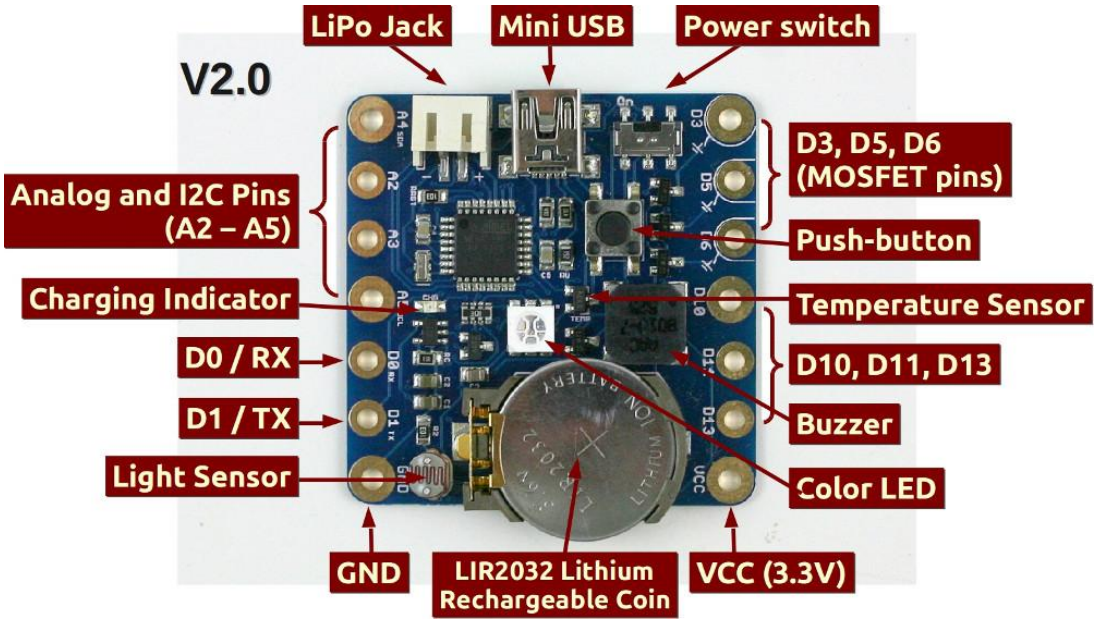
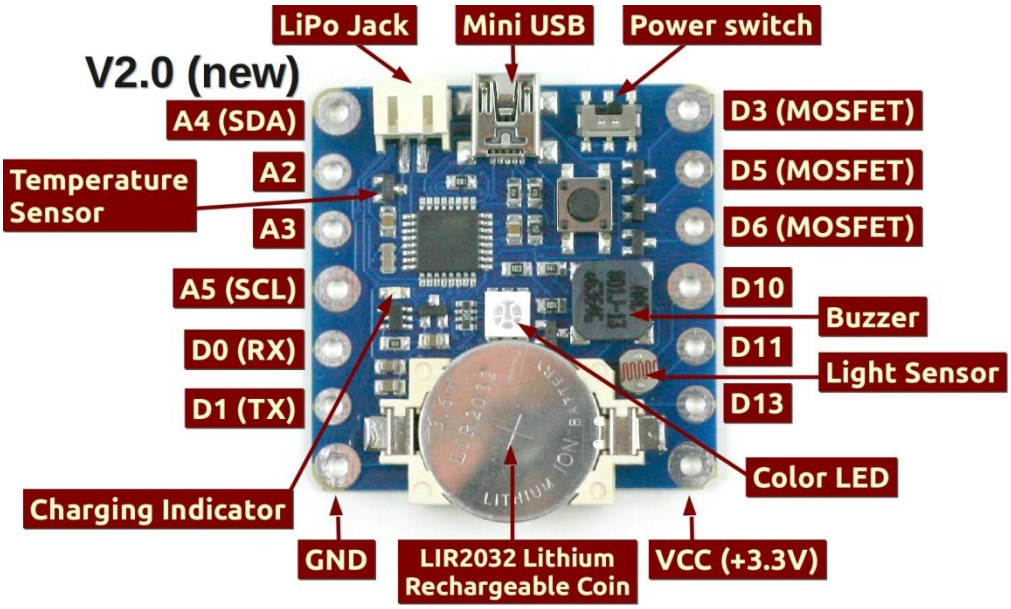
SquareWear 2.0 User Manual

- All SquareWear boards come with **pre-flashed demo programs**. Clicking on the push-button cycles through all available demos in the program. Additional demos are available in the SquareWear 2.0 Arduino library. Instructions can be found at the beginning of each program's source code.

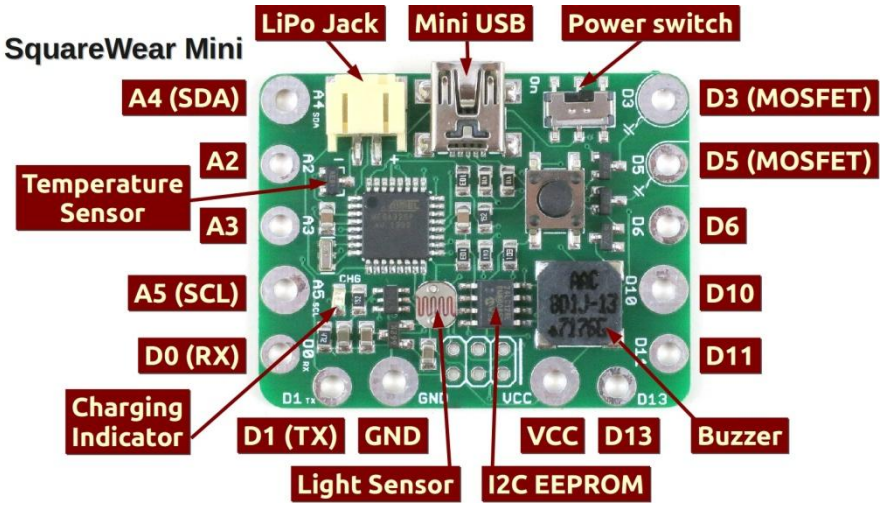
- **Hardware Interface**
  - ***Original Version (brown battery holder):***



  - ***Revised Version (white battery holder):***

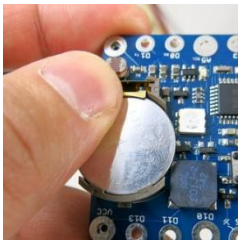


- ***SquareWear 2.0 Mini:***



- **Built-in Components**
  - ATmega328p running at 3.3V, 12MHz, pre-flashed with USBasp bootloader.
  - MCP1700 3.3V / 250mA LDO linear regulator.
  - MCP73831 lithium charging chip (configured to charge at 35mA).
  - MCP9700 temperature sensor, 10K photo-resistor (light sensor).
  - 8.5mm SMT buzzer, 6mm SMT tactile button.
  - 2.0mm JST connector for external lithium battery.
  - 2N7002 MOSFETs, SMT mini-USB port, and power switch.
  - 5mm color (RGB) LED (*not available on Mini)*
  - LIR2032 (20mm) rechargeable lithium coin battery (45mAh capacity) (*not on Mini)*
  - 24LC128 I2C EEPROM (*available on Mini only)*

- **Power Options (IMPORTANT!!! Please READ!!!)**
  - SquareWear 2.0 (except Mini) has a built-in 45mAh rechargeable lithium coin battery (LIR2032). 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 power SquareWear by an external Lithium-Polymer (LiPo) battery (such as http://rayshobby.net/cart/accessories/acc-batteries/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. If it ever breaks, use some hot glue to fix it back in place.



- The build-in lithium charger (MCP73831) is set to charge at 40mA per hour (on Mini it's set to charge at 200mA). This will fully charge the coin battery in 1 hour. The charging time for external LiPo varies depending on the battery capacity. The green indicator LED will turn off when battery is fully charged.
- SquareWear has a built-in LDO (MCP1700) to provide regulated 3.3V 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.
- Keep battery plugged in at all times. Certain functionality, such as buzzer and USB bootloader, may not work reliably if the 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** (MOSFET pins***, support PWM*)
  - **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 2.0 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+
  - **D4**: push-button (also used to enter bootloader)
  - **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

- **Sewing:** 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.

Programming SquareWear 2.0

- **Pre-Requisites:**
  - The recommended way to set up software is by downloading one of the following pre-configured Arduino installation files:
    - Mac: http://rayshobby.net/software/arduino-1.0.5-squarewear-macos.zip
    - Win: http://rayshobby.net/software/arduino-1.0.5-squarewear-windows.zip
    - Linux32: http://rayshobby.net/software/arduino-1.0.5-squarewear-linux32.zip
    - Linux64: http://rayshobby.net/software/arduino-1.0.5-squarewear-linux64.zip

    These installation files are based on Arduino 1.0.5 and have pre-installed necessary files to get you started. They also contain drivers for Windows, and SquareWear demos.
    - **Mac OS users**: if the pre-configured software package gives you an error Arduino.app is damaged and can't be opened, this is because the latest OS X requires signed apps. The temporary work-around is to turn off this requirement in system settings. The solution is described in this page.
  - After installation, run Arduino 1.0.5 from the installed folder. Then in the menu, select **Tools->Boards->SquareWear 2.0**. There are many provided example programs, which you can find in **File->Examples->SquareWear2->…**

- **Enter Bootloader and Upload a Program (IMPORTANT! PLEASE READ!!):**
  - To upload a program, plug in a mini-USB cable to the USB port. First turn off SquareWear (i.e. the power switch away from USB). Then press the pushbutton while turning on the power switch. At this point, the normal application will stop running, and the controller will present itself as a USBasp programmer to the host computer.

        - Unlike the standard Arduino, SquareWear does **not** appear as a serial port (so no COM port or tty.xxx). Instead, it bootloads into a USBasp programmer.
        - **Windows Users:** you need to install USBasp driver, which is included in the **Arduino1.0.5/drivers/usbasp** folder. Note: Windows 8 strictly enforces signed driver signature. The work-around is to disable signed digital signature enforcement. Google 'Windows 8 USBasp' to find suitable solutions.
        - **Linux Users**: 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, 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 you did not successfully enter program mode, or if you are in Linux, you don't have system permission to use the USB device.
        - 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 normal. Just ignore them.
    - To upload a new program, repeat the above steps.

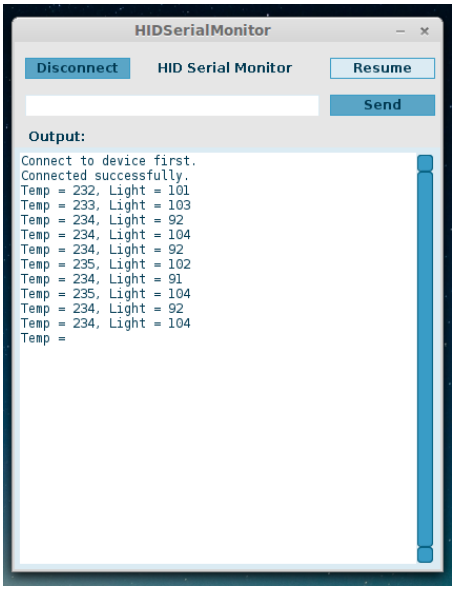
- **Using Existing Arduino Programs:**
    - Because SquareWear 2.0 is based on Arduino, you can upload any existing Arduino program. The Arduino software has numerous example programs. Just make sure that you suitably modify the program to match the pins assignment on SquareWear 2.0.

- **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***. So if the first equation does not work, try the second equation.

- **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 use of software PWM.

- **Using the HIDSerial library:**
  - SquareWear 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 Interrupt INT0:**
  - 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).

- Since 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):**
  - Because SquareWear 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.

- **Using WS2811/WS2812/WS2813 LED matrix:**
  - The SquareWear software pack has pre-installed the Neopixel library. You can use it to interface with WS2811/2812/2813 LED strips/matrices. The SquareWear examples include a few demos (ledpad-xxx). These demo programs assume the LED strip / matrix is connected as follows: VCC (red wire), GND (black wire), and pin D1 (white wire).
  - SquareWear 2.0 Mini is designed to plug in directly to a WS2812 LED matrix, and use pin D10 for data. Thus all Mini demos assume the LED data pin is connected to pin D10.