

Overview

Today you'll have an encounter with Maker culture, and be given a bunch of tools to experiment with physical computing. Where you take it from here, is up to you!

There will be presentations of the creative and often crazy **interactive objects** which people are building using microcontrollers. You'll find out how to recreate others' projects, and personalise the construction or the behaviour according to your own tastes and ideas.

You'll learn **programming basics**, and how code helps people to understand the behaviour of computers. In particular you'll learn about *values*, *types*, *lists*, *steps*, and *names*. You'll encounter *named values*, known as *variables*, and *named steps*, known as *functions*

You'll experiment with the *tools and skills* for **prototyping electronics** by creating your own minimal #Shrimp Arduino-compatible computer to take home, and creating a neat visual illusion as an example of what it can do.

Planned Activities

First of all, we will build a minimal #Shrimp project board. To prove that the circuit is correct, we'll add an LED and connect it through a resistor to Ground to create a **Blink** circuit. Then we can upload the **Blink** program to the microcontroller, which will tell it to flash the LED. We'll use this program as an example to learn programming.

Next, we'll create a **Persistence of Vision** circuit, by removing the resistor and LED and adding 8 LEDs in a row, connected back to ground with a wire. After uploading the ShrimPOV sketch we should be able to see letters painted in the air.

If we have time, we'll design our own pixelated **bitmap images**, on graph paper, calculate the binary numbers which can be used to represent them, then edit the ShrimPOV font in the program code, replacing some characters we don't need with our own images. When this modified code is uploaded, we should see our own images appearing.

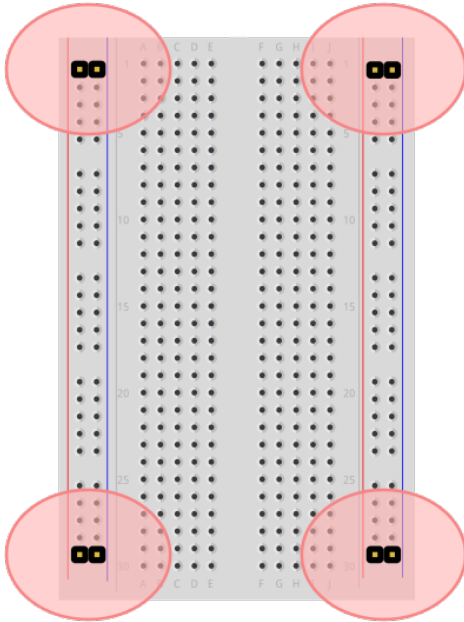
Getting started

First of all, check the components in your bag, you should have at least the following...

- 1x Paper overlay
- 1x ATMEGA328 Microcontroller
- 1x CP2102 USB to UART converter
- 1x Rainbow jumper cable
- 1x Solderless breadboard
- 1x 3AAA battery pack with switch
- 3x AAA 1.5Volt Alkaline Battery
- 1x 100 nanoFarad capacitor (marked 104)
- 1x 10 KiloOhm resistor (with Brown, Black and Orange stripes)
- 1x 100 Ohm resistor (with Brown, Black and Brown stripes)
- 1x 16MHz crystal
- 3x Green wire
- 2x Red wire
- 1x 9-pin header strip
- 4x 2-pin header strip

To construct your #Shrimp, just follow the detailed steps in the interactive workshop.

For some workshops with limited time, some of the initial stages will have already been completed for you to save time, such as *attaching the overlay*, or *inserting the ATMEGA chip*.



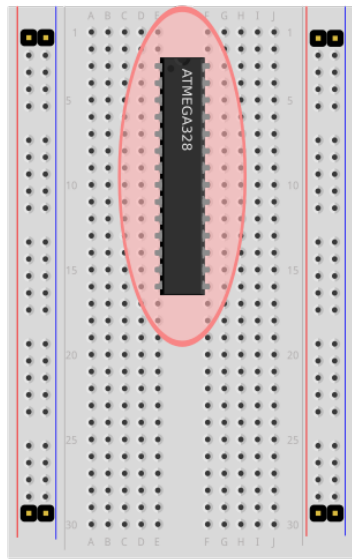
Attach overlay (2-pin headers)

First, we attach a paper overlay to the 400 tie-point solderless breadboard. We will use this as a guide to put the components in the right places by poking them through the indicated holes.

To follow along with the presentation material, the overlay should be positioned with the ATMEGA pin number 1 at the top left corner (e.g. the chip showing a half-moon shape at the top). As a double check of the overlay position, the 9 pin header with colored wires should be at the top left of the circuit.

The paper overlay shows four pairs of “header strip” pins in the corners. These have no effect on the wiring of the circuit (they don’t connect anywhere), but they will hold the overlay in place.

Carefully align a 2-pin header strip with each corner of the diagram. Push them through the paper into the two “power rails” which run down the sides of the breadboard. They will hold the paper overlay in place.



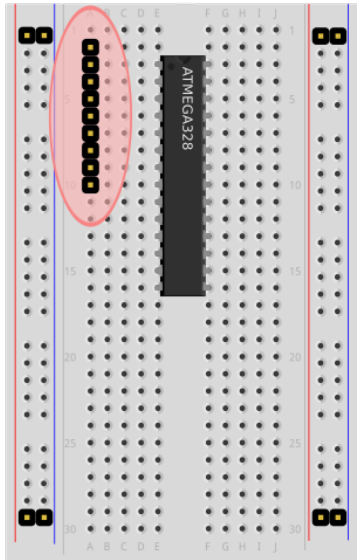
Insert the ATMEGA chip

Now is the time to add the ATMEGA microcontroller if it's not already in the breadboard. This is a black oblong with numbers printed on it, and 28 silver legs, looking a bit like an insect. It is the computer at the heart of a #Shrimp, with the ability to control inputs and outputs - sensing or triggering things out there in the world.

Check that there are holes in the paper overlay corresponding with the little silver legs, 14 holes on each side of the chip for a total of 28 holes. Punch new holes with a pin if they are not yet there.

Check that the microcontroller's legs don't splay out too much as this will make it hard to insert, or could possibly break the legs if you force them. If the legs are not at right angles to the chip, ease them into position by gently pressing one side of the chip against the table top (14 pins at a time). Measure their position by balancing the chip on the paper overlay, where the legs should line up with the picture.

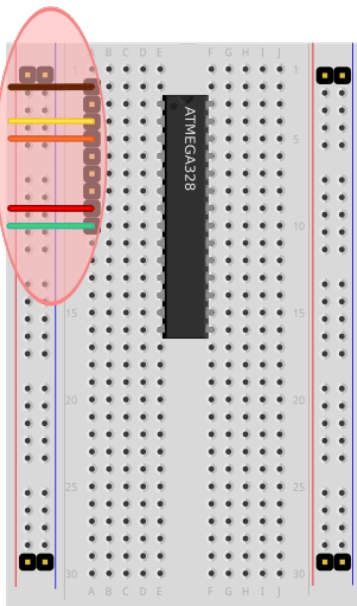
Carefully align the chip, checking the half-moon shape is at the top, with two empty breadboard rows above the chip. Check the legs pass cleanly through where they are drawn on the paper, into the breadboard below. Once the legs are all aligned. press down until the chip slides fully into the board (about 3mm movement).



Add the 9-pin power and programming header

A series of copper pins will be used to program and provide power to the #Shrimp. Find the line of 9 pins in black plastic, shown on the paper overlay with colored wires going to them. Not all of the 9 pins be used in this circuit, but putting all 9 in the right place will help you position everything else.

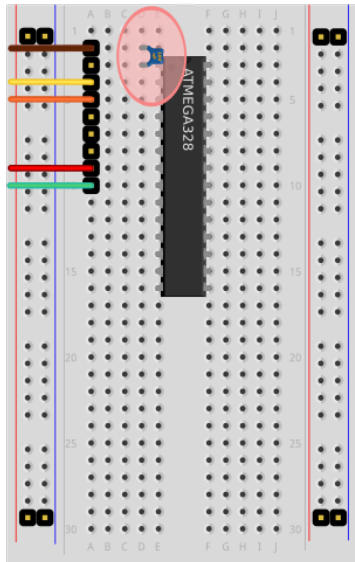
Push the 9-pin header strip into the top left corner of the board, leaving just one empty row at the top of the board



Attaching the Rainbow jumper cable

Your CP2102 module should have been provided with a 5-pin rainbow-colored cable, with Red, Orange, Yellow, Green and Brown wires.

Leaving the wires attached to each other in a strip, push the ends of the jumper cable onto the 9-pin header according to the colors in the diagram.



Add the 100 nanoFarad 'decoupling' capacitor

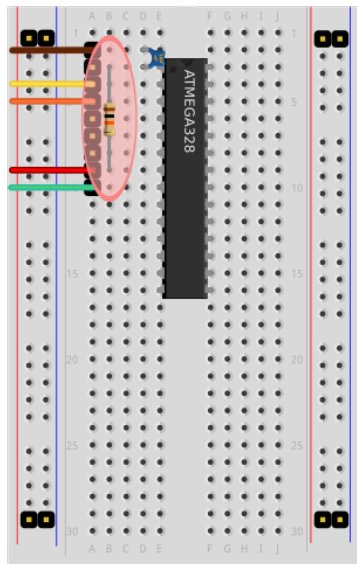
Look for the 100 nanoFarad capacitor, which is a small disc with two thin wires coming out of it. Our '104' capacitors are ceramic capacitors so it doesn't matter which way round they are inserted, each leg is the same as the other.

As a 'decoupling' capacitor, it will smooth out any random electrical spikes which appear on Pin 1, and ensures that signals sent to it are stable and reliably detected. Pin 1 accepts a 'reboot' signal for your computer. This smoothing component helps guarantee the chip will reboot only when you request it (e.g. when you're reprogramming the microcontroller) and not at other random times.

The digits 104 indicate its *capacitance* using scientific notation. It means the number of picoFarads would start '10' and then continue with a further 4 zeroes, so it would be written as 100,000 picoFarads. Since 1 nanoFarad is 1000 picoFarads, there's 100 nanoFarads in a capacitor marked '104'.

Insert the capacitor marked '104' between

- ***the header pin with the Brown wire attached***
- ***the top left leg of the chip (pin 1, immediately anticlockwise from the half-moon shape)***



Add the 10 kiloOhm ‘pull-up’ resistor

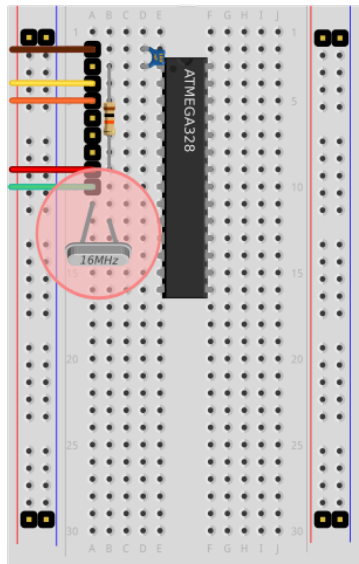
Look for a brown or blue cylinder with a wire at each end, showing colored stripes starting Brown, Black, Orange. Resistors have no orientation, and can be attached either way round.

This component is used as a *pull-up resistor* and is attached to the positive wire of the power supply, approximately 5 Volts. It therefore ‘pulls up’ the reboot pin (Pin 1) to have a positive voltage, unless the reboot pin is connected to Ground, approximately zero Volts. When reprogramming the microcontroller, the brown wire is connected to ground, and causes the chip to reboot. Before it starts running again, the new program is sent over the orange wire.

The first three colored stripes on a resistor represent the resistance of the component in just the same way as the capacitor we described above. The only difference is that colors are used instead of digits. Decoding the colors Brown=1, Black=0, Orange=3 gives us the number 103. That means the resistance in Ohms starts ‘10’ and continues with a further 3 zeroes, making it 10,000 Ohms. Since one kiloOhm is 1000 Ohms, there’s 10 kiloOhms in a resistor marked Brown, Black, Orange.

Attach the resistor with Brown, Black and Orange stripes between

- ***the header pin with the Red wire attached (positive power line, approx. 5 Volts)***
- ***the top left leg of the chip (pin 1, immediately anticlockwise from the half-moon shape)***



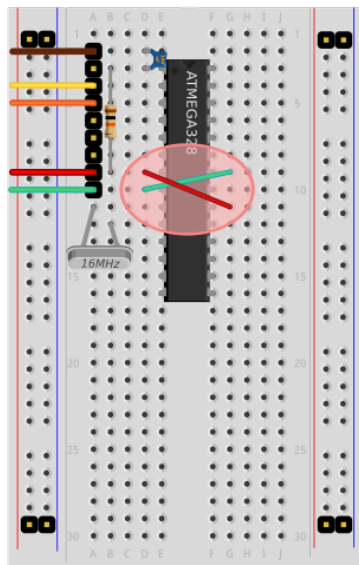
Add the 16 MHz Crystal

Look for a silver box with rounded ends, and two wires, marked 16.000.

You can think of a computer as something a bit like clockwork. In fact the first digital computer in the world was built using clock-making techniques, and had cogs representing numbers.

The 16.000 mark indicates the number of back-and-forth movements this crystal generates per second, in megaHertz. One Hertz means once per second, and one MegaHertz means one million times per second.

Insert the crystal with one leg in the row immediately below the Green wire, and the other leg in the row below that

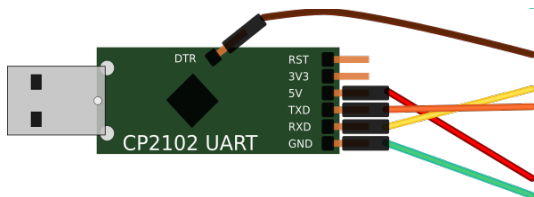


Add Power and Ground wires crossing the chip

Look for one Red and one Green wire, stripped to show silver at each end.

The ATMEGA chip is broken up internally into separate parts, each of which expects to be provided with a stable power supply. A power and ground connection will be provided to the left half of the chip by the red and green wires attached to the 9-pin header. Two additional wires are needed to connect the power supply to the correct legs on the right-hand half of the microcontroller.

Connect a Red wire from the existing Red wire, across the chip and down two rows. Connect a Green wire from the existing Green wire, across the chip and up one row.



Attach the CP2102 USB to UART module

Look for a green or blue-coloured circuit board with a USB connector at one end and 6 pins at the other end.

This device enables your desktop or laptop machine to communicate with the #Shrimp, for example to send new programs to it with the free Arduino IDE, or to exchange other information with a program when it's running on the #Shrimp.

Attach the other end of the rainbow wires to the labelled pins on the CP2102 as follows...

- ***Red -> 5V***
- ***Orange -> TXD***
- ***Yellow -> RXD***
- ***Green -> GND***
- ***Brown -> DTR (labelled on the back)***

You've made your own Computer!

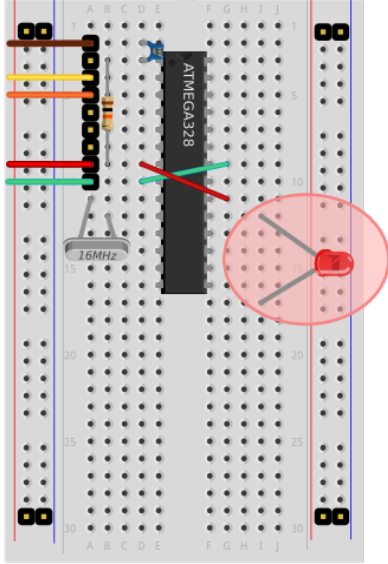
You've now finished making a Minimal #Shrimp circuit, which can act as the digital controller for many experimental projects.

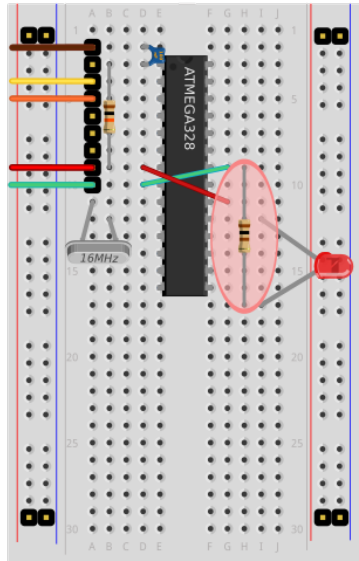
A more complete Protected #Shrimp circuit is described at <http://shrimping.it>. It has additional protective components, handy if you plan to use all the features of the ATMEGA microcontroller, or if you need to use it in an electrically noisy environment. A Protected #Shrimp can substitute for an Arduino Uno hobby board in almost any project.

You should be supplied with the additional components to turn a Minimal #Shrimp into a Protected #Shrimp in a separate bag at your workshop, for later experimentation.

Create the 'Blink' Circuit

Now we will construct a test circuit by adding an LED and a resistor, and uploading a test program onto the #Shrimp which causes the LED to flash.

	<h3>Add a Light Emitting Diode</h3> <p>Look for a red or clear dome with two wire legs coming out the bottom.</p> <p>A <i>diode</i> is a component which only allows electrical current to flow in one direction. One leg is longer than the other leg to indicate which way round the diode needs to be wired. Electricity should flow into the long leg and out of the short leg. Round LEDs also have one slightly flatter side, which corresponds with the short, negative leg of the LED.</p> <p><i>Insert an LED, inserting the long leg to the row <u>below</u> the Red line on the right hand side of the chip (power, or 5Volts) and the short leg in the first empty row below the microcontroller.</i></p>
---	--



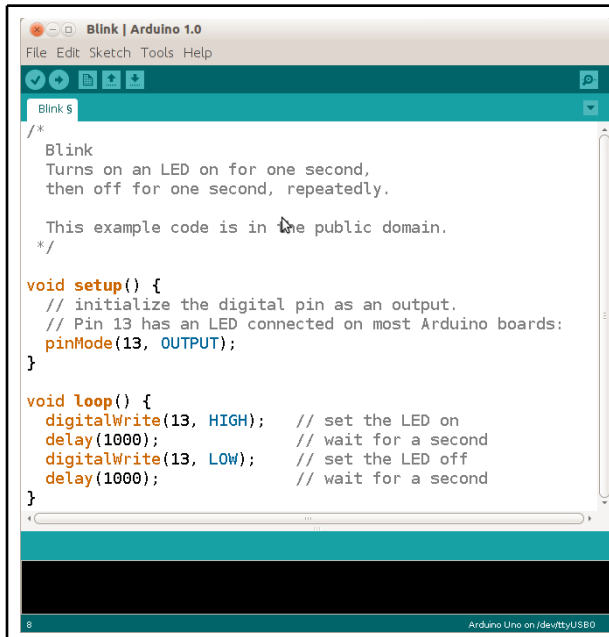
Add a 'current-limiting' resistor

Look for a brown or blue cylinder with a wire at each end, showing coloured stripes starting Brown, Black, Brown. Resistors have no orientation, and can be attached either way round.

The amount of electricity flowing through an LED needs to be controlled to prevent it from overheating and destroying itself. Our circuit is running at approximately 5 Volts, and LEDs are typically rated for just over 2 Volts. After a short while, applying 5 Volts would cause our LED to fail. We therefore need to add a resistor in series to prevent all 5 Volts from being applied to the LED. Some of the voltage will be used up by the resistor, and only a share of the Voltage is applied across the LED.

For an explanation of the coloured stripes, see the earlier section describing the pull-up resistor.

Insert the resistor between the short leg of the LED, and the row containing the Green wire on the right hand side of the chip (ground or 0Volts)

A screenshot of the Arduino IDE interface. The window title is "Blink | Arduino 1.0". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar shows icons for opening files, saving, and uploading. The main text area contains the following code:

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

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // set the LED off
  delay(1000);           // wait for a second
}
```

The status bar at the bottom indicates "8" and "Arduino Uno on /dev/tty/USB0".

Upload the 'Blink' program

To upload a new program, you need to have the CP2102 drivers installed in your machine. You can download installers for Windows and Mac at <http://shrimping.it/drivers/> Modern Linux machines should automatically connect to the CP2102 device as the drivers are built-in to the operating system.

Load 'Blink' which is in the menu under File -> Examples -> Basics -> Blink
Check there is a tick next to the correct entry in Tools->Serial Ports
Check that there is a dot next to Arduino Uno in Tools->Board
Click on the Horizontal Arrow in the toolbar to upload the program.

Change the 'Blink' code

You can change bits of your code to prove that your machine is able to send new behaviours to your microcontroller.

Why not change the number of milliseconds in the delay request so that the light blinks much more quickly, much more slowly spends longer on for a long blink, or spends longer off for a short blink.