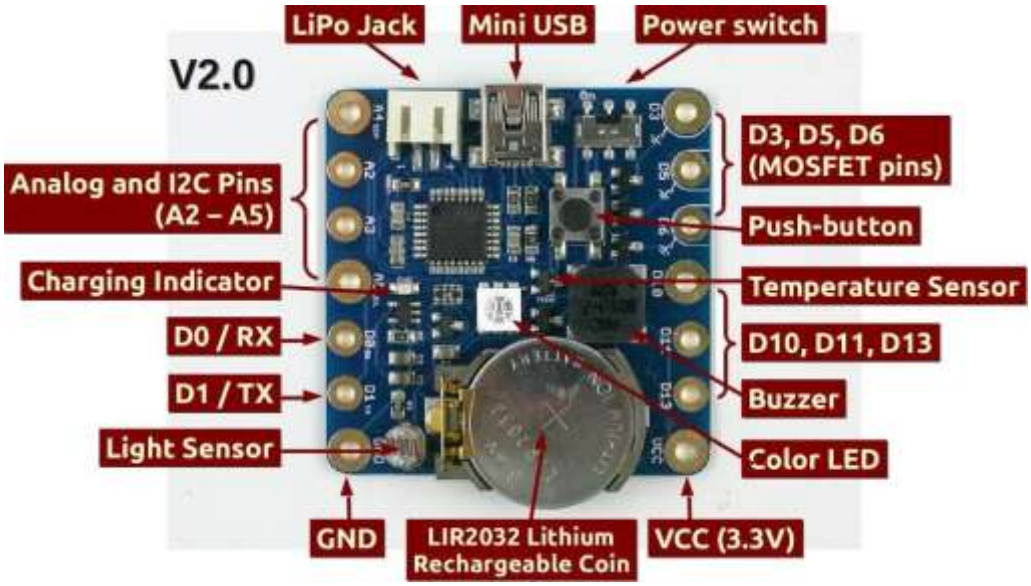
SquareWear 2.x User Manual

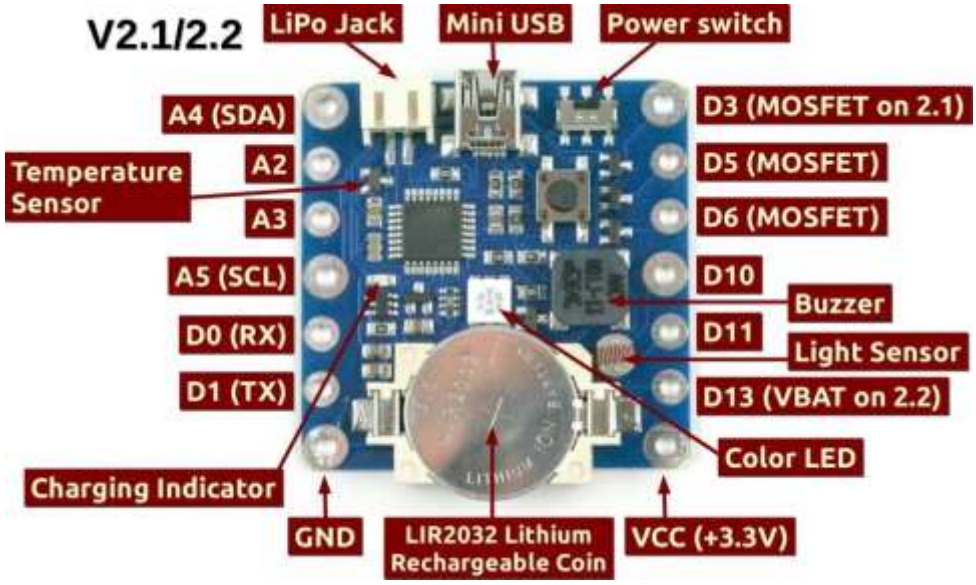
(*Last update: Feb 6, 2015*)

Hardware Interface

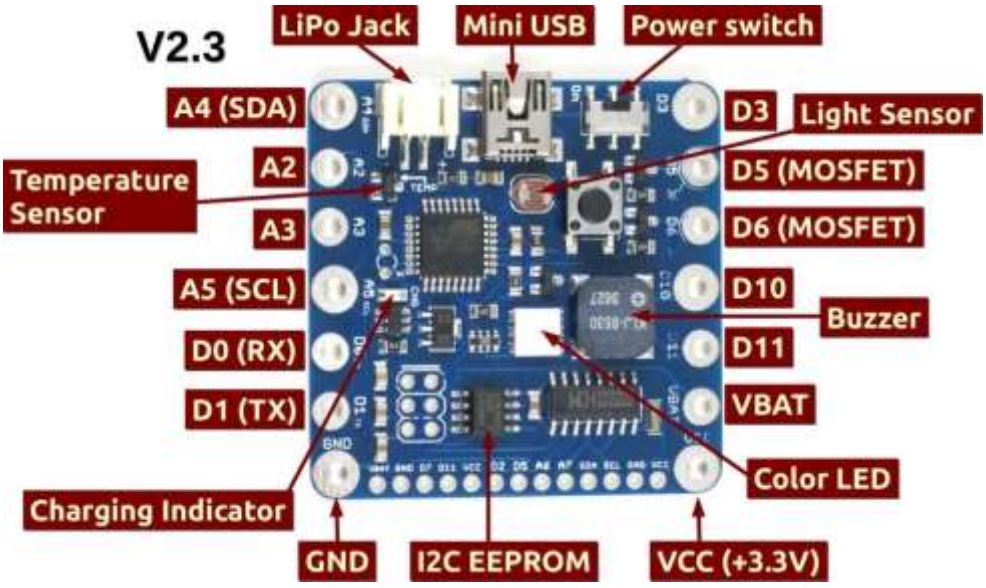
2.0 (brown battery holder):



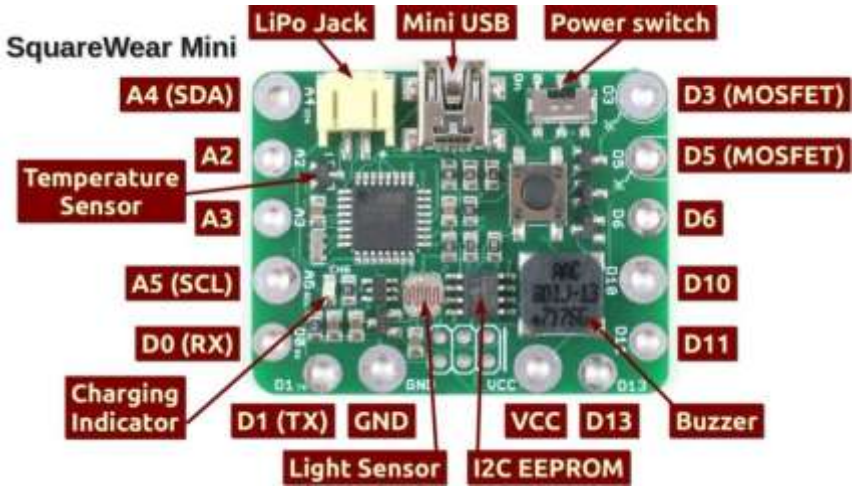
2.1/2.2 (white battery holder):



2.3 (no built-in battery)



SquareWear Mini (no built-in battery):



Built-in Components

- ***2.0/2.1/2.2/Mini***: Mega328p @ 3.3V 12MHz, with USBasp bootloader (courtesy of USnoobie).
- ***2.3***: Mega328p @ 3.3V, 8MHz, with Arduino bootloader (compatible with Lilypad and Pro Mini)
- 3.3V / 250mA Linear Regulator (LDO), Lithium charging chip and charging indicator LED.
- Temperature sensor (TDR), light sensor (LDR), Color (RGB) LED (*not on Mini)*
- Mini-buzzer, SMD tactile button, power switch, and JST connector for external lithium battery.
- N-MOSFETs (for driving high current load).
- LIR2032 rechargeable lithium coin battery (45mAh capacity) (*not on 2.3/Mini)*
- 16KB I2C EEPROM (*not on 2.0/2.1/2.2)*
- CH340 USB-serial converter (*only available on 2.3*)

Power Options (IMPORTANT!!! Please READ!!!)

- SquareWear 2.0/2.1/2.2 have built-in 45mAh rechargeable Lithium coin battery. Every time you plug in the mini-USB cable, it charges the battery automatically. Do **NOT** use it with non-rechargeable batteries such as CR2032.
- You can also using an external LiPo battery (such as http://rayshobby.net/cart/lipo-700). ***When using external LiPo, please remove the built-in coin battery.***
  - To remove the built-in LIR2032, carefully push back the metal tap on the battery holder, until you can lift the battery up. DO **NOT** force lifting the battery as it may break the battery holder..

- The build-in Lithium charger is set to charge at 35mA on 2.0/2.1/2.2, and 200mA on 2.3/Mini. This will fully charge the coin battery in about 1-2 hour. The time to fully charge external LiPo varies depending on capacity. The green indicator LED will turn off when battery is fully charged.
- SquareWear has a built-in LDO providing regulated 3.3V (up to 250mA) from battery. When battery voltage drops below 2.7V, the microcontroller will stop running. In this case, you should plug in a USB cable and charge the battery for 15 minutes before using it again.
- SquareWear 2.2/2.3 has a VBAT (battery) pin, which is connected to the battery's positive pin. If your project requires high power, you can use this pin to provide high current (more than 250mA allowed by the LDO).
- Keep battery plugged in at all times. Certain functionality, such as buzzer, may not work properly if battery is removed.

Pin Names and Functions (SquareWear uses the same pin names as Arduino)

- **Digital I/O Pins**: the board has 8 available digital I/O pins
  - **D0 / D1** (also serial RX / TX pin)
  - **D3 / D5 / D6** (support PWM*, MOSFET pins**)
  - **D10 / D11** (support PWM*)
  - **D13** (internally wired to blue LED, setting D13 high lights up the blue LED)

**** PWM (Pulse Width Modulation),*** *also referred to as analog output, provides adjustable level of voltage. You can use PWM to control the brightness of the LED, the speed of a motor etc. All PWM pins can function as standard digital I/O pins, but additionally you can use Arduino's* **analogWrite** *function to set analog voltage levels.*

***** MOSFET Pins***

D3, D5, D6 support PWM, and in addition they are internally wired through MOSFETs. The MOSFETs function as '***Power Sinks***'. This means setting the pins to logical high will connect them to GND, and setting them low will disconnect them from GND. Therefore, in order to use MOSFET pins, such as for 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. By setting the pin high or low, you can control the LED.

The main advantage of Power Sink pins is that they can drive a high amount of current, so your LEDs will look very bright, your speaker will sound loud, etc. You can also use the MOSFET pins to drive a motor, a heat wire, a muscle wire etc. Each MOSFET can drive up to 250mA current. You can combine the three MOSFET pins together to drive more current.

- **Analog Input Pins**: SquareWear 2.x has 4 available analog input pins
  - **A2** (also digital D16)
  - **A3** (also digital D17)
  - **A4** (also digital D18 and I2C SDA pin)
  - **A5** (also digital D19 and I2C SCL pin)

    Analog pins are typically used to read analog signals, such as sensor values. They can also function as digital I/O pins.

- **Internally Assigns Pins (not available for general-purpose use)**
  - **D2 / D7**: USB D- / D+ (only applies to 2.0/2.1/2.2/Mini; available for general use on 2.3)
  - **D4**: push-button (also used to enter bootloader on 2.0/2.1/2.2/Mini)
  - **D8 / D12 / D13**: red / green / blue channel of the RGB LED
  - **D9**: buzzer (set this pin LOW if not using buzzer).
  - **A0 / A1**: light / temperature sensor

- **Breadboard Pins** (**only available on 2.3**): SquareWear 2.3 has the following breadboard pins:
  - **VBAT, GND, D7, D11, VCC, D2, D5, A6, A7, SDA, SCL, GND, VCC**

- **Sewing and Touch Sensing:** SquareWear has large pin holes, allowing you to stitch conductive threads through them and attach the board to textile or fabric. You can also solder wires directly to the pin pads, or solder sew-on snaps to allow quick attachment to / detachment from textile. The large pins are also suitable for touch sensing. Please refer to the touch sensing demos in SquareWear software library. It generally helps increase the touch sensitivity by: 1)
  moistening fingers; 2) holding the VCC or GND pins.

Programming SquareWear 2.x

- **SquareWear 2.0/2.1/2.2/Mini:** (*USBasp bootloader*)

1. Installation:

The recommended installation is to use one of the following pre-configured Arduino installation files:

- **Mac**: http://rayshobby.net/software/arduino-1.0.5-squarewear-macos.zip
  - If you encounter error Arduino.app is damaged and can't be opened, try this work-around.
- **Win**: http://rayshobby.net/software/arduino-1.0.5-squarewear-windows.zip
  - Unless you have installed USBasp driver before, you will need to install the Zadig driver. Download the driver and during installation, choose to install **libusb-win32** driver.
- **Linux 32**: http://rayshobby.net/software/arduino-1.0.5-squarewear-linux32.zip
- **Linux 64**: http://rayshobby.net/software/arduino-1.0.5-squarewear-linux64.zip

After installation, run Arduino 1.0.5 from the installed folder. Then select menu **Tools->Boards->SquareWear 2.0**. There are many provided example programs in **File->Examples->SquareWear2->…**

2. Enter Bootloader on 2.0/2.1/2.2/Mini (IMPORTANT! PLEASE READ!!):

To upload a program, first enter bootloader. To do so, plug in a mini-USB cable to the USB port. First turn off SquareWear (i.e. turn power switch away from USB). Then press the pushbutton while turning on the power switch, and release the pushbutton within 1-2 seconds after the switch is on. At this point, the normal application will stop running, and the controller will present itself as an USBasp programmer to the host computer.

Unlike the standard Arduino, SquareWear 2.0/2.1/2.2/Mini do **not** appear as a serial port (so no COM port or tty.xxx). Instead, it bootloads into an USBasp programmer.
Check the video demo available at: https://www.youtube.com/watch?v=mzAbWqQ7zLE

Linux Users: you need to run Arduino in `sudo` mode or create a rule in **/etc/udev/rules.d/** (except if using the VirtualBox image) to give permission to USBasp,

3. Upload Program to 2.0/2.1/2.2/Mini:

After you've entered bootloader, to upload a program, click on the ***Upload*** button in Arduino. Once the program is uploaded, the microcontroller will start running the program 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 the board fails to enter program mode, or if you are in Linux, you don't have system permission to use the USB device. It may also be caused by low battery voltage.
- If the output reports the following warning
  avrdude: warning: cannot set sck period. please check for usbasp firmware update.
  avrdude: error: usbasp_transmit: error sending control message: Broken pipe
  It's n o rmal. Just ignore them.
  To upload a new program, enter bootloader (Step 2) again.

Temperature Sensor:

- According to the datasheet of MCP9700, if the temperature sensor reading (analog value) is ***A***, the temperature in ***Celsius*** is: ***C = (A * 3.25 / 1024 – 0.5) / 0.01***. The conversion from Celsius to Fahrenheit is: F = (C – 32) * 5 / 9.
- Some SquareWears used MCP9701, and the equation is: ***C = (A * 3.25 / 1024 – 0.4) / 0.0195***.

Using the SoftPWM Library:

The color LED is not wired to any hardware PWM pin, so you cannot use **analogWrite** directly on the color LED. But the SquareWear 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 an example.

Using WS2811/WS2812/WS2813 LED Matrix:

- The SquareWear software pack has pre-installed Adafruit's Neopixel library. You can use it to interface with WS2811/2812/2812B LED strips/matrices. The SquareWear examples include several demos under the **LED matrix** category. These demo programs assume the LED strip / matrix is connected to: VCC (red wire), GND (black wire), and pin D10 (white wire).
- SquareWear Mini is designed to plug directly into the 5x7 chainable LED matrix, and use pin D10 for data. Thus all Mini demos assume the LED data pin is connected to pin D10.
- Refer to the documentation on SquareWear chainable color LED matrix for further instructions.



Interface with External I2C and Serial Devices:

Squarewear 2.3 has breadboard pins which match the layout of several typical I2C and Serial devices, such as the MPU6050 gyro/accelerometer, and Bluetooth transceiver. You can solder breadboard pins to SquareWear 2.3 and plug in sensors accordingly. See picture on the right.

Using the HIDSerial Library (SquareWear 2.0/2.1/2.2/Mini):

- **Note**: SquareWear 2.3 has hardware USB-serial converter, so you can use standard Arduino Serial class and the Serial monitor. The discussions below only apply to 2.0/2.1/2.2/Mini.
- SquareWear 2.0/2.1/2.2/Mini does not have built-in USB-serial converter, so it does not appear as a serial port. Instead, it simulates USB functions in software, using the V-USB library. The serial communication is implemented through the HID (human interface device) protocol, which requires no driver installation.

- Check the ***hidserial*** demo for example. The library is designed to be compatible with Arduino's Serial class as much as possible.
- 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 Arduino **setup()** function, call **serial.begin()**;
- Use **serial.write(char)** to write a single character, **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 HIDSerial 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 ensure that it's called frequently.
- **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 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.
  - Once connected, whenever a string is transferred from the device to host, it will be displayed in the text area.
  - To send a string from the host to the device, type the string in the text field, and then click on ***Send*** (no more than 32 characters a time).
  - You can also ***Pause*** or ***Resume*** the device and host communication.

○ Using Interrupts INT0 (SquareWear 2.0/2.1/2.2/Mini):

- **Note**: INT0 (D2) is available on SquareWear 2.3. The discussions below only apply to 2.0/2.1/2.2/Mini.
- Because the HIDSerial library makes use of interrupt 0 (INT0 or D2), you cannot use INT0 if **HIDSerial.h** is included in your program. The work-around is to use interrupt 1 (INT1 or D3).
- On SquareWear 2.2/Mini, D3 can be used directly. On 2.0/2.1, D3 is internally wired to a MOSFET, to use INT1 you need to solder a wire from the Gate pin of the MOSFET and connect that to your interrupt source.
- If you encounter a compilation error '__vector_x' multiple definition, do the following:
  Open **arduino-1.0.5/hardware/arduino/cores/arduino/WInterrupts.c**
  and comment out all **ISR(INT0_vect)** functions, for example:

```
/*ISR(INT0_vect) {
  if(intFunc[EXTERNAL_INT_2])
  intFunc[EXTERNAL_INT_2]();
}*/
```

- All other interrupts, such as pin change interrupts, timer interrupts, all function as normal.

○ Simulate Human Interface Devices (HID) (SquareWear 2.0/2.1/2.2/Mini):

Because SquareWear 2.0/2.1/2.2/Mini uses the V-USB implementation, you can program it to simulate a mouse, a keyboard, or other HID devices. Examples can be found in the V-USB website.