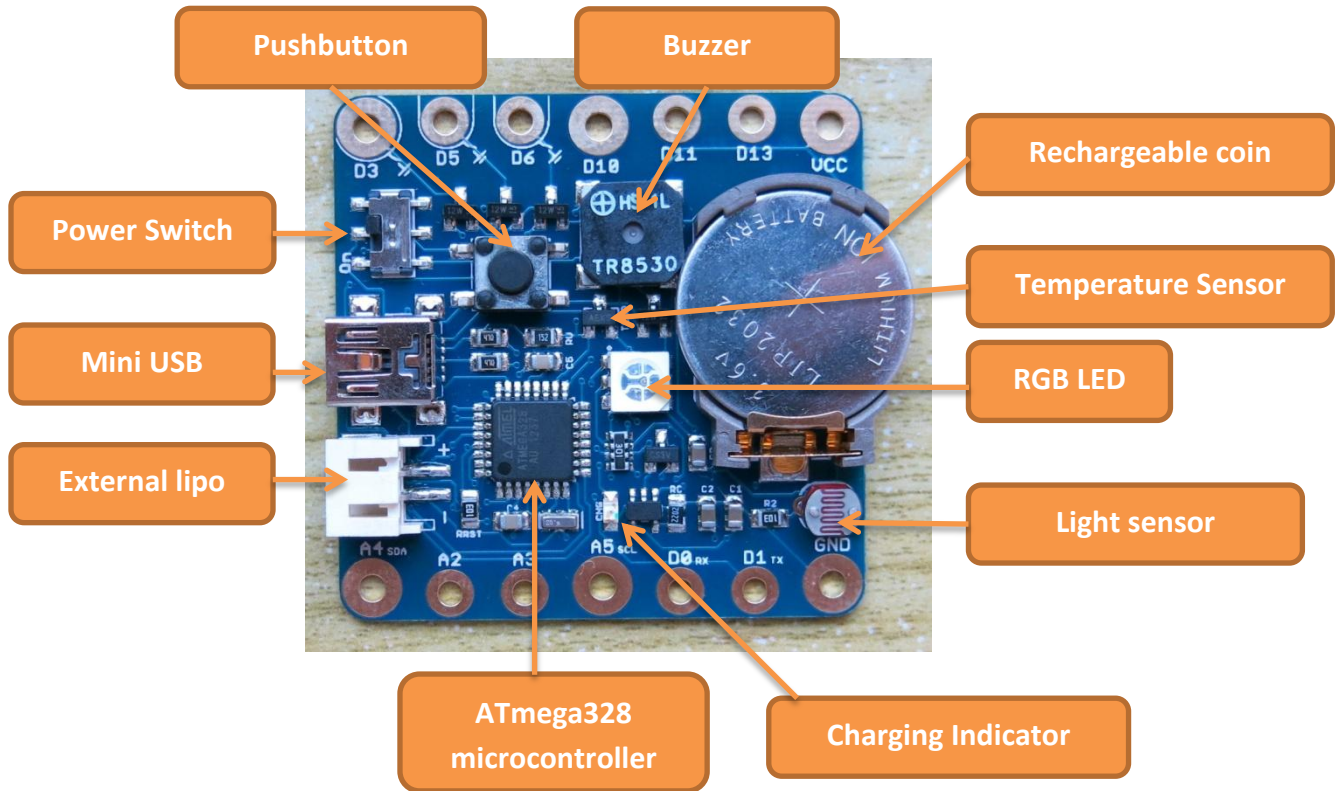


SquareWear 2.0 User Manual

- **Specification**

- ATmega328 microcontroller @ 12MHz with USBasp bootloader
- Built-in 20mm rechargeable lithium coin battery
- MCP73831 lithium charger
- Built-in RGB LED, buzzer, temperature sensor, light sensor, mini-USB port
- External lipo battery jack

- **Hardware Interface**



- **Power Options**

- SquareWear 2.0 has a built-in 20mm rechargeable lithium coin battery (type LIR2032, 40mAH), which is usually good enough for starter projects.
- You can also power it by an external Lithium-Polymer (LiPo) battery (such as <https://www.sparkfun.com/products/341>). ***If using external LiPo, please remove the built-in coin battery to avoid the batteries charging each other.***
- The build-in Lipo charger can charge either battery at 40mA per hour. This will fully charge the built-in coin in about 1-2 hour. To charge, plug in a mini-USB cable to the USB port. The charging LED will turn on. Once fully charged, the charging LED will turn off.

- **Pin Names and Functions**

- SquareWear 2.0 is based on ATmega328, which is the same microcontroller in standard Arduino. Therefore the pin names and functions are shared with standard Arduino.
- **Digital I/O pins:** the board has 8 available digital I/O pins
 - **D0** (also Arduino's serial RX pin)
 - **D1** (also Arduino's serial TX pin)
 - **D3** (supports PWM, power sink only)
 - **D5** (supports PWM, power sink only)
 - **D6** (supports PWM, power sink only)
 - **D10** (supports PWM)
 - **D11** (supports PWM)
 - **D13** (internally connected to blue LED)

***** PWM (Pulse Width Modulation), also referred to as analog output, means you can adjust the voltage level of the pin, which can be used to change the brightness of LED etc. All PWM pins can function like standard digital I/O pins, but additionally you can use Arduino's [analogWrite](#) function.**

As described above, D13 is internally connected to the blue channel of the RGB LED. Setting this pin HIGH will light up the LED.

***** Power Sink Pins**

D3, D5, D6 also support PWM, and they function additionally as **Power Sinks** only. This means setting these pins to logical high will connect them to Ground, and setting them low will disconnect them from Ground. Therefore, in order to use them, such as controlling an LED, you need to connect the positive lead of the LED to VCC, and negative lead to one of these Power Sink pins. Then by setting the pin high or low, you can control the LED.

The main advantage of Power Sink pins is that they can run a very high amount of current, so your LEDs will look very bright, your speaker will sound loud, etc.

- **Analog Input pins:** the alpha version has 4 analog input pins
 - **A2**
 - **A3**
 - **A4** (this is also the i2C SCL pin)
 - **A5** (this is also the i2C SDA pin)

Analog pins can be used to read an analog signal, such as sensors.

- **Build-in Components and Internally Assigns Pins:**
 - **D2:** used for USB communication
 - **D4:** push-button (also used for the microcontroller to enter bootloading mode)
 - **D7:** push-button (also used for USB communication)
 - **D8:** red channel of the RGB LED (active high)
 - **D9:** buzzer
 - **D12:** green channel of the RGB LED (active high)
 - **D13:** blue channel of the RGB LED (active high)
 - **A0:** light sensor
 - **A1:** temperature sensor

- **Sewing:** the SquareWear board has large pin holes, which allow you to sew conductive threads through them and attach the board to clothes or fabric.
- **Sew-on Snaps:** on the alpha version, there are 6 relatively large pin pads (D3, D13, VCC, A4, A5, GND). You can solder a small sew-on snap to these pads, which allow you to snap the board onto clothes or fabric, and take them off easily.

- **Programming the SquareWear 2.0**

Programming the SquareWear 2.0 is done through standard Arduino software.

- **Pre-requisites:**

- Download and install Arduino from <http://arduino.cc>. The recommended version is 1.0.5
- Unzip the attached SquareWear 2.0 package, copy **boards.txt** to the following path, and overwrite the existing boards.txt: **arduino-1.0.5/hardware/arduino**
- Copy SquareWear2 folder in the package to the following path: **arduino-1.0.5/libraries**
- If you are using Windows, you should install the USBasp driver from: <http://www.fischl.de/usbasp/>
You probably don't need to install it ahead of time. But the first time you enter program mode (as per instructions below) the system will ask you for the driver, and you can direct it to search in the unzipped driver folder.

- Run Arduino 1.0.5, in the menu, select **Tools->Boards->SquareWear 2.0**
- There are several provided example programs, which you can find in **File->Examples->SquareWear2->...**

- **Enter the program mode:**

- To upload a program, plug in a mini-USB cable to the USB port. Then you need to enter program mode. To do so, turn off the power switch, then **press the pushbutton while turn**

the power switch back on. At this point, the normal program will stop running, and the controller will present itself as a USBasp programmer to the host computer.

- In Windows, once you enter program mode, the system will ask you for driver if you have not installed the driver previously.
 - If you are using Linux, you need to create a file in `/etc/udev/rules.d/` to give permission to the USB device. Alternatively, run Arduino in sudo mode.
 - Next, press the Upload button in the Arduino software will upload the program to the board. Then the program will start running immediately.
 - If the Arduino output reports the following error:
`avrdude: error: could not find USB device "USBasp" with vid=0x16c0 pid=0x5dc`
That means either you did not enter program mode, or if you are in Linux, you haven't given the system permission to use this USB device.
 - The Arduino output might report the following warning, just ignore them:
`avrdude: warning: cannot set sck period. please check for usbasp firmware update.`
`avrdude: error: usbasp_transmit: error sending control message: Broken pipe`
 - To update the program, enter program mode again, and upload.
- **Using Existing Arduino Programs:**
 - Because SquareWear 2.0 is based on Arduino, you can upload any existing Arduino program. The Arduino software has numerous example programs. Just make sure that you modify the program to use the available pins on SquareWear 2.0.
 - **Using the SoftPWM library:**
 - The build-in LEDs are not connected to hardware PWM pins, so you cannot use `analogWrite` directly on those LEDs. But the SquareWear2 library includes a SoftPWM (software PWM) library that can simulate PWM on these LEDs.
 - To use the library, you need to include both `HIDSerial.h` and `SoftPWM.h` in your sketch.
 - In the `setup()` function, call `SoftPWMBegin()`;
 - Then use `SoftPWMSet(pin, value)` to set a PWM value (0 to 255) to a pin (any digital pin).
 - Check the 'fade' demo for example.
 - **Using the HIDSerial library:**
 - SquareWear2 does not have built-in USB-serial converter. However, it can simulate serial functions in software. This is done through the HID (human interface device) protocol, which requires no driver installation.
 - Check the 'hidserial' demo for example. The library is designed to be somewhat compatible with Arduino's `Serial` class.
 - To use the library, you need to include `HIDSerial.h` in your sketch.
 - Specify a global `HIDSerial` class variable. For example: `HIDSerial serial;`
 - In the `setup()` function, call `serial.begin()`;

- Use `serial.write(char)` to write a single character. Use `serial.print` or `println` to write a string.
- Use `serial.available()` to check if there is an incoming string (from host computer).
- Then use `serial.read` to read the incoming string.
- **Important:** because the library is not interrupt driven, you must call `serial.poll()` as frequently as possible to handle USB requests in time. This can be done by inserting `serial.poll()` in the inner loop and make sure it's called frequently.
 - Technically, even if you have called `serial.print`, the string is not sent immediately to the host computer, but will be sent over several iterations. Each iteration is done when `serial.poll()` is called. That's why it has to be called as often as possible.
 - Similarly, in order to detect incoming strings, since the serial function is not interrupt drive, you need to call `serial.poll()` as often as possible to not lose a string.
- **Host software:** to use the HID serial function, you need to run host software called HID Serial Monitor. It's cross platform. To use the software:
 - First click on the 'Connect' button to connect to the device (if unsuccessful, check if SquareWear is turned on and if you have followed the above descriptions for calling `serial.begin()` and `serial.poll()` in your sketch.
 - Next, once connected, if a string is transferred from device to host, it will be displayed in the text area.
 - To send a string to the device, type the string in the text field, and then click on 'Send'.
 - **Limitation:** try not to send a string longer than 8 characters. Sending a string longer than 8 characters may generate an error, this is being addressed in software.
 - You can also pause the device and host communication by clicking on the 'Pause' button.

