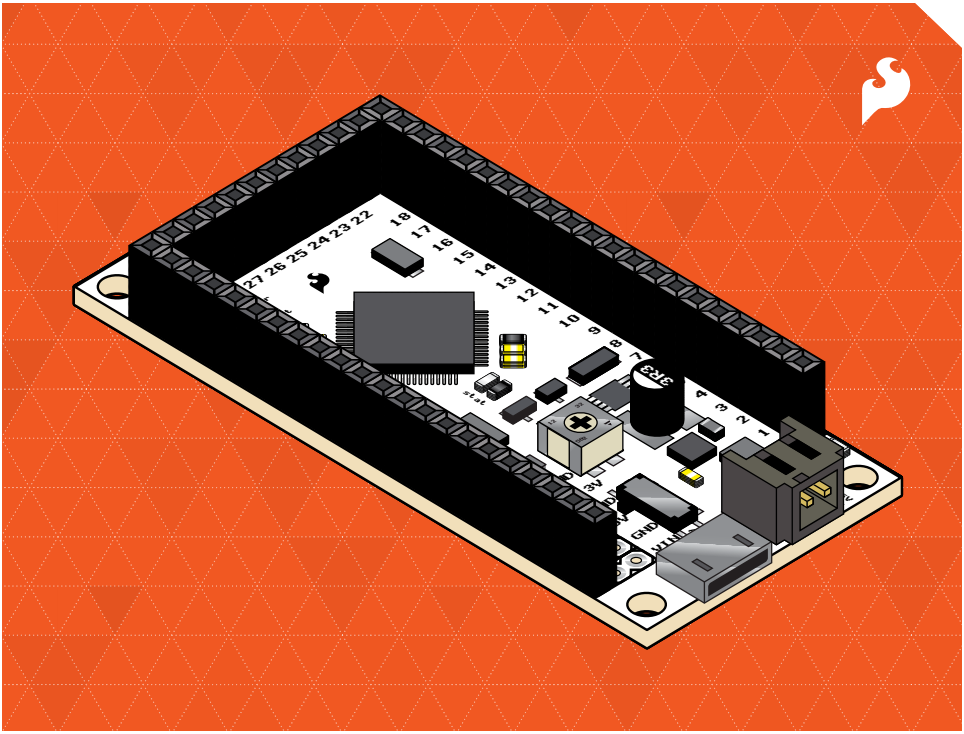
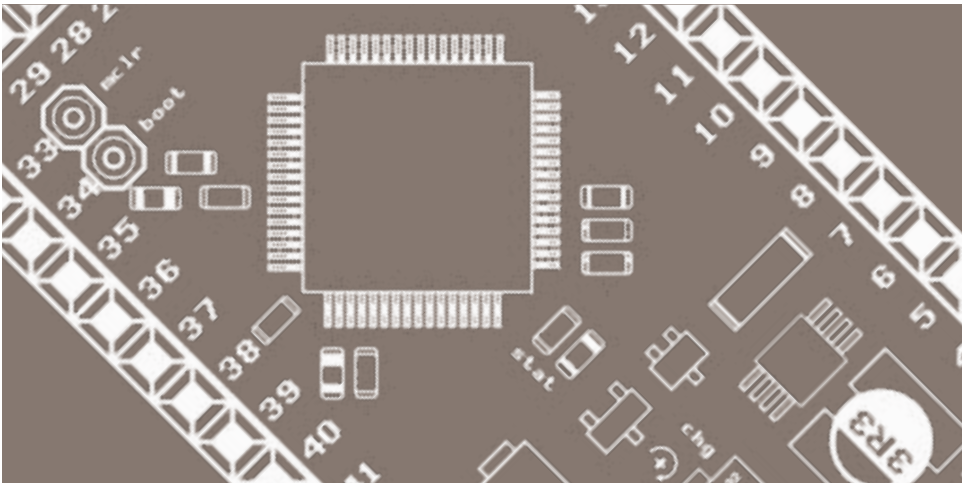
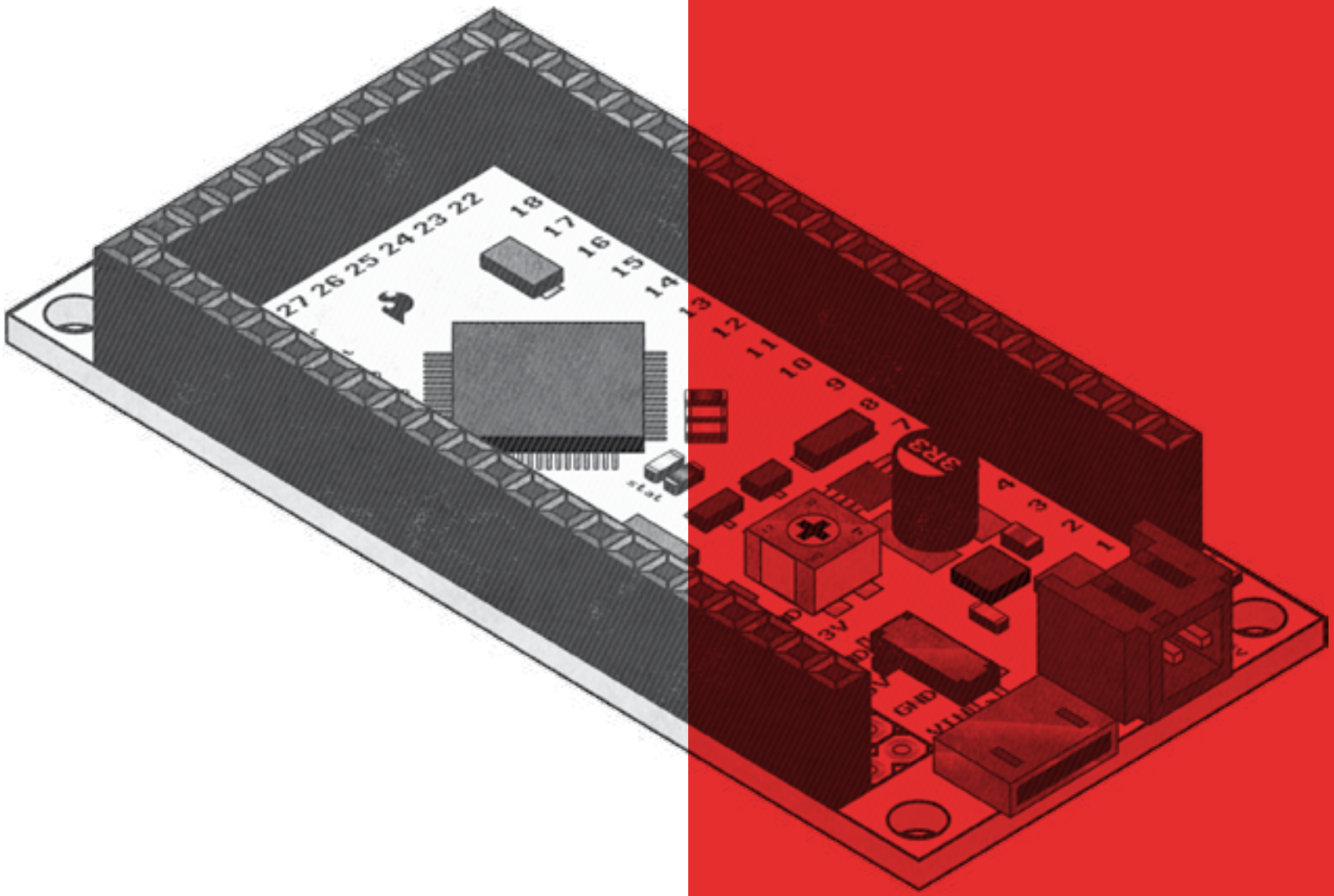


SIKIO GUIDE

Your guide to the SparkFun Inventor's Kit for IOIO-OTG



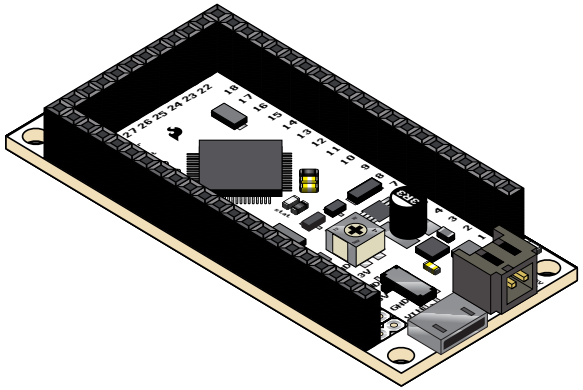

27 26 25 24 23 22
18 17 16 15 14 13 12 11 10 9 8 7
stat
3R3
4 3 2 1
3V
GND

INDEX

Introduction to SIKIO

The SparkFun Inventor's Kit for IOIO-OTG (pronounced 'yo-yo O-T-G') will teach you how to program your Android device to interface with electronic circuits using the IOIO-OTG hardware platform. In other words, you will be writing Android applications that control things like motors, LEDs, and buttons. We have prepared eight experiments that demonstrate the basic interaction of an Android device with hardware. In the process you will learn more about software development for Android, as well as the fundamentals of electronics.

We will be building our Android applications using a simple development environment called Processing. This environment allows you to get past a lot of the initial hurdles of developing for Android, and allows for more easily implemented graphics and visual effects. Utilizing the IOIO libraries we provide, we can also use the processing environment to interact with physical hardware.



The best way to get started with the SIKIO is to start the software installation first, because it takes a good amount of time to finish. Meanwhile, you can go through all of the examples in this manual to try them out. With each example, you will be presented with a schematic and hardware connection diagram. Build your circuit first, then scan the QR code included with the example with an Android. You will need to download a QR scanning application if you don't already have one. After you have scanned the QR code, the application can be installed on your phone. Now you can run the application with your Android device connected to the IOIO-OTG via USB, with your circuit connected to the IOIO-OTG hardware. This process should allow you to become comfortable with the circuit-building process.

As you explore the hardware, make sure to continue the software installation process. Once that's completed, you get to start tinkering with the code. You'll open the code in Processing and download it to your Android device. Then you'll connect your device to your IOIO-OTG while running the downloaded application to test it out. With each experiment and the help of the online wiki, you'll learn what's going on behind the scenes, and start acquiring the ability to make the applications you envision on your own.

QR scanner

The SparkFun Inventor's Kit for IOIO-OTG guide uses quick response or "QR" codes to make downloading the code for each circuit easier. You will need to download an application capable of reading QR codes if you don't have one already. One such free app is called "QR Droid."

Note:

COMPLETE SOFTWARE INSTALLATION TYPICALLY TAKES 60 MINUTES

This is a quick overview of the software installation process and various components required to develop applications to control hardware with your phone. The full guide is found online in the SIKIO wiki here:

https://sparkfun.com/sikio

To actually play with the code and develop your own applications, it is necessary to download our software bundle and work through the full guide's installation process.
This will give you a rough idea of what the installation entails.

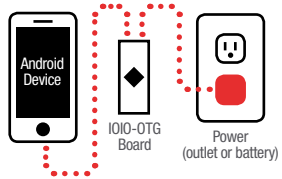
We suggest you get started on the full software installation now. As you're working on that, you can continue reading through this manual to get an overview of what you'll be installing and why. You will also be able to continue to the experiments, build the circuits, and test out the applications by utilizing the QR codes. Once the software is done installing, you can start tinkering with the code and downloading it to your Android device.

Coding an application

The following is a basic overview of the coding tool chain for the IOIO-OTG:

Running your application

In order to run your app, the IOIO-OTG board will need to be connected both to your Android device and to an appropriate power source.



Software Installation Components

As you work your way through the complete software install guide, you will first download a large bundle of software which includes the following major components to install. Each has a purpose and is necessary to get the full tool chain to work properly and allow you to control hardware with your Android device.

① Android SDK

First, the Android SDK Bundle needs to be installed. This provides you with the foundational software and tools you will need to build Android applications. Since Android programming is based on Java, the Java JDK will have to be present on your machine or installed.

② Android SDK Configuration

The Android SDK manager is what allows you to download SDK tools that enable you to build applications. By downloading our bundle, these tools will be included and already properly configured. Our configuration includes a specific version of the Google API that allows the code to be translated into commands your phone can understand as well as some supporting libraries that allow for further access to your phone.

③ Android USB Driver

A USB driver is needed to let your Android device be programmed by your computer. Your OS may install a default working one automatically and if not, try the default drivers included in the bundle. If neither works, you'll have to search for a specific driver based on the make and model on your phone or tablet.

④ Processing Environment and Libraries

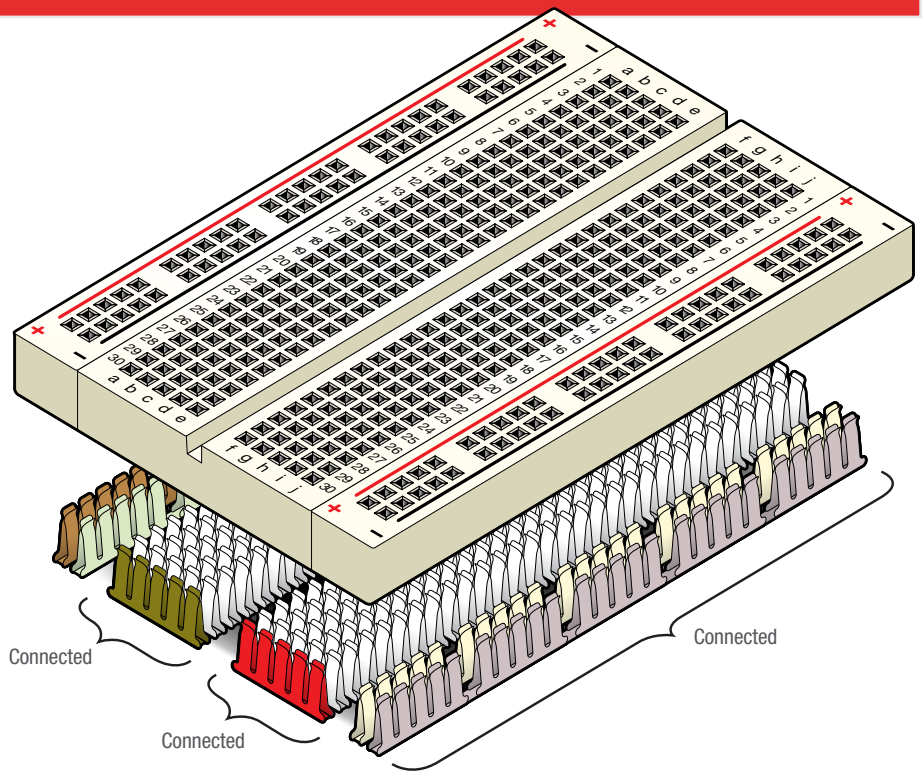
Now, a version of the Processing environment will be is included and you will install its Android mode. This will allow you to create Android applications with considerably less work than the standard method. Next, we'll install the IOIO libraries for Processing so that the IOIO hardware can be accessed by Processing programs.

Device settings

There are a couple settings that need to be enabled in order to use your Android device with the IOIO-OTG hardware. These typically can be found under Settings in the Applications, Developer, or Security sections depending on the Android version. See the wiki install for more info.

☑ **USB debugging**

☑ **Use 3rd party/unknown/non-market apps**



The solderless breadboard will allow you to assemble and disassemble circuits quickly without the use of a soldering iron. The metal connections inside are arranged in isolated groups to help you control the flow of electricity through your circuit.

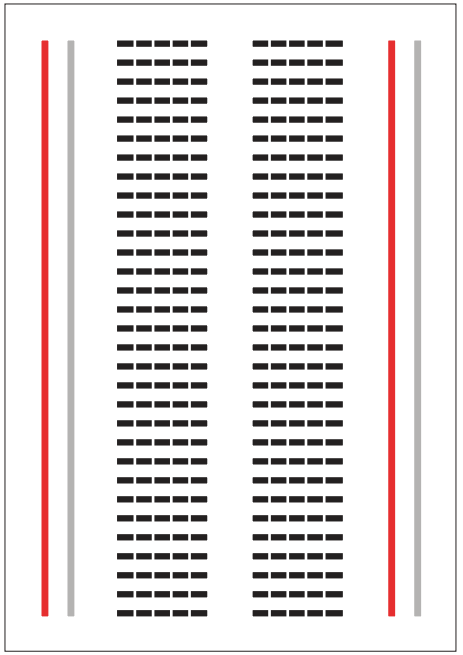
Power Buses

+ Power
Power is usually connected to all sockets in these vertical columns

- Ground
All sockets in these vertical columns are usually running to ground

▪▪▪▪▪ Horizontal rows

Each of these rows numbered 1-30 are comprised of five connected sockets per row, labeled a-e and f-j.



① USB connector (micro-AB, female)
Used to connect to host computer, an Android device or a Bluetooth dongle.

② Power jack (2-pin JST, female)
Used for power supply to the board. Voltage between 5V-15V should be supplied.

③ GND pins (10 pins)
Ground connection.

④ VIN pins (3 pins)
Used for outputting the supply voltage to your circuit, or as an alternative input to the power jack.

⑤ 5V pins (3 pins)
5V output from the on-board regulator, which can be used in your circuit.

⑥ 3.3V pins (3 pins)
3.3V output from the on-board regulator, which can be used in your circuit.

⑦ I/O pins (46 pins)
General purpose I/O pins. Some have special functions, see artwork on back of PCB.

⑧ PWR LED (red)
Lights when the IOIO-OTG is getting power.

⑨ STAT LED (yellow)
General purpose on-board LED, under application control.

⑩ MCLR pin
Not normally used. Its purpose is for programming new bootloader firmware on the IOIO-OTG board.

⑪ BOOT pin
Not normally used. Special pin used for getting the IOIO-OTG into boot-loader mode on power-up. Note that this pin is shared with the stat LED.

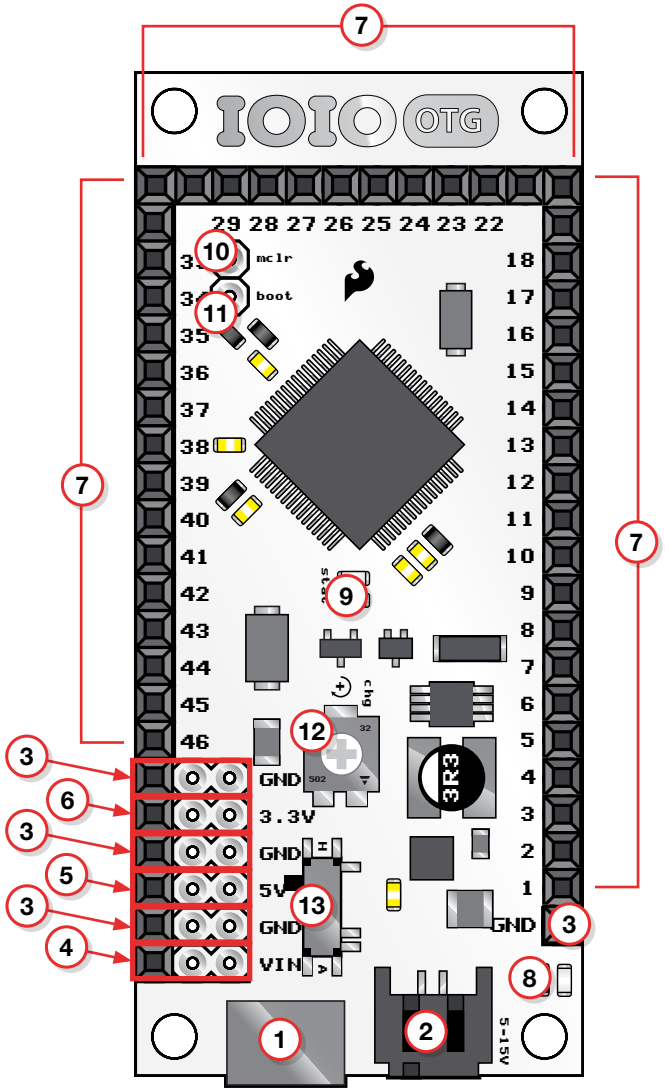
⑫ Charge current trimmer (CHG)
Adjusts the amount of current supplied on the VBUS line of the USB when acting as a USB host (or when it is connected to your Android). Typically used in battery-powered application to prevent battery drain. Turning in the (+) direction increases charge current. Generally, this can be left at the default factory setting.

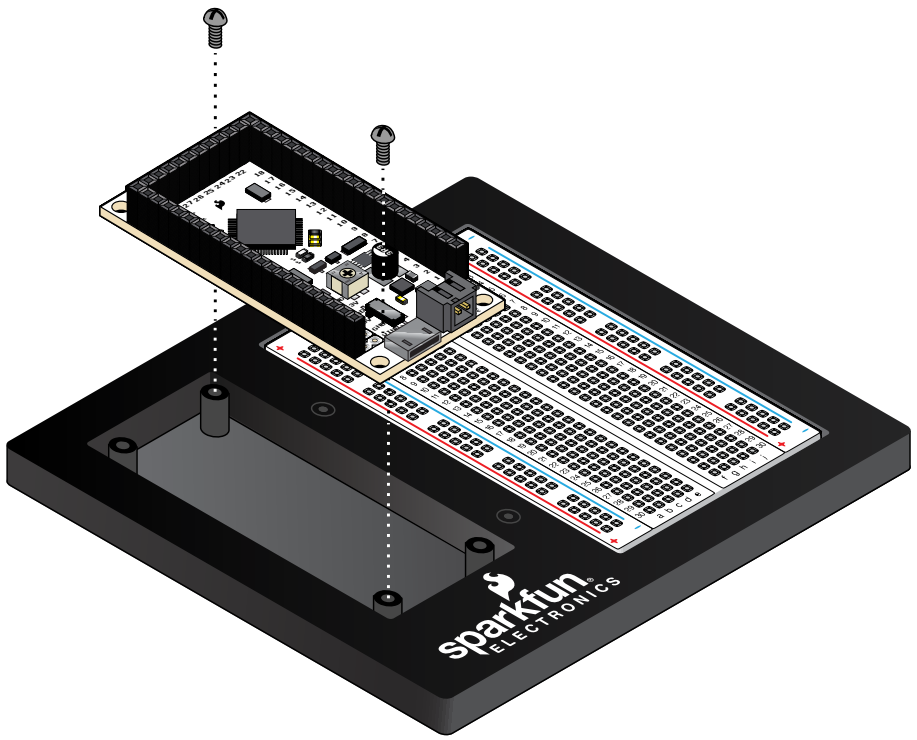
⑬ Host/Device selector switch
Not normally used. Keep switch in A mode. In “A” mode, the IOIO-OTG will detect whether it should act as host or as device automatically, according to whichever USB connector is plugged in (micro-A or micro-B). By default, this switch should be set to host mode, “H,” to support non-standard USB cables or adapters that use micro-B type.

THE ANATOMY OF THE IOIO-OTG BOARD

The IOIO-OTG board

The IOIO-OTG is a board specially designed to work with Android devices. The board provides robust connectivity to an Android device via a USB connection and is fully controllable from within an Android application using a simple and intuitive Java API.





SECURE IOIO-OTG AND BREADBOARD TO BASEPLATE

You may wish to secure your IOIO-OTG and breadboard to the provided baseplate for easier installation of circuit components. Your breadboard is self-adhesive and the IOIO-OTG board can be attached with the included phillips-head screws for easy removal later.

0

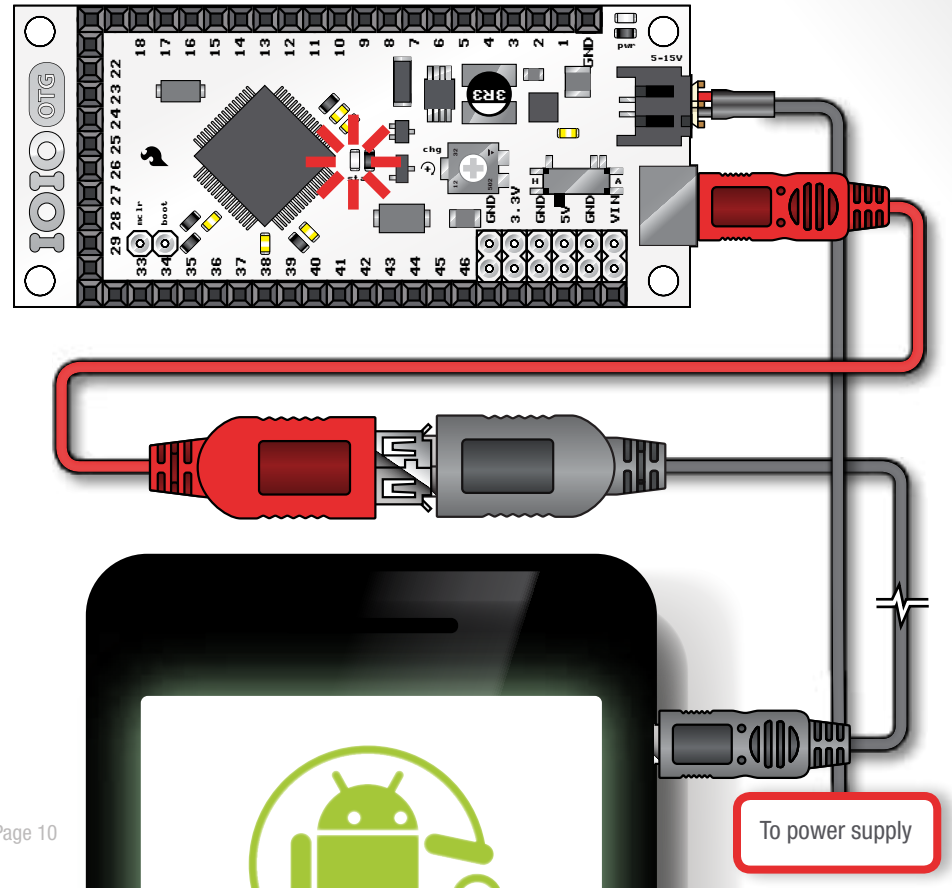
YOUR FIRST IOIO-OTG APP

Circuit 0 is a simple application that will cause the IOIO-OTG's status LED to blink on and off. The purpose of this application is simply to demonstrate how IOIO-OTG will interact with your Android device once programmed. No kit components will need to be installed.

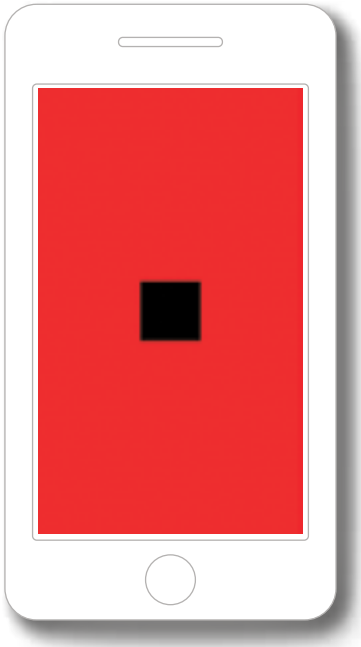
This is essentially a blink sketch intended to demonstrate the basics of programming an Android with the Processing development environment to interact with hardware via a IOIO-OTG. This example blinks an LED on the IOIO-OTG, and blinks a rectangle on the Android screen. This example shows how the main Processing sketch and the IOIO-OTG code work and interact. The main sketch file controls the Android user interface, while the IOIO-OTG tab controls the IOIO-OTG board, and the two interact through the use of global variables.

Process Flow

The basic process flow for running this application and the others is as follows. Open the experiment in Processing (github wiki contains detailed information). Compile and download the code to your Android device while it's attached to your computer. The application will now run on your device. Plug your device into your IOIO-OTG via USB as shown. It should start working properly. If not, try restarting the application or the IOIO-OTG itself. Also, try turning off USB debugging before restarting the application for newer Androids.



Every half second, the IOIO-OTG tab sends a command to the IOIO-OTG to turn the onboard LED on or off. It also sets the global variable 'lightON' at that time before going to back to sleep. This global variable is read within the draw() loop and a black or white box is drawn depending on the state of the LED.



What You Should See

Once you have loaded the application onto your device either by downloading the code or with a QR scan, the status LED should blink on the IOIO-OTG and your device screen will display a blinking rectangle. If it doesn't seem to be working, check the following Troubleshooting section.

This example doesn't require anything other than your Android and a IOIO-OTG, connected by a USB cable, and a way to power your IOIO-OTG.

Scanning this Quick Response code will install the appropriate app on your Android device

Need more?

Additional resources that explain the code's functionality for this particular application exist in our more extensive online wiki. This information can be found at **sparkfun.com/SIKIO**

Want to get into the code?

With your circuit assembled and code uploaded to your phone, your circuit is complete. If you have installed the necessary software to edit the sketch, now would be a good time to adjust some of the parameters in the code and fine-tune the way the circuit functions. Remember – if you screw up, you can always download the original again and start over.

Software installation instructions can be found on page 3

Programming with Processing

All of our code 'sketches' in the SIKIO consist of two main tabs. One tab, which we'll call our 'Main' tab, contains our setup() and draw() functions, and code that affects the Android aspects of each experiment (what appears on your phone screen). The other tab, called 'IOIO_Tab', will deal with all the functions that affect the IOIO-OTG such as connecting to the board, opening hardware pins, and using Digital and Analog inputs and outputs.

In this case, the main tab imports our IOIO libraries and creates our IOIO instance so we can talk to the hardware. It also establishes a variable called lightOn which is used to keep track of the LED's status as well as change the color of the rectangle on the screen. In our draw loop, we draw a background as well as our rectangle.

The IOIO tab connects to our IOIO-OTG device, sets up a digital output for controlling the LED. Every 500 milliseconds it toggles the status of the LED as well as the lightOn variable used to change the color of the rectangle on the screen.

Example Code

SIKIO_C0_Quickstart

Creating our IOIO instance:

```
// Allows us to interface with hardware new

new SikioManager(this).start();
```

Drawing the background and rectangle:

```
background(255,0,0); // R,G,B

rect(width/2, height/2, 100,100); // X, Y (position), width, height
```

Changing color of rectangle based on variable lightOn:

```
if (lightOn == true)

{
        fill(255); // white
}
```

IOIO_Tab

Creating a digital output for our LED, setting it as an output and turning it on:

```
DigitalOutput led; //DigitalOutput type for the onboard led

led = ioio.openDigitalOutput(IOIO.LED_PIN);

led.write(false); // for the onboard LED, false turns it on, true turns it off
```

Changing the lightOn variable so that the rectangle changes color:

```
lightOn = !lightOn;
```

Waiting for half a second before running the loop again (allows your phone to run other apps):

```
Thread.sleep(500);
```

Troubleshooting:

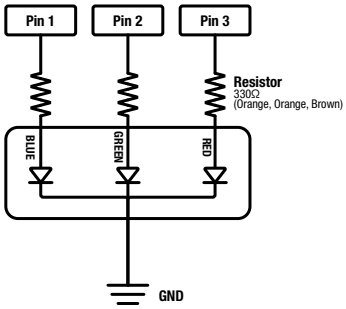
LED/Rectangle not blinking

Consider any/all of the following:

- Restart your IOIO-OTG by unplugging and replugging in power.
- For newer devices, turn off USB debugging. Make sure app is closed, then connect your Android to the IOIO-OTG. An open accessory dialogue should pop up. Select the quickstart app and "just once" to run it.
- For older devices, try simply restarting the app.
- See online wiki for additional help.

READY TO GET STARTED?

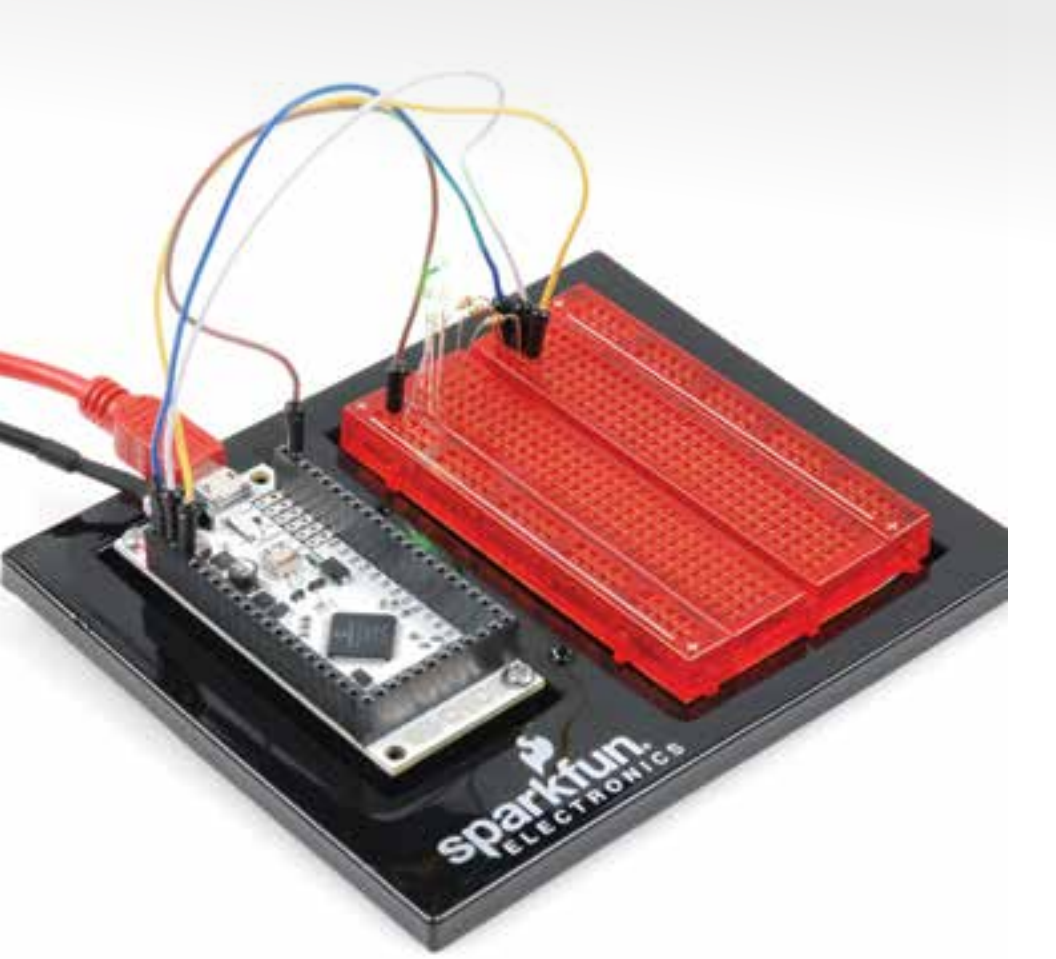
Let's get your IOIO-OTG working with external hardware

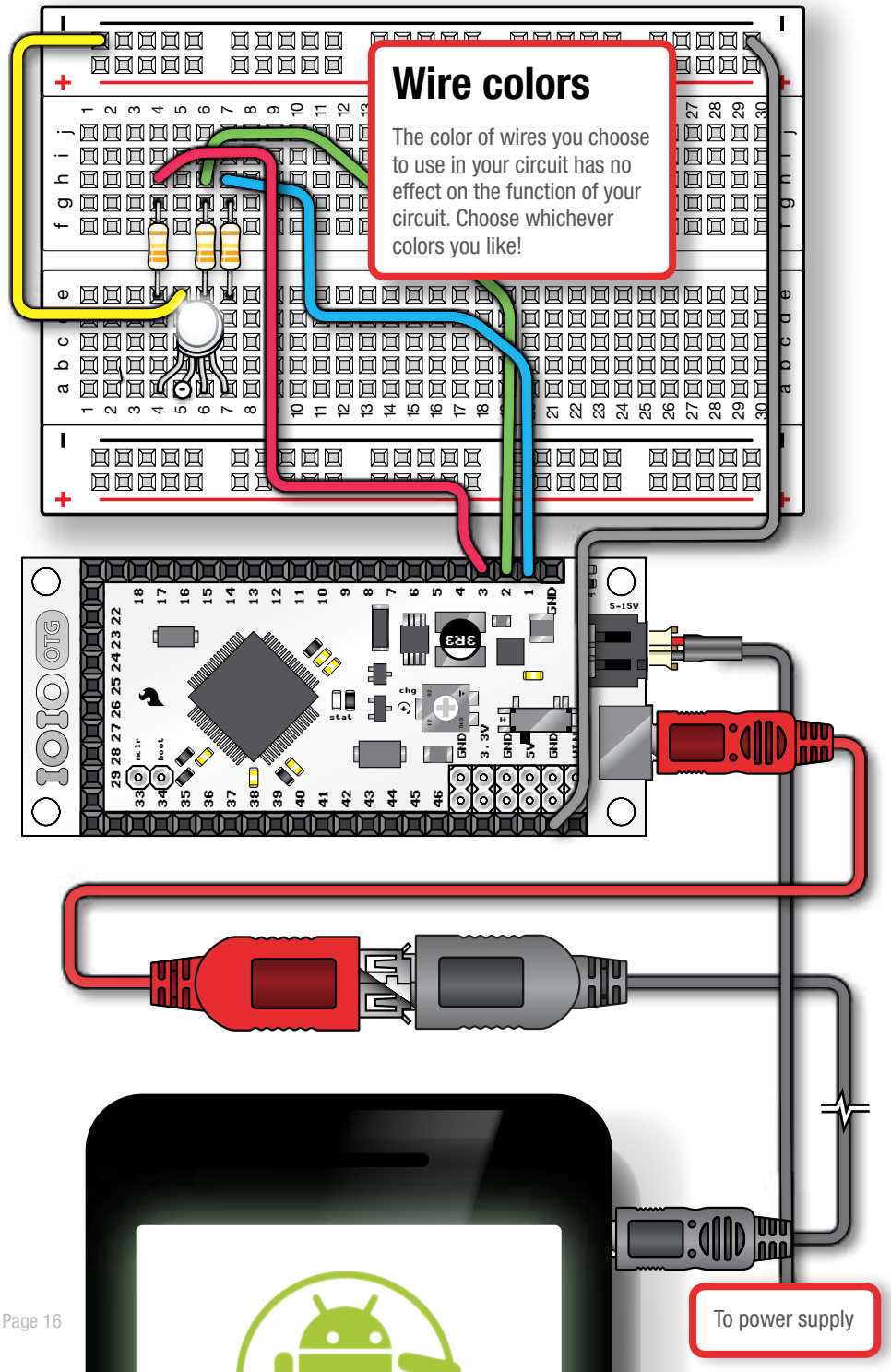


RGB LED

1

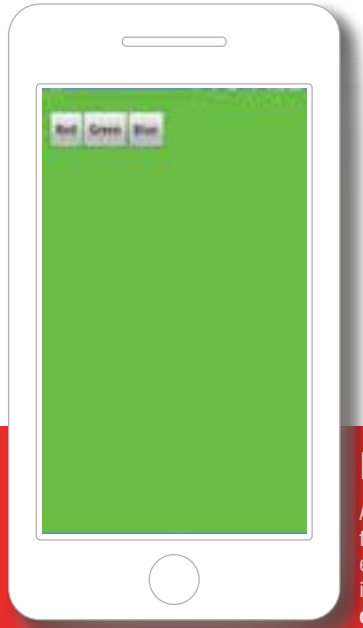
We all like things that blink. In this circuit, you will be controlling an RGB (red-green-blue) LED from your Android app. The RGB LED contains three different color-emitting diodes that can be independently controlled with *output* pins from the IOIO-OTG.





⚠ Parts marked with this symbol are polarized. Special attention should be paid to their orientation.

Component		IOIO-OTG	BREADBOARD			
5mm RGB LED ⚠			a4	a5	a6	a7
330Ω Resistor			e4	g4		
330Ω Resistor			e6	g6		
330Ω Resistor			e7	g7		
Jumper Wire			e5	(–)		
Jumper Wire		Pin 1	h7			
Jumper Wire		Pin 2	h6			
Jumper Wire		Pin 3	h4			
Jumper Wire		Gnd	(–)			



What You Should See

You will be presented with three on-screen buttons that control each LED. When you hit a button, the output pin on the IOIO-OTG gets pulled up to 3.3V, the corresponding LED turns on, and the background changes to the selected color. Simple as that.

Need more?

Additional resources that explain the code's functionality for this particular application exist in our more extensive online wiki. This information can be found at **sparkfun.com/SIKIO**

Want to get into the code?

With your circuit assembled and code uploaded to your phone, your circuit is complete. If you have installed the necessary software to edit the sketch, now would be a good time to adjust some of the parameters in the code and fine-tune the way the circuit functions. Remember – if you make a mistake, you can always download the original again and start over.

Software installation instructions can be found on page 3

Do this:

- Make the RGB LED turn purple.
- Make the RGB LED turn orange.

Try that:

- Make the RGB LED cycle through the visible spectrum (rainbow) of colors. (hint: use HSV ->RGB)
- Make the screen color the same as the LED color.
- Try moving one or more of thee buttons' position on screen.

Hint: 10 and 40 are X and Y positions on the screen, "Red" is the button label.

redButton = new APButton(10, 40, "Red");

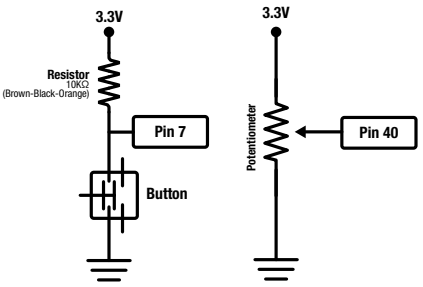
Troubleshooting:

LED remains dark or shows incorrect color

Double check the wiring and that the LED is not placed in the breadboard backward. It's easy to make a simple mistake.

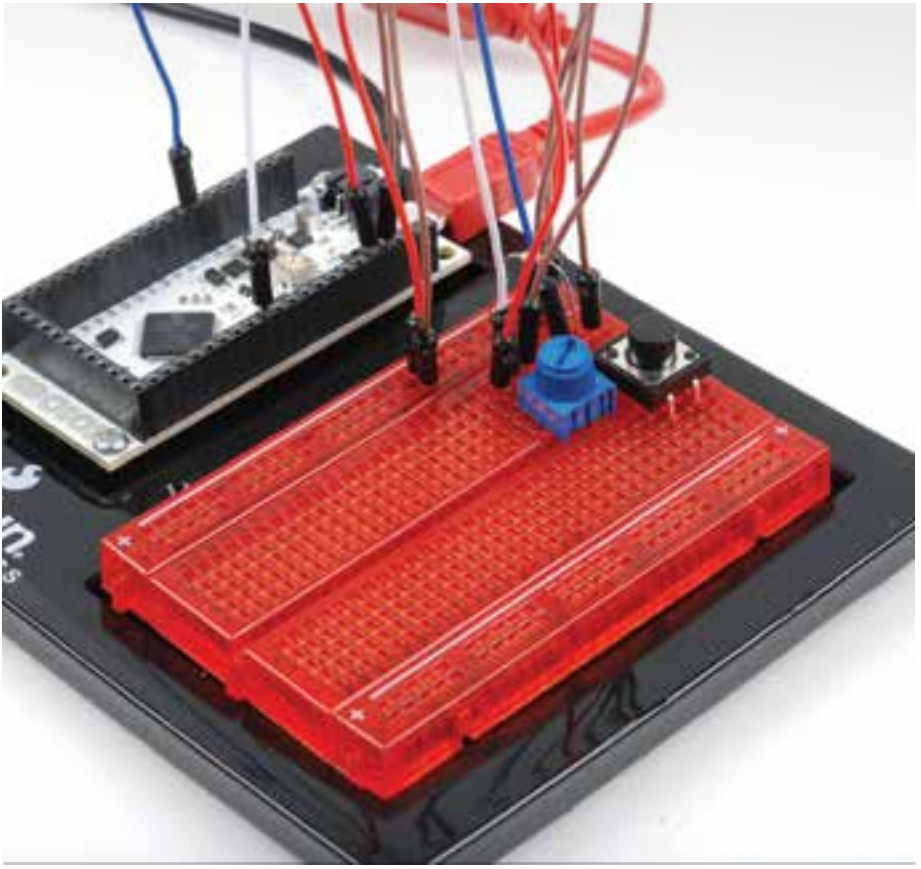
Make sure the LED is not plugged in backward. The longest pin should be connected to ground.

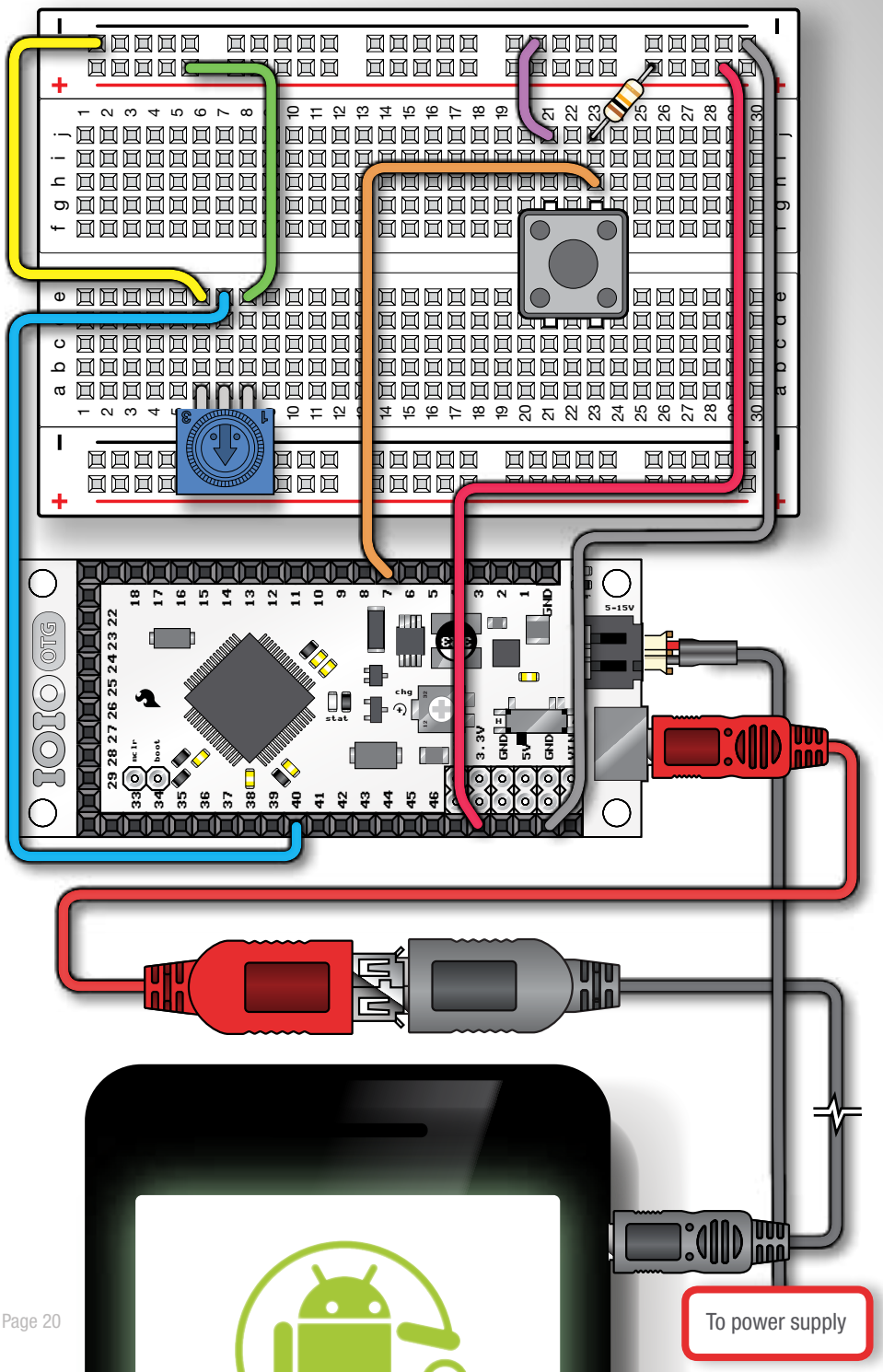
2



Button and Potentiometer

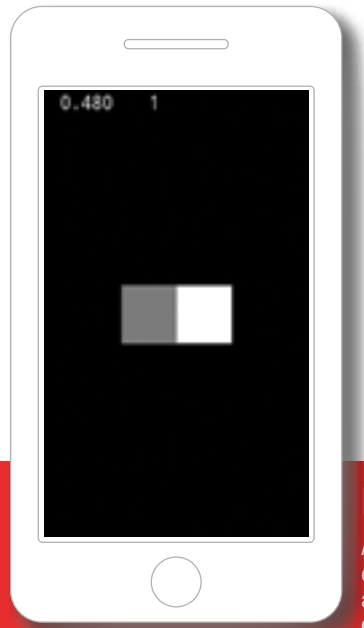
In the first example, we showed how to use *output* control to toggle an LED. What if you want your app to read some type of an *input*? In this example, the IOIO-OTG board and app will read a button-press and potentiometer value, then visually display the results. The button creates either 3.3V or 0V (GND) and is read as a *digital input* pin on the IOIO-OTG. The potentiometer is a variable resistor, which creates a voltage value between GND and 3.3V and is read by an *analog input* pin on the IOIO-OTG.





CIRCUIT 2: BUTTON AND POTENTIOMETER

Component		IOIO-OTG	BREADBOARD			
Potentiometer			a8	a7	a6	
Button			g21	g23	d21	d23
10K Resistor			j23	(+)		
Jumper Wire		Pin 7	h23			
Jumper Wire			j21	(–)		
Jumper Wire			e6	(–)		
Jumper Wire			e8	(+)		
Jumper Wire		Pin 40	e7			
Jumper Wire		3.3V	(+)			
Jumper Wire		GND	(–)			



What You Should See

Two boxes are displayed on screen. The left one corresponds to the potentiometer and the right one to the button. The right box is either on or off depending on the button state. The left box varies in brightness by turning the potentiometer. Also, both scaled values are displayed above each box. 1 corresponds to 3.3V, so every 0.1 equals 0.33V.

Need more?

Additional resources that explain the code's functionality for this particular application exist in our more extensive online wiki. This information can be found at **sparkfun.com/SIKIO**

Want to get into the code?

With your circuit assembled and code uploaded to your phone, your circuit is complete. If you have installed the necessary software to edit the sketch, now would be a good time to adjust some of the parameters in the code and fine-tune the way the circuit functions. Remember – if you screw up, you can always download the original again and start over.

Software installation instructions can be found on page 3

Do this:

- Change the rectangle the button controls to an ellipse.
- Let the potentiometer change the size of the rectangle.

Hint: instead of using rect(x,y,width,height), use ellipse(x,y,width,height)

Try that:

- Display text that counts the number of button presses.

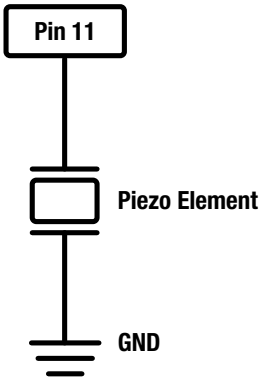
Troubleshooting:

Light not turning on

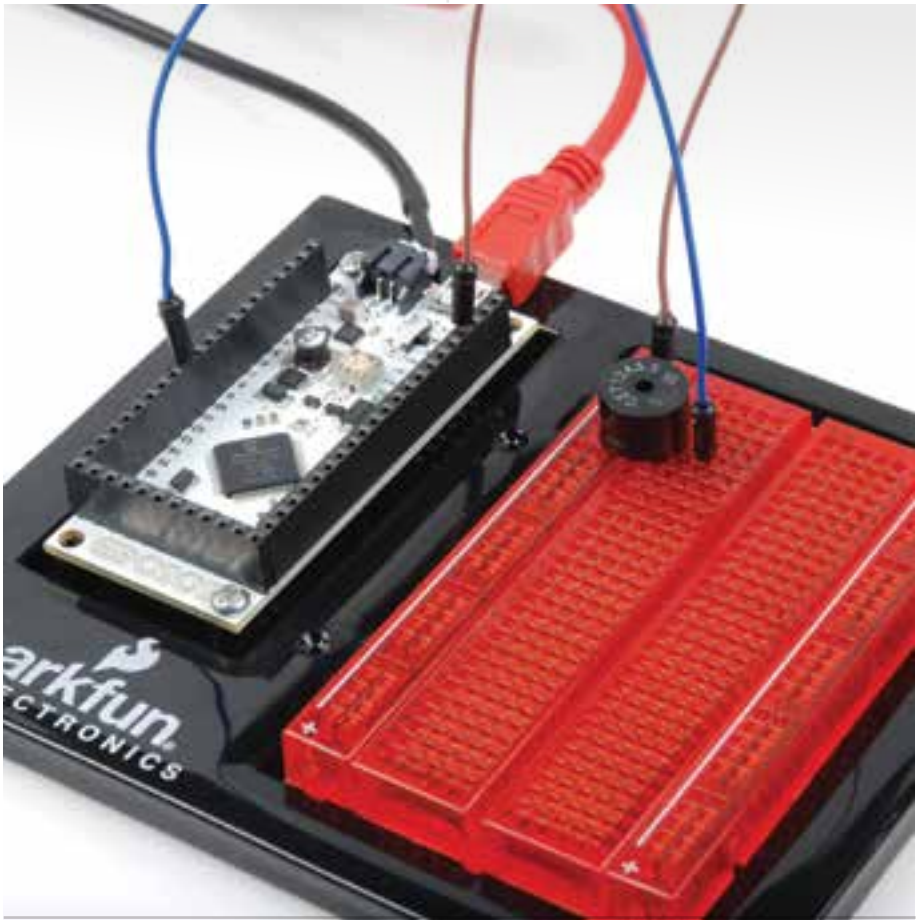
The pushbutton is square, and because of this it is easy to put it in the wrong way. Give it a 90 degree twist and see if it starts working.

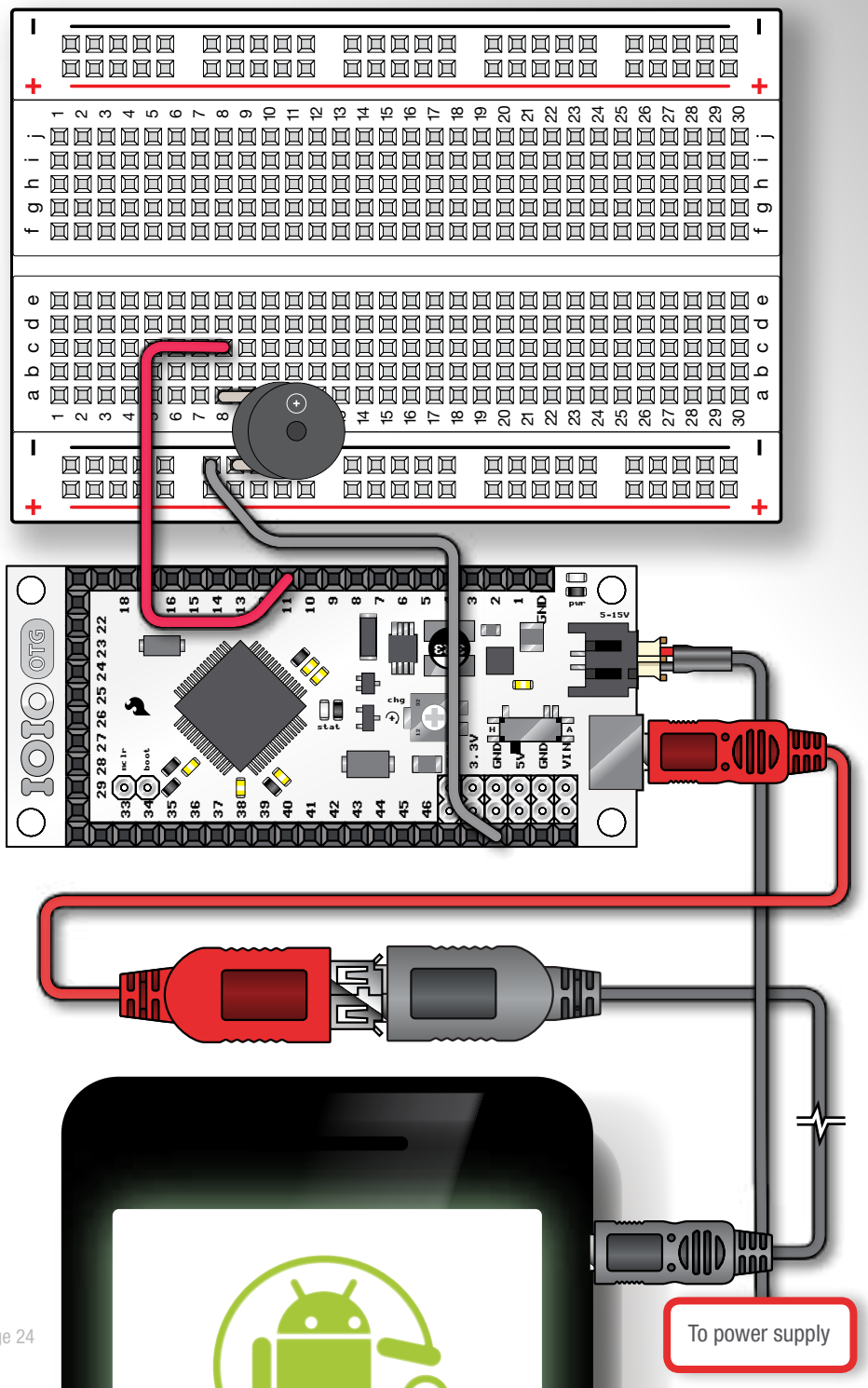
3

Buzzer



Make some noise! A buzzer is connected to a *PWM output* pin on the IOIO-OTG and GND. PWM stands for pulse width modulation, which is a fancy way of saying that the output pin turns off (GND) and on (3.3V) for a period of time at a given frequency. If you change how long the pin is on compared to how long the pin is off (this is called duty cycle and is represented by the pulse width), different sounds can be created.





CIRCUIT 3: BUZZER

⚠ Parts marked with this symbol are polarized. Special attention should be paid to their orientation.

Component		IOIO-OTG	BREADBOARD		
Buzzer ⚠			a8	(-)	
Jumper Wire		Pin 11	c8		
Jumper Wire		GND	(-)		

What You Should See

Or rather, what you should hear! In this example, we have created a keyboard that detects on-screen button presses. Each key plays a different note corresponding to a change in the PWM frequency.

Need more?

Additional resources that explain the code's functionality for this particular application exist in our more extensive online wiki. This information can be found at **sparkfun.com/SIKIO**

Want to get into the code?

With your circuit assembled and code uploaded to your phone, your circuit is complete. If you have installed the necessary software to edit the sketch, now would be a good time to adjust some of the parameters in the code and fine-tune the way the circuit functions. Remember – if you screw up, you can always download the original again and start over.

Software installation instructions can be found on page 3

Do this:

- Change the notes to a higher or lower octave.
- Try making the keyboard fit your entire screen.

Hint: The 'freq' variable is used to store the frequency of the note to be played. A note in the next higher octave is double the frequency.

Try that:

- Adjust settings so the app plays a melody on startup.
- Even better, use buttons to select between a few melodies to play.

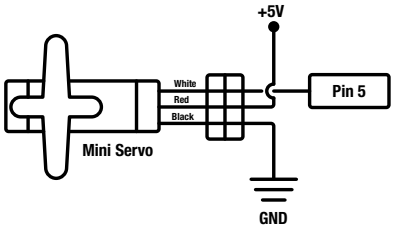
Troubleshooting:

No sound

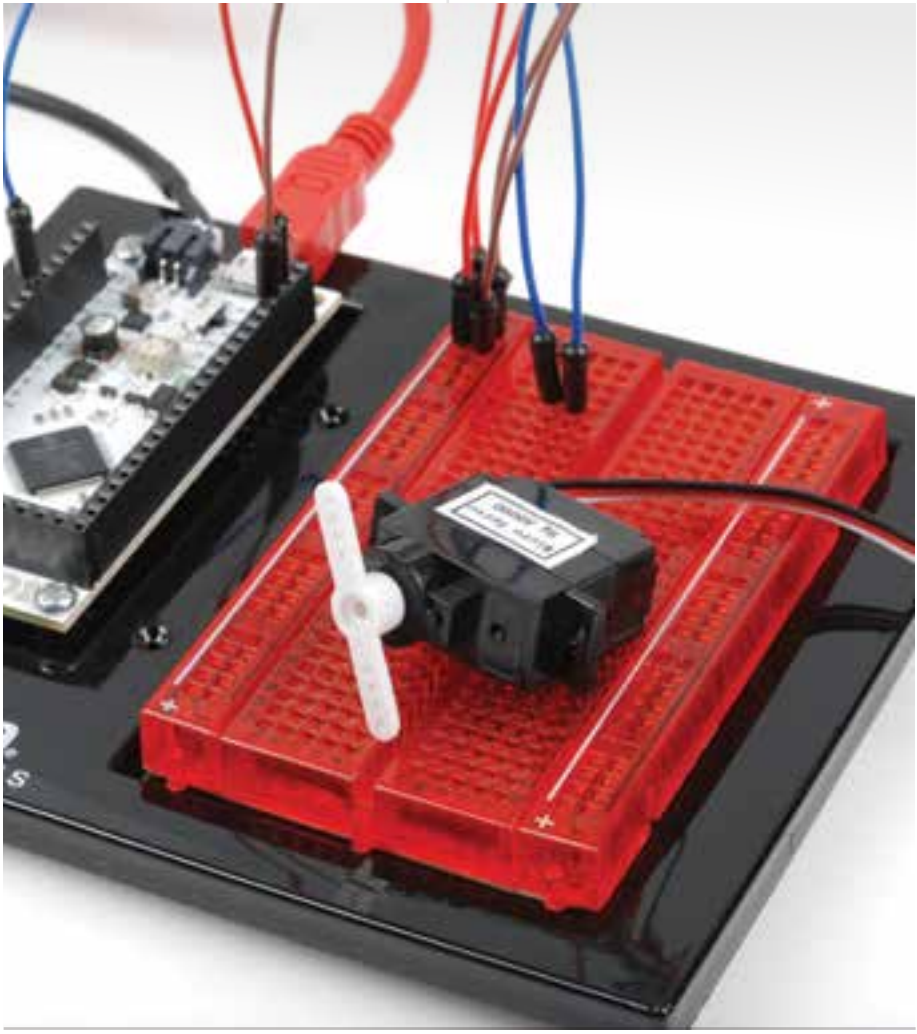
Given the size and shape of the piezo element, it is easy to miss the right holes on the breadboard. Double check the placement and make sure it's not installed backward.

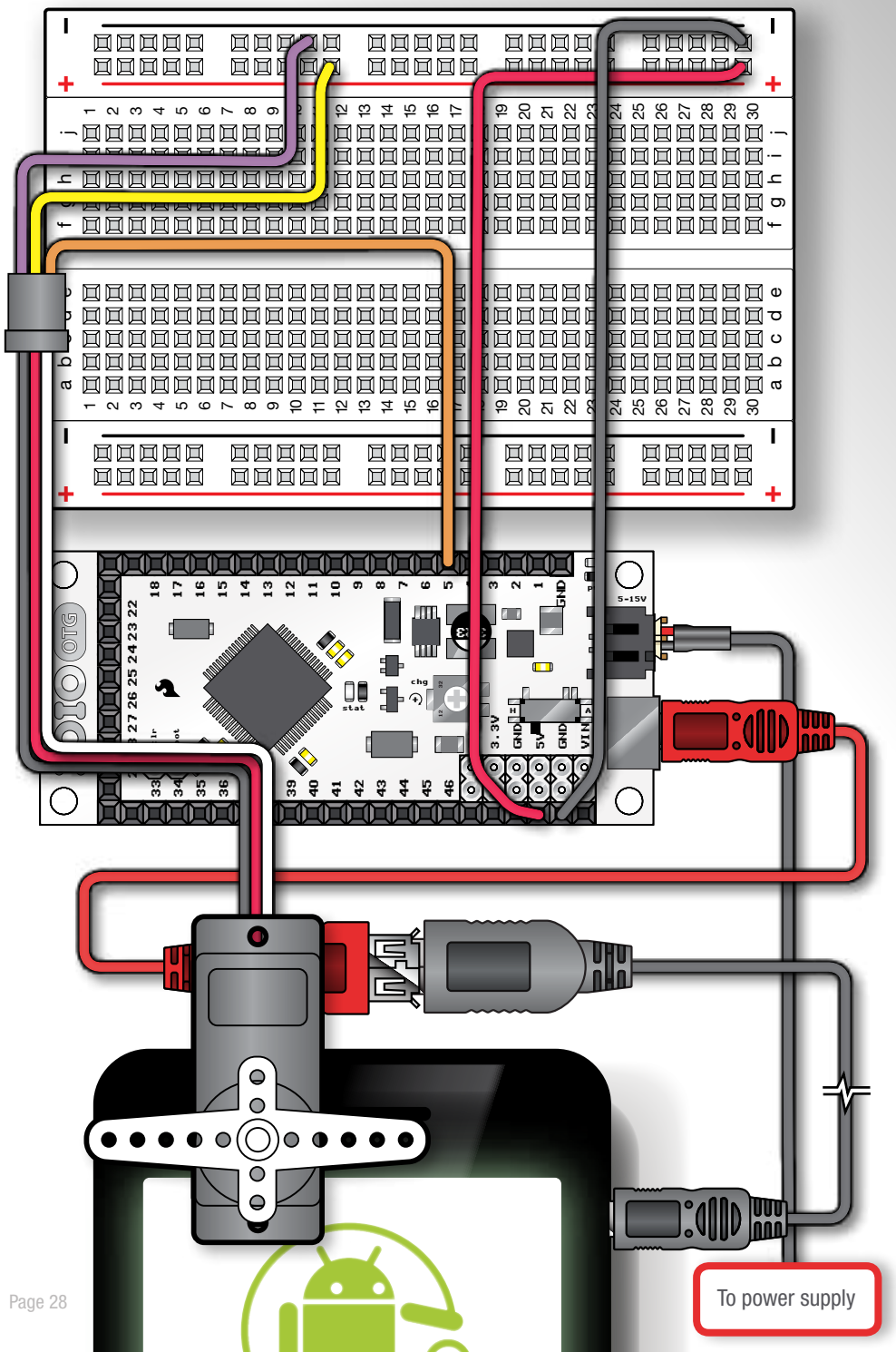
4

Servo



Another use for *PWM output* control is controlling a servo.
A servo is a type of motor that allows you to accurately control the position of the rotation. The servo needs power (be sure to use 5V, not 3.3V), GND, and a PWM output pin from the IOIO-OTG. When the duty cycle on the PWM output pin is changed, the servo position changes.





CIRCUIT 4: SERVO

Component		IOIO-OTG	BREADBOARD
Servo Motor			
Jumper Wire			(+)
Jumper Wire			(-)
Jumper Wire		Pin 5	
Jumper Wire		Gnd	(-)
Jumper Wire		5V	(+)

5V Circuit!

Heads up! Most of the circuits in the IOIO-OTG SparkFun Inventor's Kit require 3.3 volts. This circuit requires 5 volts.

What You Should See

You should be able to drag the on-screen red box left and right with your finger to control the servo position. Also, the duty cycle and a few other metrics are displayed.

Need more?

Additional resources that explain the code's functionality for this particular application exist in our more extensive online wiki. This information can be found at **sparkfun.com/SIKIO**

Want to get into the code?

With your circuit assembled and code uploaded to your phone, your circuit is complete. If you have installed the necessary software to edit the sketch, now would be a good time to adjust some of the parameters in the code and fine-tune the way the circuit functions. Remember – if you screw up, you can always download the original again and start over.

Software installation instructions can be found on page 3

Do this:

- Change the size and color of the text.

Hint: Color of objects on the screen is changed using the fill(red,green,blue) command. Font type and size is set by the createFont("font",size) function.

Try that:

- Adjust settings so the servo responds to the y axis of the on-screen box instead of the x axis.

Troubleshooting:

Servo not twisting

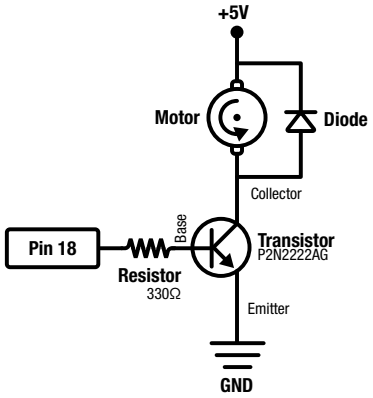
Even with colored wires, it is still easy to wire a servo incorrectly. This may be the case.

Still not working

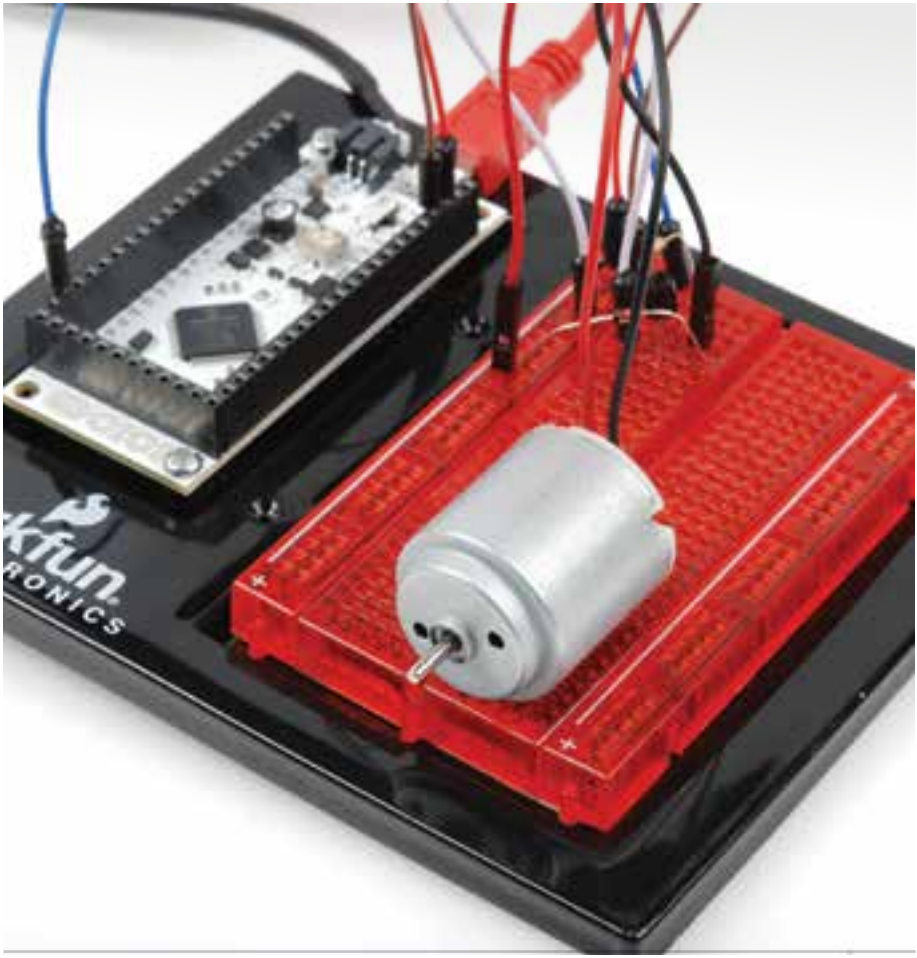
Double check power is connected to 5V, not 3.3V.

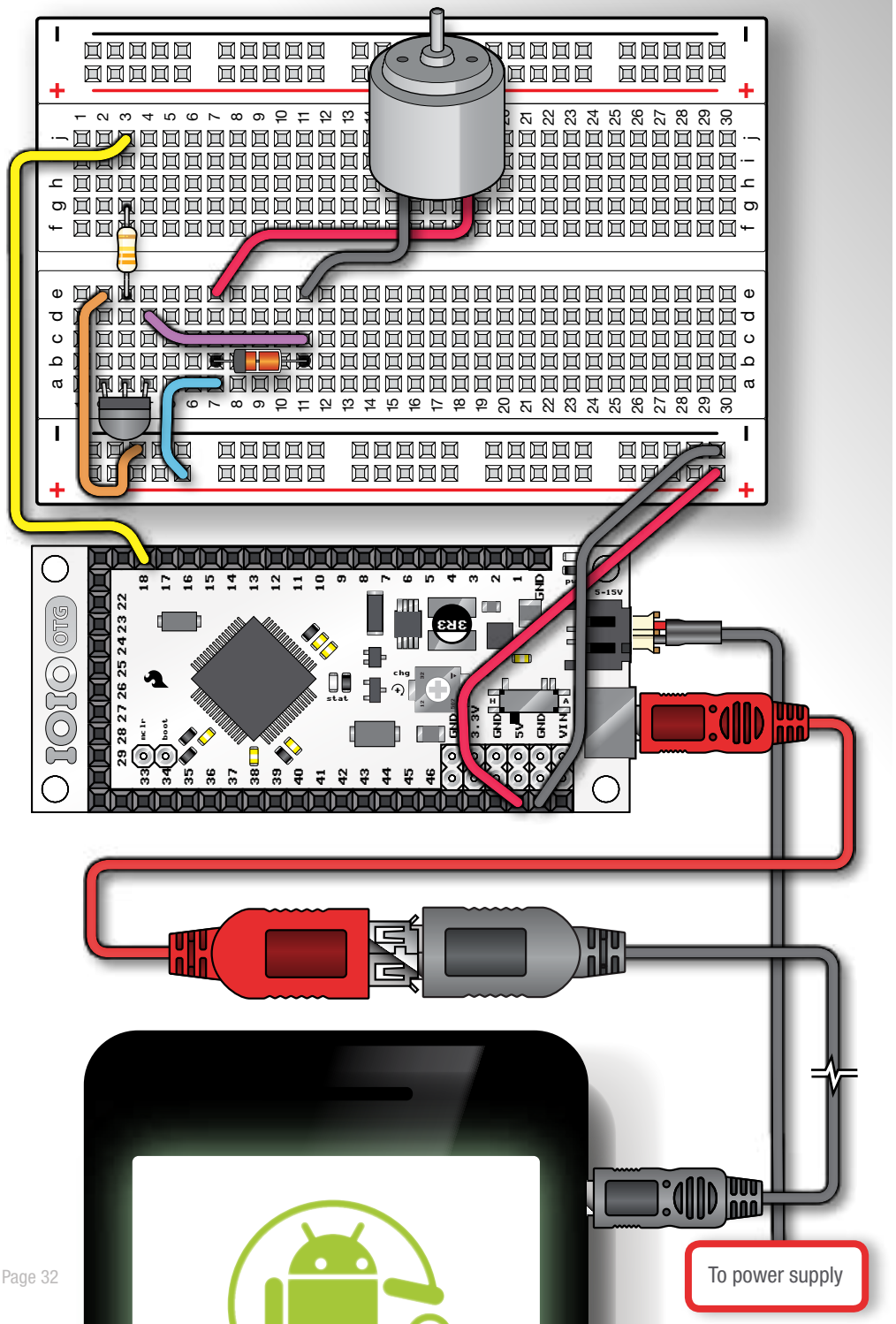
5

Motor



Motors can be used in many applications, but often require some hardware to get up and running. This example uses a transistor, which acts as a switch controlled by an *output* pin on the IOIO-OTG to turn the motor on and off. Since the motor needs a lot of current to operate, the transistor allows an output pin on the IOIO-OTG to turn on (5V) and off (GND) current to the motor.





⚠ Parts marked with this symbol are polarized. Special attention should be paid to their orientation.

Component		IOIO-OTG	BREADBOARD			
⚠ Transistor			E a2	B a3	C a4	
Motor			e7	e11		
⚠ Diode			b11	b7		
330Ω Resistor			e3	g3		
Jumper Wire		Pin 18	j3			
Jumper Wire			d4	c11		
Jumper Wire			e2	(-)		
Jumper Wire			a7	(+)		
Jumper Wire		5V	(+)			
Jumper Wire		Gnd	(-)			

What You Should See

Here you can use a text field to enter the number of seconds to turn on the motor. When you tap on the text field, you should get a number pad. Enter the running time in seconds and as soon as you hit done, the motor should start spinning.

Need more?

Additional resources that explain the code's functionality for this particular application exist in our more extensive online wiki. This information can be found at **sparkfun.com/SIKIO**

Want to get into the code?

With your circuit assembled and code uploaded to your phone, your circuit is complete. If you have installed the necessary software to edit the sketch, now would be a good time to adjust some of the parameters in the code and fine-tune the way the circuit functions. Remember – if you screw up, you can always download the original again and start over.

Software installation instructions can be found on page 3

Do this:

- Change the location of the text field.

Hint: Change the APEditText(x,y,width,height) function.

Try that:

- Make a simple on/off screen button for the motor.
- Make several buttons, each of which turns on the motor for a different number of seconds.

Troubleshooting:

Motor not spinning

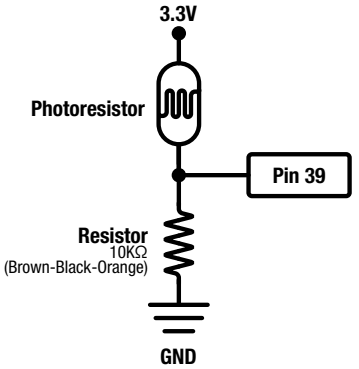
If you sourced your own transistor, double check with the data sheet that the pinout is compatible with a P2N2222AG (many are reversed).

Still no luck

If you sourced your own motor, double check that it will work with 5 volts and that it does not draw too much power.

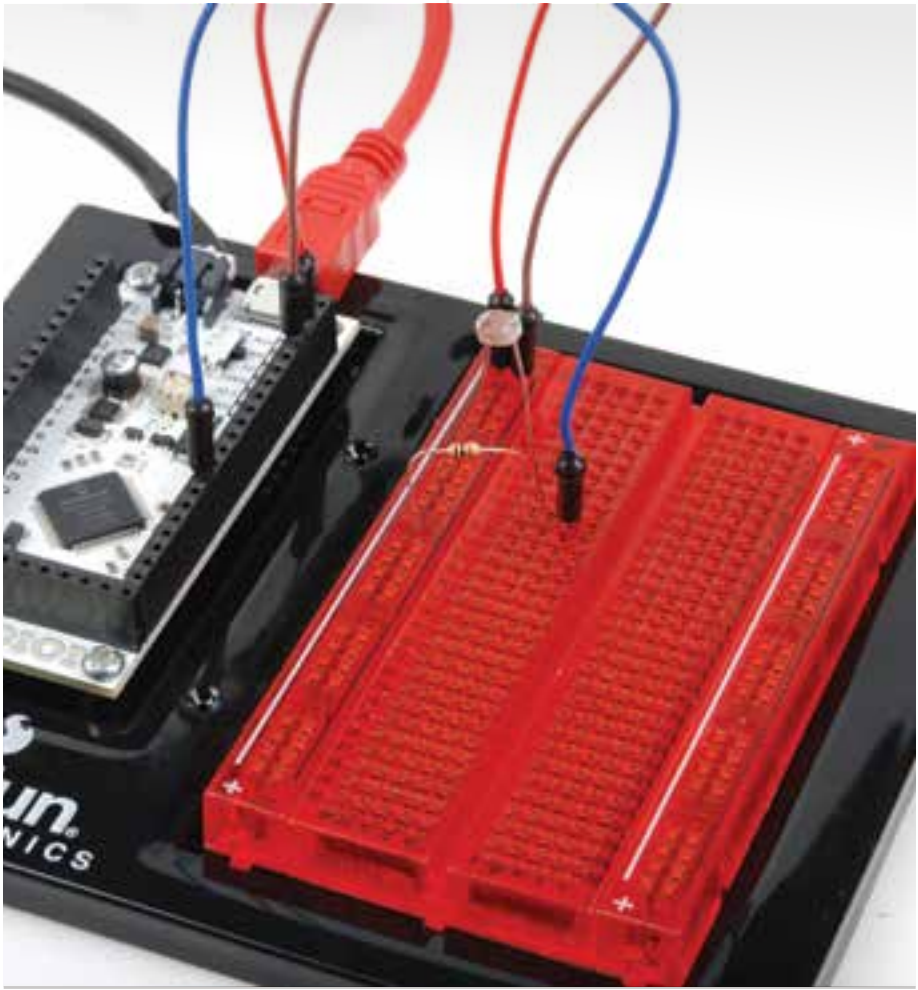
Check the orientation of the diode, making sure it is not installed backward.

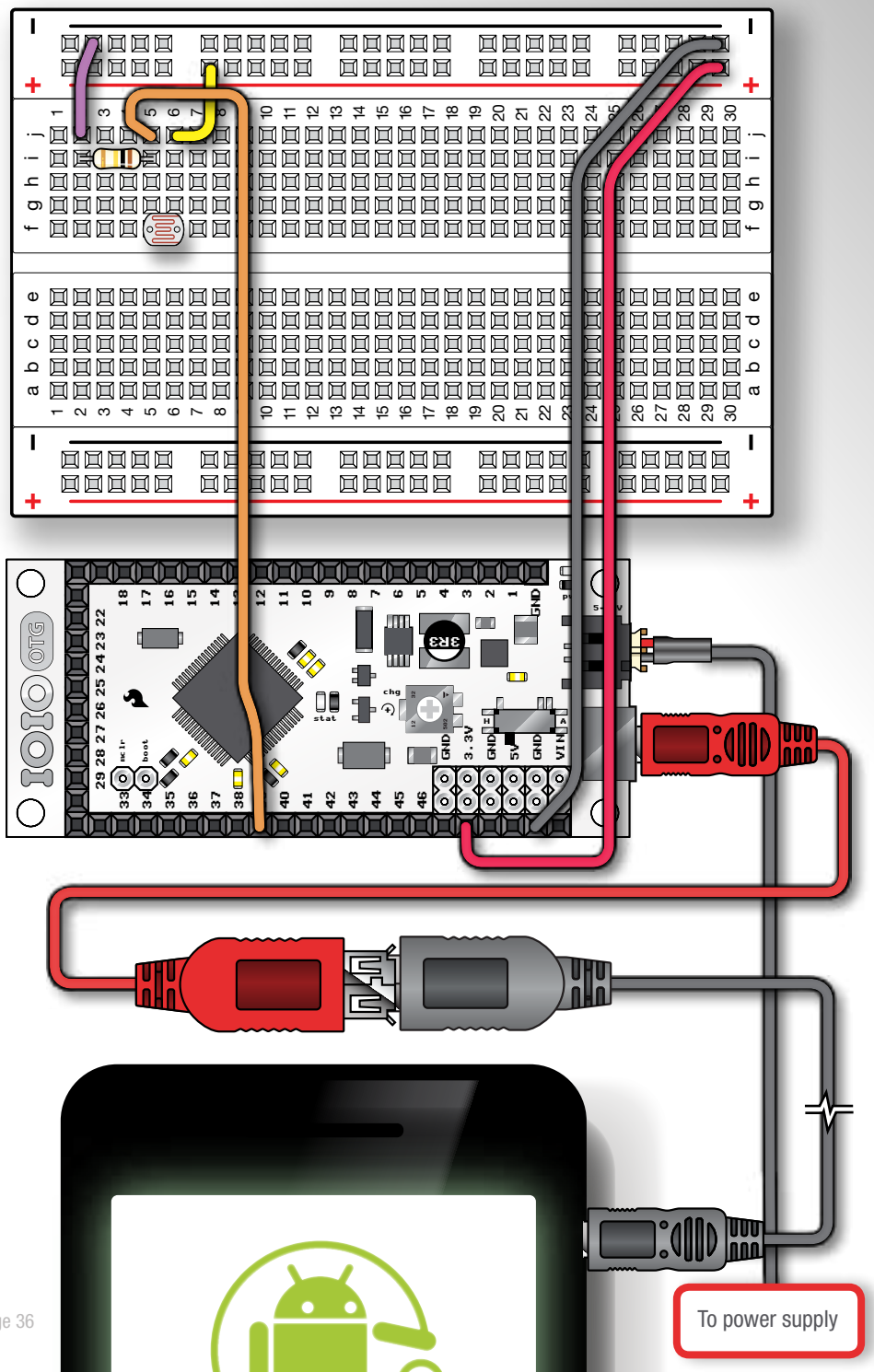
6



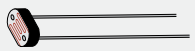
Photoresistor

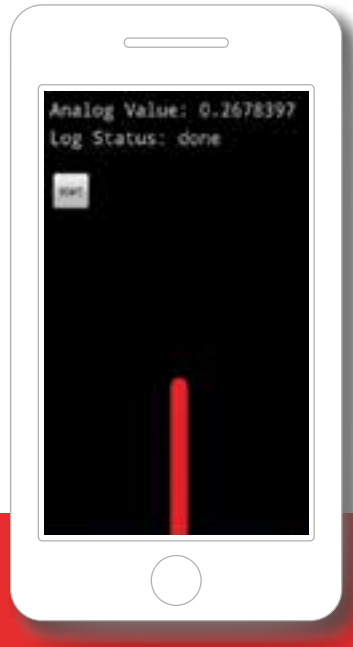
If you want to measure light, an easy way to do it is with a photoresistor. As the light intensity changes, the resistance across the photoresistor changes and can be read by an *analog input* pin on the IOIO. The hardware consists of a common voltage divider, where one of the resistors is the photoresistor.





CIRCUIT 6: PHOTORESISTOR

Component		IOIO-OTG	BREADBOARD	
Photoresistor			f5	f6
10KΩ Resistor			i2	i5
Jumper Wire			j6	(+)
Jumper Wire			j2	(-)
Jumper Wire		Pin 39	j5	
Jumper Wire		3.3V	(+)	
Jumper Wire		GND	(-)	



What You Should See

TEST: The Android app displays a vertical red line acting as a bar graph. When the intensity of light increases, the line gets smaller. There is also a button on the screen that you can hit to log data to the SD card. A file called sensorValues.txt is created in the Sikio folder in the root directory of your storage device and contains the logged analog values from the photoresistor.

Need more?

Additional resources that explain the code's functionality for this particular application exist in our more extensive online wiki. This information can be found at **sparkfun.com/SIKIO**

Want to get into the code?

With your circuit assembled and code uploaded to your phone, your circuit is complete. If you have installed the necessary software to edit the sketch, now would be a good time to adjust some of the parameters in the code and fine-tune the way the circuit functions. Remember – if you screw up, you can always download the original again and start over.

Software installation instructions can be found on page 3

Do this:

- Instead of drawing a line, use the light input to control brightness of the background.

Hint: Use a combination of the following: the background(red,green,blue) function, the map(value,initial_min,initial_max,final_min,final_max) function and the variable thread1.potVal which corresponds to the photoresistor input.

Try that:

- Use variables to track highest and lowest record amount of light.

Troubleshooting:

It isn't responding to changes in light

Given that the spacing of the wires on the photoresistor is not standard, it is easy to misplace it. Double check it's in the right place.

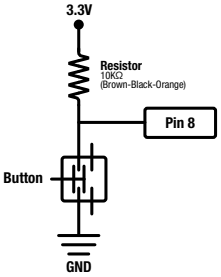
Still not quite working

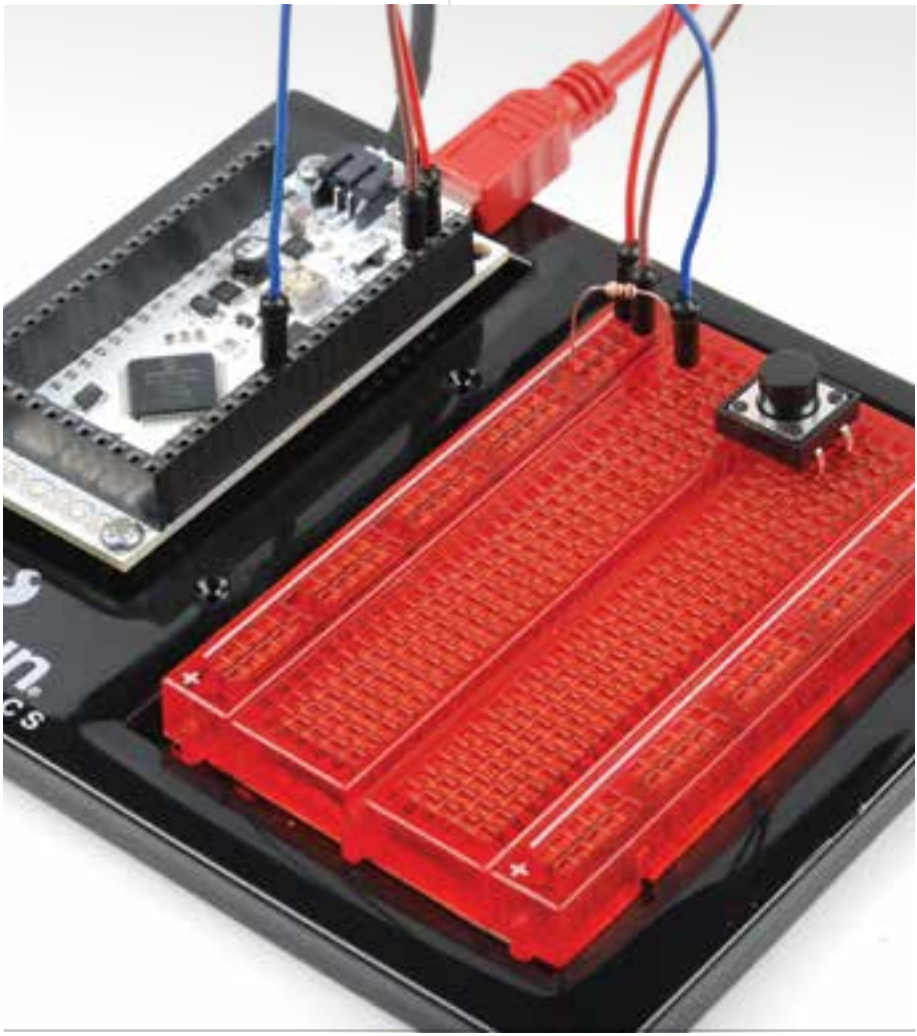
You can also try a flashlight or try covering the photoresistor with your fingers to test if you can see a change in the sensor readings.

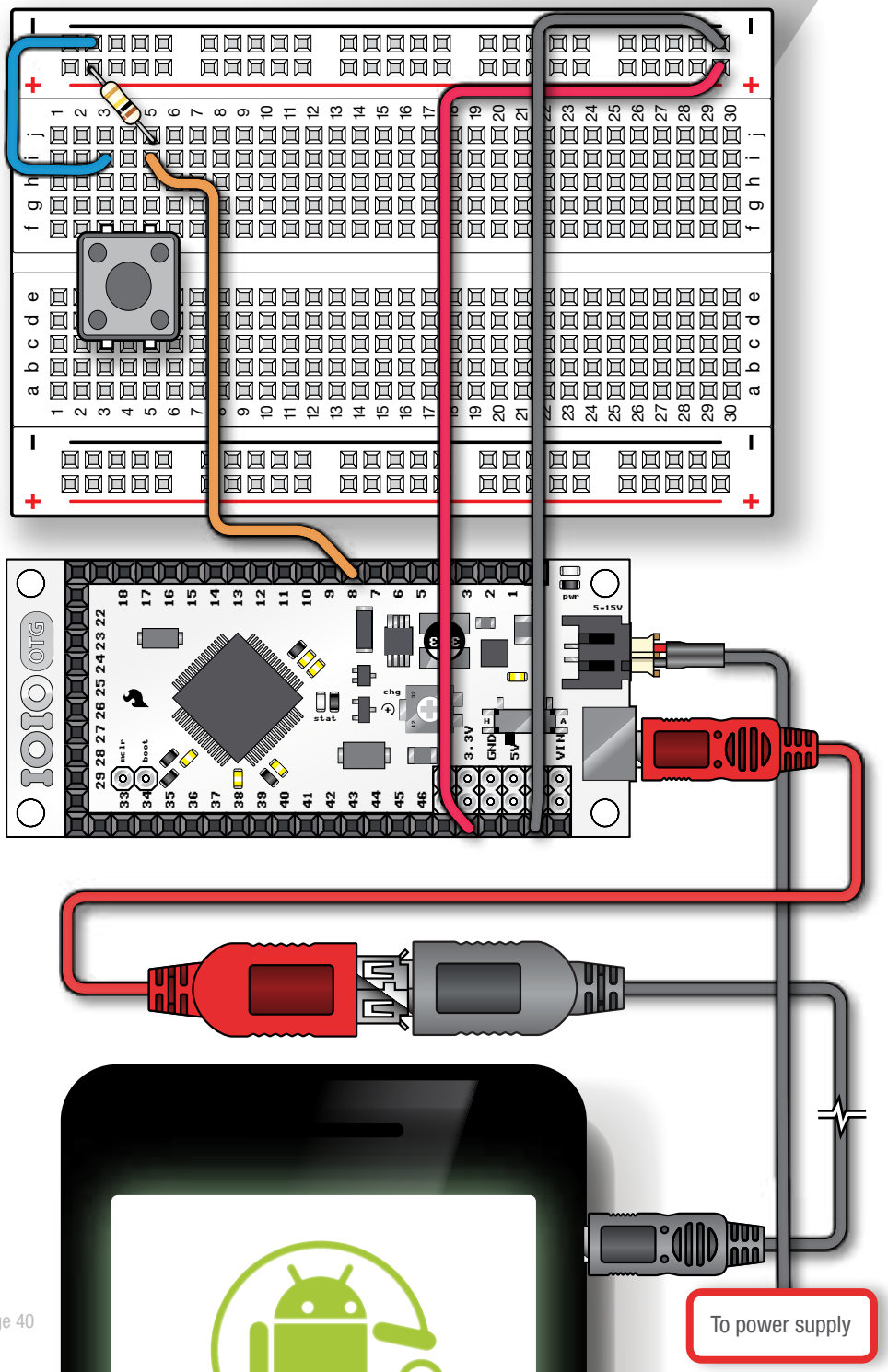
Camera

7

This example uses a push button to take a picture with your Android. All we need to do is read a button press with an *input* pin on the IOIO-OTG. When the button is hit, a picture is taken and saved to the internal memory of your Android.

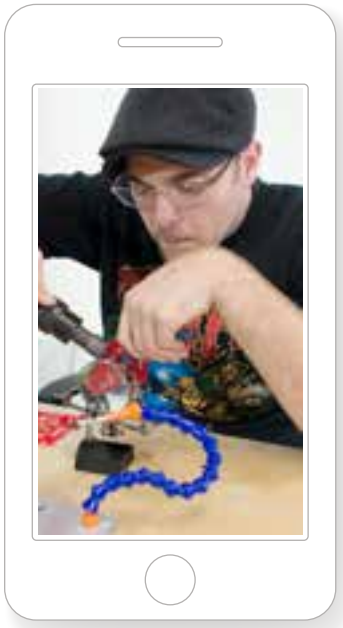






CIRCUIT 7: CAMERA

Component		IOIO-OTG	BREADBOARD			
Button			c3	c5	f3	f5
10KΩ Resistor			j5	(+)		
Jumper Wire			i3	(–)		
Jumper Wire		Pin 8	i5			
Jumper Wire		3.3V	(+)			
Jumper Wire		GND	(–)			



What You Should See

When the app starts, you should see the live image on your display. To take a picture, you can hit the external push button. The image file, called 'image.jpg', is saved in the Sikio folder which is found in the root directory of your Android's storage device.

Need more?

Additional resources that explain the code's functionality for this particular application exist in our more extensive online wiki. This information can be found at **sparkfun.com/SIKIO**

Want to get into the code?

With your circuit assembled and code uploaded to your phone, your circuit is complete. If you have installed the necessary software to edit the sketch, now would be a good time to adjust some of the parameters in the code and fine-tune the way the circuit functions. Remember – if you screw up, you can always download the original again and start over.

Software installation instructions can be found on page 3

Do this:

- Use a potentiometer or other analog input device to use as the photo trigger.

Hint: Have a threshold value for which a photo is taken if the reading of the input exceeds that. Also, consider having it only check the value once every 10 seconds (or longer) so that it doesn't just constantly take pictures. You could do this roughly by counting its sleep cycles (which are 100 ms each).

Try that:

- Use a sensor (like an IR sensor) to trigger the shutter instead of a button.
- Use a potentiometer to control the brightness of the photo being taken.

Troubleshooting:

Image not saving

The pushbutton is square, and because of this it is easy to put it in the wrong way. Give it a 90 degree twist and see if it starts working.

Image not displaying

In the camera section of the code, there are some lines you may need to comment out depending on what device you have. See code comments and the C7 wiki for more detail.

SIKIO Wiki Homepage: http://www.sparkfun.com/SIKIO

SparkFun IOIO-OTG Product Page: https://www.sparkfun.com/products/11343

Ytai's Official IOIO Resources: https://github.com/ytai/ioio/wiki

Processing for Android Wiki: http://wiki.processing.org/w/Android

Processing Coding Reference: http://processing.org/reference/

Android SDK Home: http://developer.android.com/sdk/index.html

Android Coding Reference: http://developer.android.com/reference/packages.html

Unleash the Power of your Android

This kit will guide you through experiments of varying difficulty as you learn all about embedded systems, physical computing, programming and more! Explore the power of the IOIO-OTG platform and your Android device.

The SparkFun Inventor's Kit for IOIO-OTG requires a knowledge of programming. An Internet connection and Android device is also required.

You will also learn to assemble 7 basic physical electronic circuits, but **no soldering is required.** No previous experience is necessary!

KIT INCLUDES

- IOIO-OTG Board
- Breadboard
- Instruction booklet
- Small servo
- RGB LED
- Temperature sensor
- DC motor
- Push button switches
- Potentiometer
- Photoresistor
- Transistor
- Jumper wires
- Micro B USB cable
- Female A to Micro A USB adapter
- Signal diode
- 10k Ohm resistors
- 330 Ohm resistors
- Piezo buzzer
- Baseplate
- 5V Wall Wart
- Phillips Screwdriver