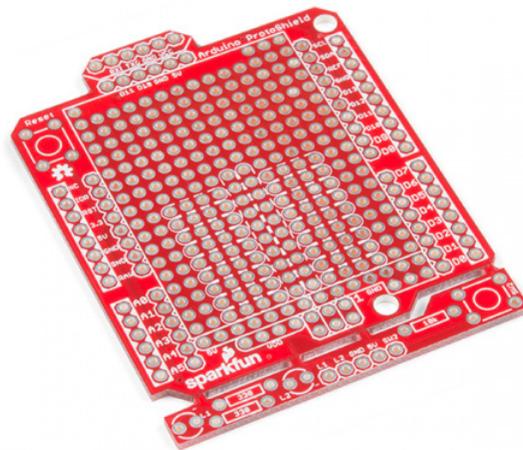


SparkFun Arduino ProtoShield Hookup Guide

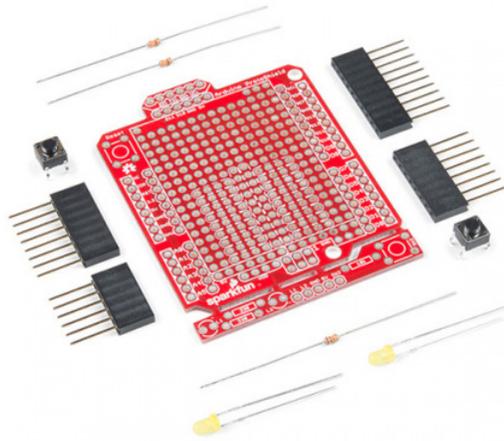
Introduction

👤 Heads Up! This tutorial is for the latest version of the SparkFun Arduino ProtoShield. If you have an older version, please consult the retired Arduino ProtoShield Quickstart Guide

The SparkFun Arduino ProtoShield PCB and ProtoShield kit lets you customize your own Arduino shield using whatever custom circuit you can come up with! This tutorial will go over its features, hardware assembly, and how to use shield.



SparkFun Arduino ProtoShield - Bare PCB
O DEV-13819



SparkFun ProtoShield Kit

DEV-13820

Suggested Materials

To follow along with this project tutorial, you will need the following materials. Depending on what you have, you may not need everything listed here. Add it to your cart, read through the guide, and adjust the cart as necessary.

SparkFun Arduino ProtoShield Hookup Guide SparkFun Wish List



SparkFun ProtoShield Kit
DEV-13820

Riding on the waves of the very popular Arduino USB board is this small prototyping shield. Originally spawned wit...



Breadboard - Mini Modular (Red)
PRT-12044

This red Mini Breadboard is a great way to prototype your small projects! With 170 tie points there's just enough ro...



Breadboard - Mini Modular (Green)
PRT-12046

This green Mini Breadboard is a great way to prototype your small projects! With 170 tie points there's just enough ...



Breadboard - Mini Modular (Blue)
PRT-12045

This blue Mini Breadboard is a great way to prototype your small projects! With 170 tie points there's just enough r...



Breadboard - Mini Modular (Black)
PRT-12047

This black Mini Breadboard is a great way to prototype your small projects! With 170 tie points there's just enough r...



SparkFun RedBoard - Programmed with Arduino
DEV-13975

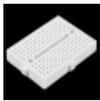
At SparkFun we use many Arduinos and we're always looking for the simplest, most stable one. Each board is a bit...



Stackable Header - 3 Pin (Female, 0.1")

PRT-13875

This is a simple 3 pin female PTH header. This header is in a configuration that makes it ideal to be used with the [...]



Breadboard - Mini Modular (White)

PRT-12043

This white Mini Breadboard is a great way to prototype your small projects! With 170 tie points there's just enough r...



Female Headers

PRT-00115

Single row of 40-holes, female header. Can be cut to size with a pair of wire-cutters. Standard .1" spacing. We use ...



Hook-Up Wire - Assortment (Solid Core, 22 AWG)

PRT-11367

An assortment of colored wires: you know it's a beautiful thing. Six different colors of solid core wire in a cardboard ...



(2) SparkFun Bluetooth Modem - BlueSMiRF Silver

WRL-12577

The BlueSMiRF Silver is the latest Bluetooth wireless serial cable replacement from SparkFun Electronics! This ver...



Break Away Male Headers - Right Angle

PRT-00553

A row of right angle male headers - break to fit. 40 pins that can be cut to any size. Used with custom PCBs or gen...



SparkFun USB Mini-B Cable - 6 Foot

CAB-11301

This is a USB 2.0 type A to Mini-B 5-pin cable. You know, the mini-B connector that usually comes with USB Hubs, ...



Jumper Wires Standard 7" M/M - 30 AWG (30 Pack)

PRT-11026

If you need to knock up a quick prototype there's nothing like having a pile of jumper wires to speed things up, and ...

Tools

You will need a soldering iron, solder, general soldering accessories, and the tools listed below for prototyping.



Solder Lead Free - 100-gram Spool

● TOL-09325



Needle Nose Pliers

○ TOL-08793



Diagonal Cutters

🕒 TOL-08794



Weller WLC100 Soldering Station

🕒 TOL-14228



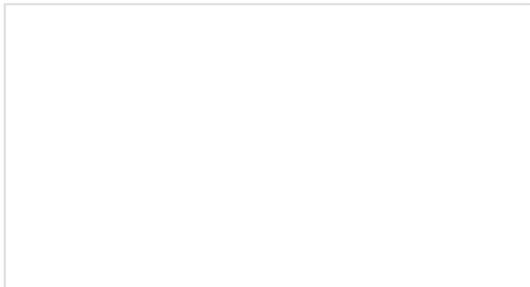
Wire Strippers - 30AWG (Hakko)

🕒 TOL-12630

Retired

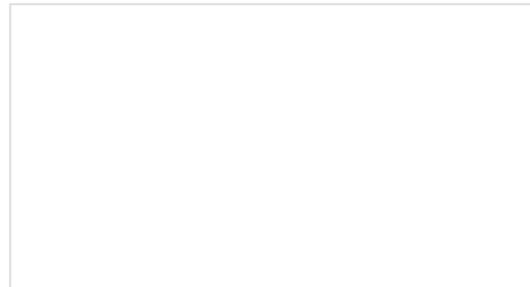
Suggested Reading

If you aren't familiar with the following concepts, we recommend checking out these tutorials before continuing.



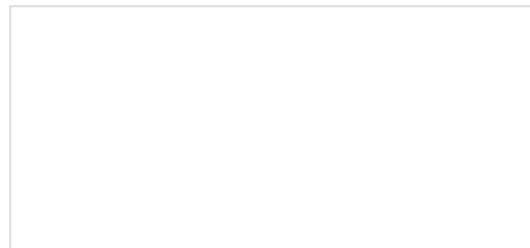
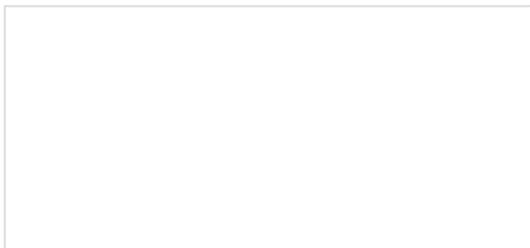
How to Solder: Through-Hole Soldering

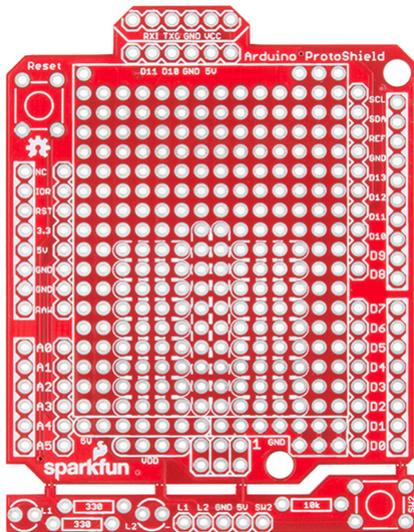
This tutorial covers everything you need to know about through-hole soldering.



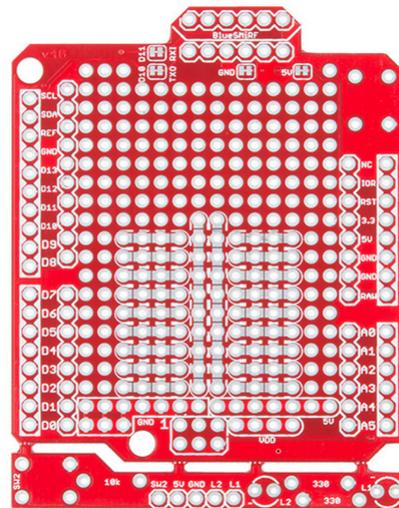
What is an Arduino?

What is this 'Arduino' thing anyway? This tutorial dives into what an Arduino is and along with Arduino projects and widgets.





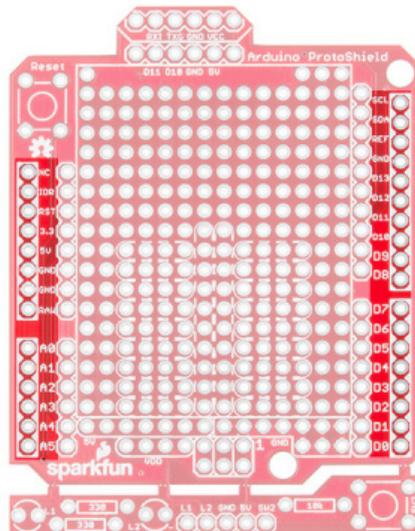
Top View



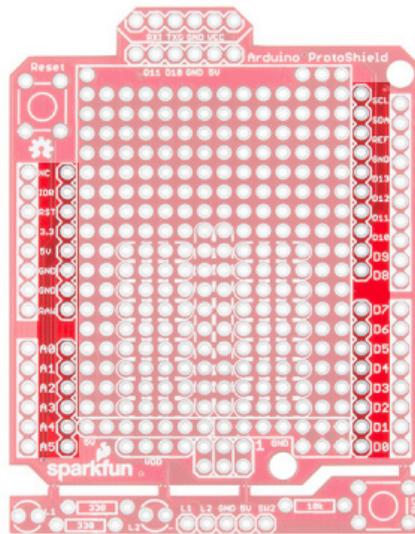
Bottom View

Stackable Headers for Arduino Uno R3 Footprint

The Arduino ProtosShield is based off the Arduino R3's footprint. Headers can be installed on the pins located closest to the edge of the board. You will notice that the location of the headers are highlighted in a rectangular silkscreen. For those using the stackable headers included in the kit, make sure to insert the header from the top side and solder from the bottom.

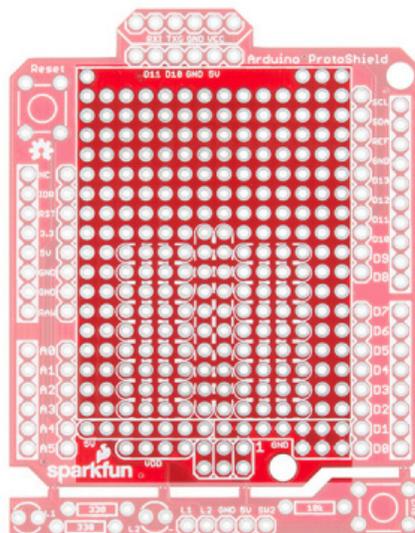


These pins are also broken out on the other side of its labeling. You will notice that the 1x10 header on the upper right side of the board is slightly offset from the Arduino R3 footprint. Don't worry, this was intentional so that you can place the board on a standard breadboard!



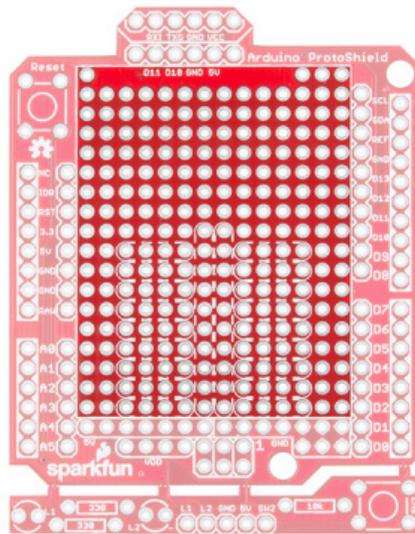
Prototyping Area

Next up is the sea of plated through holes. Look at all that great space for prototyping projects!



Solderless Mini-Breadboard

Do you have a small circuit on a mini-breadboard connected to your RedBoard or Arduino Uno? The ProtoShield has a spot to place a mini-breadboard on top to keep everything together. Peel the adhesive off the mini-breadboard, align it to the silkscreen, and stack it on!



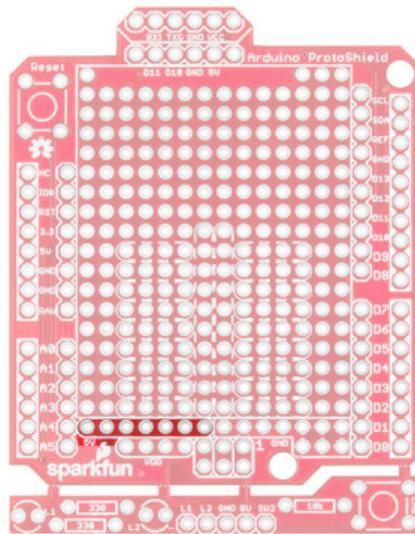
Solderable-Like Breadboard

Once you are done prototyping your circuit on the mini-breadboard, there is an option to solder the circuit directly to the board for a more secure connection. The bottom half of this area was designed with a breadboard in mind. You will not notice too much on the top side. Flip the board over and you will see open jumper pads between each through hole to make a connection like a breadboard. Once you add a component, simply add a solder jumper between holes to make a connection. For those that prefer the standard prototyping pads, we left the other side as is.

<p><i>Solderable-Like Breadboard (Top View)</i></p>	<p><i>Solderable-Like Breadboard w/ Open Jumpers (Bottom View)</i></p>

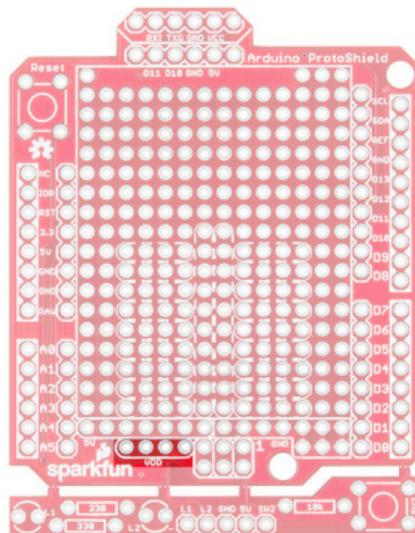
ICSP (MISO, MOSI, and SCK NC)

Need to access the Arduino's ICSP pins? The pins are broken out if you do not want to remove the shield every time you need the ICSP pins or if you decide to stack another shield on top of the ProtoShield. You will need add two 1x3 stackable headers.



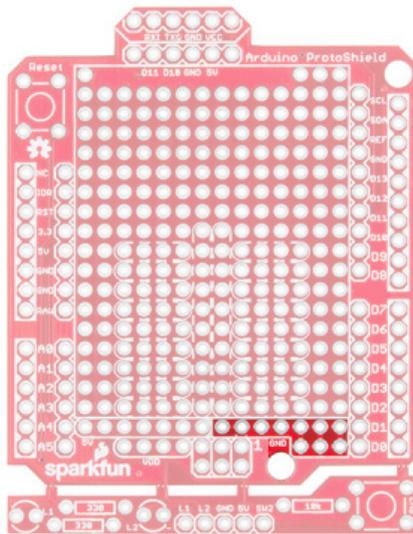
VDD Rail (NC)

Need a different voltage? We squeezed a VDD rail just below. It is currently not connected to anything. We left it up to the user to connect to Arduino's 3.3V pin or another regulated voltage (i.e. 2.8V or 1.8V).



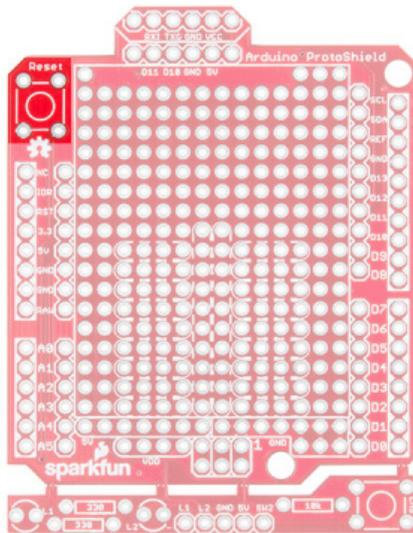
GND Rail

Need to ground more circuits? There are few more ground connections on the board as indicated by the image below.



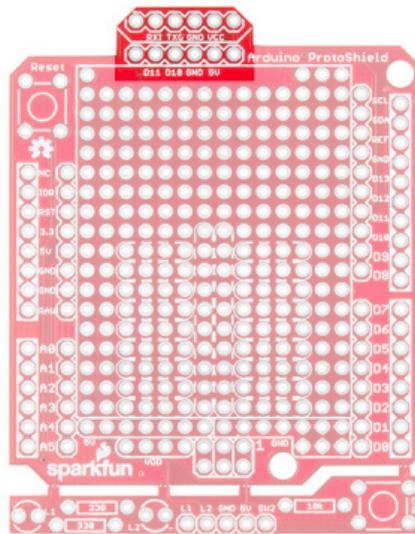
Reset Button

Stacking a shield on an Arduino can make it hard to reset the microcontroller. Like other Arduino shields, the reset button has been rerouted on the ProtoShield.

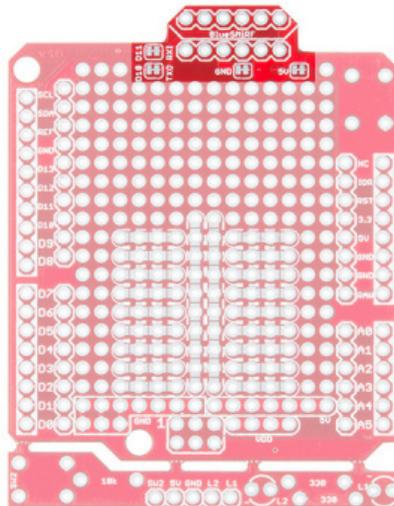


Software Serial UART Port

The SparkFun Arduino ProtoShield breaks out a software serial UART port. The pins are arranged in such a way as to connect to our BlueSMiRF if you decide to go wireless.

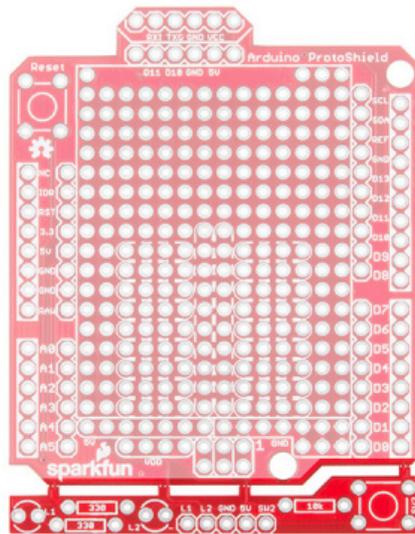


If you decide that you want to connect a different serial UART device or use different pins for serial, we added closed jumpers on the back. Simply cut the traces and reroute the connection by adding wires to the headers.



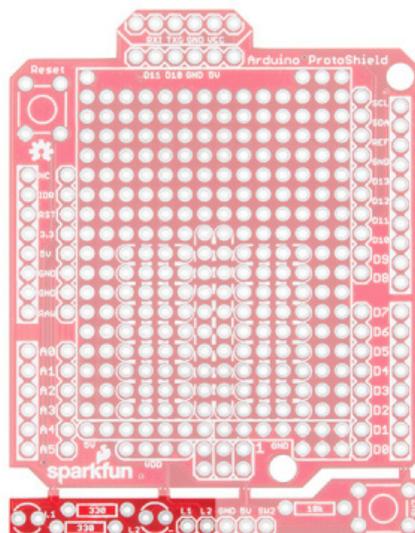
Oh Snap! ProtoSnap-able Prototyping Hardware

The board includes prototyping hardware that is based on the ProtoSnap design. When you are finished testing and want to use bigger components, simply snap off the bottom of the board using pliers.



3mm LEDs (NC)

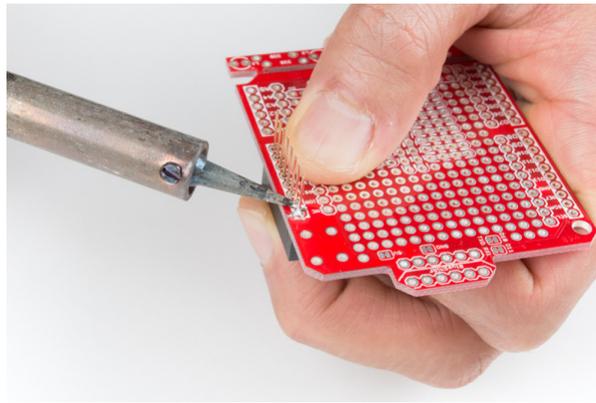
When viewing the board from the top, there are two locations for 3mm LEDs and their 330 Ω current limiting resistors.



Note: The LED pins **L1** and **L2** are not connected (NC). To control, simply solder a wire from **L1** or **L2** to an I/O pin. Ground is currently connected to the rest of the shield between the mousebites. We left it up to the user to choose and solder these connections.

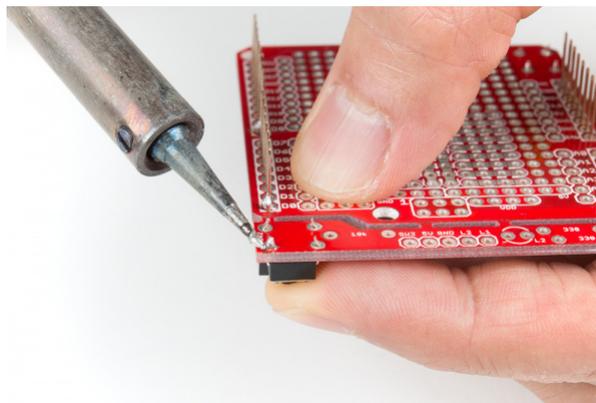
Momentary Pushbutton (NC)

There is a spot for a momentary pushbutton and 10k Ω pull up resistor on the other side of the LEDs.



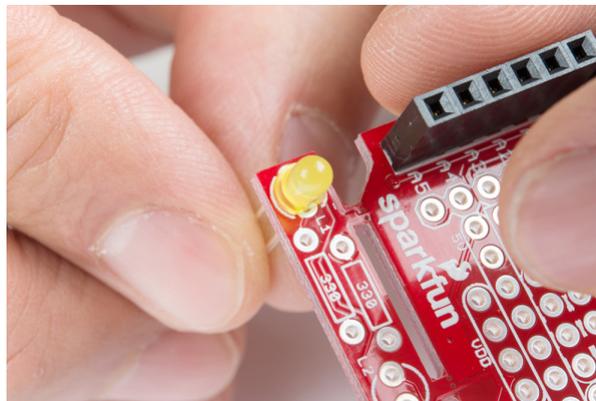
Buttons

Insert the buttons for reset and prototyping. You may need to bend the buttons to fit into the through holes. Careful not to snap off the prototyping area! Hold the button against the board and tack some solder on one pin.



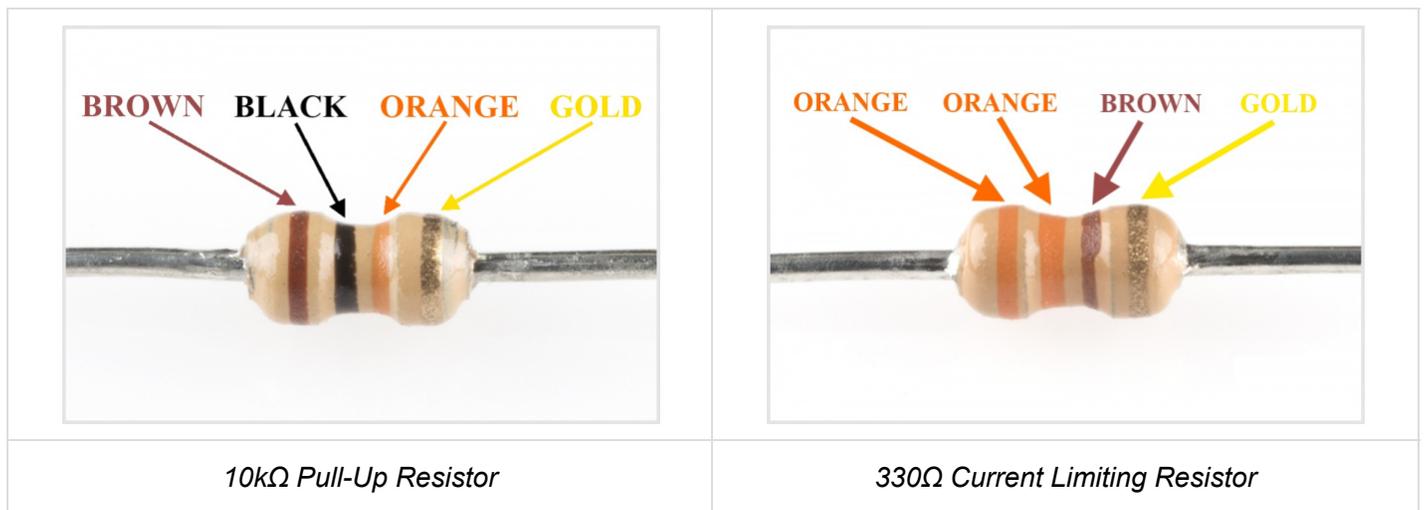
LEDs

Find the 3mm LED's cathode side and align it with its silkscreen labeled with a "—" sign. Slide the LED in from the top side and tack on a pin. Repeat for the second LED.

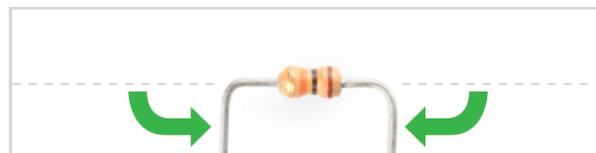


Resistors

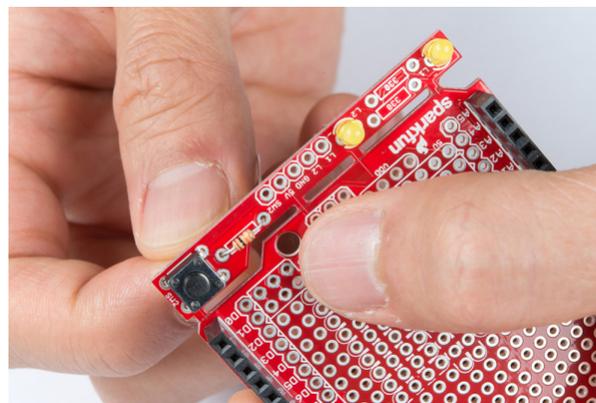
The kit includes resistors with two different values. There is one $10\text{k}\Omega$ resistor [color band = BROWN, BLACK, ORANGE, GOLD] that is used as a pull-up resistor. The two 330Ω resistors [color band = ORANGE, ORANGE, BROWN, GOLD] are used to limit current to the LEDs.



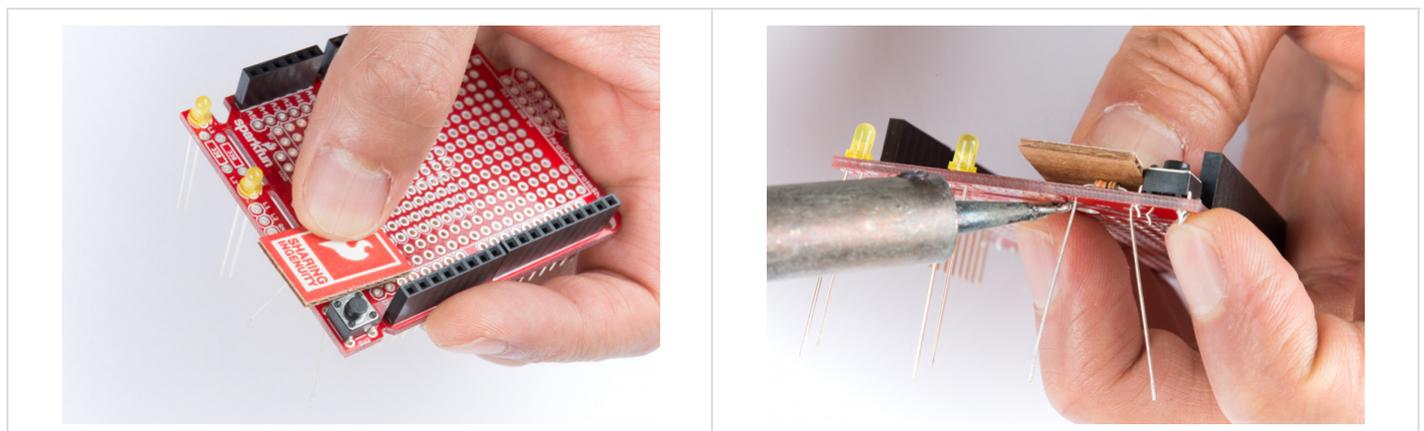
Let's start with the pull-up resistor. Bend the 10kΩ resistor's terminals.



Then slide the pins where it says "10k".



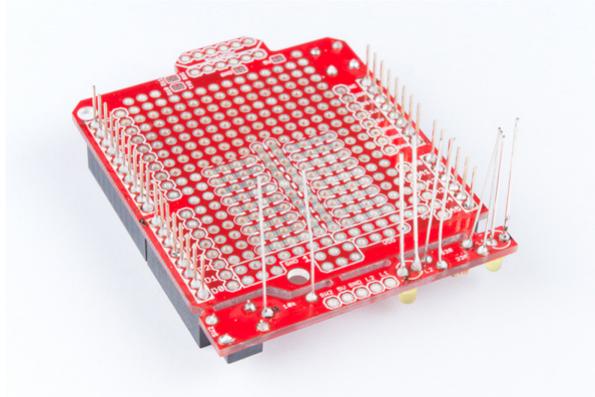
Bend the terminals to hold the resistor in place for soldering. The resistor can get hot to the touch when holding the component down during soldering. At this point, feel free to grab a small piece of cardboard to hold the resistor down. When you are ready, tack one of the 10kΩ resistor's pins. Repeat for the 330Ω resistors.



Check Alignment Before Soldering Away!

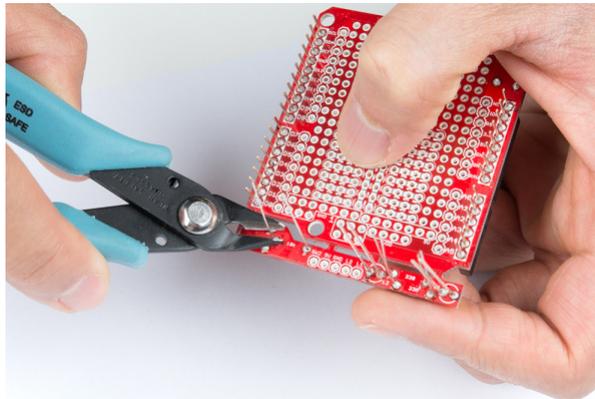
At this point, it would be good to check the alignment of the components. Make sure that they are flush against the board and they are in its correct location. Did you add solder for the components on the bottom side? All of the components included in the kit should be soldered on the side of the jumpers.

Everything good? Solder away! This is the fun part.



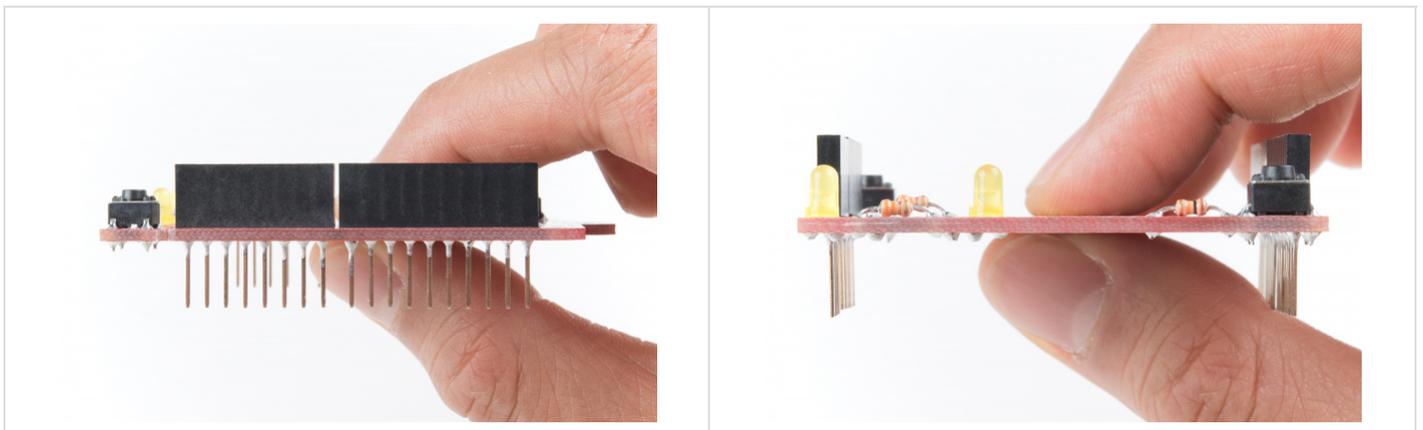
Cut Excess Leads

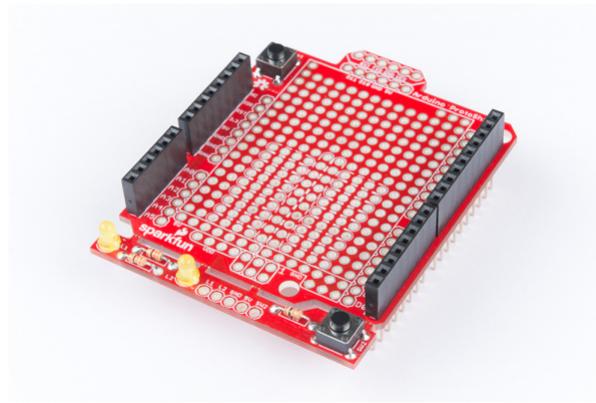
Grab a flush cutter and trim off excess leads connected to the prototyping hardware. Careful not to cut off the stackable headers!



Finished ProtoShield

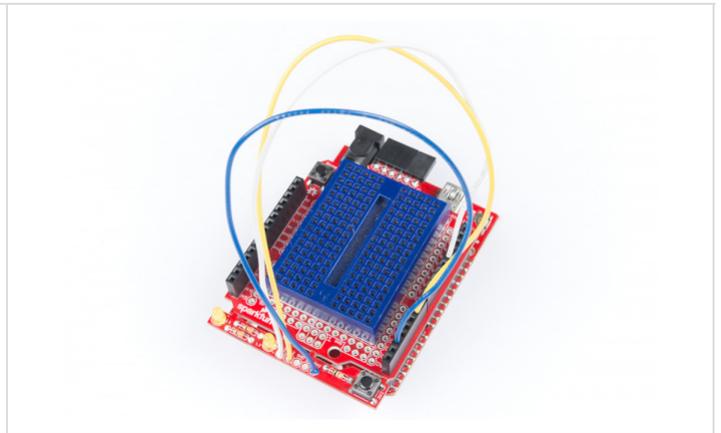
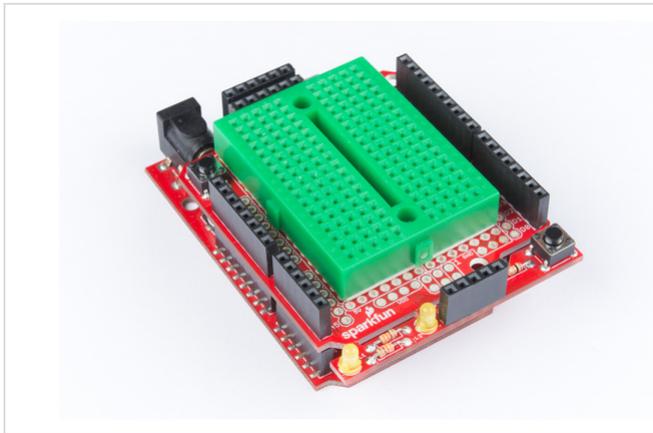
If you used water soluble flux, clean the board off. Otherwise, bask in the glow of your new ProtoShield for Arduino! Your board should look similar to the images below with the terminals soldered and cut.



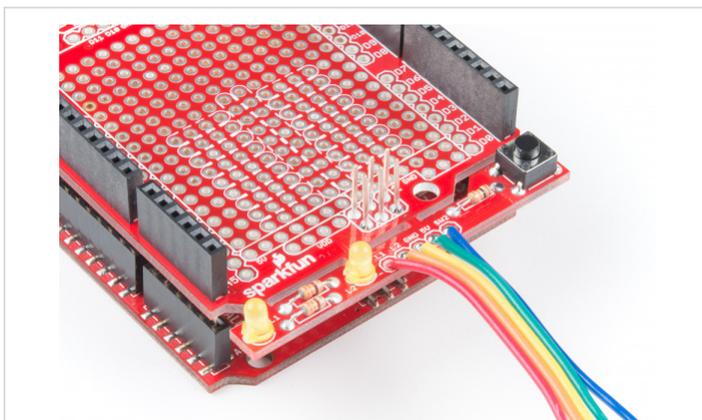


Customize

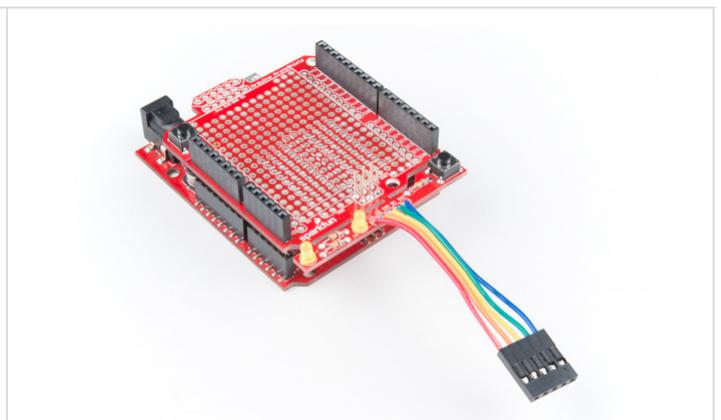
So what do you do from here? There are a few options! We left this up to the user to choose depending on personal preference. You can stick a mini-breadboard on the shield for prototyping small circuits. Additional headers can be added on however you would like for serial UART or in the prototyping area. For a lower profile, you can also strip hookup wire and solder cables against the plated through holes.



For those sandwiching the ProtoShield with another shield, you can solder two 1x3 stackable headers for the ICSP pins. If you choose this route, you will need to go with a lower profile, using wires to connect the circuit.



Close Up of ICSP Pins w/ Stackable Headers



ProtoShield w/ Stackable Headers

⚡ **Oh snap!** This shield was designed for the Arduino Uno footprint. If you are using a variant with 3.3V logic level on the ICSP pin (i.e. SAMD21 Development Board or Arduino Pro 3.3V/8MHz), you will need to cut the two traces connecting to the 5V pin on the 1x3 ICSP header.

Finished Prototyping?

If you are finished prototyping, you can remove the mini-breadboard and solder your custom circuit to the prototyping area for a more secure connection. Want to use a different LED or a bigger button in your project? You can snap off the prototyping area using needle nose pliers and replace it with the hardware of your choice!



Big Dome Pushbutton - Red
○ COM-09181



Metal Pushbutton - Momentary (16mm, Red)
● COM-11966



NovelKeys Big Switch - Tactile
● COM-14583



Super Bright LED - Red 10mm
● COM-08862

Example Code

Note: The following examples assumes you are using the latest version of the Arduino IDE on your desktop. If this is your first time using Arduino, please review our tutorial on installing the Arduino IDE.

Blinking LED

Let's connect L1 to pin 13 using a wire. Assuming that you soldered to **L1**, connect it to pin 13.

Prototyping Hardware	Arduino
L1	13

Copy the code below and upload to your Arduino!

```
/*BLINKING AN LED
```

Turn an LED on for one second, off for one second,
and repeat forever.

This sketch was written by SparkFun Electronics,
with lots of help from the Arduino community.

This example is based on the blinking an LED
example in the SparkFun Inventor's Kit v3.3.

This code is completely free for any use.

Visit <https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v33/experiment-1-blinking-an-led>

for more information.

Visit <http://www.arduino.cc> to learn about the Arduino.

Version 2.0 6/2012 MDG

```
*****/
```

```
// The LED is connected to digital pin 13
```

```
// Change the pin number depending on
```

```
// what L1 is connected to.
```

```
const int L1 = 13;
```

```
void setup()
```

```
{
```

```
  pinMode(L1, OUTPUT); //Set L1 to output
```

```
}
```

```
void loop()
```

```
{
```

```
  digitalWrite(L1, HIGH); // Turn on the LED
```

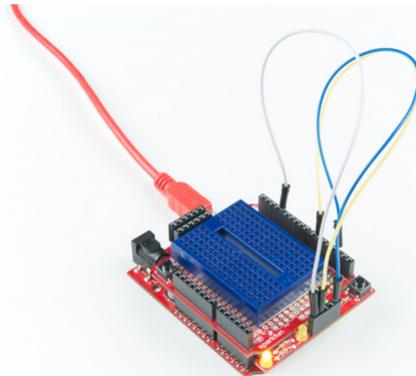
```
  delay(1000); // Wait for one second
```

```
  digitalWrite(L1, LOW); // Turn off the LED
```

```
  delay(1000); // Wait for one second
```

```
}
```

LED labeled as L1 should begin blinking.



Fading LED

Let's connect L2 to pin 6 and to fade in and out. Assuming that you soldered to **L2**, connect it to pin 6.

Prototyping Hardware	Arduino
L2	6

Copy the code below and upload to your Arduino!

```
/* Fading LED
```

```
Use for-loops to smoothly vary the brightness of an LED
```

```
This sketch was written by SparkFun Electronics,  
with lots of help from the Arduino community.  
This example is based on the Fading LEDs example  
in the LilyPad Development Board Activity Guide.
```

```
This code is completely free for any use.
```

```
Visit https://learn.sparkfun.com/tutorials/lilypad-development-board-activity-guide/4-fading-1  
eds
```

```
for more information
```

```
Visit http://www.arduino.cc to learn about the Arduino.
```

```
*****/
```

```
// Create integer variable for the LED pin we'll be using:
```

```
const int L2 = 6;
```

```
// Create a new integer variable called brightness:
```

```
int brightness;
```

```
void setup()
```

```
{
```

```
  // Set the LED pins to be output:
```

```
  pinMode(L2, OUTPUT);
```

```
}
```

```
void loop()
```

```
{
```

```
  // The two "for loops" below will make a LED fade on and off in a "breathing" pattern.
```

```
  // Now we'll have the program automatically change the value of brightness
```

```
  // using a command called "for".
```

```
  // for is like a tiny version of loop. The for command has several parts:
```

```
  // 1. something to do before starting (brightness = 0)
```

```
  // 2. a test to decide whether to keep going (brightness <= 255)
```

```
  // 3. a block of commands to run (everything within the {} below the for)
```

```
  // 4. a command to run before doing it again (brightness = brightness + 1)
```

```
  // Here's a for command which will start brightness at 0, check to see if it's less than
```

```
  // or equal to 255, run the commands after it, then add one to brightness and start over:
```

```
for (brightness = 0; brightness <= 255; brightness = brightness + 1)
```

```
{
```

```
  // Within the loop, we'll use brightness variable to control the brightness of the LEDs:
```

```
  analogWrite(L2, brightness);
```

```
  // NOTE that not all pins work with analogWrite!
```

```
  // The ones with a "~" in front of them will change brightness,
```

```

// the others will only turn on if brightness > 128.
// Both types are used above, run the code and note the difference between them.

// The delay command controls the speed - if you make the delay larger,
// it will slow down the loop. Smaller, and it will run faster:

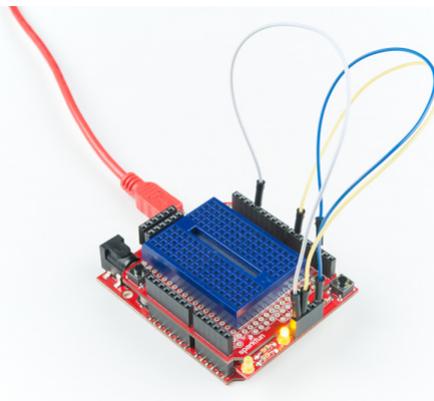
delay(5);
}

// What if we want the LED to start at full brightness and fade to black?
// We can easily set up the for loop to run in reverse:

for (brightness = 255; brightness >= 0; brightness = brightness - 1)
{
  analogWrite(L2, brightness);
  delay(5);
}
}

```

LED labeled as L2 should begin fading in and out. Try connecting and redefining the LED to another pin on your Arduino to see if you can still fade.



Momentary Push Button

Let's connect SW2 to pin 2 to read a push button. For feedback, we will be using both of the LEDs in our prototyping hardware. Assuming that you soldered to **SW2, L2, and L1**, connect it to their respective pins on 2, 6, and 13.

Prototyping Hardware	Arduino
SW2	2
L2	6
L1	13

Copy the code below and upload to your Arduino!

```
/*Momentary Push Button
```

```
Use momentary pushbuttons for digital input
```

```
This sketch was written by SparkFun Electronics,  
with lots of help from the Arduino community.  
This example is based on the push button  
example in the SparkFun Inventor's Kit v3.3
```

```
This code is completely free for any use.
```

```
Visit https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v33/experiment-5-push-buttons  
for more information.
```

```
Visit http://www.arduino.cc to learn about the Arduino.
```

```
Version 2.0 6/2012 MDG
```

```
Version 2.1 9/2014 BCH
```

```
*****/
```

```
const int SW2 = 2; // pushbutton 1 pin  
const int L1 = 13; // LED pin  
const int L2 = 6; // LED pin
```

```
int button1State; // variables to hold the pushbutton states
```

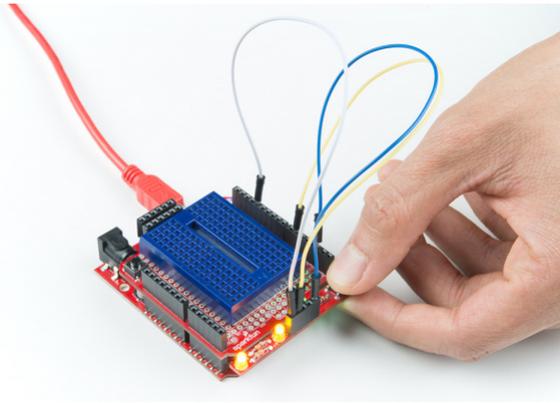
```
void setup()
```

```
{  
  // Set up the pushbutton pins to be an input:  
  pinMode(SW2, INPUT);  
  // Set up the LED pin to be an output:  
  pinMode(L1, OUTPUT);  
  pinMode(L2, OUTPUT);  
}
```

```
void loop()
```

```
{  
  button1State = digitalRead(SW2);  
  
  // if SW2 is pressed  
  if (button1State == LOW) {  
    digitalWrite(L1, HIGH); // turn the LED on  
    digitalWrite(L2, HIGH); // turn the LED on  
  }  
  else {  
    digitalWrite(L1, LOW); // turn the LED off  
    digitalWrite(L2, LOW); // turn the LED off  
  }  
}
```

Pressing on the button should light up both LEDs simultaneously. Removing your finger from the button should turn them off.



Bluetooth Serial Passthrough

Let's try to send a character between two BlueSMiRF Silvers (i.e. the RN42s). In this example, we will need two bluetooths, ProtoShields, and Arduinos. Assuming that the boards are soldered, connect the BlueSMiRFs to the Software Serial UART port. We will be using the default connection to software serial pins 10 and 11. Simply align the silkscreen as indicated in the hookup table below.

BlueSMiRF Silver	ProtoShield Software Serial UART port
RX-I	RXI (D11)
TX-O	TXO (D10)
GND	GND (GND)
VCC	VCC (5V)

⚡ **Oh snap!** This shield was designed for the BlueSMiRF footprint. If you are using the Bluetooth Mate, you will need to cut the traces and reroute the respective pins. For more information about the differences, make sure to check out the comparison in the Using the BlueSMiRF tutorial.

We will also be using the same connection to the prototyping hardware.

Prototyping Hardware	Arduino
SW2	2
L2	6
L1	13

Power both up, copy the code below, and upload to both Arduinos!

```

/*
  Example Bluetooth Serial Passthrough Sketch
  modified by: Ho Yun "Bobby" Chan
  date: May 17, 2018
  by: Jim Lindblom
  date: February 26, 2013
  SparkFun Electronics
  license: Public domain

  This example sketch converts an RN-42 bluetooth module to
  communicate at 9600 bps (from 115200), and passes any serial
  data between Serial Monitor and bluetooth module.
  *****/

#include <SoftwareSerial.h>

int bluetoothTx = 10; // TX-0 pin of bluetooth mate, Arduino D10
int bluetoothRx = 11; // RX-I pin of bluetooth mate, Arduino D11

SoftwareSerial bluetooth(bluetoothTx, bluetoothRx); // (Arduino SS_RX = pin 10, Arduino SS_TX = p
in 11)

void setup()
{
  Serial.begin(9600); // Begin the serial monitor at 9600bps

  bluetooth.begin(115200); // The Bluetooth Mate defaults to 115200bps
  bluetooth.print("$"); // Print three times individually
  bluetooth.print("$");
  bluetooth.print("$"); // Enter command mode
  delay(100); // Short delay, wait for the Mate to send back CMD
  bluetooth.println("U,9600,N"); // Temporarily Change the baudrate to 9600, no parity
  // 115200 can be too fast at times for NewSoftSerial to relay the data reliably
  bluetooth.begin(9600); // Start bluetooth serial at 9600
}

void loop()
{
  if (bluetooth.available()) // If the bluetooth sent any characters
  {
    // Send any characters the bluetooth prints to the serial monitor
    Serial.print((char)bluetooth.read());
  }
  if (Serial.available()) // If stuff was typed in the serial monitor
  {
    // Send any characters the Serial monitor prints to the bluetooth
    bluetooth.print((char)Serial.read());
  }
  // and loop forever and ever!
}

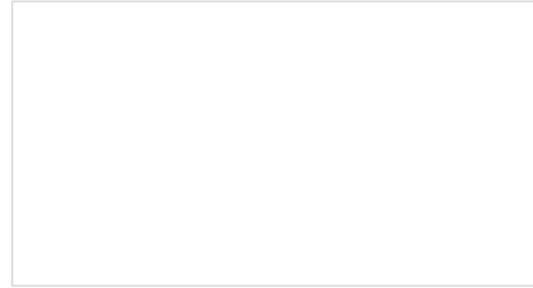
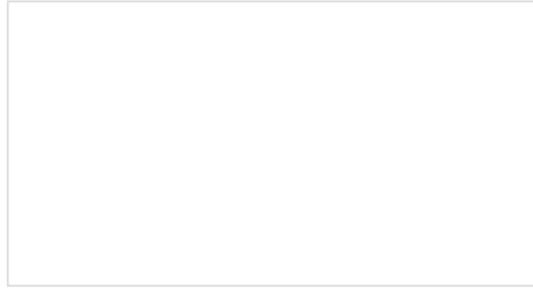
```

Installing Arduino IDE

A step-by-step guide to installing and testing the Arduino software on Windows, Mac, and Linux.

Using the BlueSMiRF

How to get started using the BlueSMiRF and Bluetooth Mate Silvers.



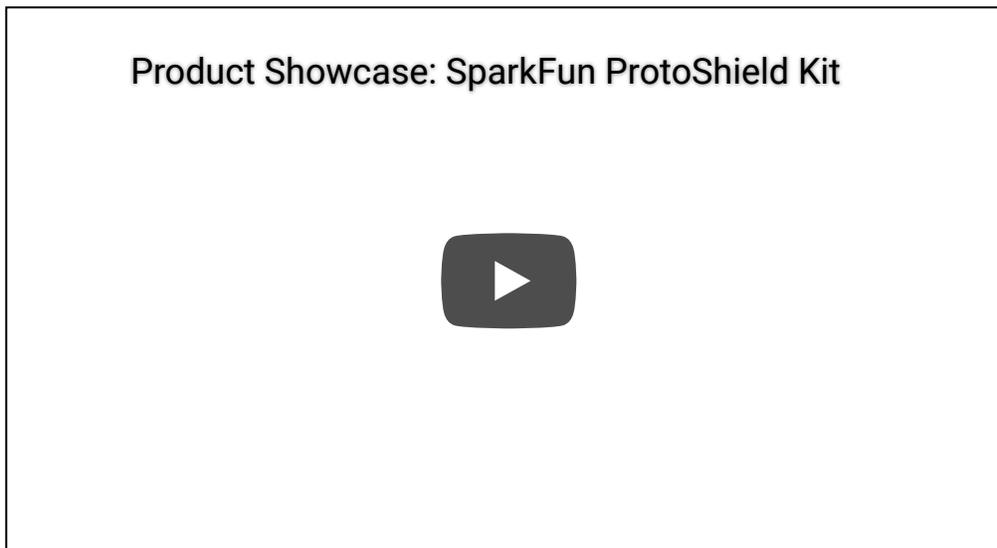
How to Work with Jumper Pads and PCB Traces

Handling PCB jumper pads and traces is an essential skill. Learn how to cut a PCB trace, add a solder jumper between pads to reroute connections, and repair a trace with the green wire method if a trace is damaged.

Arduino Shields v2

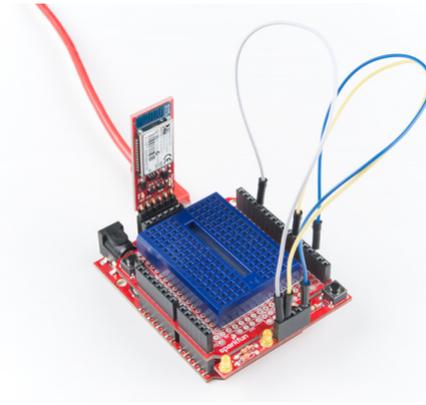
An update to our classic Arduino Shields Tutorial! All things Arduino shields. What they are and how to assemble them.

Suggested Viewing



Hardware Overview

There is a lot going on with the ProtoShield so it is useful to know what side we are referencing when soldering components to the board. We will refer to the top side based on the board's name on the upper right hand corner. The bottom will be the side with the jumpers.



Discovering, Pair, and Autoconnecting Bluetooths

Once uploaded,

- open the Arduino Serial Monitor set at **9600 baud** with **No line ending**
- send **\$\$\$** and hit the **Send** button to set the bluetooth in command mode
- change the Arduino Serial Monitor from *No Line Ending* to **Newline**
- send the autoconnect command **sm,3**
- send the inquiry scan command **i** to scan for the other bluetooth in range

You should see something similar to the output below.

```
CMD
AOK
Inquiry, COD=0
Found 1
000666643FBF,RN42-3FBF,1F00
Inquiry Done
```

The other RN42 bluetooth that was in range came up with address of `000666643FBF` . You should see something similar when connecting a RN-41 or RN-42. Type the following command and change the address to the one that you obtained:

- **c,000666643FBF** and hit the **Send** button

This will pair and connect both bluetooths together. The address will be saved in memory and auto connect every time there is a power cycle. Open a serial terminal (since you can only have one Arduino Serial Monitor open at a time) set at **9600 baud** and connect the other Arduino/ProtoShield/BlueSMiRF. Sending any character from the Arduino Serial Monitor should pop up on the serial terminal and vice versa!

Combining It All to Send a Message!

Let's send a simple message between the two bluetooths now that they are configured to autoconnect. Copy the code below, and upload to both Arduinos!

```

/*
  Example Serial Bluetooth Messenger
  by: Ho Yun "Bobby" Chan
  SparkFun Electronics
  date: May 16, 2018
  license: Public domain

  This example sketch converts an RN-42 bluetooth module to
  communicate at 9600 bps (from 115200). Assuming that the
  two bluetooths are paired and configured to autoconnect,
  a message is sent with the push of a button. Any
  received characters from the other bluetooth will display
  on the serial monitor.

  This sketch was written by SparkFun Electronics.
  This example is based on the Example Bluetooth Serial
  Passthrough Sketch by Jim Lindblom.

  This code is completely free for any use.
  Visit https://learn.sparkfun.com/tutorials/using-the-bluesmirf
  for more information.
  *****/

#include <SoftwareSerial.h>

const int SW2 = 2; // pushbutton 1 pin
const int L1 = 13; // LED pin for push button
const int L2 = 6; // LED pin for received character

int button1State; // variables to hold the pushbutton states

int bluetoothTx = 10; // TX-0 pin of bluetooth mate, Arduino D10
int bluetoothRx = 11; // RX-I pin of bluetooth mate, Arduino D11

SoftwareSerial bluetooth(bluetoothTx, bluetoothRx); // (Arduino SS_RX = pin 10, Arduino SS_TX = p
in 11)

void setup()
{
  // Set up the pushbutton pins to be an input:
  pinMode(SW2, INPUT);
  // Set up the LED pin to be an output:
  pinMode(L1, OUTPUT);
  pinMode(L2, OUTPUT);

  Serial.begin(9600); // Begin the serial monitor at 9600bps

  bluetooth.begin(115200); // The Bluetooth Mate defaults to 115200bps
  bluetooth.print("$"); // Print three times individually
  bluetooth.print("$");
  bluetooth.print("$"); // Enter command mode
  delay(100); // Short delay, wait for the Mate to send back CMD
  bluetooth.println("U,9600,N"); // Temporarily Change the baudrate to 9600, no parity

```

```

// 115200 can be too fast at times for NewSoftSerial to relay the data reliably
bluetooth.begin(9600); // Start bluetooth serial at 9600
}

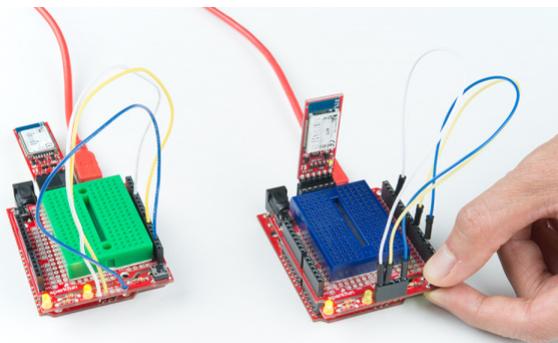
void loop()
{
  button1State = digitalRead(SW2);

  //Send a character
  if (button1State == LOW) //if SW2 is pressed
  {
    // Send characters to bluetooth to transmit
    bluetooth.println("Hi!");
    digitalWrite(L1, HIGH); // turn the LED on
  }
  else {
    digitalWrite(L1, LOW); // turn the LED off
  }

  // If the bluetooth received any characters, print to the serial monitor
  if (bluetooth.available())
  {
    // Send any characters the bluetooth prints to the serial monitor
    Serial.print((char)bluetooth.read());
    digitalWrite(L2, HIGH); // turn the LED on
  }
  else {
    digitalWrite(L2, LOW); // turn the LED on
  }
  // and loop forever and ever!
}

```

Pressing a button on one of the Arduinos (as indicated by the blue mini-breadboard) should send a message and light up the LED labeled as L1 on the transmitting node. The receiving node (as indicated by the green mini-breadboard) will light up the LED labeled as L2 and print a message (in this case "Hi!") to a serial monitor or serial terminal. The other Arduino will do the same when you press on its button.



Note: Take turns pressing on the push buttons to transmit. The message may get garbled when both bluetooths are transmitting simultaneously.

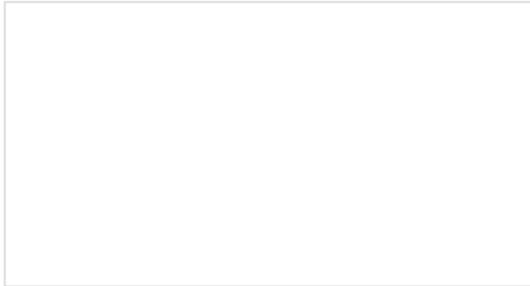
Resources and Going Further

Now that you've successfully got your protoshield for Arduino up and running, it's time to incorporate it into your own project!

For more information, check out the resources below:

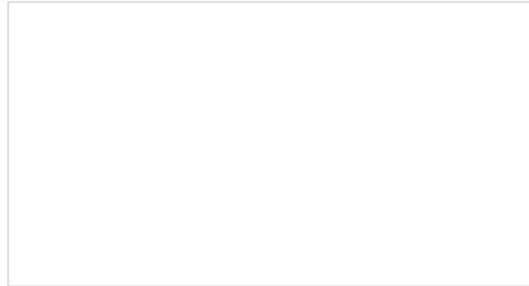
- Schematic (PDF)
- Eagle Files (ZIP)
- GitHub Repo
- SparkFun Product Showcase: SparkFun ProtoShield Kit
- RN-42 and RN-41 Command Reference and User's Guide (PDF)

Need some inspiration for your next project? Check out some of these related tutorials:



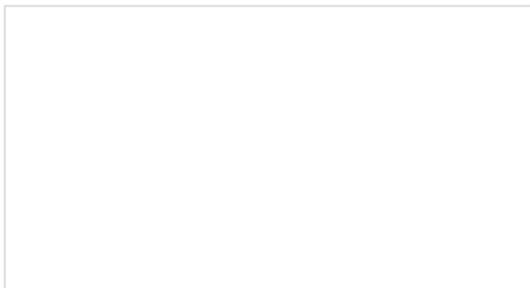
Working with Wire

How to strip, crimp, and work with wire.



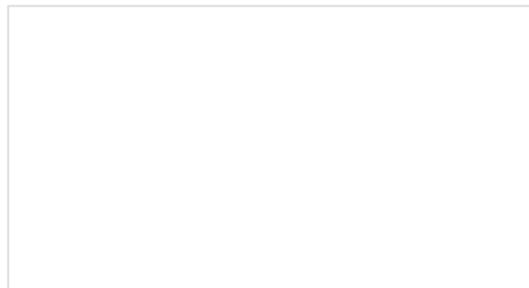
Prototype Wearable LED Dance Harness

A project tutorial to add an extra effect for dancers performing a choreographed piece. The harness can be added quickly under a costume.



Proto Pedal Assembly and Theory Guide

Getting started with the SparkFun Proto Pedal. We'll assemble the board, then discuss some of the details of the circuit.



Fuse Breakout Board Hookup Guide

Protect your project by adding a fuse in your circuit to save it from molten destruction.