**SMART WEARABLE CONTROLLING SYSTEM**

**BY HAND AND FINGERS GESTURE RECOGNITION**

HARDWARE MANUAL

Version: December 2017

Tran Viet Cuong

Contents

[1. Overview 3](#_Toc501902065)

[2. Processing module 3](#_Toc501902066)

[2.1 Teensy 3.2 4](#_Toc501902067)

[2.2 I2C Multiplexer (TCA9548A) 7](#_Toc501902068)

[2.3 IMU – BNO055 8](#_Toc501902069)

[2.4 PPG filter 9](#_Toc501902070)

[2.5 Battery changer (MCP73831T) 10](#_Toc501902071)

[3. IMU module 11](#_Toc501902072)

[4. IMU\_PPG module 12](#_Toc501902073)

# Overview

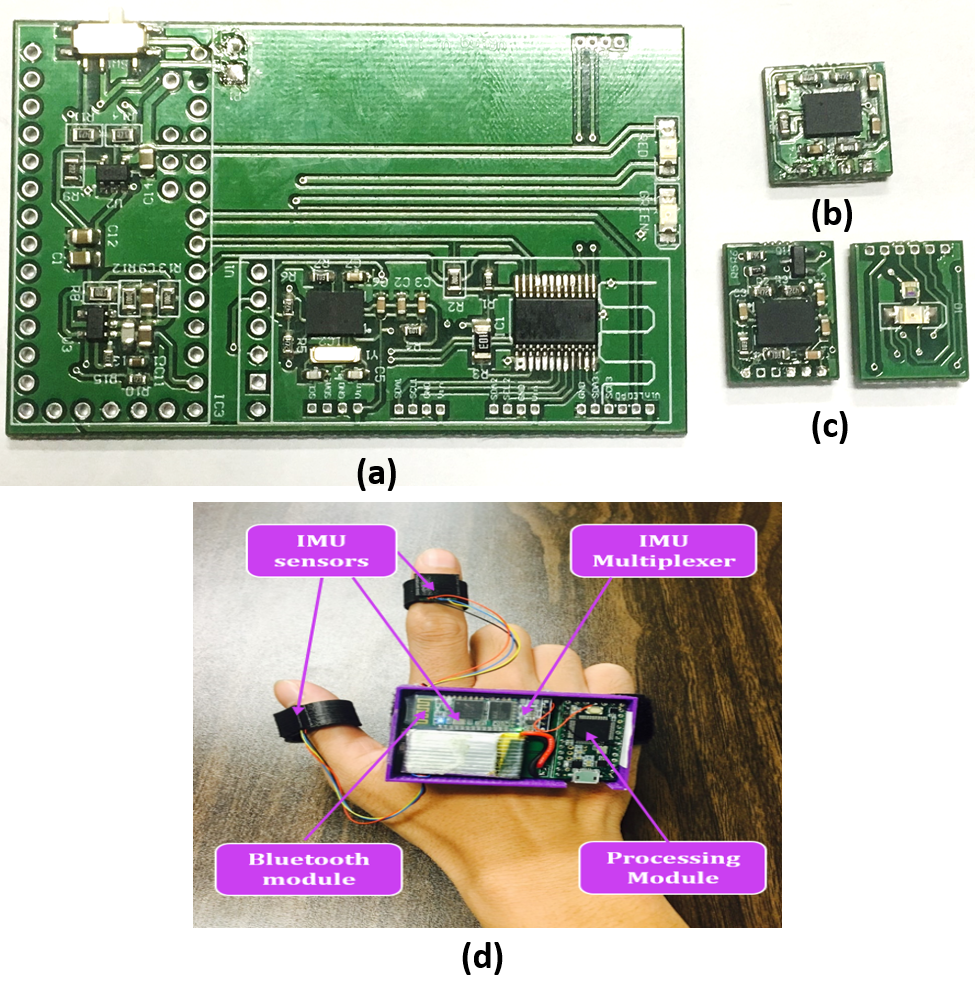


Figure 1‑1 Smart wearable controller system

Smart wearable controller system consists of three modules: processing module, IMU module and IMU\_PPG module.

# Processing module

This processing module includes a microcontroller unit (MCU) – Teensy 3.2, an IMU (BNO055), an I2C multiplexer (TCA9548A), Bluetooth module (HC-06), PPG filter and battery charger (MCP73831T) as shown in Figure 2‑1.

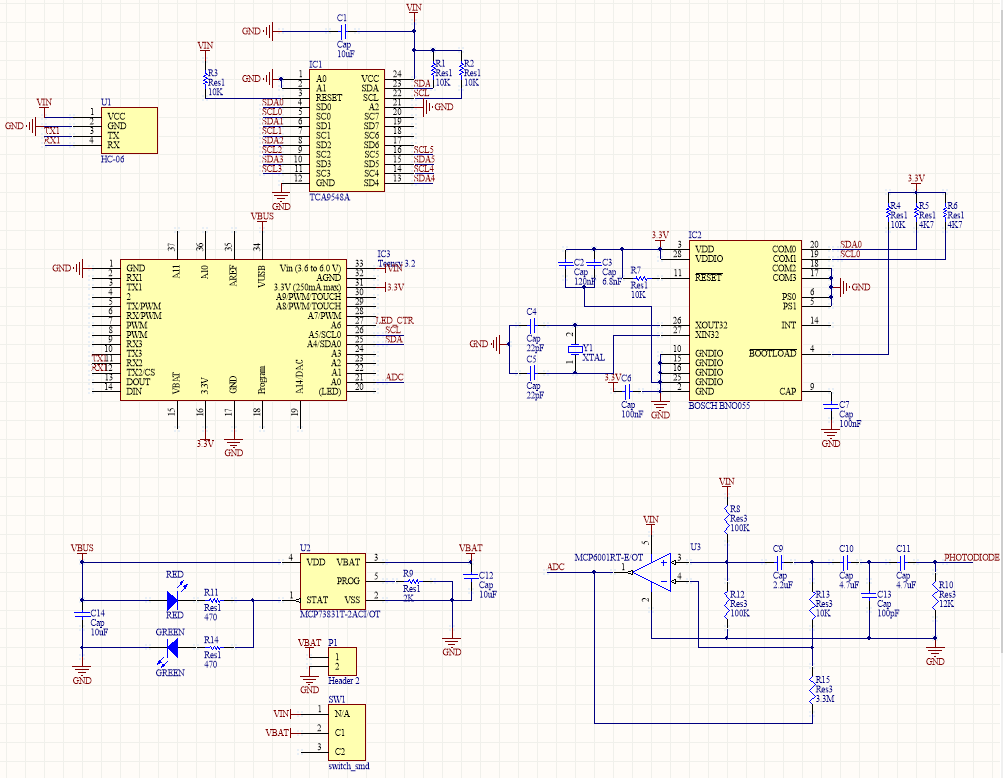
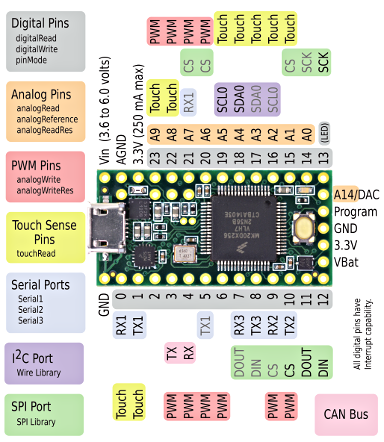


Figure 2‑1 Schematic of processing module

## Teensy 3.2

Technical specifications:

* 32 bit ARM Cortex-M4 72MHz CPU (M4 = DSP extensions)
* 256K Flash Memory, 64K RAM, 2K EEPROM
* 21\* High Resolution Analog Inputs (13 bits usable, 16 bit hardware)
* 34\* Digital I/O Pins (21 shared with analog)
* 12 PWM outputs
* 1 12-bit DAC output
* 8 Timers for intervals/delays, separate from PWM
* USB with dedicated DMA memory transfers
* CAN bus
* 3 UARTs (serial ports)
* SPI, I2C, I2S, IR modulator
* I2S (for high quality audio interface)
* Real Time Clock (with user0added 32.768 crystal and battery)
* 16 general purpose DMA channels (separate from USB)
* Touch Sensor Inputs



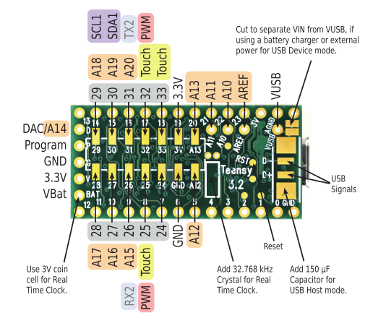


Figure 2‑2 Teensy 3.2 pinout

**Please note:** Teensy 3 and 2 are not official Arduino-brand products. Although the Teensyduino IDE has been adapted so that many simple Arduino projects will work with the Teensy, there will still be a lot of libraries and shields that will not work with this device!

Here is tutorial for integrating Arduino IDE and Teensy 3.2 ([link](https://www.pjrc.com/teensy/tutorial.html))

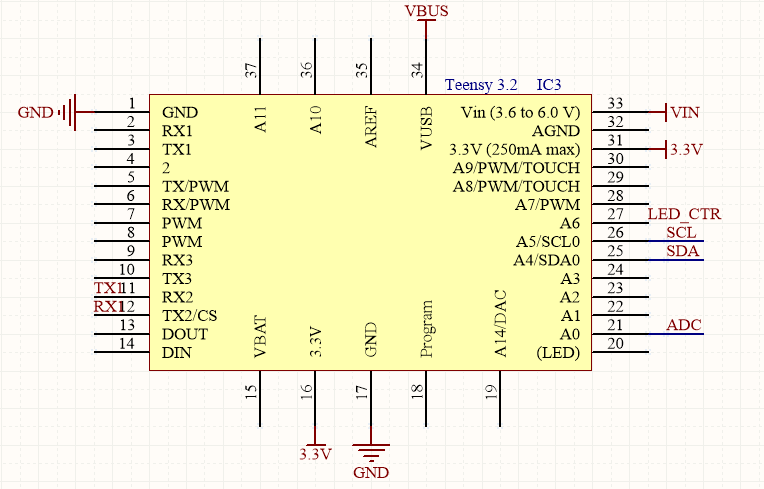


Figure 2‑3 Teensy 3.2 pinouts for system

**Power Pins:**

* VUSB (VBUS – 34) – this is the 5V from USB, using for a battery charger or external power
* VIN (Vin – 33) – this is the power pin from 3.7V – 250 mA battery
* 3.3V (3.3V – 16) – this is the 3.3V pin voltage supply for sensors
* GND (GND – 1, 17) – common ground for power and logic

**Bluetooth Pins:**

* Tx2 (12): this pin is connected to HC-06 Rx pin
* Rx2 (11): this pin is connected to HC-06 Tx pin

**I2C Pins:**

* SCL (SCL0): this pin is connected to the I2C multiplexer (TCA9548A)
* SDA (SDA0): this pin is connected to the I2C multiplexer (TCA9548A)

**PPG Pins:**

* LED\_CTR (27): this pin controls led from IMU\_PPG module
* ADC (A0-21): This pin is connected to PPG filter to get heart beat signal

## I2C Multiplexer (TCA9548A)

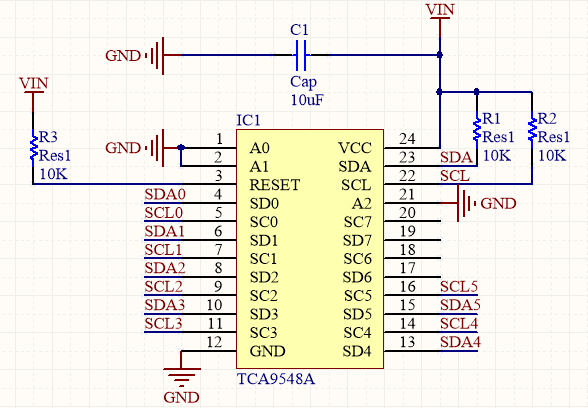


Figure 2‑4 TCA9548A schematic

Using it is fairly straight-forward: the multiplexer itself is on I2C address 0x70 (but can be adjusted from 0x70 to 0x77) and you simply write a single byte with the desired multiplexed output number to that port, and bam - any future I2C packets will get sent to that port. In theory, you could have 8 of these multiplexers on each of 0x70-0x77 addresses in order to control 64 of the same-I2C-addressed-part.

**Power Pins:**

* Vin - this is the power pin. Since the sensor chip uses 3-5 VDC. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V
* GND - common ground for power and logic

**I2C Control-Side pins:**

* SCL - this is the I2C clock pin for the chip itself, connect to your microcontrollers I2C clock line.
* SDA - this is the I2C data pin for the chip itself, connect to your microcontrollers I2C data line.
* RST - this is the reset pin, for resetting the multiplexer chip. Pulled high by default, connect to ground to reset
* A0 A1 A2 - these are the address selection pins for the multiplexer. By default the multiplexer is at address 0x70 and these three pins are pulled low. Connect them to Vin to set the address to 0x71 - 0x77.
* A0 is the lowest-significant bit (if it is pulled high, it will increase the address by 1).
* A1 is the 2nd-lowest-significant bit (if it is pulled high, it will increase the address by 2).
* A2 is the 3rd-lowest-significant bit (if it is pulled high, it will increase the address by 4).

**I2C Multiplexed-Side pins:**

* SDx and SCx: There are 8 sets of SDx and SCx pins, from SD0/SC0 to SD7/SC7. These are the multiplexed pins. Each one is a completely seperate I2C bus set. So you have have 8 I2C devices with identical addresses, as long as they are on one I2C bus each.
* These pins do not have any pullups installed, so if you are using a chip or breakout without i2c pullups be sure to add them! Nicely, you can have Vin be 3.3V and have these pins pulled up to 5V (that is, they are 5V compliant)

This tutorial will guide how to program with I2C multiplexed ([link](https://learn.adafruit.com/adafruit-tca9548a-1-to-8-i2c-multiplexer-breakout?view=all#wiring-and-test))

## IMU – BNO055

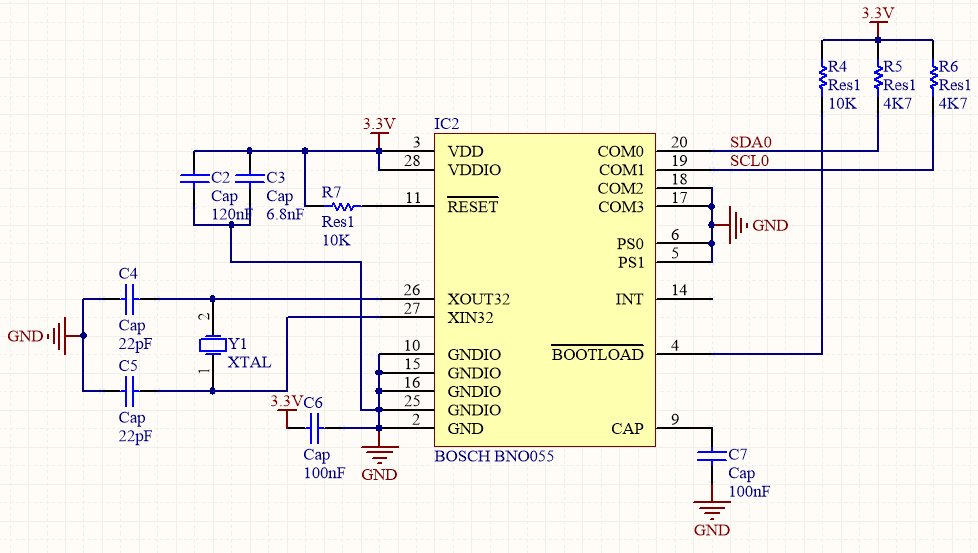


Figure 2‑5 IMU - BNO055 schematic

IMU sensor on the processing module utilized external crystal of 32.768 kHz to obtain high accuracy of sensor reading and in the meantime performed better with low power consumption.

**Power Pins:**

* VIN: 3.3-5.0V power supply input
* 3VO: 3.3V output from the on-board linear voltage regulator, you can grab up to about 50mA as necessary
* GND: The common/GND pin for power and logic

**I2C Pins:**

* SCL - I2C clock pin, connect to SCL0 from TCA9548A. This pin can be used with 3V or 5V logic, and there's a 10K pullup on this pin.
* SDA - I2C data pin, connect to SDA0 from TCA9548A. This pin can be used with 3V or 5V logic, and there's a 10K pullup on this pin.

## PPG filter

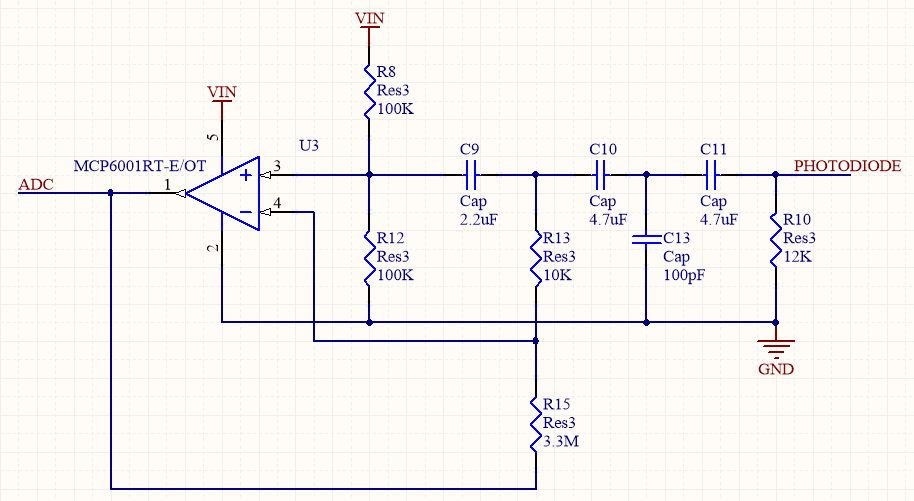


Figure 2‑6 PPG filter schematic

The basic PPG filter, we are using the same ambient light sensor (APDS-9008) (PHOTODIODE pin), and the same green super bright reverse mount LED from IMU\_PPG module ([link](https://pulsesensor.com/pages/open-hardware)).

The Current to Voltage Converter is a well known Op Amp circuit that invariably uses a photodiode as a current source, and is often used as a starting point for developing optical heart-rate monitors. After breadboarding a few using different Op Amps and Photodiodes, we designed a fairly universal Low Pass Filter for the output (passive RC. R: 100 C: 4.7uF). The next step was to shrink the hardware by designing with surface mount parts. In our search for SMT photodiodes, we came across a whole pile of ambient light sensors often used in phones and laptops to adjust LCD brightness. These things are sensitive to the visible spectrum, and because of the space requirements in handhelds, they are integrated photodiode / op amp / feedback network in a super teeny tiny itsy bitsy package (!!!). After testing a few models, we picked the Avago APDS-9008 (datasheet) because it has the best form factor for our needs. The peak sensitivity for this sensor is 565 nm, hence the green LED.

In deciding on which LED to use, we came across the reverse mount package from Kingbright (datasheet) purely by accident. We were looking for a super-bright green that matched my price point and had a really small profile. There were already problems with other surface mount LEDs because they are mostly taller than the sensor, which can cause the photodiode to saturate when your finger gets out of alignment. Getting the finger placement just right made it work, but it was too tedious for the user. Turns out Kingbright sells both standard mount and reverse-mount versions of their super-bright green, and we ordered a reverse-mount without knowing it. Now, usually I would curse this kind of mix up between the-part-you-thought-you-ordered and the-part-you-got (TPYTYO v TPYG). It is almost never a good thing. This time I stood up, paced around a bit, then did a little dance, and thanked my dog. The reverse mount LED solders to the back (duh) of the PCB and shines through a small hole (~2mm). This means that the only thing on the top of the board, the side you put your finger on, is the sensor. With this design, we can place the sensor very close to the skin, blocking out ambient light, and it becomes much more immune to noise resulting from movement.

## Battery changer (MCP73831T)

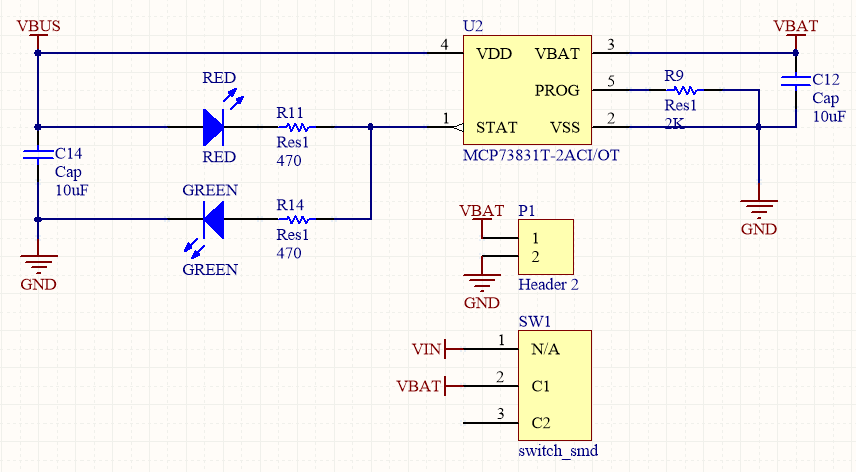


Figure 2‑7 Batter changer schematic

**Charge Indictator LEDs (**There are two LEDs)

* Red - this means the chip has detected a cell and is charging it
* Green - this means the chip has completed the charge cycle and the battery is ready to go

This battery charger used 500mA charge rate.

# IMU module

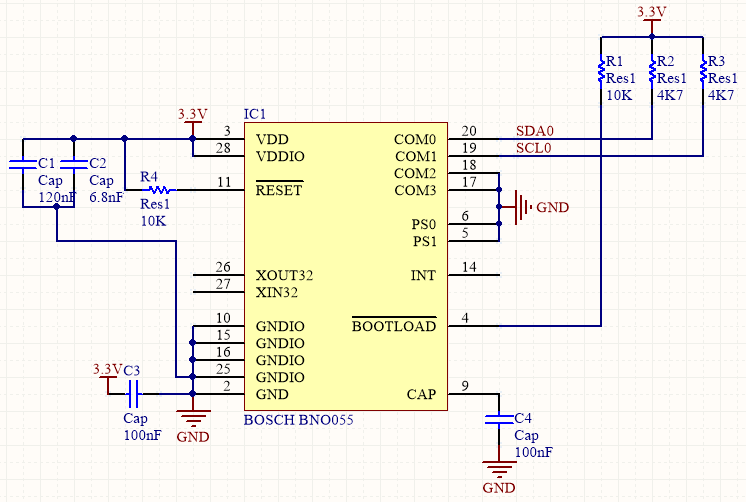


Figure 3‑1 IMU module schematic

IMU sensors in sensor module used their default internal clock with deviation of 3%.

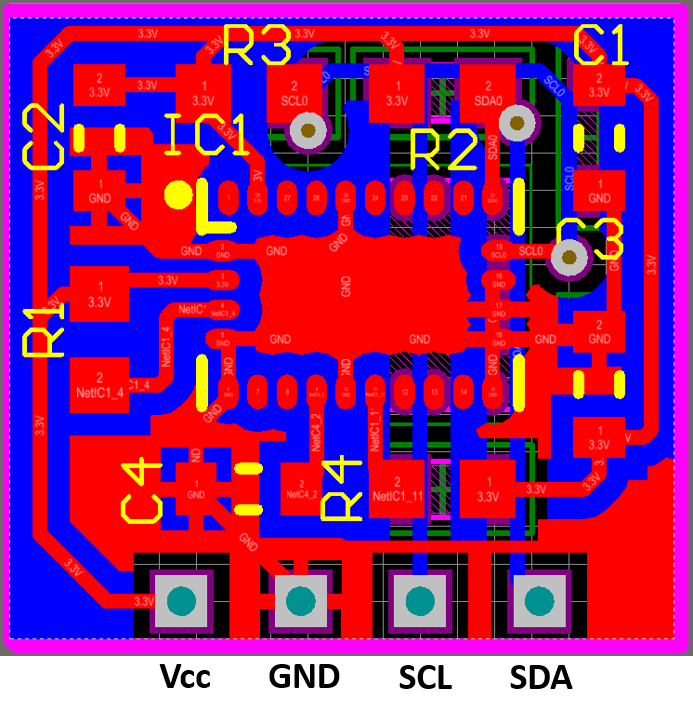


Figure 3‑2 IMU module pinout

# IMU\_PPG module

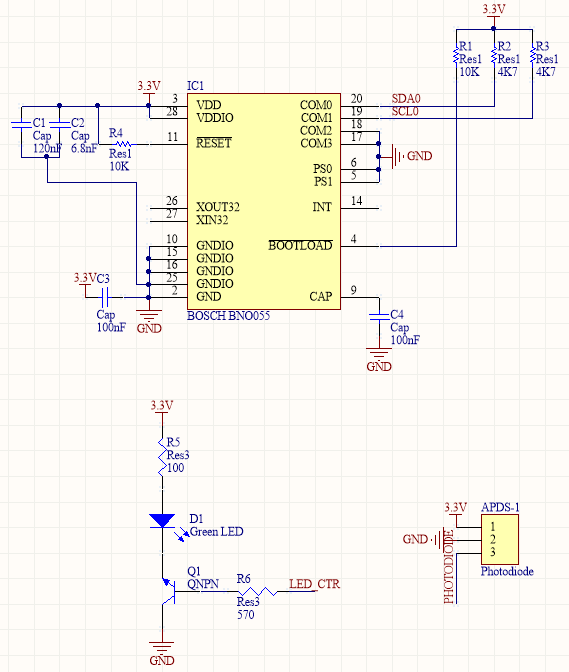


Figure 4‑1 IMU\_PPG module schematic

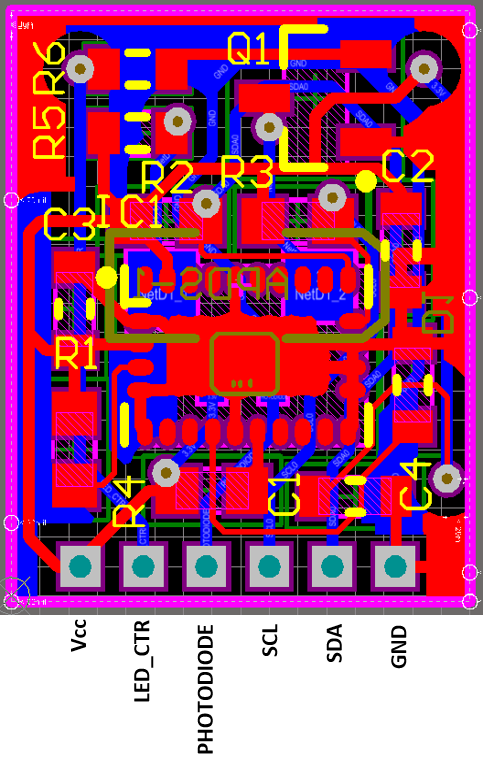


Figure 4‑2 IMU\_PPG pinouts