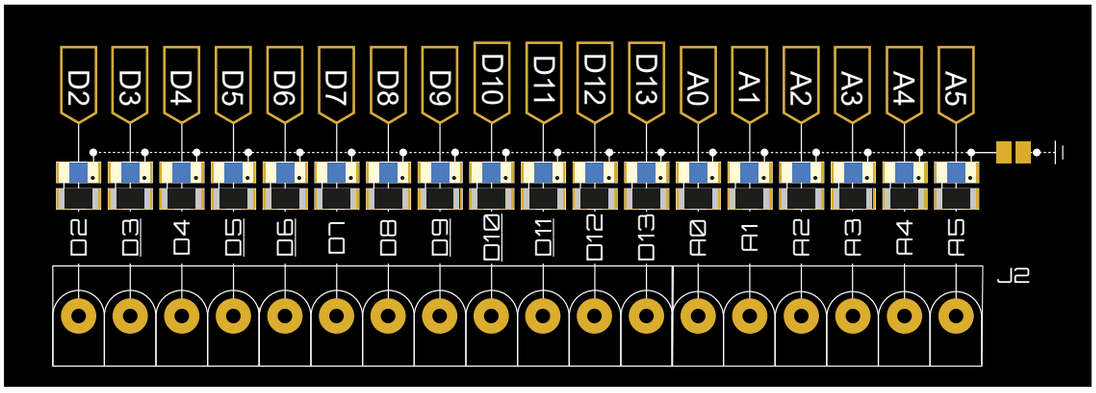
As space is a concern, and because they aren’t needed for normal functionality, we decided to use surface mount, 0805 (2012 in metric - 2.0mm x 1.2mm) resistors and LEDs. Because of this, it is rather difficult to solder by hand. We recommend a steady hand and to watch our quick how-to video before tackling this upgrade.



There is no need to worry about the polarity of the resistor, but the LED must be oriented correctly. The indicator triangle, line or other mark on the LED indicates the cathode, or ground terminal. Make sure this indicator is pointing in the direction of ground in the circuit (follow the red line from the A5 pin to Ground in fig.1). It doesn’t matter if the resistor or the LED comes first in the circuit so check the two images on the page for the different ways to install the components (fig.2, fig.3).

To turn on the LED circuits, bridge the jumper pads with solder. If you have successfully installed these components and decide you don’t need them any more, you can desolder the jumper pads to disconnect them from ground and break the circuit.

As the pin is acting as the positive end of the circuit, when the pin goes HIGH (5V) the LED will turn on. When the pin goes LOW (0V), the LED will turn off.



Circuit - FTDI and Bluetooth Headers

Bluetooth Header

Want to control your project from your phone? How about programming it wirelessly? Well, here's a handy header to plug in a Bluetooth module to make those things possible. This module connects via serial communication. Rx on the microcontroller (Pin 2) connects to Tx on the module, and Tx (Pin 3) to Rx, just like the FTDI module.

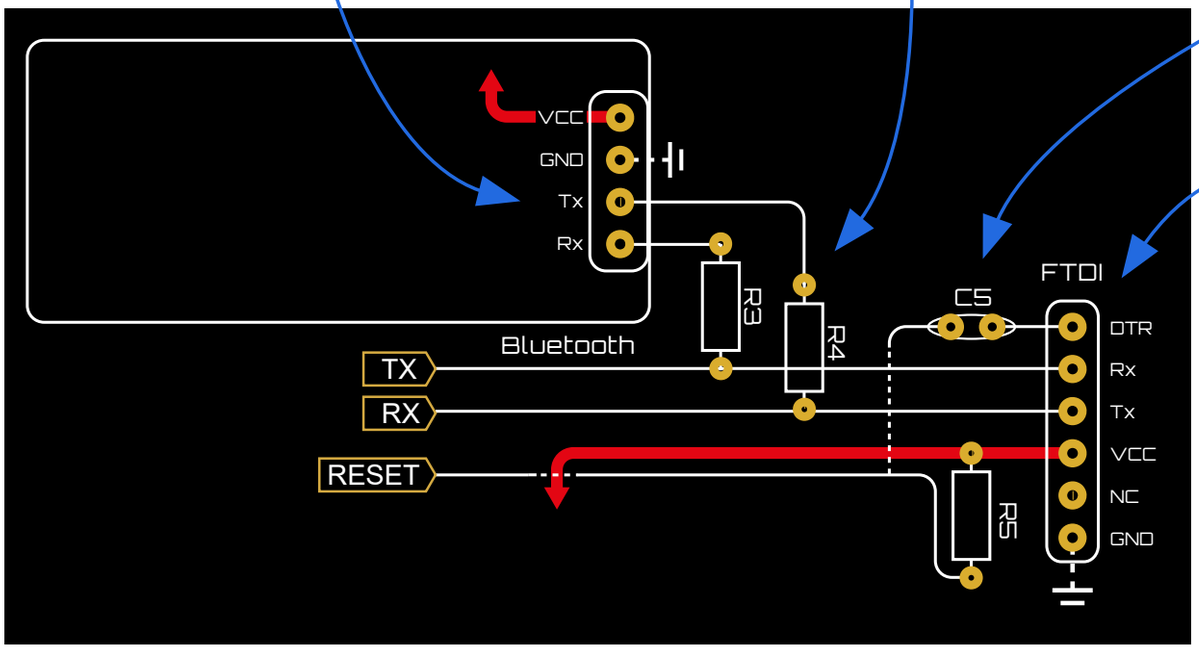
When purchasing a BT module, look for an HC-06 module with 4 pins.

Resistors

We don't recommend connecting a Bluetooth module and using an FTDI module to program the chip at the same time. If Tx on the BT module were to go high (5V) while Tx on the FTDI module went low (0V) while connected, you have 5V connected directly to ground, which is a short circuit. Because of this, we have put 10k resistors between the FTDI and Bluetooth's data lines. If there is a short, the amount of current would be minimal.

Capacitor

When the FTDI module communicates with the microcontroller, the microcontroller needs to be reset at a specific time. To do this, the RESET pin of the microcontroller needs to be set low then high, just like pressing and releasing the reset button manually. When DTR on the FTDI chip goes low, the capacitor is drained and the RESET pin goes low. After a short time, the capacitor will be charged up through the pull-up resistor (R5), back to 5v, which is the same as releasing the button.



FTDI Header

There are a few ways to communicate with the microcontroller to program its functionality. You could take the chip and plug it into a compatible Arduino Uno that has the USB functionality on board, then plug that into a PC. You could program it via another device over Bluetooth (using a Bluetooth module). Or you could use a USB interface chip and accompanying circuitry (The FTDI module for example). We'd love to put that on board for you to build, but all these chips are surface mount, so not easy to DIY. We've added an FTDI module header instead. These modules are cheap, easy to find and there is plenty of documentation available. Make sure the pins match when buying and inserting the module.

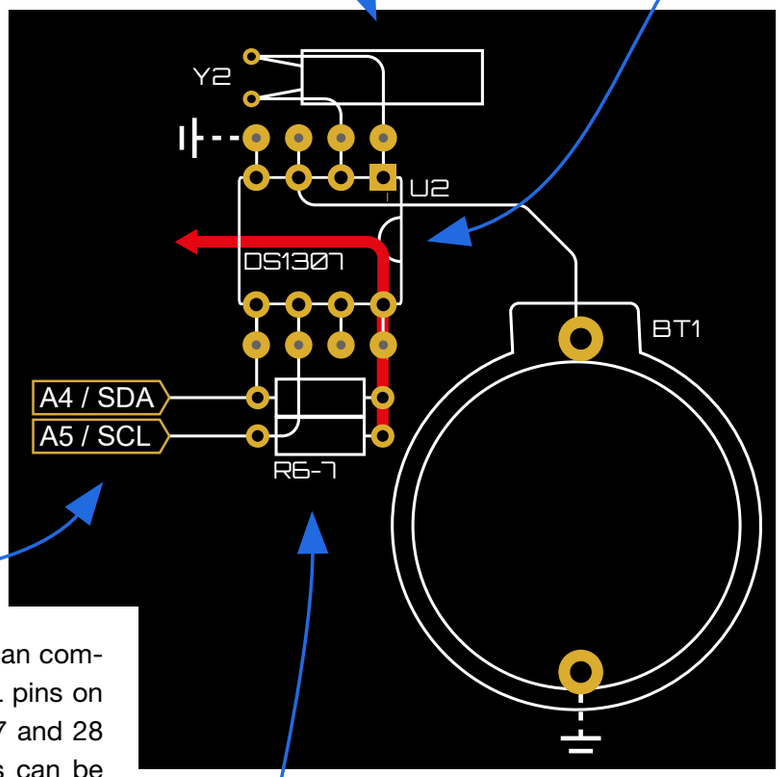
Circuit - Real Time Clock

Crystal

Just like the ATMega328P, the DS1307 needs a crystal to help it keep time. In this case, the chip requires a 32.768MHz crystal. The membrane in the crystal vibrates at 32,768,000 times a second. This oscillator circuit does not need extra capacitors like the ATMega328P's as this is dealt with inside the Integrated Circuit (IC).

DS1307 RTC IC

An RTC (Real Time Clock) module has an RTC chip, a crystal, resistors and a coin cell. Its job is to keep time even if the rest of the device has lost power. RTC chips run on very little current, so they can keep time for a long time on one coin cell. This is great for when you want to build a clock using the Digitiser kit, or even a graphical clock with the RGB Matrix. It is also useful for scheduling functions. Maybe you want the RGB LEDs to turn on and slowly change brightness as you wake up... So many possibilities!



I²C Communication

I²C is another way certain devices can communicate. It uses the SDA and SCL pins on a microcontroller. These are pins 27 and 28 on the ATmega328P. Many devices can be connected to the same I2C pins on a microcontroller. So go ahead and use the SDA and SCL screw terminals to add other I2C devices. Just remember, if the two devices have the same address, then they can't be used together. Check the datasheets first. Some I2C devices have an address select pin to choose between two addresses. Connecting that pin to ground or VCC will give you one address or the other.

Pull-Up Resistors

Just like the reset switch, these are pull up resistors that hold SDA and SCL signal lines high (at 5V) until they are pulled low by the communicating devices.

2032 Battery Holder

This holds the 2032 coin cell battery (sold separately). This is used as backup power for the RTC. It enables the chip to store the correct time even if power is removed from the board.

Assembly Instructions - Tips

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's 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. Too hot and the tip will oxidise too quickly,..

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.

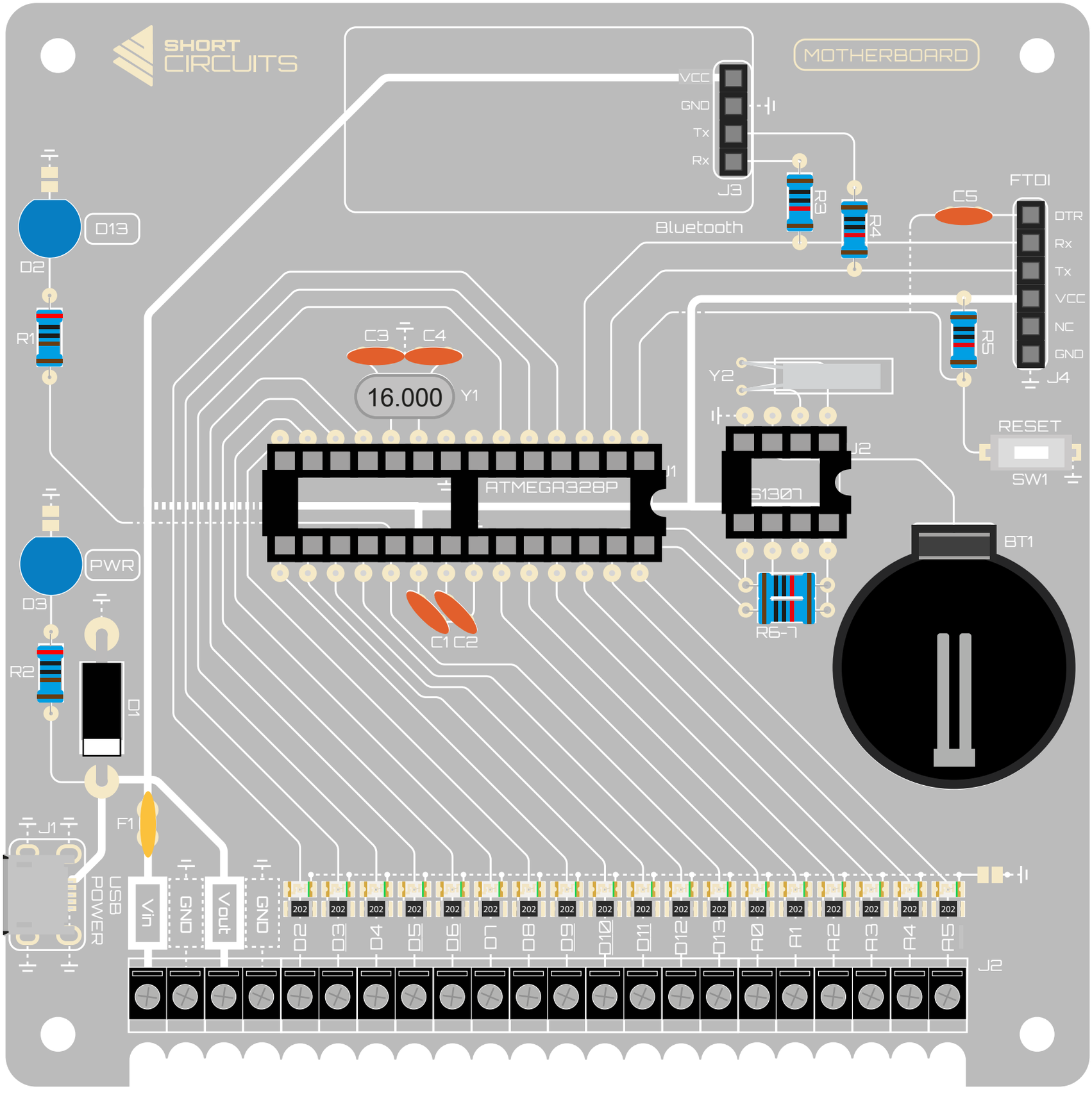
Thinner is better. Working with a thick wire of solder can get messy. Use 0.4-0.5mm solder for more control. You will have to feed more into the solder joint, but you have more control when there are other pins close by that you want to avoid. This is essential when soldering surface mount components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

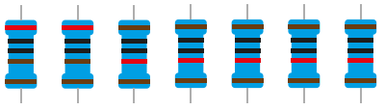

SHORT CIRCUITS
MOTHERBOARD
VCC
GND
Tx
Rx
J3
Bluetooth
R3
R4
C5
FTDI
DTR
Rx
Tx
VCC
NC
GND
J4
D13
D2
R1
C3
C4
16.000
Y1
Y2
R5
RESET
SW1
J2
ATMEGA328P
J1
S1307
BT1
PWR
D3
R2
D1
C1
C2
R6-7
F1
J1
USB POWER
Vin
GND
Vout
GND
202
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
A0
A1
A2
A3
A4
A5
J2

ASSEMBLY

Assembly Instructions

There are many ways to assemble a PCB that uses through hole components. One is to solder all the lowest profile components first. This makes sure they are supported when you flip the board. Using sticky tape or tack to hold components in place negates the need, so we'll order them by difficulty and ease of access. Check the bill of materials to make sure you have everything you need. Also, MAKE SURE YOU PUT THE COMPONENTS IN THE FRONT OF THE BOARD, NOT THE BACK!!!

Resistors



Always remove resistors from the packaging by snipping the leads, not by pulling them out of the packaging. This prevents any glue from being transferred to the holes in the PCB, or into the holes in your breadboard.

Bend the leads as close to the resistor as possible to form a U shape. Then insert each lead into the holes nearest the specific designator (R1, etc.). Insert all the resistors and use sticky tape or sticky tack to hold them in place. Resistors are not polarized, so they can be inserted in either orientation.

Flip the board so the leads and the bottom side of the pads are easily accessible. Tin your iron by applying some solder, then wiping it clean. The tip should be shiny. Now, with solder in one hand, and iron in the other, apply heat with the iron to the underside of the pad and the lead sticking through, while touching the solder to the pad. Solder should wick around the lead forming a mountain and covering the pad. Now remove the iron from the pad. This should all take no longer than a few seconds. Try to avoid holding heat to the pads for too long. A hotter iron is safer in this respect, as you shouldn't have to hold it on to wait for the solder to melt. After it cools, the joint should remain relatively shiny. Repeat the process for the rest of the resistor leads. Flip the board back over and remove the tape/tack. If any of the resistors aren't flush with the PCB, apply a little heat to the underside while pushing the resistor from the top side. Now snip the leads just above the solder.

Capacitors

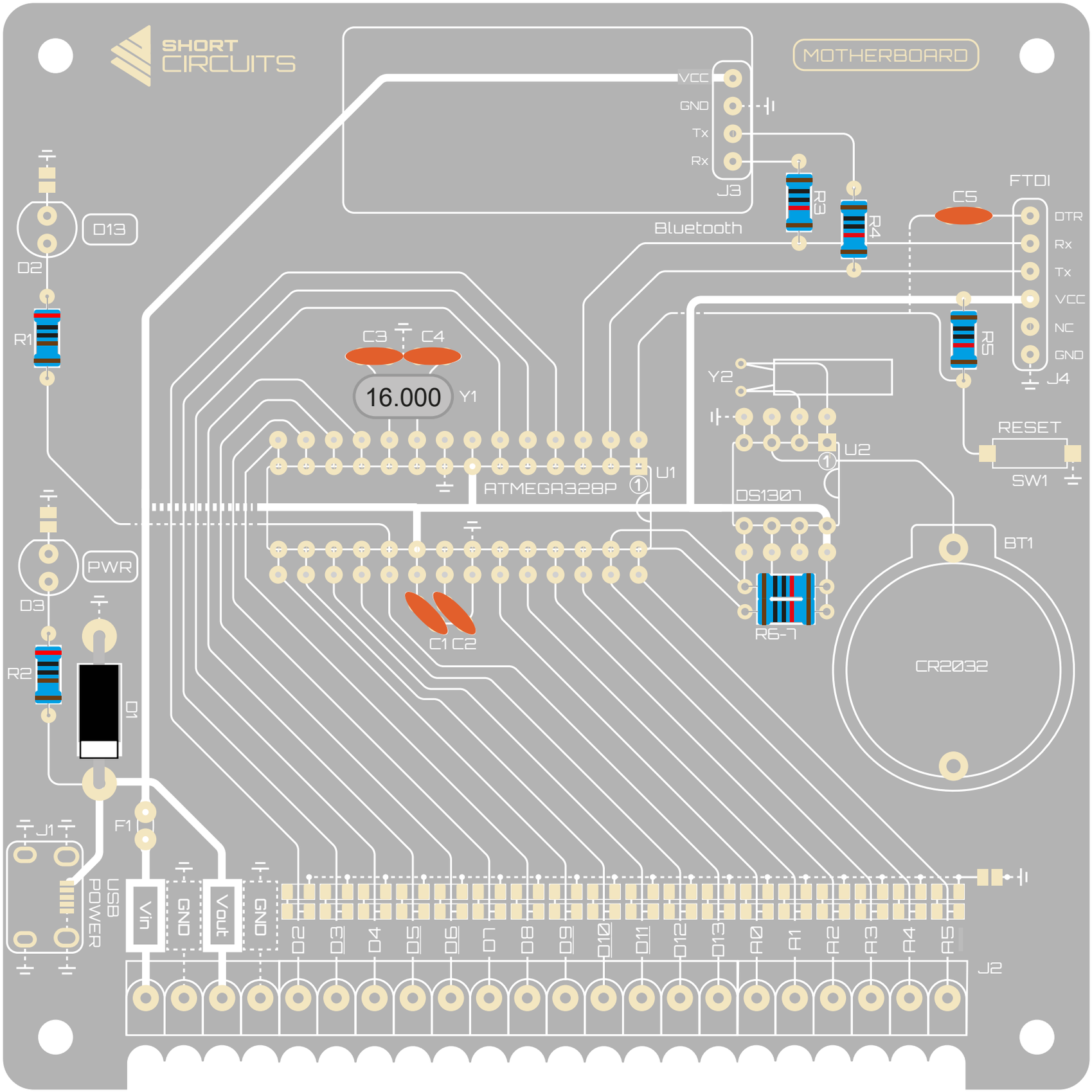
The capacitors are of the ceramic variety and thus polarity is of no concern. There are however, two different valued capacitors in this kit. The 22pF caps are for the crystal oscillator circuit (C3 and C4). The 100nF (0.1uF) caps are placed at C1, C2 and C5. Check the BOM while assembling PCBs so you get the right value components in the right places.

Zener Diode

The Zener Diode has very thick leads. So you may have to turn your iron up a little. Short and hot is better than long and cold when soldering. Bend the leads into a U shape and insert them through the holes near the D1 designation. Pay close attention to the polarity. Match the white band on the diode with the white band on the silkscreen on the PCB. The white band indicates the negative side of the diode. Yes the positive side is connected to ground. This is because we are using the breakdown voltage value of the diode. If we exceed the breakdown voltage, current will flow in the opposite direction to the normal flow, through the diode, towards ground, rather than through our microcontroller or other components.

16MHz Crystal

These are the next shortest components. Start with the Crystal. Polarity doesn't matter with this component, but there is writing on it, so, if you love chaos, put it in upside down, or don't. Not much to this one, just solder it like you did the resistors. These have thicker leads, so a little more heat may be needed. Check it's flush with the board and apply pressure and heat if it isn't.


SHORT CIRCUITS
MOTHERBOARD
VCC
GND
Tx
Rx
J3
Bluetooth
R3
R4
C5
FTDI
DTR
Rx
Tx
VCC
NC
GND
J4
D13
D2
R1
C3
C4
16.000
Y1
Y2
R5
RESET
SW1
U2
DS1307
U1
ATMEGA328P
BT1
CR2032
PWR
D3
R2
D1
C1
C2
R6-7
J1
F1
USB POWER
Vin
GND
Vout
GND
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
A0
A1
A2
A3
A4
A5
J2
ASSEMBLY

Assembly Instructions

32.768MHz Crystal

Thread the legs of the crystal through the holes marked Y2 then lay the crystal flat on the PCB within the rectangle. You can use a piece of sticky tack to hold it in place while you solder. Flip the board over and solder.

LEDs

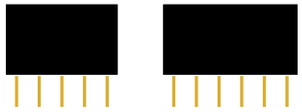
LEDs are polarity sensitive. The short lead is negative and the long lead is positive. You can also tell by finding the flat edge on the rim of the LED, that's the negative side. The flat part of the symbol on the PCB is also negative, so match them when inserting the LED. Solder the leads from the underside like all the other components.

DIP28 + DIP8 Socket

You could solder the chip directly to the board, but, there are a few disadvantages to this. If you accidentally solder it in the wrong way, then you have to desolder it. Which is a terribly laborious process. If you decide to use an Arduino Uno to program the chip, or you accidentally fry the chip, you need to be able to switch it out. This is why we use a DIP socket.

Soldering the 28 pins may seem like a daunting task, but it's actually pretty easy. Insert the socket into the area marked U1. Pay attention to the semi-circle indicator and match the one on the socket with the one on the board. Now, stick it in place and solder 1 corner pin, then the diagonally opposite corner. Now check alignment and whether its flush with the board. Add pressure and heat where needed to get it in the perfect spot. Now you can solder all the other pins knowing that the socket will stay exactly where it needs to be. The DIP8 Socket would be treated the same as the previous socket.

Headers

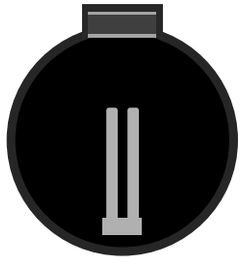


You can treat this like the DIP Socket. Solder an end pin, then the other end pin. Align it with heat on the underside and pressure from the component side. Then solder the inner pins.

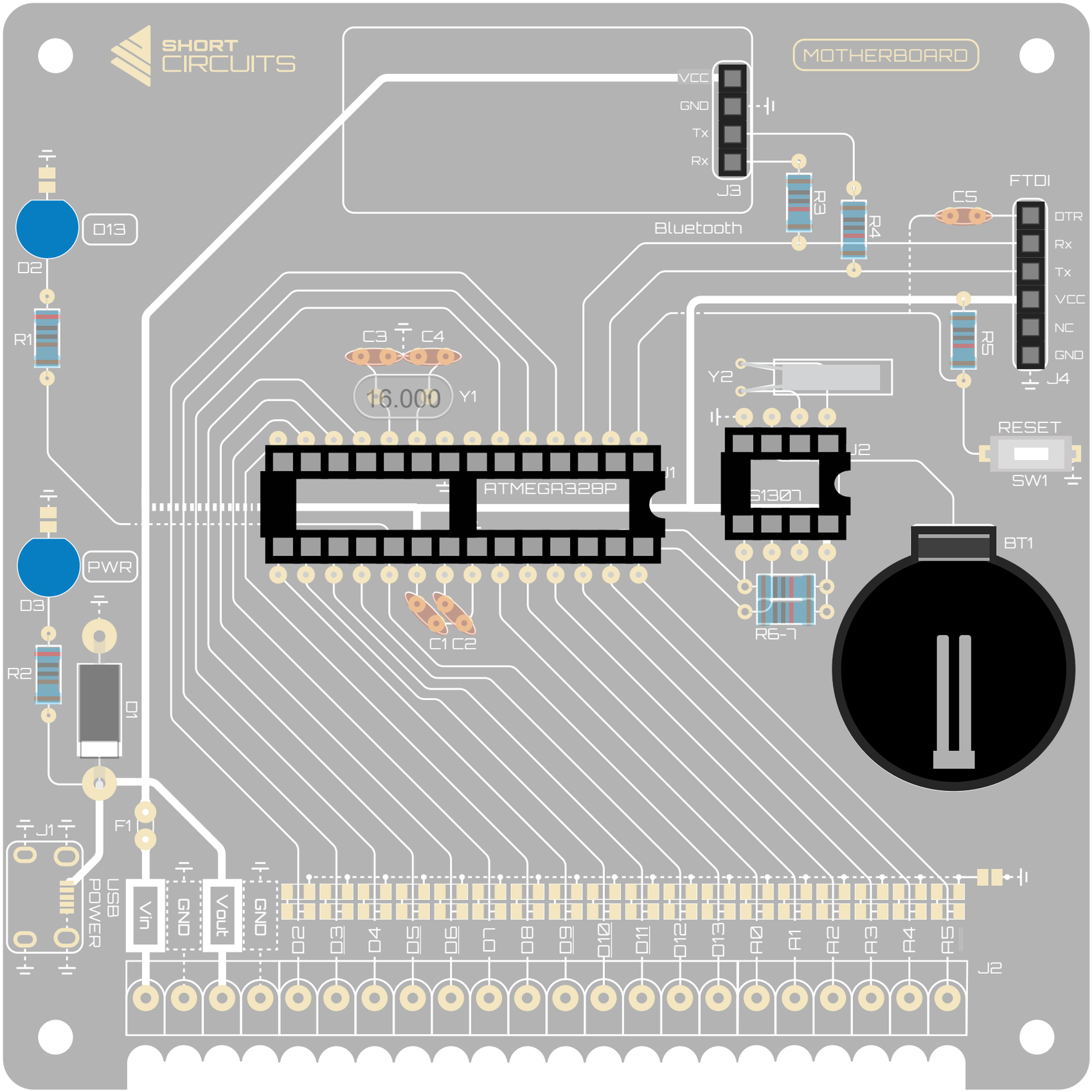
Switch

The switch is a surface mount component, so entails a different assembly process. Without leads to hold the component in place, and having to solder on the same side as the component, it's a little tricky to hold surface mount components in place while soldering. It is usually a good idea to solder one pad first, no matter how many pins the component has. This ensures the component will stay in place while soldering the other pins. First, apply some solder to the pad. Then hold the switch in place with tweezers and heat the solder thats on the pad. The switch is not polarized, so either orientation is acceptable. When you're happy with the position, solder the other pin in place.

Battery Holder



Position the battery holder to match the picture on the board. Sticky tack in place and solder those pins. Easy.


SHORT CIRCUITS
MOTHERBOARD
VCC
GND
Tx
Rx
J3
Bluetooth
R3
R4
C5
FTDI
DTR
Rx
Tx
VCC
NC
GND
J4
D13
D2
R1
C3
C4
16.000
Y1
Y2
R5
RESET
SW1
J2
ATMEGA328P
J1
S1307
BT1
PWR
D3
R2
D1
C1
C2
R6-7
F1
J1
USB POWER
Vin
GND
Vout
GND
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
A0
A1
A2
A3
A4
A5
J2

ASSEMBLY

Assembly Instructions

Micro USB Socket

The micro USB socket is by far the hardest component to solder in this kit. The pins are tiny, and it involves both through hole and surface mount soldering. It's tricky. Patience will come in handy with this one. If you mess it up, don't worry, you can still power the board using the terminal blocks. Just cut an old USB cable and insert the red and black wires into the VCC In and GND terminals respectively.

Insert Inspirational Yoda Quote

Fortunately, the socket should snap in to the through holes and hold itself in place. So, snap it in (making sure the surface mount pins are aligned with the pads), flip the board over and solder the through hole pins in place. Now flip the board and add solder to the top of the pins. Hold the iron flat against the case of the usb so the solder sticks to it.

Once the through holes are soldered, it's time to tackle the surface mount pins. You could just solder pin 1 (GND) and pin 5 (VCC), as that is all we are using. But, we are here to learn, so you may as well solder them all. The suggested technique tackles all the pins at once anyway. Clean your iron (again, you should be doing this before, after, and during and soldering activity), then apply a little heat to all the pins and pads. After a few seconds, touch the solder to the pads, adding just a tiny amount. Now wipe the iron away from the pins to make sure each pin is soldered. Add a little more if it needs. If there is too much solder, clean your iron, touch the pads, then quickly drag back away from the socket. This should pull the excess solder away with the iron. Repeat until each pin has enough solder, but not enough to bridge between two pins. Giving the area a quick scrub with a toothbrush and some Isopropyl Alcohol will clear any excess flux that may cause issues if we were using the socket for data. A good habit to get in even though we are only using the socket for power.

Breathe.

IO LEDs & Resistors (Optional)

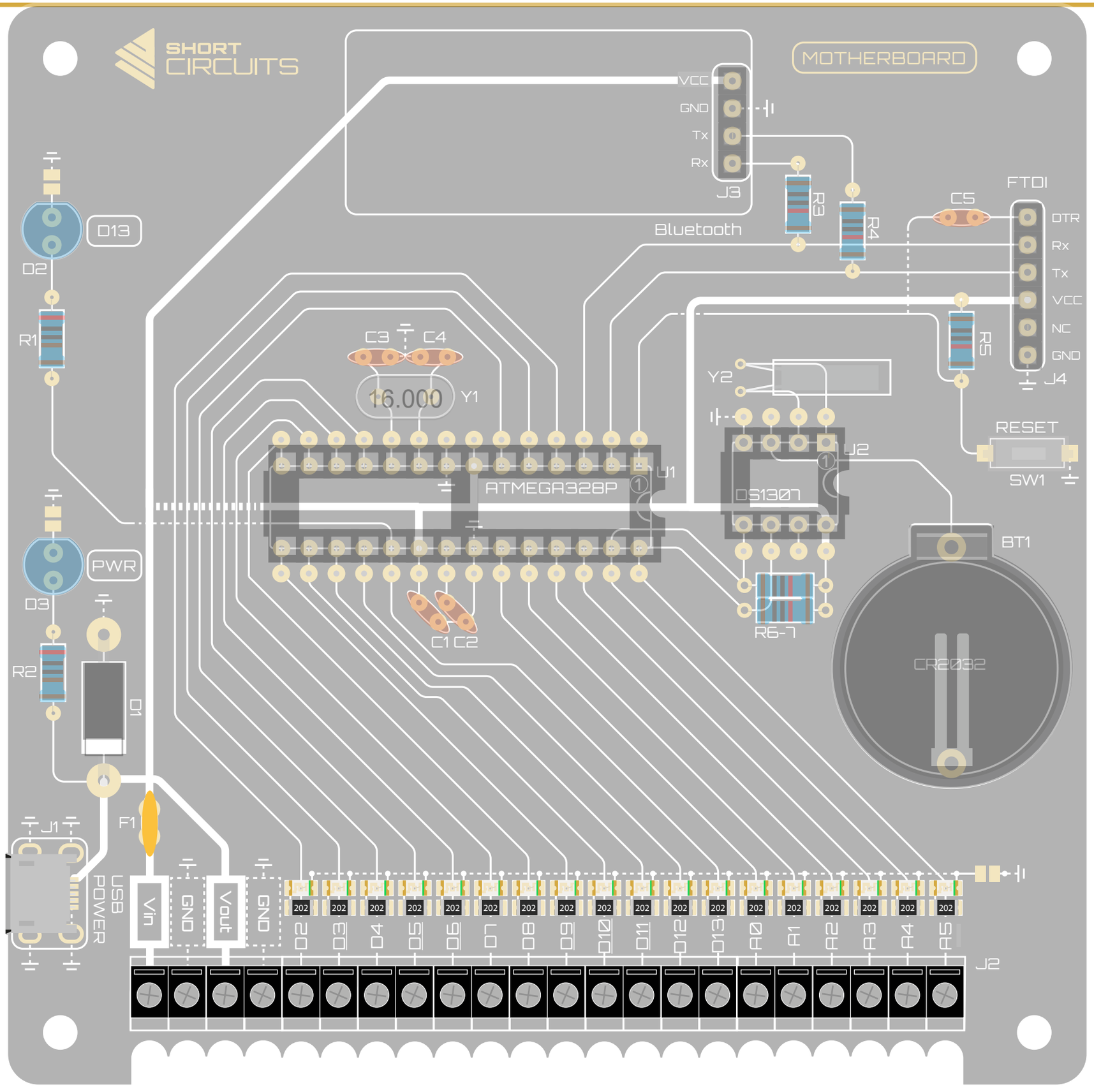
Now would be a good time to solder the optional surface mount IO LEDs. After you solder the terminal blocks they will be a little harder to get to. Add a little solder to a clean iron tip. Then apply the tip to one side of each of the LED and Resistor footprints. Now hold a surface mount LED in some tweezers (take note of the polarity), offering it up to the pad with solder on it. Apply some heat with the iron and the LED should attach itself. Make sure it is seated in the correct position. You may need to apply dabs of heat to get it in the right place. Don't hold the iron to the pad for too long. When in place, solder the other pad. This should be easy as the LED will be stuck in place. Hold the iron to the LED for too long and it will melt the other solder joint, so use quick bursts again.

Screw Terminals

We have supplied you with 11 x 2 position screw terminal blocks. To make the 22 position block, just connect them all together. If you had a project that only uses a few digital pins, you can just solder those blocks in and save the other blocks for another project. This means you have flexibility, and we only have to order 1 type of terminal block. Win! They can be a bit loose sometimes, so make sure you align them with the first and last pins, just like the headers.

Fuse

The fuse is not polarity sensitive. The resettable fuse, or PTC (or polyfuse), has bent leads to prevent you from inserting the component too far. It leaves an air gap between the PCB and the fuse itself. As the fuse is thermally sensitive, the gap helps ensure that it functions within its designed parameters. You would definitely get some kind of soldering badge by now if you were in the scouts, so we'll let you figure this last one out...


SHORT CIRCUITS
MOTHERBOARD
VCC
GND
Tx
Rx
J3
Bluetooth
R3
R4
C5
FTDI
DTR
Rx
Tx
VCC
NC
GND
J4
D13
D2
R1
C3
C4
16.000
Y1
Y2
R5
RESET
SW1
J2
U1
ATMEGA328P
DS1307
BT1
CR2032
PWR
D3
R2
D1
R6-7
C1
C2
J1
F1
USB POWER
Vin
GND
Vout
GND
202
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
A0
A1
A2
A3
A4
A5
J2

CIRCUIT

Testing for faults

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

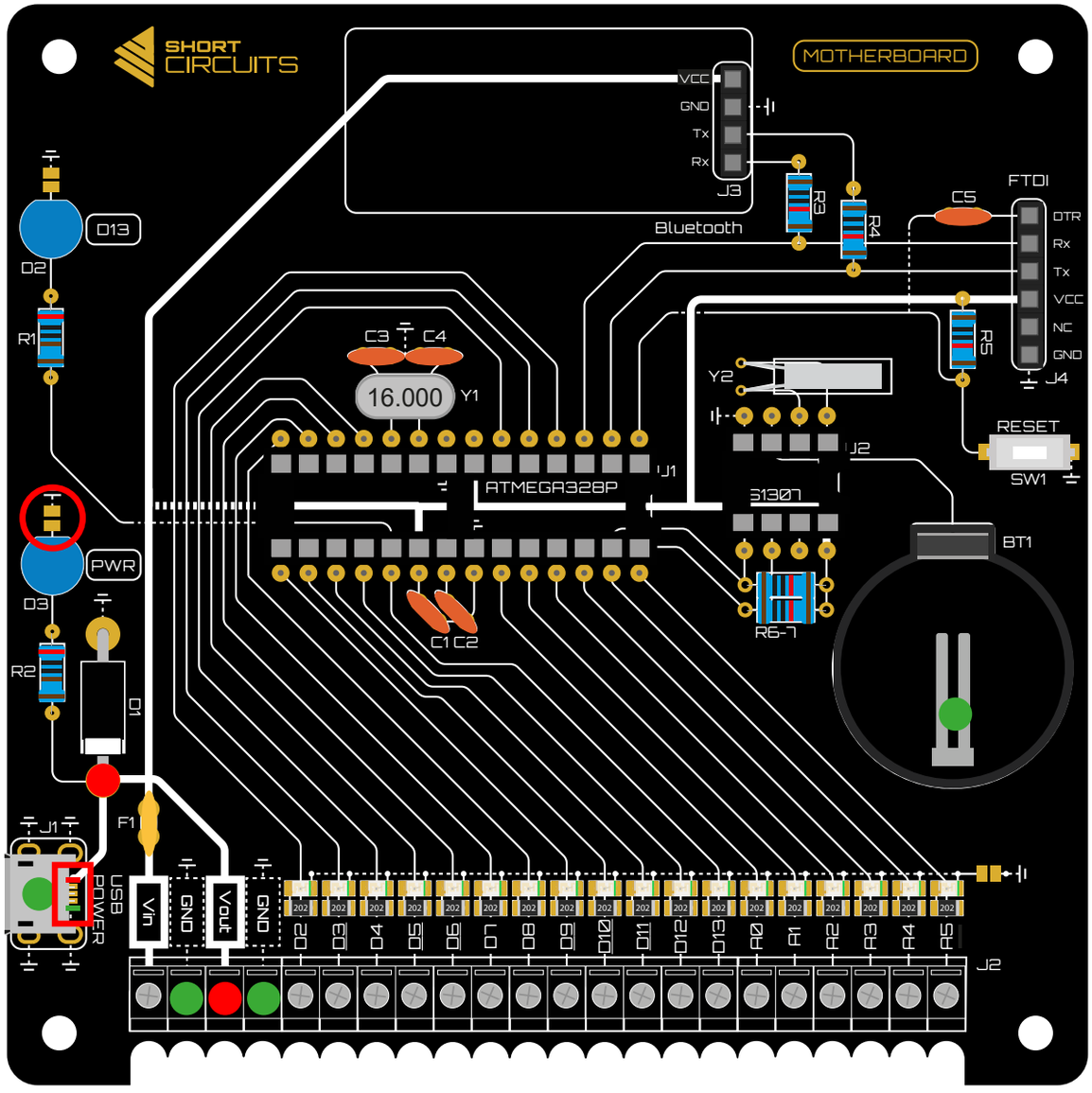
Visual Check

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.

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

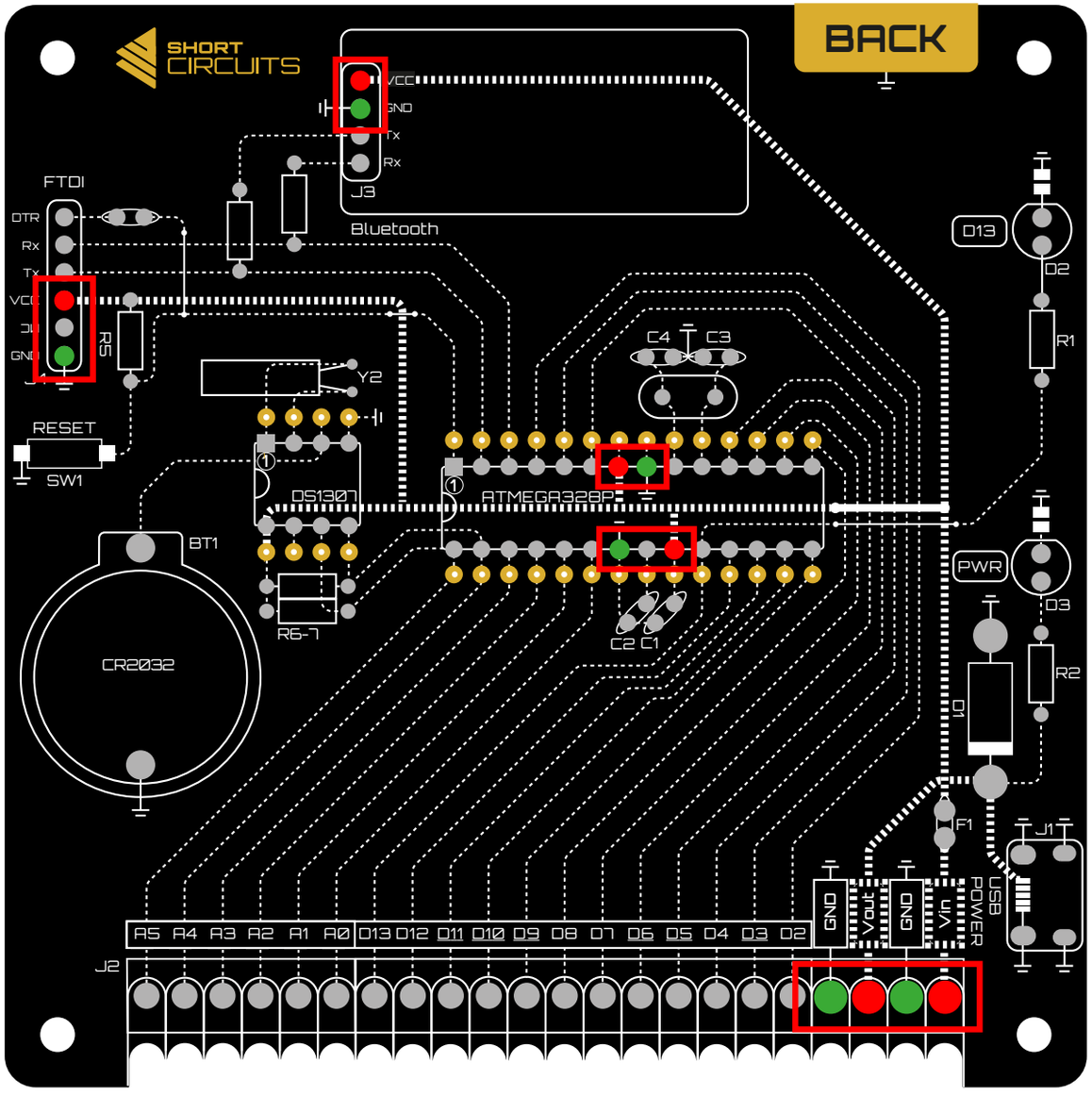
Now we need to check for short circuits (the bad kind). If we have a short circuit somewhere on the board and we plug a USB power cable in with no surge protection, you will destroy something. So to check for shorts, get your multimeter and put it in continuity mode. Here's the symbol: Make sure the 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. Note the case of the USB socket is a nice big target and further away from where your other probe will go. Keep it held there while you press the red probe onto the screw of the Vout terminal block (the metal screw is connected to ground). 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. If everything looks good then it may be the USB socket. Flip the board back over and check the pads on the socket. Clean it with more Isopropyl Alcohol then drag your iron over them from the back of the pads towards you. Repeat the continuity test.

Polarity

Check all the components that are polarity dependent. In this circuit, that would be the LED's and the Zener Diode. The Zener's cathode (white line) should be connected to Vcc. If this was the other way around, there would be very little resistance through the diode and would cause a short. If the components match the silkscreen under them, they should be good to go. Repeat the continuity test. If there is still a problem, go back and check everything again.



You should not proceed if there is a short between Vcc and GND.

Power Test

We can now plug a USB cable in to see if we can light the Power LED. Make sure the jumper pads are soldered together (circled in the diagram on previous page). Plug the USB cable into the socket and the other end into a 5V supply. The PWR LED should light up indicating the circuit is functioning. If not, first check power is getting to the board. Set your multimeter to DC Voltage: Put your red multimeter lead on Vcc (Vout terminal block screw or the marked end of the Zener Diode) and the black lead on GND (USB socket case or GND terminal block screw). The multimeter should read somewhere between 4.6V and 5V. If not, check the USB cable, if it works in another device it should be fine. Check a different cable, give it a wiggle. If that doesn't work, follow the path on the board, from the USB socket's Vcc line (the thick one) through R2, the LED and the Jumper Pads. Check everything that should be connected is, and everything that shouldn't isn't. Now check if any components are polarity dependent. If they are, check they are the right way round. The LED's flat edge should be pointing towards the Jumper Pads and GND.

Install Integrated Circuits

Unplug power before next step! Now, we can push the ICs into their sockets. Grab the ATMega328P and the DS1307 chips and place them carefully in their respective sockets. You may have to bend the legs inwards to make it fit. Do this on a flat surface so all the legs stay parallel. Make sure you pay close attention to the polarity. The semi-circles on one end of the chips need to correspond to the semi-circle on the silkscreen (the picture on the board) and the socket. Be careful, the socket could be installed the wrong way, you can always rely on the silkscreen though.

Power the circuit through the USB or 5V through the Vin and GND terminal blocks. We have uploaded a "blink" sketch onto the ATMega328P so, if you have soldered the jumpers near the D13 LED then the LED should start blinking. If it doesn't, hook up an FTDI to USB Serial adapter to the board. Double check the orientation of this, the pin labels should match. If they don't and you power the board, you could fry the FTDI adapter.

Upload a sketch using the Programming section of this manual and check if the LED blinks rapidly when you hit upload in the Arduino IDE. If it does, everything is working as it should. Follow the Programming sections tips to start coding and take control of your creation!

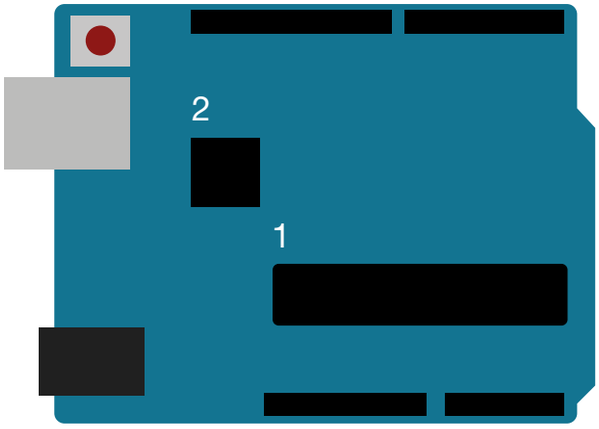
Still having problems? Head over to the forums on our website for help and support. www.shortcircuits.cc

Programming

The ATMega328P cannot communicate directly over USB. So when we go to program the chip, we need to use a go between. There are a few ways to achieve this. The easiest way would be to plug an FTDI module into the FTDI header and program the chip while it's sitting in the MOTHERBOARD.

NOTE: There are many FTDI modules out there, whether you use one from us or somewhere else, always make sure that the pin names match when connecting it to the board. If you power it on when it's connected incorrectly, you may fry the module completely.

Another option would be to remove the chip from the board and install it in the DIP28 socket on a compatible Arduino Uno (1). Arduino Uno's have an extra tiny chip (2) that translates the USB signal so the ATMega328P can understand it.

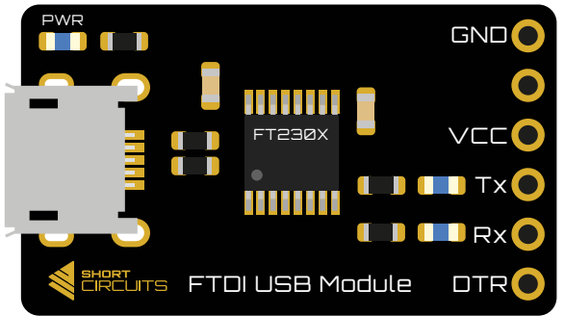


The most complicated option, but probably the most interesting, would be to use a Bluetooth module to program the chip wirelessly. This will be covered in a later guide.

To program the chip, using whichever method, you will need to download the Arduino IDE. This software compiles written C++ code into machine code that the ATMega328P can understand. Either visit the download page HERE and download the software for your operating system, or download it from the Windows app store if you are using Windows. Consider donating to the development of the Arduino IDE software, as their work has done amazing things for the electronics and maker communities, opening up so many possibilities. As they say "Open source is love!"

To get started without having to learn to code, check out the example sketches under File > Examples. As you add different libraries to the software (Through Tools > Manage Libraries...) You will see new examples relating to the libraries you download. Libraries are pre-written code written by lovely people that you can use in your projects. They are usually written with specific components in mind.

FTDI



To program the chip using an FTDI module:

Plug the module into the FTDI header on the board. WARNING: Double check the FTDI module is oriented correctly, with the pin names matching on both the MOTHERBOARD silkscreen and the FTDI module Silkscreen.

Plug a USB cable into the FTDI module and into a Computer.

Run the Arduino IDE and select Tools, then Port.

Your ATmega board should show up as "COM#" (the number after COM varies depending on what's connected etc.) Click to select it.

Then, under Boards, select "Arduino Uno", as the MOTHERBOARD is Arduino Uno compatible.

Now you can get programming! Read our quick coding guide for this board, download some of our example sketches from HERE, or find a tutorial online and follow it. There are plenty of Youtubers with great tutorials. Lots of written tutorials online as well. Check out Instructables, the Arduino website and others.

Arduino Compatible Pins

Pins / Terminals / IO

The ATMega328P has lots of different inputs and outputs that have different capabilities. As we are using the Arduino IDE (the easiest and best documented software for this), we will need to use the pin references they use when programming the MOTHERBOARD. Here are some of the different types of pins:

INT0	External Interrupt 0
INT1	External Interrupt 1

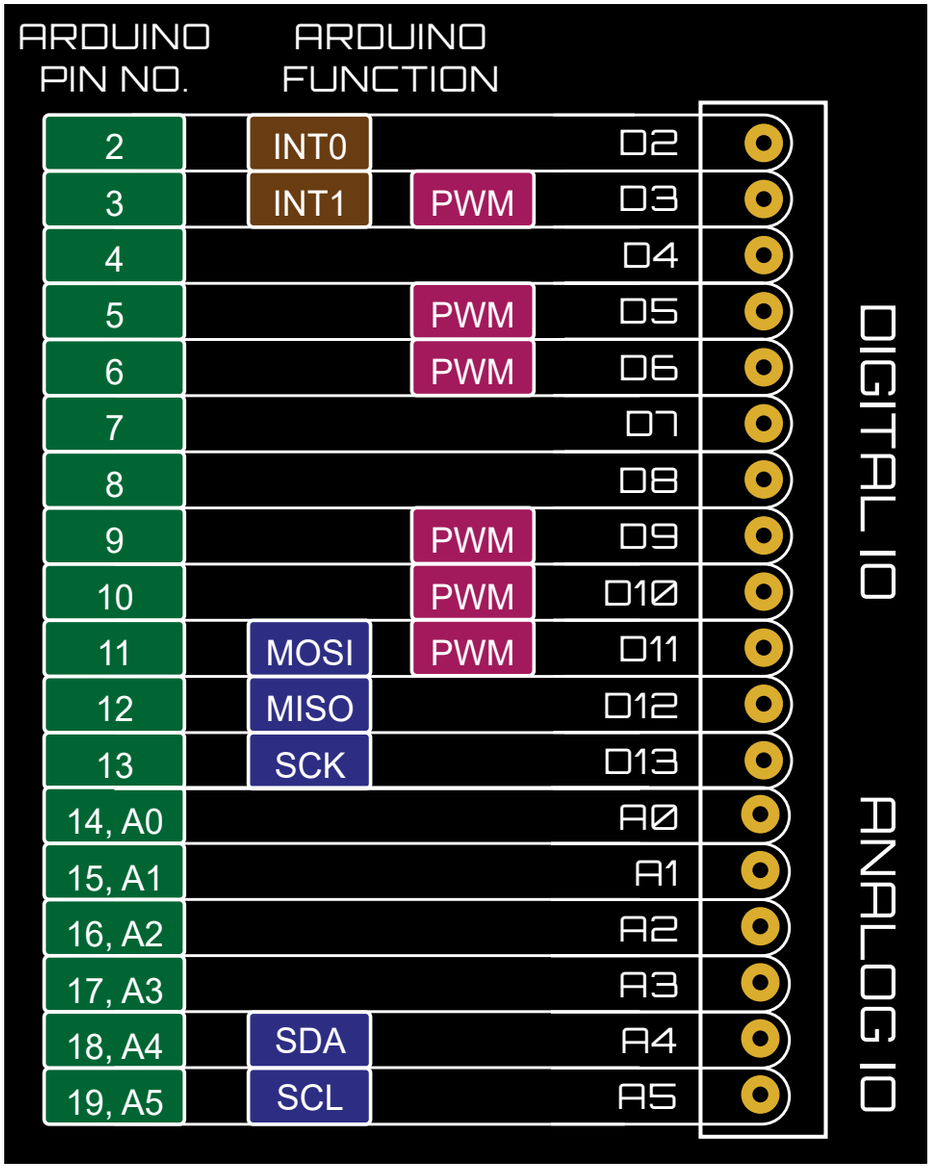
These are external interrupt pins. They allow the microcontroller to carry on with all of its tasks until something triggers one of these pins. Without them, the microcontroller would have to constantly check the pins for change, which would get in the way of other functions.

MOSI	Master Out, Slave In (SPI)
MISO	Master In, Slave Out (SPI)
SCK	Serial Clock (SPI)

These pins are used for the microcontroller's Serial Peripheral Interface. They are used to communicate between multiple devices. Unlike the Tx and Rx pins used for FTDI and BT modules, the SPI bus is synchronous. This means both Master and Slave are in perfect time with each other.

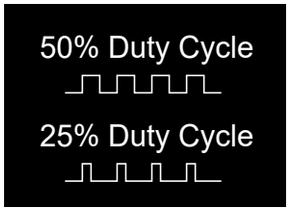
SDA	Serial Data (I2C)
SCL	Serial Clock (I2C)

I2C is the Inter-Intergrated Circuit protocol. It allows multiple peripherals to be connected to just 2 signal wires (1008 devices to be exact). I2C sends bits to indicate which peripheral it wants to talk to and whether it wants to send or receive data. This means it's a little slower than SPI. But it's faster than Tx Rx, and only uses 2 pins.



PWM	Pulse Width Modulation

Pulse Width Modulation varies how much time a signal is pulled high. This can be used to dim LEDs or to control servos with greater control. To dim an LED to 50% for example, you would send a PWM signal with a 50% duty cycle. Which means it is on 50% of the time, and off the other 50%. As this signal pulses at a very high rate, we see the LED dim rather than flicker.



Coding Basics - Blink

Lets get started with some code. This assumes you have no prior knowledge and starts with the basics. This sketch will flash the LED repeatedly.

The first part of your code will handle any preprocessor directives like:

- Defining pins with #define
- Including any libraries that your code will call upon with #include

Also, any global (accessible by any part of the code) variables that we will use to store data will be declared here.

For our “Blink” sketch, we will need to declare which pin the LED is connected to.

There are two ways of doing this and either will do nicely:

```
1  const int LED = 13;
2
1  #define LED 13
2
```

keyword that defines the variable as constant (read-only).

data type: Integer (signed number from -32768 to 32768).

Arduino pin number.

Used to end a statement.

A Variable named “LED” that stores the number of the pin that is connected to an LED.

preprocessor directive that basically swaps any reference to “LED” with the value 13 before the program starts.

The C coding language and the Arduino IDE use functions to organise code. A function holds code within it and can be initialised by other bits of code. 2 functions that Arduino use in almost all sketches are “setup” and “loop”. Setup runs once at the start of the sketch and is used to set up different things. Loop repeats indefinitely.

```
3  void setup()
4  {
5     pinMode(LED,OUTPUT);
6  }
7
```

indicates that the function setup will not return any information.

part of the Arduino structure. Executes once.

a predefined function that defines the mode of the pin.

function that puts the pin in a low impedance state.

we can use “LED” or “13” here as all instances of LED will be replaced with 13.

Variables

A value that represents something in your code. This could be a value that changes or is fixed.

Keywords

Reserved words in C programming that are part of the syntax/language.

Functions

Predefined blocks of code. Data is passed into a function to perform certain actions. Good for repeating the same action again and again.

Data Types

There are many data types in C++. Each type holds a different range of numbers. See the table on the next page for a list of data types commonly used while programming 8-bit microcontrollers with the Arduino IDE

Preprocessor Directive

Dealt with before the program starts

Structure

The elements used as part of the Arduino (C++) code.

In this case, we need to tell the ATMega328P that D13 will act as an output. This will allow more current to be supplied to it, as it will be in a low impedance state (low resistance). We use setup to do this as it only has to happen once at the beginning.

To blink the LED, we will need to turn it on, then off, then repeat the process indefinitely. We can use loop to achieve this.

The positive lead of our LED is connected to the pin, so pulling the pin to 5V will turn it on, pulling it to 0V will turn it off.

The function digitalWrite() asks for a pin reference and either HIGH or LOW.

Without a delay between turning the LED on and off, and off and on, we would not be able to see anything happen. As the loop will last just a few milliseconds.

delay() asks for a number, in milliseconds that tells it how long to delay all other processes. The downside is that it will stop everything, so if you are trying to multiplex the display on the DIGITISER kit for example, you will need to use an alternative.

```
 8
 9  void loop()
10  {
11     digitalWrite(LED,HIGH);
12     delay(1000);
13     digitalWrite(LED,LOW);
14     delay(1000);
15  }
16
```

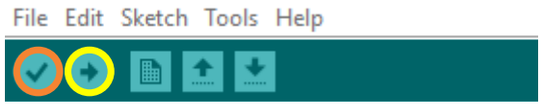
part of the Arduino structure, a function that repeats indefinitely.

a function that pulls a pin LOW or HIGH.

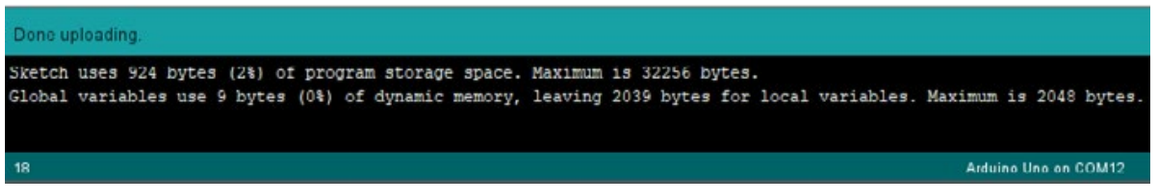
This stops all processes for 1000ms.

LOW represents pulling a pin to GND (0V).

Curly brackets are used to contain any statements within a function, loop or conditional statement. Always close what has been opened!



Now hit the compile button to check to see if you have anything wrong that the software can identify. The software will prompt you to save the sketch. If everything is good, and no warnings are shown in the console at the bottom of the window, hit upload! The Tx/Rx lights on the FTDI module should flicker, as well as the D13 LED. If they don't and your code does not upload, check the troubleshooting section.



Data Types:

(Using ATMega 8bit chips)

array
(collection of variables)

bool
(true / false)

byte
(0-255)

char
(stores ASCII characters)

float
(numbers with decimals
-3.4028235E+38 to 3.4028235E+38)

int
(-32,768 to 32,767)

long
(-2,147,483,648 to 2,147,483,647)

short
(-32,768 to 32,767)

string / String()
(text strings)

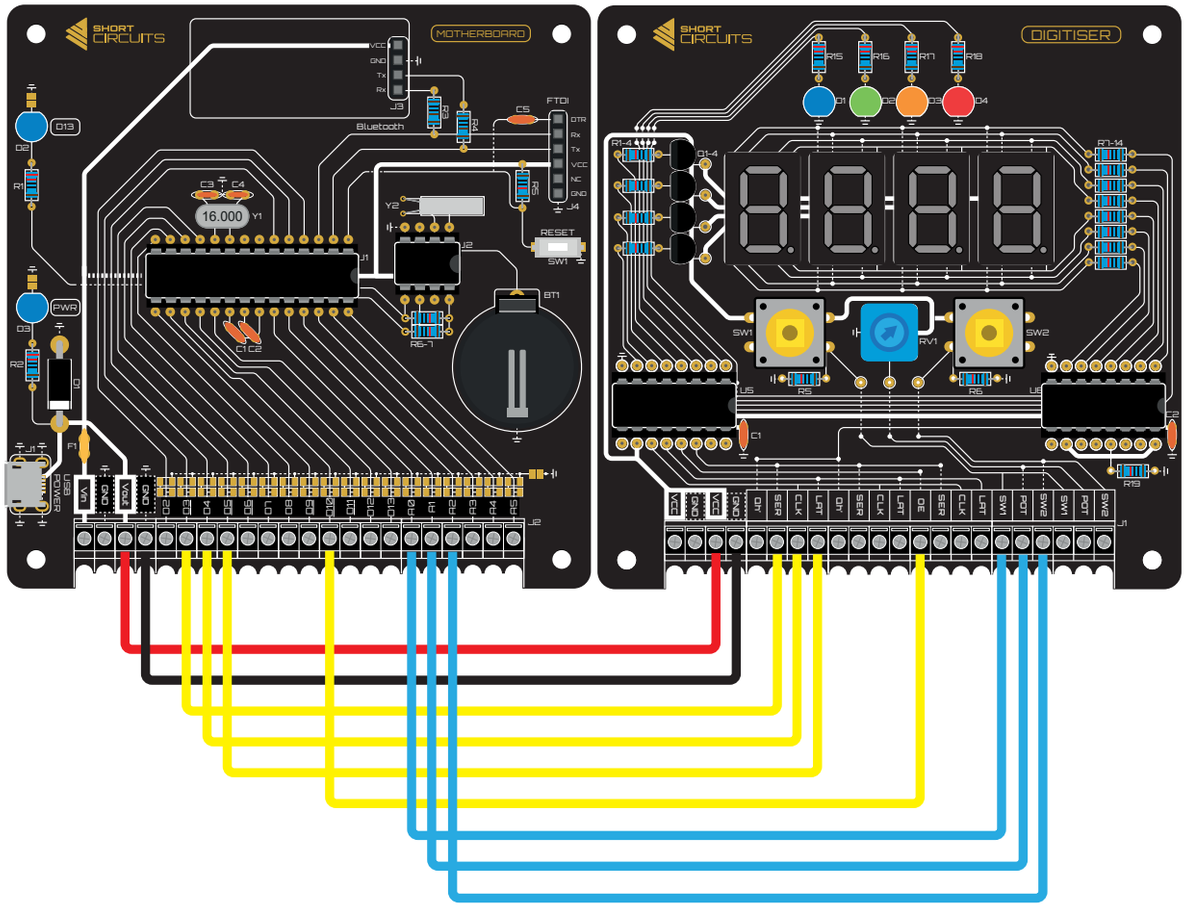
unsigned char
(0-255)

unsigned int
(0-65,535)

unsigned long
(0-4,294,967,295)

word
(0-65,535)

Project Ideas - Clock



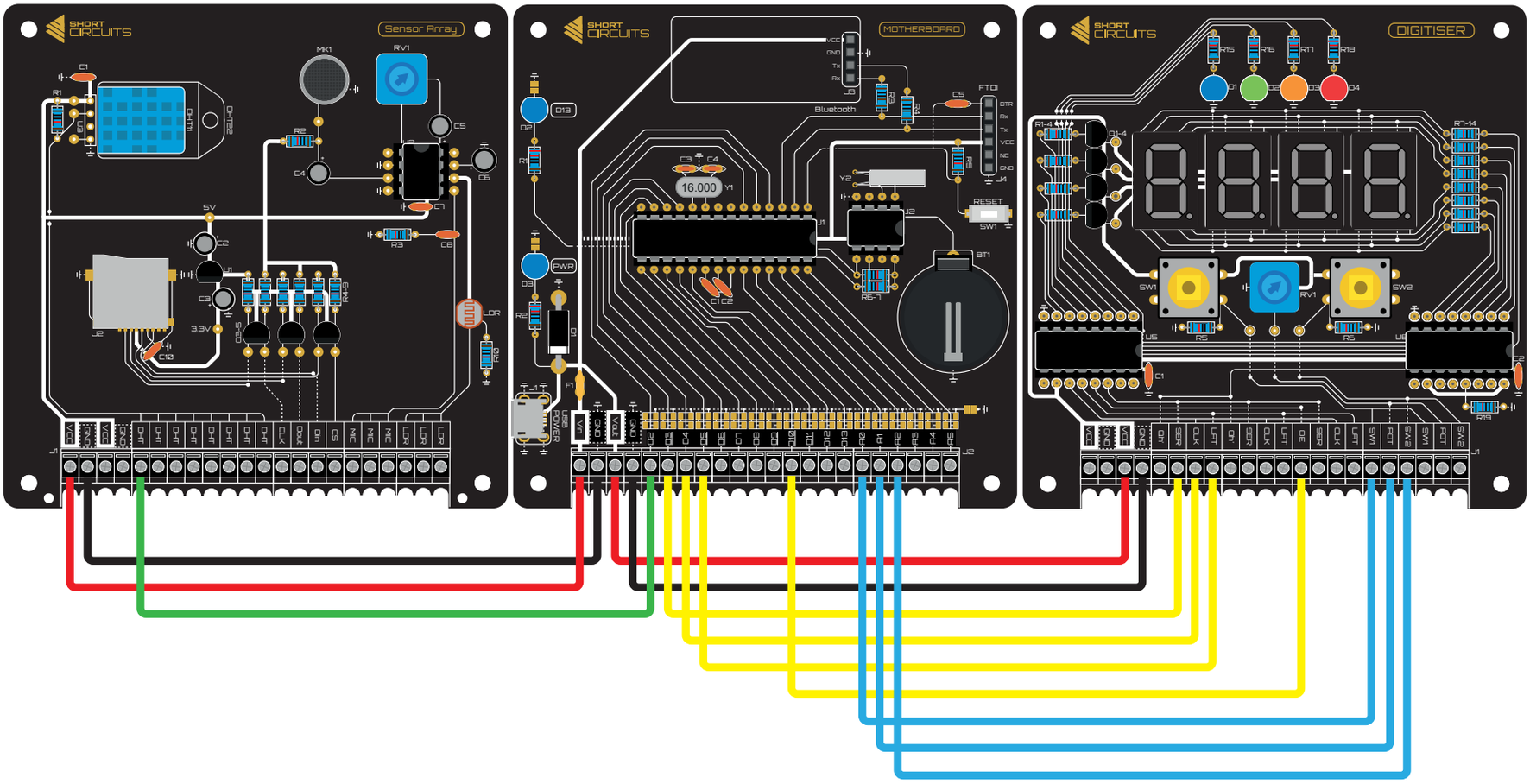
To build a digital clock capable of displaying the time, date and year, you will need the MOTHERBOARD and the DIGITISER kits.

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 website and download the DigiClock sketch and upload it using the Arduino IDE and the instructions in the Programming section of this manual.

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

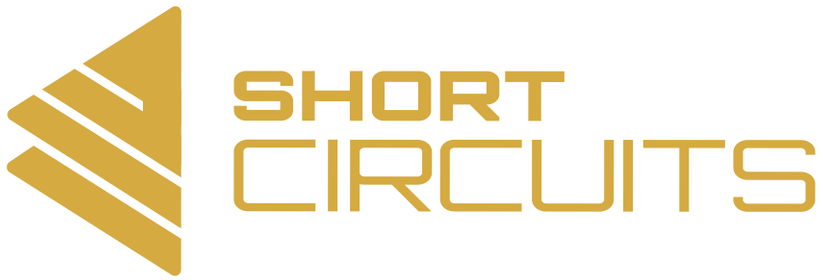


Adding the SENSOR ARRAY to the previous build enables you to add temperature and relative humidity to the digital display. You can add sound levels and light levels too, but we're sticking with a simple clock and environment sensor display in this project.

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.
Either write your own code from the hints in each kit's manual, or head over to the website 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.



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.