SensiEDGE
Simplify & Accelerate

CommonSense

Sony’s Spresense Sensor Board

Datasheet

by SensiEDGE and Edge Impulse

Version 1.0

Release Date 14.3.23

List of Content

1 Overview

1.1 General information

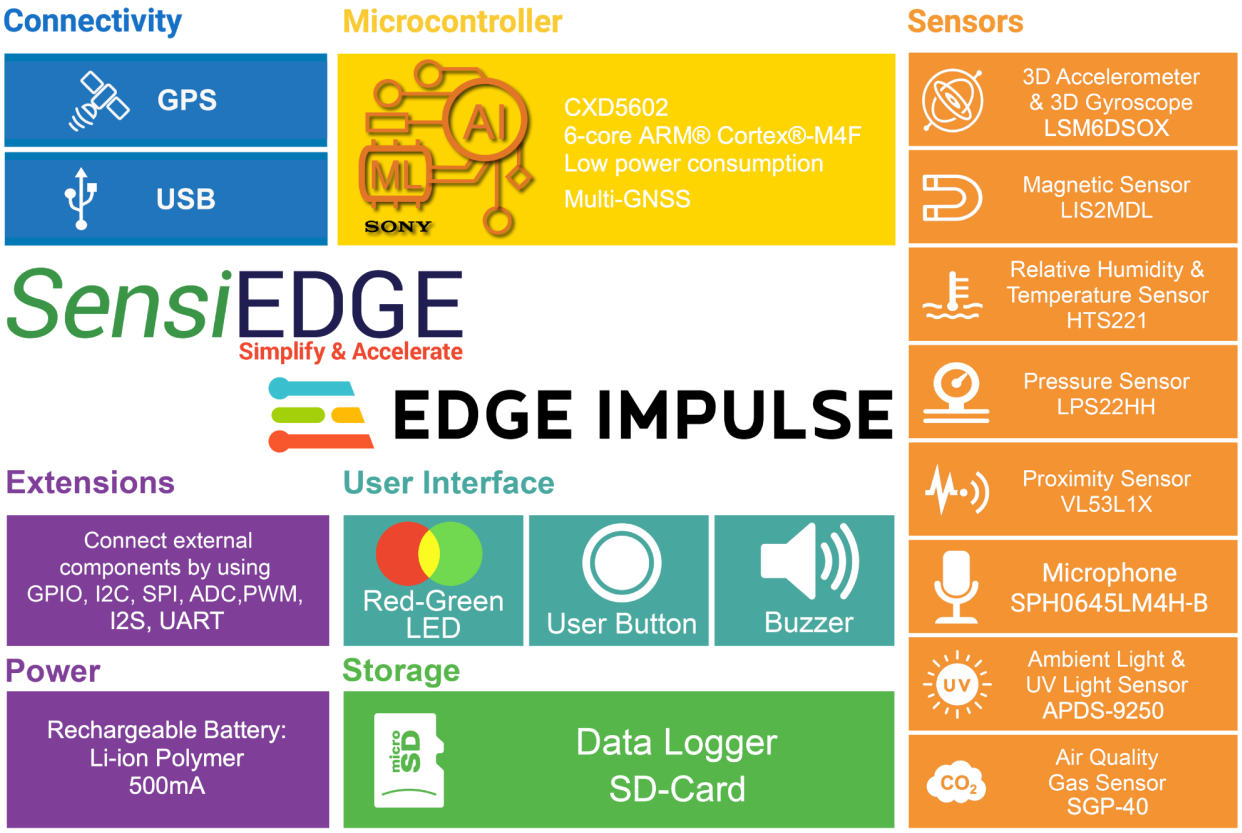

Figure 1. SensiEDGE Block Diagram

System consist (Fig. 1) from the main board and sensor board
The main board is Spresense CXD5602PWBMAIN1C
The sensor board consists from sensors and GPIO expander. The button and LEDs are connected to GPIO expander.

1.2 CXD5602 pinout

CXD6502 pinout and pin functions are in (Fig. 2) and (Tabl. 1).

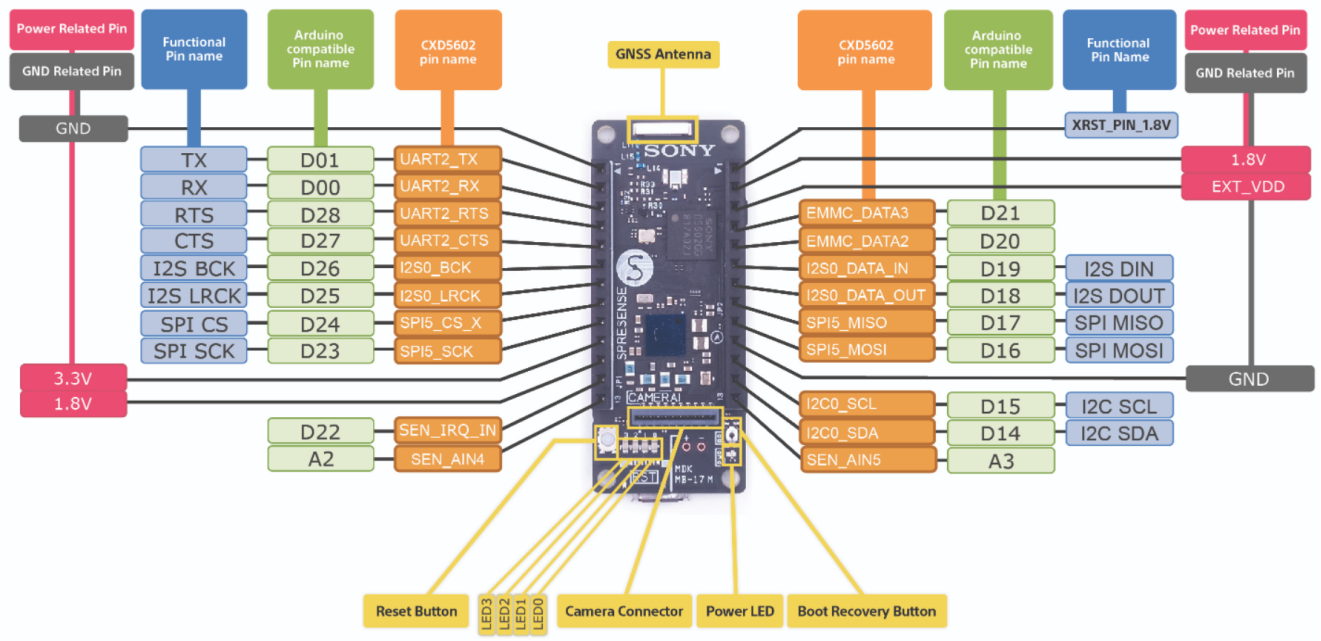

Figure 2. CXD6502 board general view and pinout

Table 1. Pin functions

J1:

Pin\Mode	0	1	2	3
1	GND			
2	GPIO	UART2_TXD	-	GPIO
3	GPIO	UART2_RXD	-	GPIO
4	GPIO	UART2_RTS	-	GPIO
5	GPIO	UART2_CTS	-	GPIO
6	GPIO	I2S_BCK	-	GPIO
7	GPIO	I2S_LRCK	-	GPIO
8	GPIO	EMMC_CMD	SPI5_CS_X	GPIO
9	GPIO	EMMC_CLK	SPI5_SCK	GPIO
10	3,3V			
11	1,8V			
12	GPIO	SEN_IRQ_IN	-	-
13	SEN_AIN4			

J2:

Pin\Mode	0	1	2	3
1	RST			
2	1,8V			
3	EXT_VDD			
4	GPIO	EMMC_DATA3	-	GPIO
5	GPIO	EMMC_DATA2	-	GPIO
6	GPIO	I2S0_DATA_IN	-	GPIO
7	GPIO	I2S0_DATA_OUT	-	GPIO
8	GPIO	EMMC_DATA1	SPI5_MISO	GPIO
9	GPIO	EMMC_DATA0	SPI5_MOSI	GPIO
10	GND			
11	GPIO	I2C0_BCK	-	-
12	GPIO	I2C0_BDT	-	-
13	SEN_AIN5			

1.3 Sensor board pin functions

Sensor board pinout and pin functions are in (Fig. 3) and (Tabl. 2).

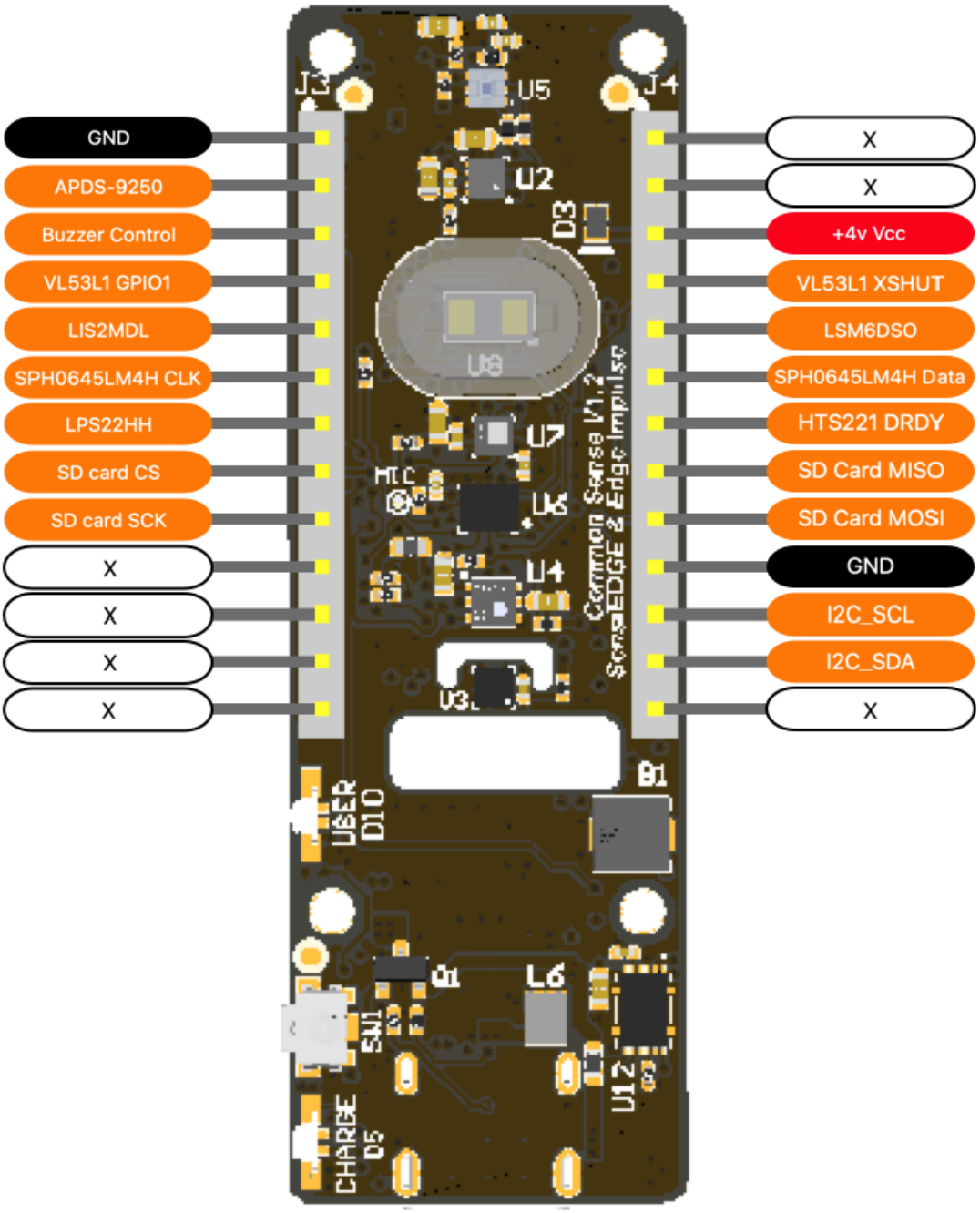

Figure 3. Sensors board pinout

Table 2. Sensors board pinout

J1:

Pin	Mode	Note
1	GND	Power
2	GPO	APDS-9250 INT
3	GPI\PWM	Buzzer control input
4	GPIO	VL53L1 GPIO1
5	GPO	LIS2MDL INT
6	I2S_BCK	SPH0645LM4H-B I2S CLK
7	GPO	LPS22HH INT\DRDY
8	SPI_CS	SD card CS
9	SPI_SCK	SD card SCK
10	x	
11	x	
12	x	
13	x	

J2:

Pin	Mode	Note
1	x	
2	x	
3	+4V	Power
4	GPI	VL53L1 XSHUT
5	GPO	LSM6 INT1
6	I2S_DATA_OUT	SPH0645LM4H-B data
7	GPO	HTS221 DRDY
8	SPI_MISO	SD card MISO
9	SPI_MOSI	SD card MOSI
10	GND	Power
11	I2C_SCL	Sensors and extra GPIO I2C SCL
12	I2C_SDA	Sensors and extra GPIO I2C SDA
13	x	

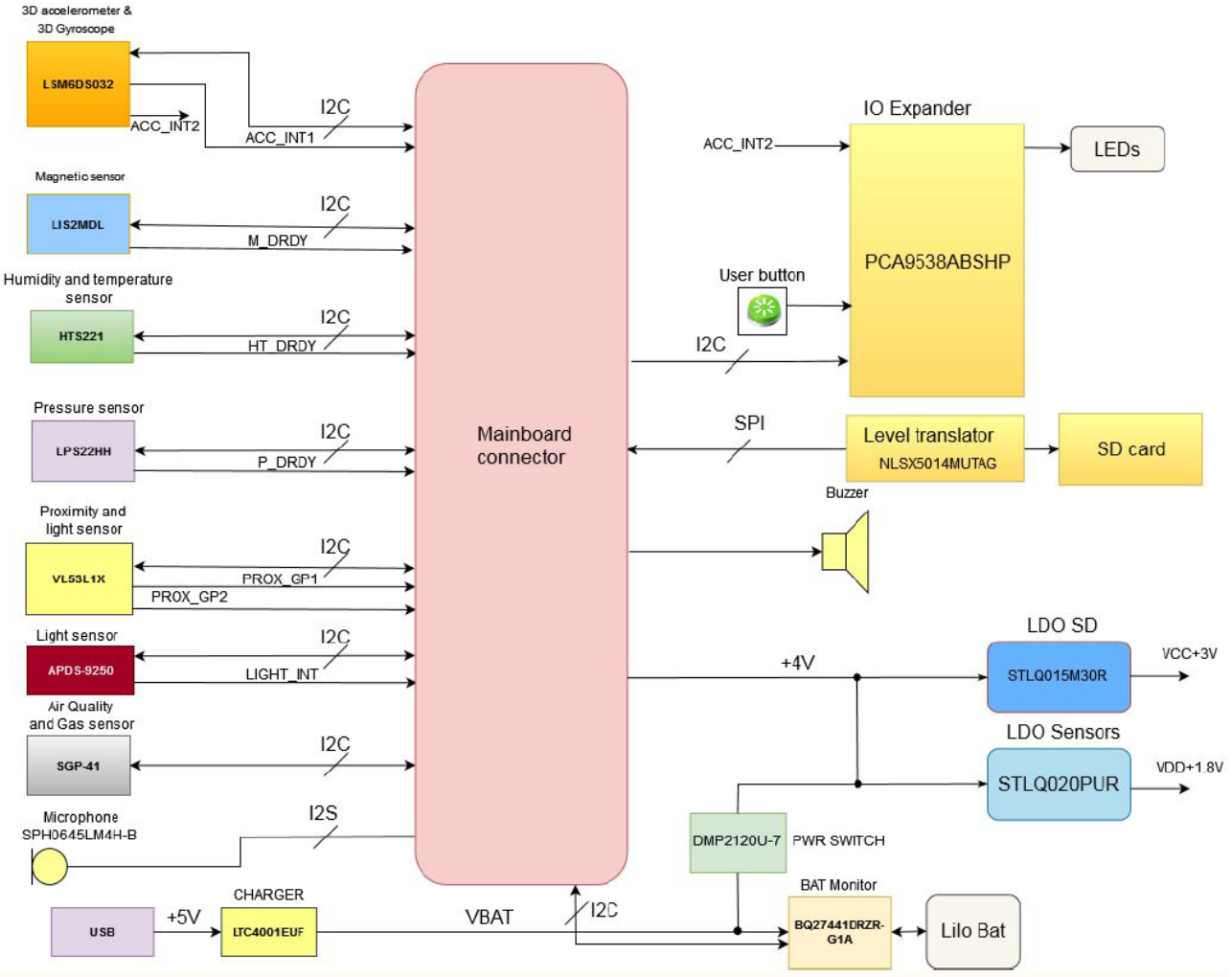

Figure 4. System board pinout

2 Main Hardware Components

The system consist from the main board based on CXD5602 microcontroller and sensor board with sensors and LEDs and button (Fig. 4). The LEDs, button works from port expander PCA9538. Pin INT2 of LSM6DSOX is also connected to port expander.

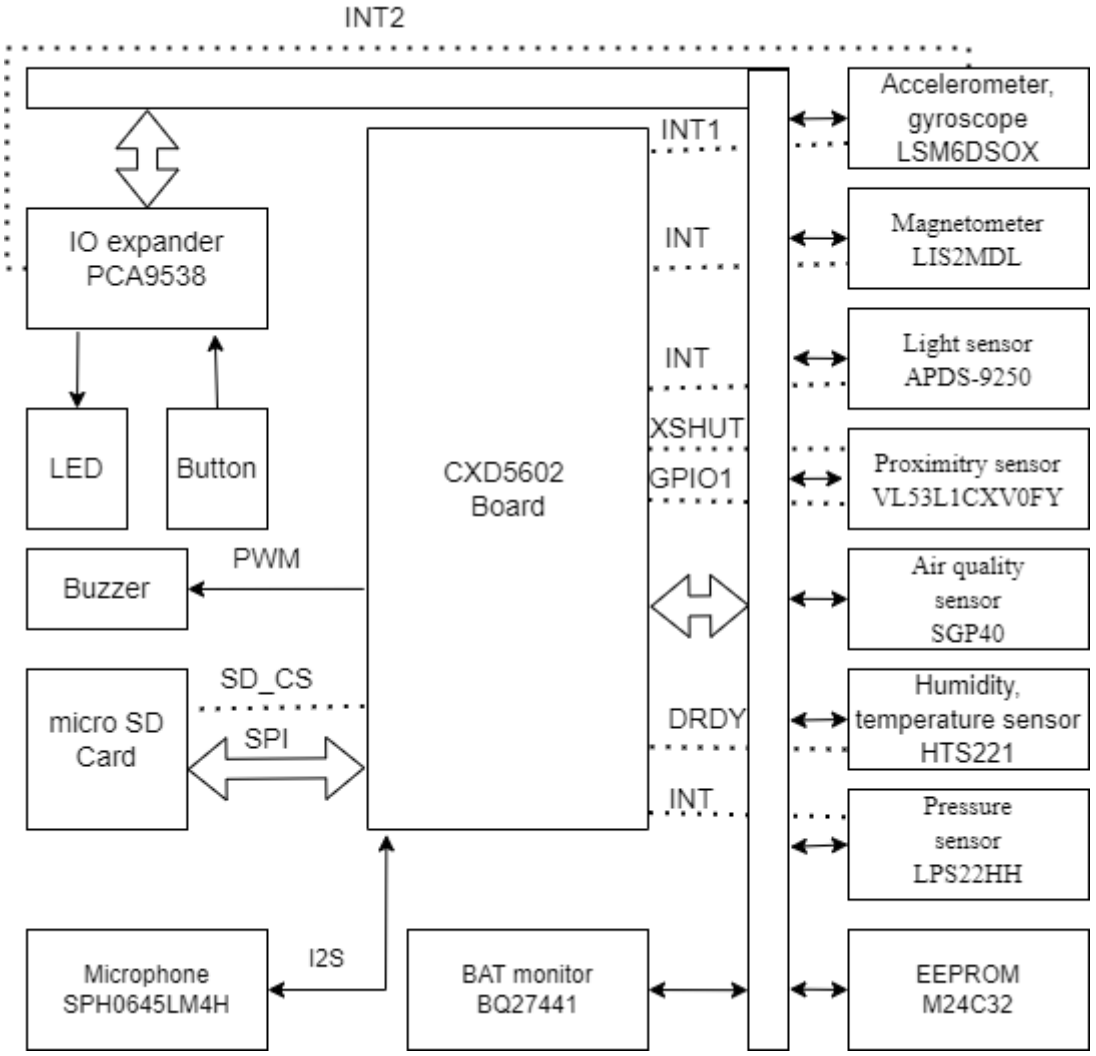

Figure 5. Systems block diagram

2.1 CXD5602 microcontroller

CXD5602GF/GG is a 32 bit RISC low-power microprocessor solution for wearable applications. It is based on the Arm® Cortex®-M4 processor with FPU 32 bit RISC and It integrates Arm® Cortex®-M0+ 32 bit RISC specifically for the system controller (power management, clock, reset) and I/O processor. It incorporates embedded 1.5 MByte of SRAM, 64 KByte of backup SRAM, and 256 KByte of I/O processor SRAM. The Arm® Cortex®-M4 processor with FPU and Arm® Cortex®-M0+ are power-gated by the Power Management Unit, respectively, that is, the CPUs are powered off by internal power switches. Processor SRAMs are able to retain data, and it's possible to restart processors quickly. To provide optimized hardware performance for sensor fusion and audio

processing services, the device integrates ultra-low power GNSS Domain, Audio Codec Domain and Sensor Domain. Integrated Audio Codec Domain supports digital noise cancelling and digital equalizer. Sensor Domain provides the specialized engine for sensor processing.

2.2 Sensors

The *Sensi*BLE module contains verity of sensors :

- ST's 3D accelerometer and 3D gyroscope
- ST's 3-Axis Magnetometer
- ST's humidity and temperature
- ST's pressure sensor
- ST's Proximity sensor
- Sensirion's Air quality sensor
- Avago's Digital RGB, IR and Ambient Light Sensor
- NXP's Port expander
- Kingston's micro SD card
- ST's EEPROM
- TI's Battery monitor
- LT's Battery charger
- Knovel's microphone

2.2.1 LSM6DS3: inertial module: 3D accelerometer and 3D gyroscope

The LSM6DS3 is a system-in-package featuring a 3D digital accelerometer and a 3D digital gyroscope. Enabling always-on low-power features for an optimal motion experience.

2.2.2 LIS2MDL: 3-Axis Magnetometer

The LIS2MDL is an ultra low-power high-performance 3-Axis Magnetometer. This device offers the unique flexibility for designers to implement movement and position detection in space-constrained products such as personal navigation devices.

2.2.3 HTS221TR: humidity and temperature

The HTS221 is an ultra compact sensor for relative humidity and temperature. It includes a sensing element consists of a polymer dielectric planar capacitor structure and a mixed signal ASIC to provide the measurement information through digital serial interfaces.

2.2.4 LPS22HH: pressure sensor

The LPS22HH is an ultra compact absolute piezoresistive pressure sensor. It includes a monolithic sensing element capable to detect.

2.2.5 VL53L1X: Proximity sensor

The VL53L1X is a state-of-the-art, Time-of-Flight (ToF), laser-ranging sensor, enhancing the ST FlightSense product family. It is the fastest miniature ToF sensor on the market with accurate ranging up to 4 m and fast ranging frequency up to 50 Hz.

2.2.6 SGP-40: Air quality sensor

The SGP40 is a digital gas sensor designed for easy integration into air purifier s or demand controlled ventilation systems. Sensirion’s CMOSens ® technology offers a complete , easy to use sensor system on a single chip featuring a digital I 2 C interface and a temperature controlle d micro hotplate, providing a humidity compensated VOC based indoor air quality signal . The output signal can be directly processed by Sensirion’s powerful VOC Algorithm to translate the raw signal into a VOC Index as a robust measure for indoor air quality. The VOC Algorithm automatically adapts to the environment the sensor is exposed to.

2.2.7 APDS-9250: Digital RGB, IR and Ambient Light Sensor

The Avago APDS-9250 is a low-voltage digital RGB, IR and ambient light sensor device that converts light intensity to digital output signal. The color-sensing feature is useful in applications such as LED RGB backlight control, solid-state lighting, reflected LED color sampler and fluorescent light color temperature detection. With the IR sensing feature, the device can be used to read the IR content in certain lighting condition and detect the type of light source.

2.2.8 PCA9538: Port expander

The PCA9538 is an 8-bit I/O expander of general purpose parallel input and output (I/O) expansion for the two-line bidirectional I2C bus (or SMBus) protocol. This device can operate with a power supply range from 2.3 V to 5.5 V. This device supports both 100-kHz

(Standard-mode) and 400-kHz (Fast-mode) clock frequencies. This device, along with other I/O expanders, provides a simple solution when additional I/Os are needed for switches, sensors, push-buttons, LEDs, fans, and so on.

2.2.9 SDCS/16: micro SD card

16 Gb micro SD card.

2.2.10 M24C32 EEPROM

The M24C32 is a 32-Kbit I2C-compatible EEPROM (Electrically Erasable PROgrammable Memory) organized as 4 K × 8 bits.

The M24C32-W can operate with a supply voltage from 2.5 V to 5.5 V, the M24C32-R can operate with a supply voltage from 1.8 V to 5.5 V, and the M24C32-F and M24C32-DF can operate with a supply voltage from 1.7 V to 5.5 V, over an ambient temperature range of -40 °C / +85 °C; while the M24C32-X can operate with a supply voltage from 1.6 V to 5.5 V over an ambient temperature range of -20 °C / +85 °C.

The M24C32-D offers an additional page, named the Identification Page (32 byte).

2.2.11 BQ27441DRZR Battery monitor

The Texas instruments fuel gauge is a microcontroller peripheral that provides system-side fuel gauge for single-cell Li-Ion batteries. The device requires minimal user configuration and system microcontroller firmware development.

2.2.12 SPH0645LM4H microphone

The SPH0645LM4H-B is a miniature, low power, bottom port microphone with an I2S digital output. The solution consists of a proven high performance SiSonic™ acoustic sensor, a serial Analog to Digital converter, and an interface to condition the signal into an industry standard 24-bit I2S format. The I2S interface simplifies the integration in the system and allows direct interconnection to digital processors, application processors and microcontrollers. Saving the need of an external audio codec, the SPH0645LM4H-B is perfectly suitable for portable applications where size and power consumption are a constraint.

2.3 User Interface

The *Sensi*EDGE module contains variety of user interfaces :

- Buzzer TE044003-1
- Button SS304BS
- RG-LED 5988510207F

2.3.1 TE044003-1: Magnetic Buzzer

Buzzers Transducer, Externally Driven Electromechanical 3 V 90mA 4kHz 70dB @ 3V, 10cm Surface Mount Solder Pads

2.3.2 SS304BS: Button

Surface mount type tact switch. Outline dimension 4.5mm x 2.3mm and 1.8 mm height. Horizontal operating direction Anti-fl ux penetration (1100J)

2.3.3 LTST-C195KGJRKT Dual color chip LED

Green, Red 565nm Green, 635nm Red LED Indication - Discrete 2V Green, 2V Red 2-SMD, No Lead

2.4 LTC4001EUF Battery charger

The LTC®4001-1 is a 2A Li-Ion battery charger intended for 5V wall adapters. It utilizes a 1.5MHz synchronous buck converter topology to reduce power dissipation during charging. Low power dissipation, an internal MOSFET and sense resistor allow a physically small charger that can be embedded in a wide range of handheld applications. The LTC4001-1 includes complete charge termination circuitry, automatic recharge and a ±1% 4.1V float voltage. Input short-circuit protection is included so no blocking diode is required.

3 CXD5602 Microcontroller

3.1 Introduction

CXD5602GF/GG is a 32 bit RISC low power microprocessor solution for wearable applications. It is based on the Arm® Cortex®-M4 processor with FPU 32 bit RISC and It integrates Arm® Cortex®-M0+ 32 bit RISC specifically for the system controller (power management, clock, reset) and I/O processor. It incorporates embedded 1.5 MByte of SRAM, 64 KByte of backup SRAM, and 256 KByte of I/O processor SRAM. The Arm® Cortex®-M4 processor with FPU and Arm® Cortex®-M0+ are power-gated by the Power Management Unit, respectively, that is, the CPUs are powered off by internal power switches. Processor SRAMs are able to retain data, and it's possible to restart processors quickly. To provide optimized hardware performance for sensor fusion and audio processing services, the device integrates ultra-low power GNSS Domain, Audio Codec Domain and Sensor Domain. Integrated Audio Codec Domain supports digital noise cancelling and digital equalizer. Sensor Domain provides the specialized engine for sensor processing. Eliminating the need for a discrete sensor hub, these features enable various sensor applications (activity recognition, voice recognition, etc.) low power audio applications such as music playback (MP3 decode, Bluetooth A2DP, etc.) and hands-free communication (Bluetooth HFP).

3.2 Features

The features of CXD5602GF/GG are:

- Application Processor

Arm® Cortex®-M4 processor with FPU 32 bit RISC
Operating frequency up to 156 MHz

- 1.5 MByte Application Memory

- Application Multi-layer Bus

32 bit Multi-layer bus architecture
Application Domain for Arm® Cortex®-M4 processor with FPU 32 bit RISC, Audio Codec, Connectivity, Storage and Image Block

- Audio Codec

Digital Equalizer (DEQ)
Two I2S Interfaces supported
Unique Audio Data Format (Pulse Density Modulation) between CXD5602GF/GG and CXD5247GF

- 8 bit parallel Camera Interface supported

- 2D Graphics Acceleration

BitBLT, Rotate, Scaling, Blender

- Connectivity/Storage Interface

On-chip USB2.0 Device supported
eMMC 4.41 for eMMC Device
SD3.0 Host Controller interface
SPI and SDIO support for external Wi-Fi transceivers
UART support for external Bluetooth transceivers
Quad SPI-FLASH Interface

- Display Interface

SPI Interface up to 40.96 Mbps
8/16/24/32 bpp LCD or E-Ink recommend up to QVGA resolution

- System and I/O Processor (SYSCPU)

Arm® Cortex®-M0+ 32 bit RISC
Operating frequency up to 100 MHz at 1.0 V
256 KByte SRAM
128 KByte ROM for secure booting

- System and IOP Domain multi-layer bus

32 bit Multi-layer bus architecture
SYSIOP for Arm® Cortex®-M0+ 32 bit RISC, PMU, GNSS, Sensor engine, HostIF, Configurable IO

- Power management

I2C and GPIO interface connections to Power Management IC (assuming CXD5247GF)
power on reset
power gate control

- Clock and Reset management

X'tal, RTC, RCOSC, PLL

- 64 KByte Backup SRAM

- Timer

RTC
A general-purpose 32 bit timer each Processor Unit

- Host Interface

I2C, SPI or UART interface
1 KByte Host communication Memory

- Sensor engine

SPI and Two I2C Interfaces
40 KByte Sensory Data FIFO
Pre-processing unit for sensor fusion
Up to Four PWMs

- ADCs

Four channel 10 bit low power ADC
Two channels 10 bit high performance ADC

- Multi-GNSS Controller

Arm® Cortex®-M4 processor with FPU 32 bit RISC
Operating frequency up to 98.208 MHz
64 KByte ROM
640 KByte SRAM
CORDIC engine for GNSS support

- Multi-GNSS receiver

GPS (L1 C/A)
GLONASS (L1OF)
QZSS (L1 C/A, L1 SAIF)
SBAS (L1 C/A)
WAAS, EGNOS, MSAS
BeiDou (B1)
Galileo (E1 CBOC)

- Configurable I/O

I2C/SPI/GPIO Interfaces

- Debug

Serial wire debug (SWD), Embedded Trace Macrocell
UART supported

4 Sensors

4.1 3D accelerometer and 3D gyroscope

4.1.1 General Description

The LSM6DSOX is a system-in-package featuring a 3D digital accelerometer and a 3D digital gyroscope boosting performance at 0.55 mA in high-performance mode and enabling always-on low-power features for an optimal motion experience for the consumer. The LSM6DSOX supports main OS requirements, offering real, virtual and batch sensors with 9 kbytes for dynamic data batching. ST's family of MEMS sensor modules leverages the robust and mature manufacturing processes already used for the production of micromachined accelerometers and gyroscopes. The various sensing elements are manufactured using specialized micromachining processes, while the IC interfaces are developed using CMOS technology that allows the design of a dedicated circuit which is trimmed to better match the characteristics of the sensing element. The LSM6DSOX has a full-scale acceleration range of ±2/±4/±8/±16 g and an angular rate range of ±125/±250/±500/±1000/±2000 dps. The LSM6DSOX fully supports EIS and OIS applications as the module includes a dedicated configurable signal processing path for OIS and auxiliary SPI, configurable for both the gyroscope and accelerometer. The LSM6DSOX OIS can be configured from the Auxiliary SPI and primary interface (SPI / I²C & MIPI I3CSM). High robustness to mechanical shock makes the LSM6DSOX the preferred choice of system designers for the creation and manufacturing of reliable products. The LSM6DSOX is available in a plastic land grid array (LGA) package.

4.1.2 Features

• Power consumption: 0.55 mA in combo high-performance mode

• “Always-on" experience with low power consumption for both accelerometer and gyroscope

• Smart FIFO up to 9 kbyte

• Android compliant

• ±2/±4/±8/±16 g full scale

• ±125/±250/±500/±1000/±2000 dps full scale

• Analog supply voltage: 1.71 V to 3.6 V

• Independent IO supply (1.62 V)

• Compact footprint: 2.5 mm x 3 mm x 0.83 mm

• SPI / I²C & MIPI I3CSM serial interface with main processor data synchronization

• Auxiliary SPI for OIS data output for gyroscope and accelerometer

• OIS configurable from Aux SPI, primary interface (SPI / I²C & MIPI I3CSM)

• Advanced pedometer, step detector and step counter

• Significant Motion Detection, Tilt detection

• Standard interrupts: free-fall, wakeup, 6D/4D orientation, click and double-click

• Programmable finite state machine: accelerometer, gyroscope and external sensors

• Machine Learning Core

• S4S data synchronization

• Embedded temperature sensor

• ECOPACK®, RoHS and “Green” compliant

4.1.3 Schematic connections

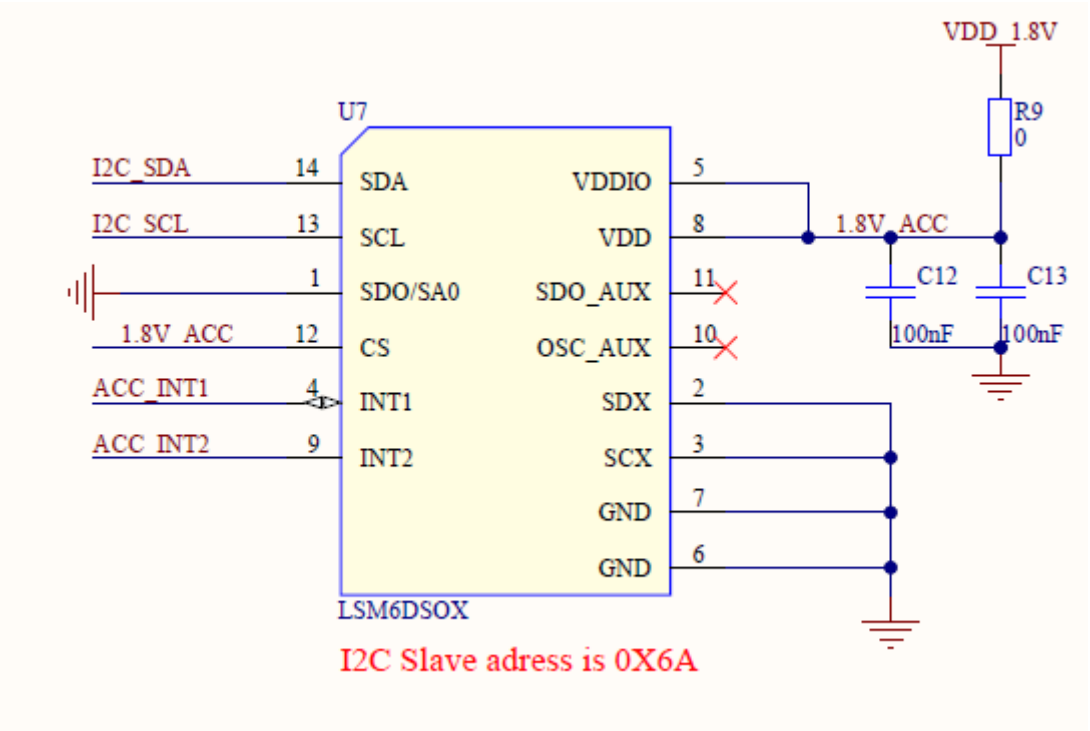

Figure 6. LSM6DSOX connection

For normal sensor work LSM6DSOX is connected to I2C0, for extra functions INT1 is connected to J2.5, and INT2 is connected to P1 of the port expander (Fig. 6).

4.2 3-Axis magnetometer

4.2.1 General description

The LIS2MDL is an ultra-low-power, high-performance 3-axis digital magnetic sensor. The LIS2MDL has a magnetic field dynamic range of ±50 gauss. The LIS2MDL includes an I2C serial bus interface that supports standard, fast mode, fast mode plus, and high-speed (100 kHz, 400 kHz, 1 MHz, and 3.4 MHz) and an SPI serial standard interface.
The device can be configured to generate an interrupt signal for magnetic field detection.
The LIS2MDL is available in a plastic land grid
array package (LGA) and is guaranteed to operate over an extended temperature range from -40 °C to +85 °C.

4.2.2 Features

- 3 magnetic field channels
- ±50 gauss magnetic dynamic range
- 16-bit data output
- SPI/I2C serial interfaces
- Analog supply voltage 1.71 V to 3.6 V
- Selectable power mode/resolution
- Single measurement mode
- Programmable interrupt generator
- Embedded self-test
- Embedded temperature sensor
- ECOPACK®, RoHS and “Green” compliant

4.2.3 Schematic connection

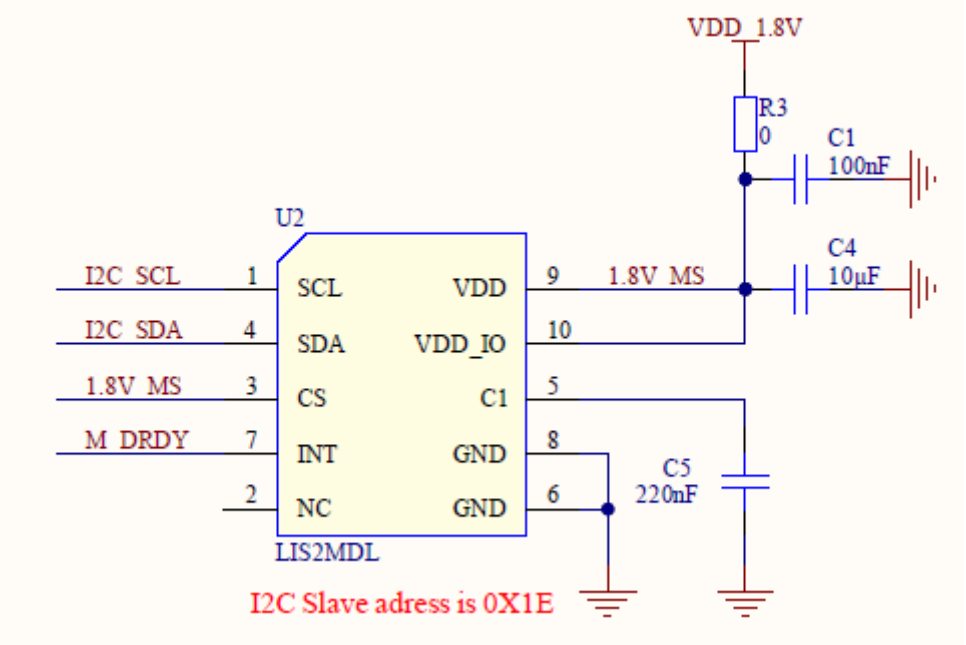

Figure 7. LIS2MDL connection

For normal sensor work LIS2MDL is connected to I2C0, for extra functions INT is connected to UART2_CTS (Fig. 7).

4.3 Humidity and temperature sensor

4.3.1 General description

The HTS221 is an ultra-compact sensor for relative humidity and temperature. It includes a sensing element and a mixed signal ASIC to provide the measurement information through digital serial interfaces. The sensing element consists of a polymer dielectric planar capacitor structure capable of detecting relative humidity variations and is manufactured using a dedicated ST process.
The HTS221 is available in a small top-holed cap land grid array (HLGA) package guaranteed to operate over a temperature range from -40 °C to +120 °C.

4.3.2 Features

- 0 to 100% relative humidity range
- Supply voltage: 1.7 to 3.6 V
- Low power consumption: 2 µA @ 1 Hz ODR
- Selectable ODR from 1 Hz to 12.5 Hz
- High rH sensitivity: 0.004% rH/LSB
- Humidity accuracy: ± 3.5% rH, 20 to +80% rH
- Temperature accuracy: ± 0.5 °C,15 to +40 °C
- Embedded 16-bit ADC
- 16-bit humidity and temperature output data
- SPI and I²C interfaces
- Factory calibrated
- Tiny 2 x 2 x 0.9 mm package
- ECOPACK® compliant

4.3.3 Schematic connection

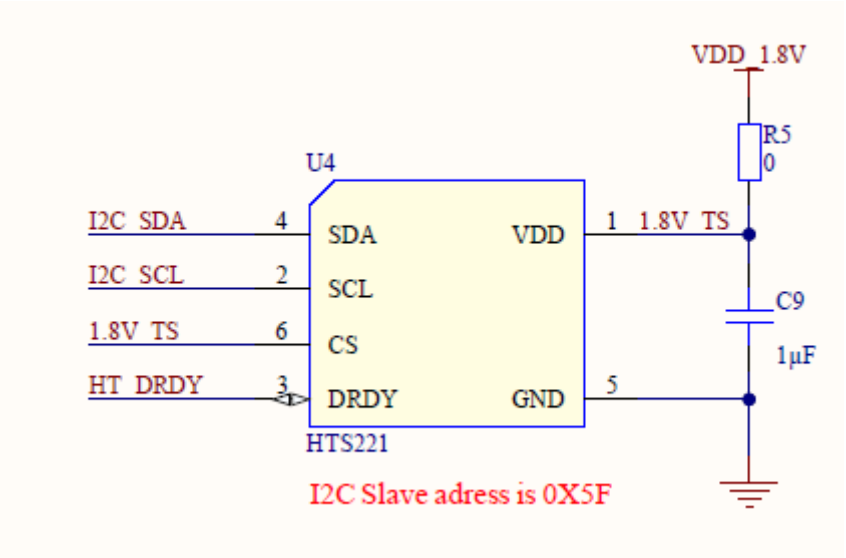

Figure 8. HTS221 connection

For normal sensor work HTS221 is connected to I2C0, for extra functions DRDY is connected to I2S0_DATA_OUT (Fig. 8).

4.4 Pressure sensor

4.4.1 General description

The LPS22HH is an ultra-compact piezoresistive absolute pressure sensor which functions as a digital output barometer. The device comprises a sensing element and an IC interface which communicates through I²C, MIPI I3CSM or SPI from the sensing element to the application. The sensing element, which detects absolute pressure, consists of a suspended membrane manufactured using a dedicated process developed by ST.
The LPS22HH is available in a full-mold, holed LGA package (HLGA). It is guaranteed to operate over a temperature range extending from -40 °C to +85 °C. The package is holed to allow external pressure to reach the sensing element.

4.4.2 Features

- 260 to 1260 hPa absolute pressure range
- Current consumption down to 4 µA
- Absolute pressure accuracy: 0.5 hPa
- Low pressure sensor noise: 0.65 Pa
- High-performance TCO: 0.65 Pa/°C
- Embedded temperature compensation
- 24-bit pressure data output
- ODR from 1 Hz to 200 Hz
- SPI, I²C or MIPI I3CSM interfaces
- Embedded FIFO
- Interrupt functions: Data-Ready, FIFO flags,
- pressure thresholds
- Supply voltage: 1.7 to 3.6 V
- High shock survivability: 22,000 g
- Small and thin package
- ECOPACK® lead-free compliant

4.4.3 Schematic connection

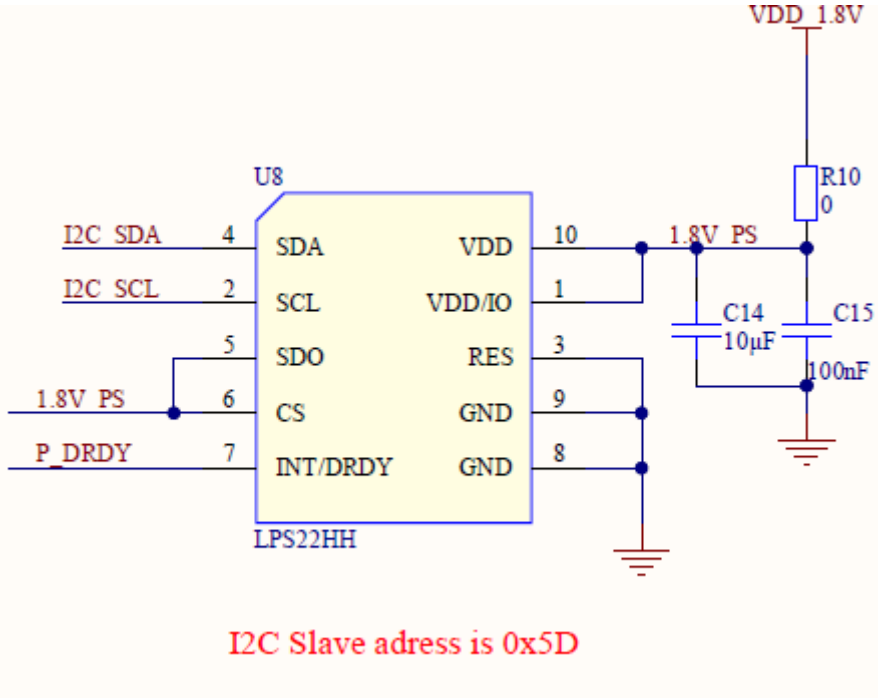

Figure 9. LPS22HH connection

For normal sensor work LPS22HH is connected to I2C0 (Fig. 9).

4.5 Proximity sensor

4.5.1 General description

The VL53L1X is a state-of-the-art, Time-of-Flight (ToF), laser-ranging sensor, enhancing the ST FlightSense product family. It is the fastest miniature ToF sensor on the market with accurate ranging up to 4 m and fast ranging frequency up to 50 Hz
Housed in a miniature and reflowable package, it integrates a SPAD receiving array, a 940 nm invisible Class1 laser emitter, physical infrared filters, and optics to achieve the best ranging performance in various ambient lighting conditions with a range of cover window options.
Unlike conventional IR sensors, the VL53L1X uses ST's latest generation ToF technology which allows absolute distance measurement whatever the target color and reflectance.
It is also possible to program the size of the ROI on the receiving array, allowing the sensor FoV to be reduced.

4.5.2 Features

Fully integrated miniature module
– Size: 4.9x2.5x1.56 mm
– Emitter: 940 nm invisible laser (Class1)
– SPAD (single photon avalanche diode)
receiving array with integrated lens
– Low-power microcontroller running advanced
digital firmware
Pin-to-pin compatible with the VL53L0X
FlightSense ranging sensor
Fast and accurate long distance ranging
– Up to 400 cm distance measurement
– Up to 50 Hz ranging frequency
Typical full field-of-view (FoV): 27°
Programmable region-of-interest (ROI) size on
the receiving array, allowing the sensor FoV to be
reduced
Programmable ROI position on the receiving
array, providing multizone operation control from
the host
Easy integration
– Single reflowable component
– Can be hidden behind many cover window
materials
– Software driver and code examples for
turnkey ranging
– Single power supply (2v8)
– I²C interface (up to 400 kHz)

4.5.3 Schematic connection

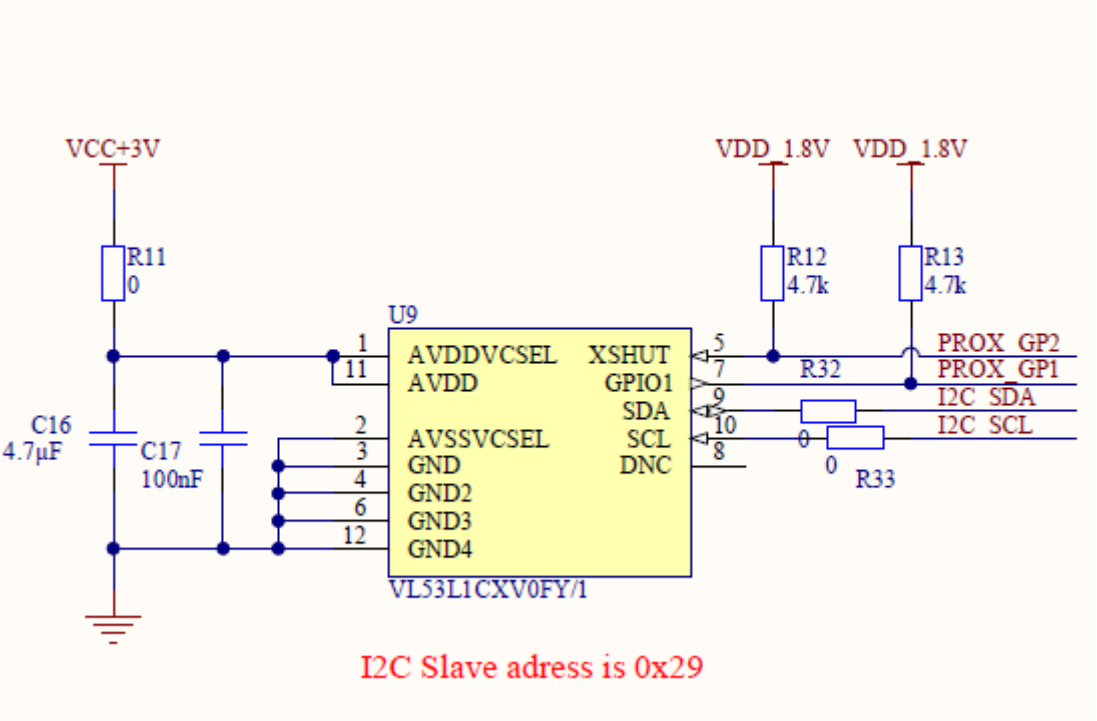

Figure 10. VL53L1X connection

For normal sensor work VL53L1X connected to I2C0, for extra functions XSHUT connected to UART2_RTS and GPIO1 connected to J2.4 (Fig. 10).

4.6 Air quality sensor

4.6.1 General description

The SGP40 is a digital gas sensor designed for easy integration into air purifier s or demand controlled ventilation systems. Sensirion’s CMOSens ® technology offers a complete , easy to use sensor system on a single chip featuring a digital I 2 C interface and a temperature controlle d micro hotplate, providing a humidity compensated VOC based indoor air quality signal . The output signal can be directly processed by Sensirion’s powerful VOC Algorithm to translate the raw signal into a VOC Index as a robust measure for indoor air quality. The VOC Algorithm automatically adapts to the environment the sensor is exposed to. Both sensing element and
VOC Algorithm feature an unmatched robustness against contaminating gases present in real world applications enabling a unique long term stability as well as low drift and device to device variation. The very small 2.44 x 2.44 x 0.85 mm 3 DFN package enables applications in limited spaces. Sensirion’s state of the art production process guarantees high reproducibility and reliability. Tape and reel packaging together with suitability for standard SMD a ssembly processes make the SGP40 prede stined for high volume applications.

4.6.2 Schematic connection

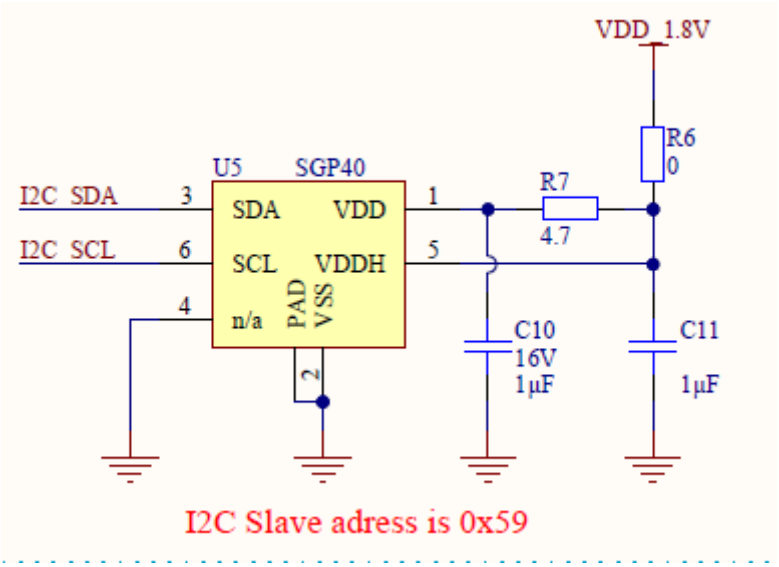

Figure 11. SGP40 connection

For normal sensor work SGP40 connected to I2C0 (Fig. 11).

4.7 Digital RGB, IR and Ambient Light Sensor

4.7.1 General description

The APDS-9250 device uses 4 individual channels of red, green, blue, and IR (RGB+IR) in a specially designed matrix arrangement. This allows the device to have optimal angular response and accurate RGB spectral response with high lux accuracy over various light sources. APDS-9250 supports the I2C interface and has a programmable interrupt controller that frees up micro-controller resources.
The device detects light intensity under a variety of lighting conditions and through a variety of attenuation materials, including dark glass. APDS-9250 could be configured as Ambient Light Sensor and RGB+IR Sensor. The color-sensing feature is useful in applications such as LED RGB backlight control, solid-state lighting, reflected LED color sampler, or fluorescent light color temperature detection. The integrated IR blocking filter makes this device an excellent ambient light sensor and color temperature monitor sensor together with the temperature compensation that allows output to have less variation over the temperature.

4.7.2 Features

• Colour and Ambient Light Sensing (CS-RGB and ALS)
- Accuracy of Correlated Color Temperature (CCT)
- Individual channels for Red, Green, Blue and Infared
- Approximates Human Eye Response with Green Channel
- Red, Green, Blue, Infrared and ALS Sensing
- High Sensitivity in low lux condition – Ideally suited for Operation Behind Dark Glass
- Wide Dynamic Range: 18,000,000: 1
- Up to 20-Bit Resolution
• Power Management
- Low Active Current – 130 µA typical
- Low Standby Current – 1µA typical
• I2C-bus Fast Mode Compatible Interface
- Up to 400 kHz (I2C Fast-Mode)
- Dedicated Interrupt Pin
• Small Package L 2.0 × W 2.0 × H 0.65 mm

4.7.3 Schematic connection

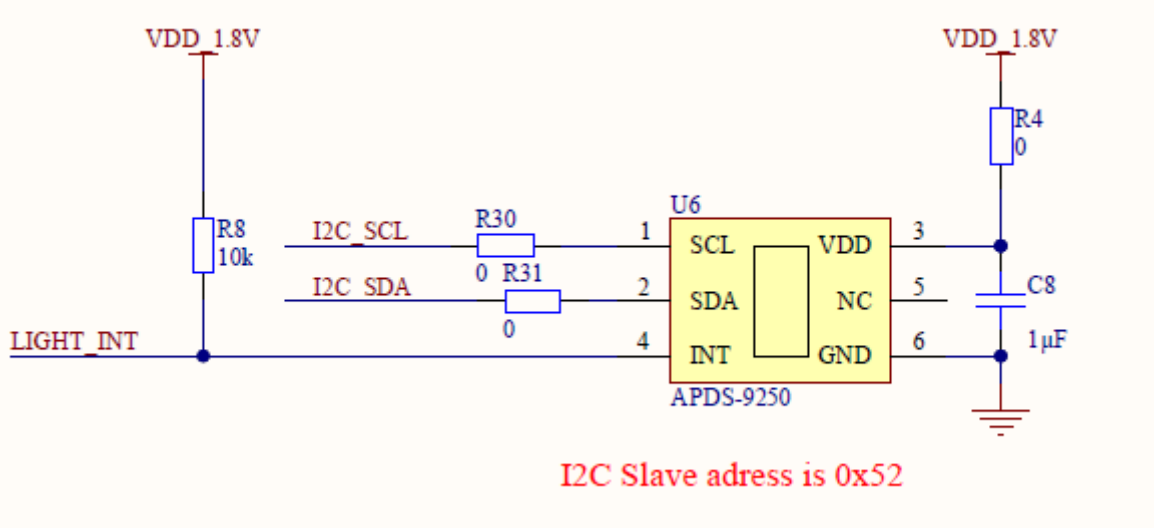

Figure 12. APDS-9250 connection

For normal sensor work APDS-9250 connected to I2C0, for extra functions INTconnected to UART2_TX (Fig.12).

4.8 Port expander

4.8.1 General description

The PCA9538 is an 8-bit I/O expander of general purpose parallel input and output (I/O) expansion for the two-line bidirectional I2C bus (or SMBus) protocol. This device can operate with a power supply range from 2.3 V to 5.5 V. This device supports both100-kHz (Standard-mode) and 400-kHz (Fast-mode) clock frequencies. This device, along with other I/O expanders, provides a simple solution when additional I/Os are needed for switches, sensors, push-buttons, LEDs, fans, and so on.
The features of PCA9538 include an interrupt that is generated on the INT pin whenever an input port changes state. The A0 and A1 hardware selectable address pins allow up to four PCA9538 devices on the same I2C bus. This device can also be reset to its default state by using the RESET feature or by cycling the power supply to cause a power-on reset.
INT can be connected to the interrupt input of a microcontroller. By sending an interrupt signal on this line, the remote I/O can inform the microcontroller if there is incoming data on its ports without having to communicate via the I2C bus. Thus, the PCA9538 can remain a simple slave device.
The device outputs (latched) have high-current drive capability for directly driving LEDs. It has low current consumption.
Two hardware pins (A0 and A1) are used to program and vary the fixed I2C address and allow up to four devices to share the same I2C bus or SMBus.

4.8.2 Features

• Low standby current consumption of 1 μA max
• I2C to parallel port expander
• Open-drain active-low interrupt output
• Active-low reset input
• Operating power-supply voltage range of 2.3 V to 5.5 V
• 5-V Tolerant I/O ports
• 400-kHz Fast I2C bus
• Two hardware address pins allow up to four devices on the I2C/SMBus
• Input and output configuration register
• Polarity inversion register
• Power-up with all channels configured as inputs
• No glitch on power up
• Noise filter on SCL/SDA inputs
• Latched outputs with high-current drive maximum capability for directly driving LEDs
• Latch-up performance exceeds 100 mA Per JESD 78, class II
• ESD protection exceeds JESD 22

– 2000-V Human-body model (A114-A)
– 200-V Machine model (A115-A)
– 1000-V Charged-device model (C101)

4.8.3 Schematic connection

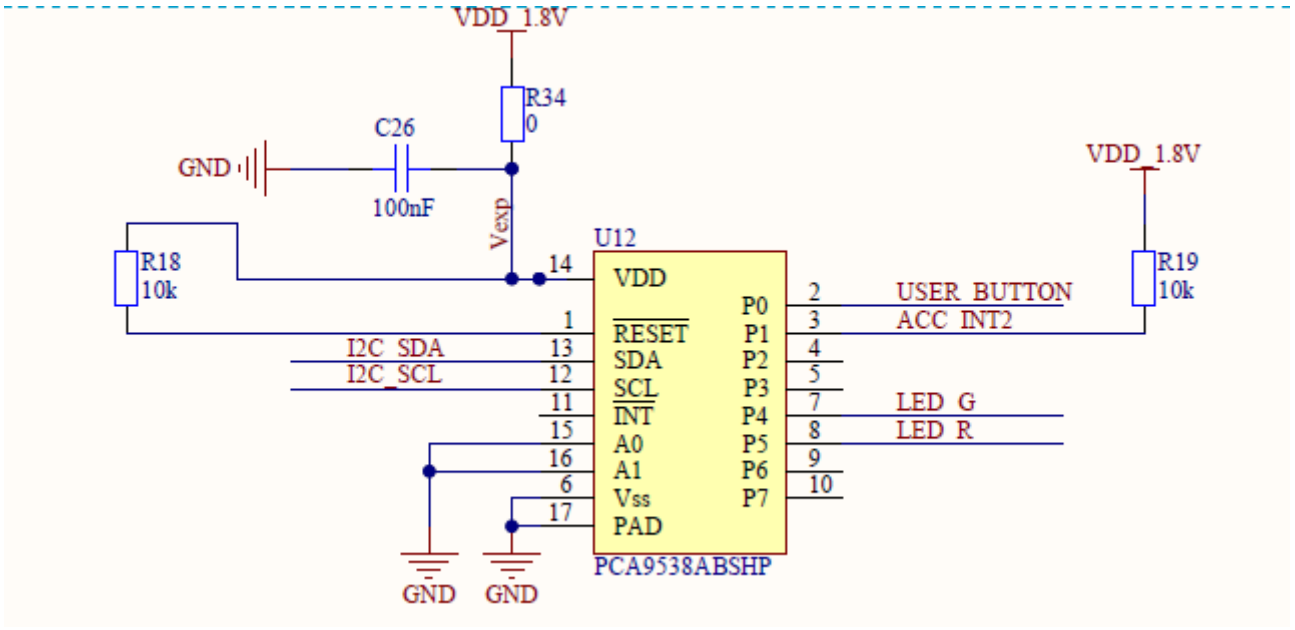

Figure 13. APDS-9250 connection

For normal sensor work PCA9538 is connected to I2C0 (Fig.13).

4.9 Micro SD card

4.9.1 General description

Should be used micro SD card 3.3V. Software developed and tested with Kingston micro SD card with 16 Gb of memory. The system might not work with some kinds of micro SD cards.

4.9.2 Schematic connection

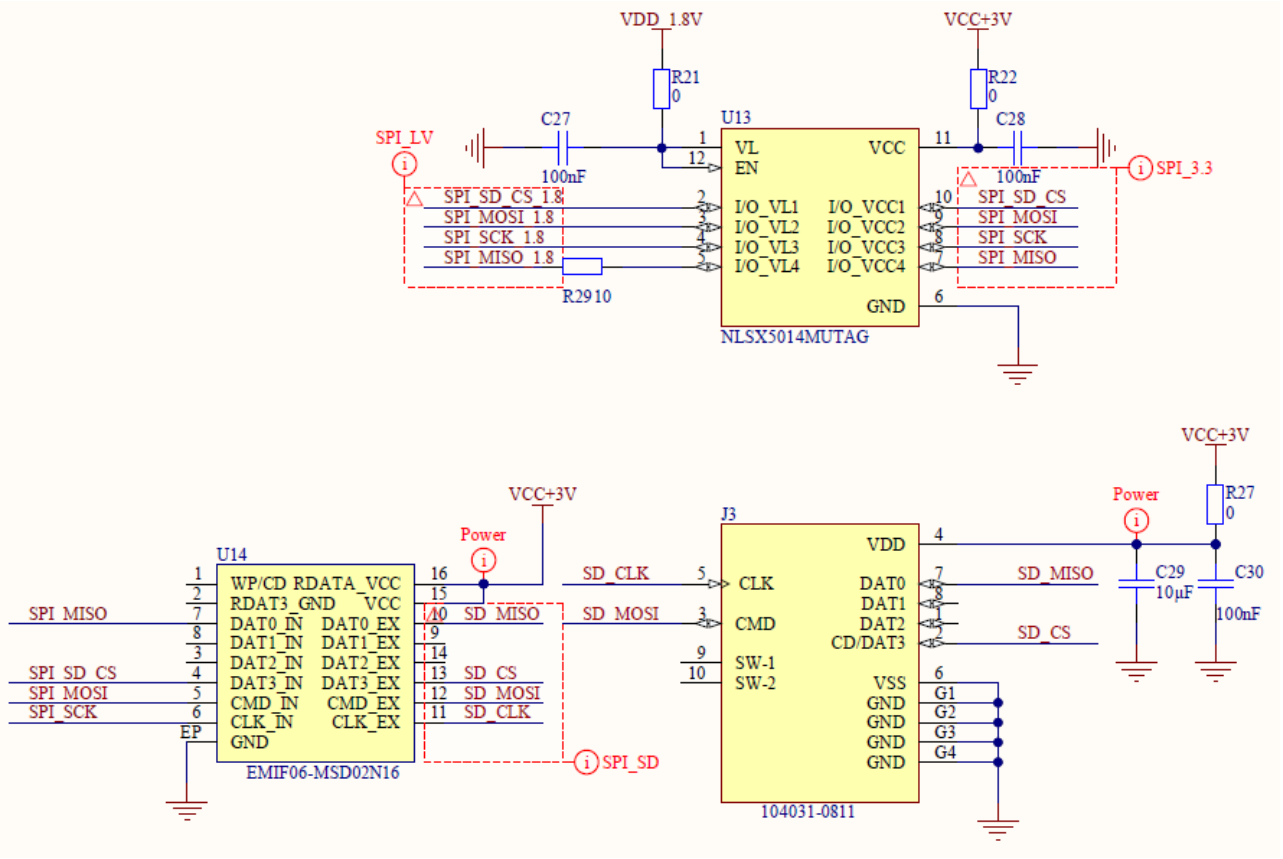

Figure 14. micro SD card connection

For normal sensor work micro SD card connected to SPI5, micro SD CS connected to SPI5_CS_X (Fig.14).

4.10 M24C32 EEPROM

4.10.1 General description

The M24C32 is a 32-Kbit I2C-compatible EEPROM (Electrically Erasable PROgrammable Memory) organized as 4 K × 8 bits.
The M24C32-W can operate with a supply voltage from 2.5 V to 5.5 V, the M24C32-R can operate with a supply voltage from 1.8 V to 5.5 V, and the M24C32-F and M24C32-DF can operate with a supply voltage from 1.7 V to 5.5 V, over an ambient temperature range of -40 °C / +85 °C; while the M24C32-X can operate with a supply voltage from 1.6 V to 5.5 V over an ambient temperature range of -20 °C / +85 °C.
The M24C32-D offers an additional page, named the Identification Page (32 byte). The Identification Page can be used to store sensitive application parameters which can be (later) permanently locked in Read-only mode.

4.10.2 Features

• Compatible with all I2C bus modes:
– 1 MHz
– 400 kHz
– 100 kHz
• Memory array:
– 32 Kbit (4 Kbyte) of EEPROM
– Page size: 32 byte
– Additional Write lockable page (M24C32-D
order codes)
• Single supply voltage:
– 1.7 V to 5.5 V over –40 °C / +85 °C
– 1.6 V to 5.5 V over –20 °C / +85 °C
• Write:
– Byte Write within 5 ms
– Page Write within 5 ms
• Random and sequential Read modes
• Write protect of the whole memory array
• Enhanced ESD/Latch-Up protection
• More than 4 million Write cycles
• More than 200-years data retention

4.10.3 Schematic connection

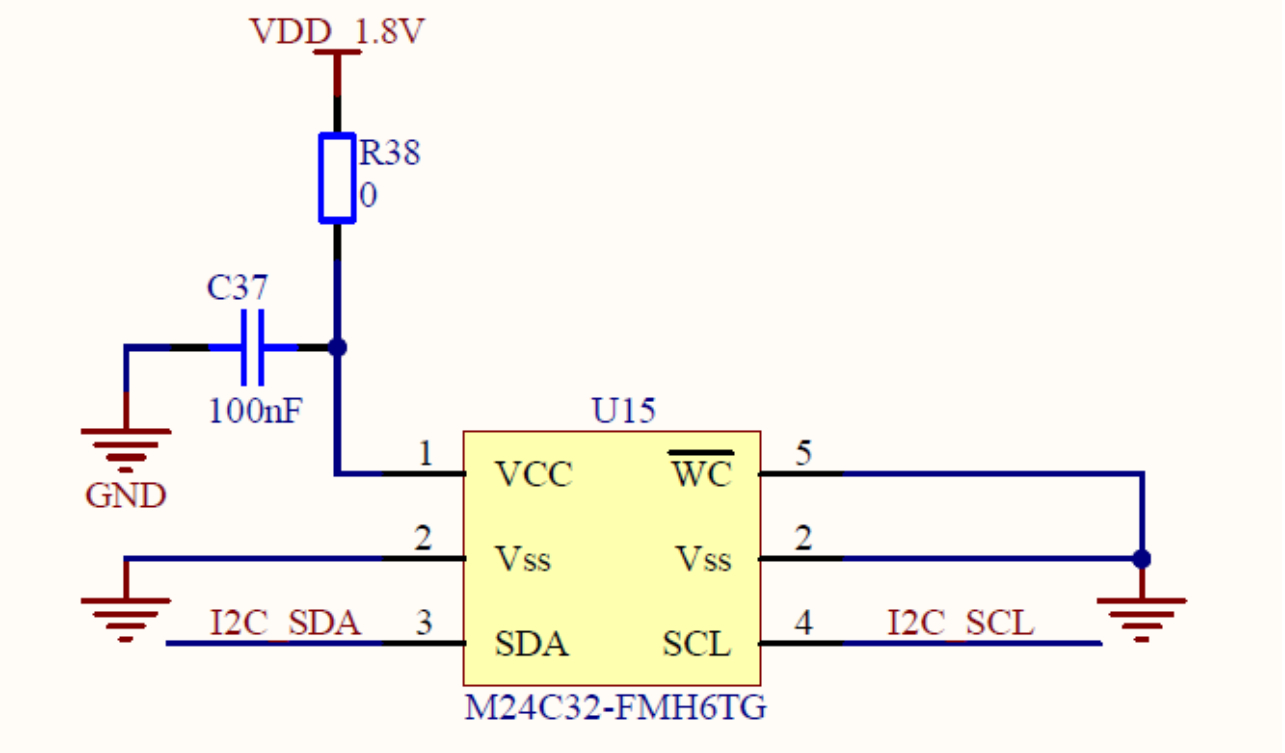

Figure 15. M24C32 connection

For normal sensor work M24C32 connected to I2C0 (Fig.15).

4.11 BQ27441DRZR Battery monitor

4.11.1 General description

The Texas Instruments bq27441-G1 fuel gauge is a microcontroller peripheral that provides system-side fuel gauging for single-cell Li-Ion batteries. The device requires minimal user configuration and system microcontroller firmware development.
The bq27441-G1 battery fuel gauge uses the patented Impedance Track™ algorithm for fuel gauging, and provides information such as remaining battery capacity (mAh), state-of-charge (%), and battery voltage (mV).
Battery fuel gauging with the bq27441-G1 fuel gauge requires connections only to PACK+ (P+) and PACK– (P–) for a removable battery pack or embedded battery circuit. The tiny, 12-pin, 2.50 mm × 4.00 mm, small outline no-lead (SON) package is ideal for space-constrained applications.

4.11.2 Features

• Single Series Cell Li-Ion Battery Fuel Gauge
– Resides on System Board
– Supports Embedded or Removable Batteries
– Powered Directly from Battery with Integrated LDO
– Supports a Low-Value External Sense Resistor(10 mΩ)
• Battery Fuel Gauging Based on Patented Impedance Track™ Technology
– Reports Remaining Capacity and State-of-Charge (SOC) with Smoothing Filter
– Automatically Adjusts for Battery Aging, Self-discharge, Temperature, and Rate Changes
– Battery State-of-Health (Aging) Estimation
• Microcontroller Peripheral Supports:
– 400-kHz I2C Serial Interface
– Configurable SOC Interrupt or
Battery Low Digital Output Warning
– Internal Temperature Sensor or
Host-Reported Temperature

4.11.3 Schematic connection

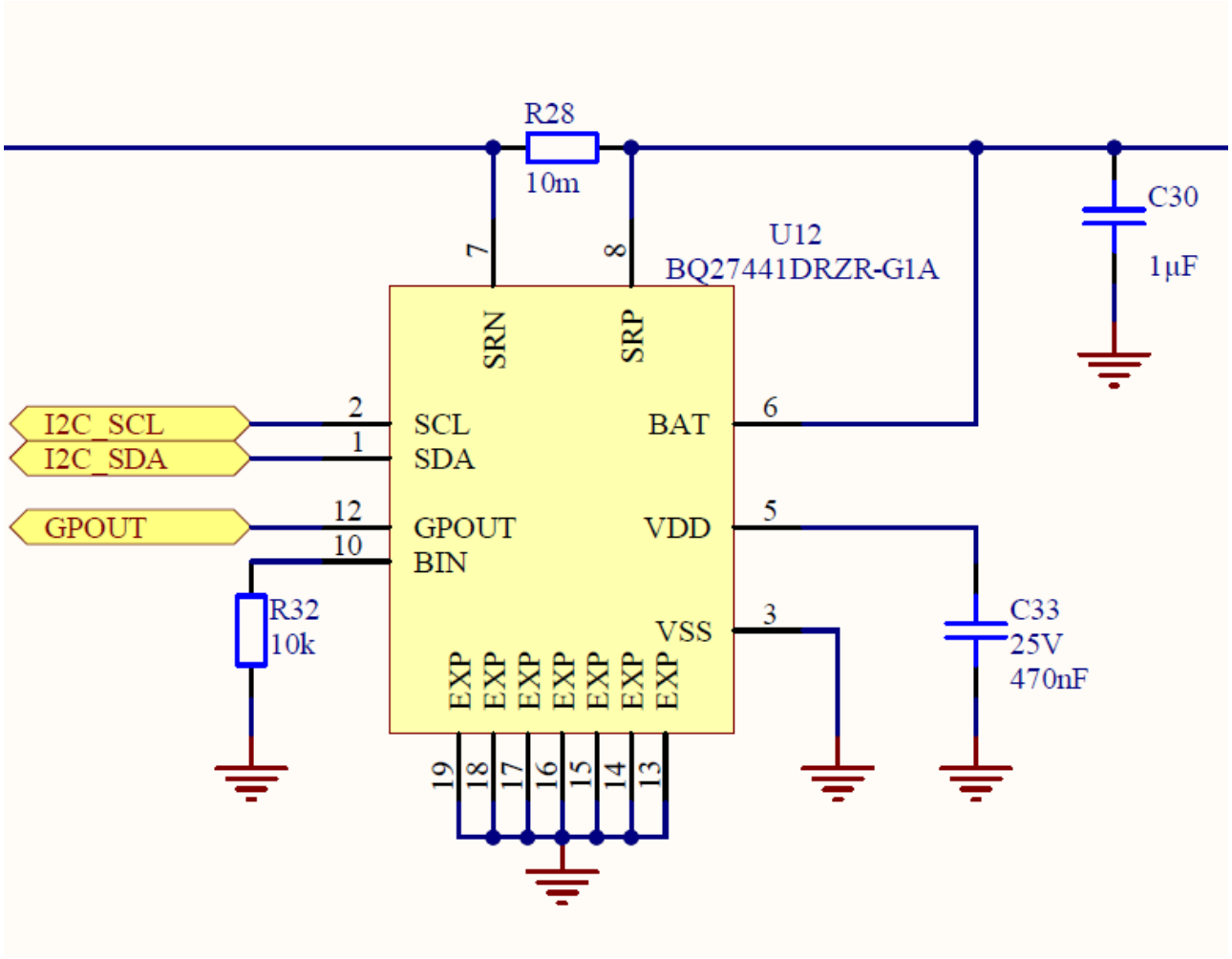

Figure 16. BQ27441 connection

For normal sensor work BQ27441 connected to I2C0, for extra functions GPOUT to P3 of port expander (Fig.16).

4.12 SPH0645LM4H microphone

4.12.1 General description

The SPH0645LM4H-1 is a miniature, low power, bottom port microphone with an I2S digital output. The solution consists of a proven high performance SiSonic™ acoustic sensor, a serial Analog to Digital convertor, and an interface to condition the signal into an industry standard 24-bit I2S format. The I2S interface simplifies the integration in the system and allows direct interconnect to digital processors, application processors and microcontrollers. Saving the need of an external audio codec, the SPH0645LM4H-1 is perfectly suitable for portable applications where size and power consumption are a constraint.

4.12.2 Features

• High SNR of 65dB(A)
• Low Current of typ. 600µA
• I2S Output: Direct attach to µP
• Multi modes: standard >1MHz,
• 600uA / sleep <1kHz, 3uA
• Flat Frequency Response
• RF Shielded
• Supports Dual Microphones
• Ultra-Stable Performance
• Standard SMD Reflow
• Omnidirectional
• Bottom-Ported Microphone
• 3.50 x 2.65 x 0.98 mm SPH Package Size

4.12.3 Schematic connection

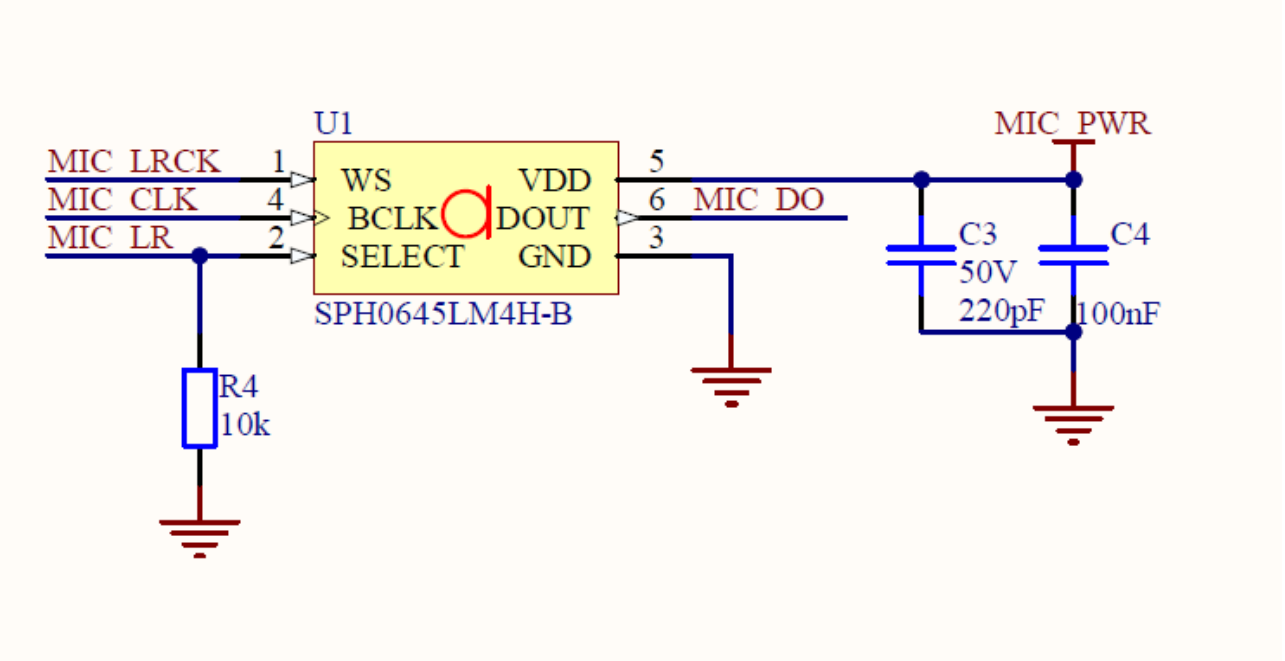

Figure 17. SPG0645LM4H connection

For normal work SPG0645LM4H connected to I2S0 (Fig.17).

4.13 LTC4001EUF Battery charger

4.13.1 General description

The LTC® 4001 is a 2A Li-Ion battery charger intended for 5V wall adapters. It utilizes a 1.5MHz synchronous buck converter topology to reduce power dissipation during charging. Low power dissipation, an internal MOSFET and sense resistor allow a physically small charger that can be embedded in a wide range of handheld applications. The LTC4001 includes complete charge termination circuitry, automatic recharge and a ±1% 4.2V float voltage. Input short-circuit protection is included so no blocking diode is required. Battery charge current, charge timeout and end-of-charge indication parameters are set with external components. Additional features include shorted cell detection, temperature qualified charging and overvoltage protection. The LTC4001 is available in a low profile (0.75mm) 16-lead (4mm × 4mm) QFN package.

4.13.2 Features

- Low Power Dissipation
- 2A Maximum Charge Current
- No External MOSFETs, Sense Resistor or Blocking Diode Required
- Remote Sensing at Battery Terminals
- Programmable Charge Termination Timer
- Preset 4.2V Float Voltage with ±0.5% Accuracy
- Programmable Charge Current Detection/Termination
- Automatic Recharge
- Thermistor Input for Temperature Qualified Charging
- Compatible with Current Limited Wall Adapters
- Low Profile 16-Lead (4mm × 4mm) QFN Package

4.13.3 Schematic connection

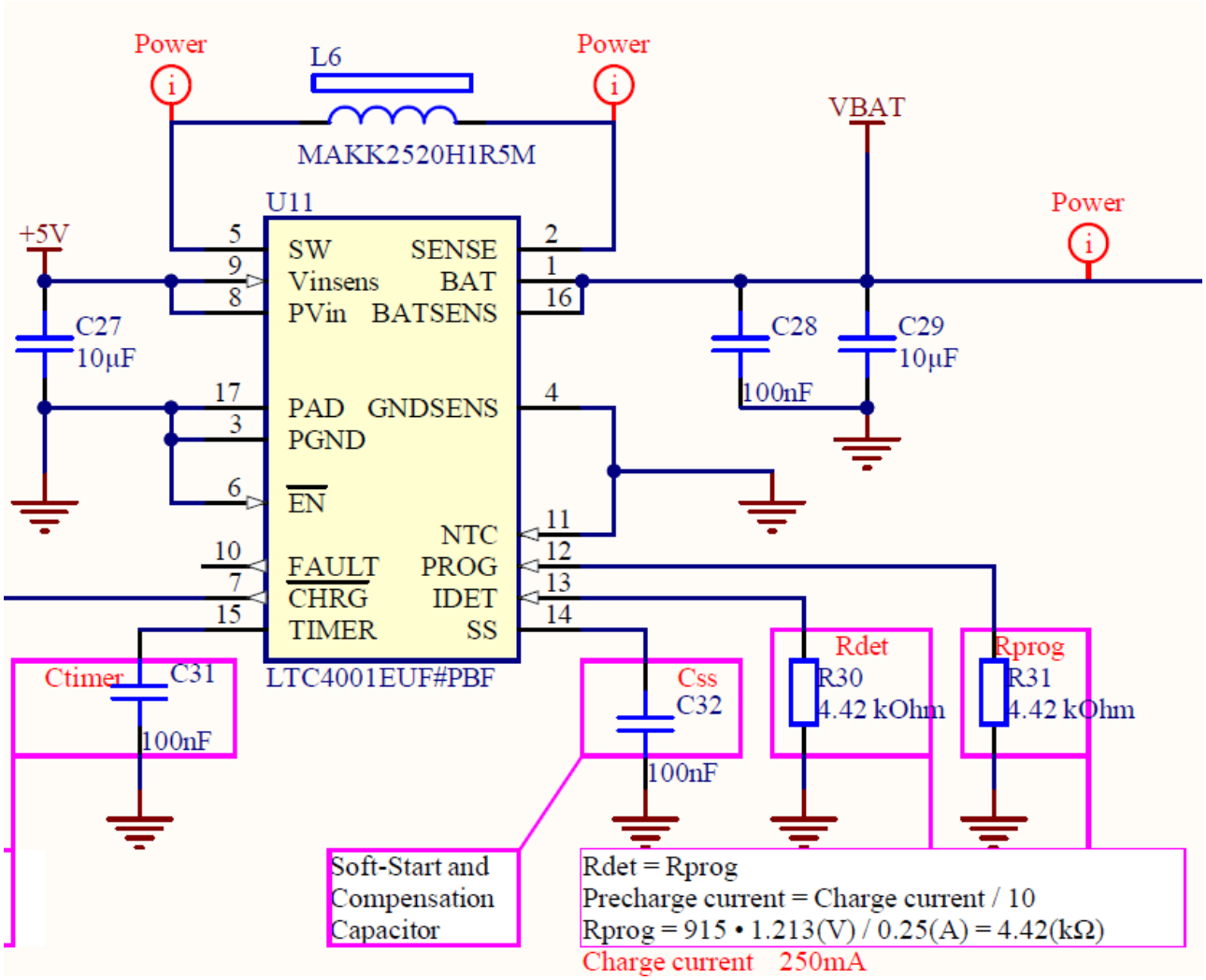

Figure 18. LTC4001EUF connection

LTC4001EUF connected to power source and battery, and has no any data connection (Fig.18).

4.14 Connectors, LED and sound information

There are two connectors:
1 - micro USB on Sony board (Fig 19);
2 - type C on the sensor board (Fig 20).

Micro USB connector options:

- power the device;
- firmware downloading.

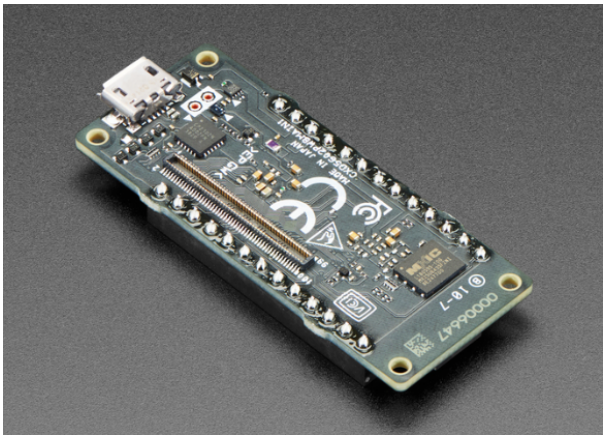

Figure 19. Micro USB on Sony board

Type C connector options:

- battery charging.

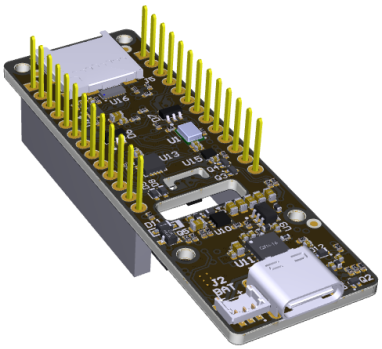

Figure 20. Type C connector on the sensor board

There are few LEDs on the sensor board. One paired LED with green and red colors. It works during battery charging. Red while the battery is charging up and green when the battery is charged. The other paired LED with green and red colors.The red LED blinking once in demo mode. If the device in measure mode and measurements doesn't run the green led blinking once. If the measurement runs, the green and red LEDs are blinks. The device modes are also described in (8. Functional description). Sound plays once during device run. Once in measure mode during running and stopping. Three short beeps if error.

5 Examples of use

In this part added some examples of usage of onboard devices. All tests are enabled in default project and going one by one. For separate test you have to comment “#define ALL_TESTS” in Tests.cpp file and uncomment just one from defines below.
If you want to use some equipment in your app see recommendations below.

5.1 LEDs

LEDs works with port expander
For working with LEDs you have to include Led.h header file.
After that you have to init the LEDs:

```
led_init();
```

After that you can use for functions to control LEDs:

```
led_green_off();
led_red_on();
led_red_off();
led_green_on();
```

5.2 Button

Button works with port expander
For working with button you have to include Led.h header file.
After that you have to init the button:

```
button_init ();
```

After that you can check button state with:

```
button_is_pressed();
```

5.3 Speaker

The speaker works with PWM
For work with PWM you have to include Pwm.h
To set PWM parameters you have to use:

```
pwm_run(10, 20, 500);
```

where 10 – frequency 1 – 1000 Hz, 20 – duty cycle 1 – 100 %, 500 – time in ms.

For PWM work

```
    pwm_act();
```

should be run in loop, when PWM is finished function returns false.

5.4 HTS221TR: humidity and temperature

To run HTS221 you have to include Hts221.h and I2c.h

Variables example and other functions:

```
typedef struct {
    float x0;
    float y0;
    float x1;
    float y1;
} lin_t;

static uint8_t hts221_whoamI = 0;
static int16_t hts221_data_raw_humidity;
static int16_t hts221_data_raw_temperature;
static float hts221_humidity_perc;
static float hts221_temperature_degC;
static lin_t lin_hum;
static lin_t lin_temp;

static float linear_interpolation(lin_t *lin, int16_t x) {
  return ((lin->y1 - lin->y0) * x + ((lin->x1 * lin->y0) -
                                     (lin->x0 * lin->y1)))
         / (lin->x1 - lin->x0);
}
```

To init the device you can use:

```
    i2c_init();
    hts221_device_id_get(nullptr, &hts221_whoamI);
    printf("hts221_whoamI = %d\r\n", hts221_whoamI);

    /* Read humidity calibration coefficient */
    hts221_hum_adc_point_0_get(nullptr, &lin_hum.x0);
    hts221_hum_rh_point_0_get(nullptr, &lin_hum.y0);
    hts221_hum_adc_point_1_get(nullptr, &lin_hum.x1);
    hts221_hum_rh_point_1_get(nullptr, &lin_hum.y1);
    /* Read temperature calibration coefficient */
    hts221_temp_adc_point_0_get(nullptr, &lin_temp.x0);
    hts221_temp_deg_point_0_get(nullptr, &lin_temp.y0);
    hts221_temp_adc_point_1_get(nullptr, &lin_temp.x1);
    hts221_temp_deg_point_1_get(nullptr, &lin_temp.y1);
```

```
    /* Enable Block Data Update */
    hts221_block_data_update_set(nullptr, PROPERTY_ENABLE);
    /* Set Output Data Rate */
    hts221_data_rate_set(nullptr, HTS221_ODR_1Hz);
    /* Device power on */
    hts221_power_on_set(nullptr, PROPERTY_ENABLE);
```

For get data from senor you can use:

```
    /* Read output only if new value is available */
    hts221_reg_t reg;
    hts221_status_get(nullptr, &reg.status_reg);

    if (reg.status_reg.h_da) {
        /* Read humidity data */
        memset(&hts221_data_raw_humidity, 0x00, sizeof(int16_t));
        hts221_humidity_raw_get(nullptr, (uint8_t*)
&hts221_data_raw_humidity);
        hts221_humidity_perc = linear_interpolation(&lin_hum,
hts221_data_raw_humidity);
        printf("Hymidity_raw = %d\r\n", hts221_data_raw_humidity);
        hts221_humidity_perc = 0.0 - hts221_humidity_perc;
        if (hts221_humidity_perc < 0) {
            hts221_humidity_perc = 0;
        }

        if (hts221_humidity_perc > 100) {
            hts221_humidity_perc = 100;
        }

        printf("Humidity [%%]:%3.2f\r\n", hts221_humidity_perc);
        reg.status_reg.h_da = 0;
    }

    if (reg.status_reg.t_da) {
        /* Read temperature data */
        memset(&hts221_data_raw_temperature, 0x00,
sizeof(int16_t));
        hts221_temperature_raw_get(nullptr, (uint8_t*)
&hts221_data_raw_temperature);
        hts221_temperature_degC = linear_interpolation(&lin_temp,
hts221_data_raw_temperature);
        printf("Temperature [degC]:%6.2f\r\n",
hts221_temperature_degC );
        reg.status_reg.t_da = 0;
    }
```

5.5 LIS2MDL: 3-Axis Magnetometer

To work with LIS2MDL you have to include Lis2mdl.h and I2c.h

Variables example:

```
static int16_t data_raw_magnetic[3];
static int16_t lis2_data_raw_temperature;
static float magnetic_mG[3];
static float lis2_temperature_degC;
static uint8_t lis2_whoamI, lis2_rst;
```

To intialisation you may use:

```
    i2c_init();
    /* Check device ID */
    lis2mdl_device_id_get(nullptr, &lis2_whoamI);

    printf("LIS2MDL_ID = %d\r\n", lis2_whoamI);

    if (lis2_whoamI != LIS2MDL_ID) {
        while (1) {
            /* manage here device not found */
        }
    }

    /* Restore default configuration */
    lis2mdl_reset_set(nullptr, PROPERTY_ENABLE);

    do {
        lis2mdl_reset_get(nullptr, &lis2_rst);
    } while (lis2_rst);

    /* Enable Block Data Update */
    lis2mdl_block_data_update_set(nullptr, PROPERTY_ENABLE);
    /* Set Output Data Rate */
    lis2mdl_data_rate_set(nullptr, LIS2MDL_ODR_10Hz);
    /* Set / Reset sensor mode */
    lis2mdl_set_rst_mode_set(nullptr,
LIS2MDL_SENS_OFF_CANC_EVERY_ODR);
    /* Enable temperature compensation */
    lis2mdl_offset_temp_comp_set(nullptr, PROPERTY_ENABLE);
    /* Set Low-pass bandwidth to ODR/4 */
    //lis2mdl_low_pass_bandwidth_set(nullptr, LIS2MDL_ODR_DIV_4);
    /* Set device in continuous mode */
    lis2mdl_operating_mode_set(nullptr, LIS2MDL_CONTINUOUS_MODE);
    /* Enable interrupt generation on new data ready */
    lis2mdl_drdy_on_pin_set(nullptr, PROPERTY_ENABLE);
```

To get data and pint:

```
    uint8_t reg;
    /* Read output only if new value is available */
    lis2mdl_mag_data_ready_get(nullptr, &reg);

    if (reg) {
        /* Read magnetic field data */
```

```
        memset(data_raw_magnetic, 0x00, 3 * sizeof(int16_t));
        lis2mdl_magnetic_raw_get(nullptr, (uint8_t*)
data_raw_magnetic);
        magnetic_mG[0] = lis2mdl_from_lsb_to_mgauss(
data_raw_magnetic[0]);
        magnetic_mG[1] = lis2mdl_from_lsb_to_mgauss(
data_raw_magnetic[1]);
        magnetic_mG[2] = lis2mdl_from_lsb_to_mgauss(
data_raw_magnetic[2]);
        printf("Mag field [mG]:%4.2f\t%4.2f\t%4.2f\r\n",
magnetic_mG[0], magnetic_mG[1], magnetic_mG[2]);
        /* Read temperature data */
        memset(&lis2_data_raw_temperature, 0x00, sizeof(int16_t));
        lis2mdl_temperature_raw_get(nullptr, (uint8_t*)
&lis2_data_raw_temperature);
        lis2_temperature_degC =
        lis2mdl_from_lsb_to_celsius(lis2_data_raw_temperature);
        printf("Temperature [degC]:%6.2f\r\n",
lis2_temperature_degC);
```

5.6 LPS22HH: pressure sensor

To work with LPS22HH you have to include Lps22hh.h and I2c.h

Variables example:

```
static uint32_t data_raw_pressure;
static int16_t lps22_data_raw_temperature;
static float pressure_hPa;
static float lps22_temperature_degC;
static uint8_t lps22_whoamI, lps22_rst;
```

To initialization:

```
    i2c_init();
    /* Check device ID */
    lps22_whoamI = 0;
    lps22hh_device_id_get(nullptr, &lps22_whoamI);

    printf("LPS22HH_ID = %d", lps22_whoamI);

    if (lps22_whoamI != LPS22HH_ID ) {
        printf("LPS22HH_ID Init Error\r\n");
        while (1); /*manage here device not found */
    }

    /* Restore default configuration */
    lps22hh_reset_set(nullptr, PROPERTY_ENABLE);

    do {
```

```
        lps22hh_reset_get(nullptr, &lps22_rst);
    } while (lps22_rst);

    /* Enable Block Data Update */
    lps22hh_block_data_update_set(nullptr, PROPERTY_ENABLE);
    /* Set Output Data Rate */
    lps22hh_data_rate_set(nullptr, LPS22HH_10_Hz_LOW_NOISE);
```

To get data and pint:

```
    lps22hh_reg_t reg;
    /* Read output only if new value is available */
    lps22hh_read_reg(nullptr, LPS22HH_STATUS, (uint8_t *)&reg, 1);

    if (reg.status.p_da) {
        memset(&data_raw_pressure, 0x00, sizeof(uint32_t));
        lps22hh_pressure_raw_get(nullptr, (uint8_t *)
&data_raw_pressure);
        pressure_hPa = lps22hh_from_lsb_to_hpa(
data_raw_pressure);
        printf("pressure [hPa]:%6.2f\r\n", pressure_hPa);
    }

    if (reg.status.t_da) {
        memset(&lps22_data_raw_temperature, 0x00,
sizeof(int16_t));
        lps22hh_temperature_raw_get(nullptr, (uint8_t *)
&lps22_data_raw_temperature);
        lps22_temperature_degC = lps22hh_from_lsb_to_celsius(
                                 lps22_data_raw_temperature );
        printf("temperature [degC]:%6.2f\r\n",
lps22_temperature_degC );
    }
```

5.7 LSM6DS3: inertial module: 3D accelerometer and 3D gyroscope

To work with LSM6DS3 you have to include Lsm6dso32.h and I2c.h

Variables example:

```
static int16_t data_raw_acceleration[3];
static int16_t data_raw_angular_rate[3];
static int16_t lsm6_data_raw_temperature;
static float acceleration_mg[3];
static float angular_rate_mdps[3];
static float lsm6_temperature_degC;
static uint8_t lsm6_whoamI, rst;
static uint8_t tx_buffer[1000];
static stmdev_ctx_t dev_ctx;
```

To initialization:

```
    i2c_init();
    /* Check device ID */
    lsm6dso32_device_id_get(&dev_ctx, &lsm6_whoamI);

    printf("LSM6DS32_ID = %d\r\n", lsm6_whoamI);

    if (lsm6_whoamI != LSM6DSO32_ID)
    while (1);

    /* Restore default configuration */
    lsm6dso32_reset_set(&dev_ctx, PROPERTY_ENABLE);

    do {
        lsm6dso32_reset_get(&dev_ctx, &rst);
    } while (rst);

    /* Disable I3C interface */
    lsm6dso32_i3c_disable_set(&dev_ctx, LSM6DSO32_I3C_DISABLE);
    /* Enable Block Data Update */
    lsm6dso32_block_data_update_set(&dev_ctx, PROPERTY_ENABLE);
    /* Set full scale */
    lsm6dso32_xl_full_scale_set(&dev_ctx, LSM6DSO32_4g);
    lsm6dso32_gy_full_scale_set(&dev_ctx, LSM6DSO32_2000dps);
    /* Set ODR (Output Data Rate) and power mode*/
    lsm6dso32_xl_data_rate_set(&dev_ctx,
LSM6DSO32_XL_ODR_12Hz5_LOW_PW);
    lsm6dso32_gy_data_rate_set(&dev_ctx,
                               LSM6DSO32_GY_ODR_12Hz5_HIGH_PERF);
```

To get data and pint:

```
    lsm6dso32_reg_t reg;
    /* Read output only if new data is available */
    lsm6dso32_status_reg_get(&dev_ctx, &reg.status_reg);

    if (reg.status_reg.xlda) {
        /* Read acceleration data */
        memset(data_raw_acceleration, 0x00, 3 * sizeof(int16_t));
        lsm6dso32_acceleration_raw_get(&dev_ctx, (uint8_t*)
data_raw_acceleration);
        acceleration_mg[0] =
        lsm6dso32_from_fs4_to_mg(data_raw_acceleration[0]);
        acceleration_mg[1] =
        lsm6dso32_from_fs4_to_mg(data_raw_acceleration[1]);
        acceleration_mg[2] =
        lsm6dso32_from_fs4_to_mg(data_raw_acceleration[2]);
        printf("Acceleration [mg]:%4.2f\t%4.2f\t%4.2f\r\n",
                acceleration_mg[0], acceleration_mg[1],
acceleration_mg[2]);
    }
```

```
    if (reg.status_reg.gda) {
        /* Read angular rate field data */
        memset(data_raw_angular_rate, 0x00, 3 * sizeof(int16_t));
        lsm6dso32_angular_rate_raw_get(&dev_ctx, (uint8_t*)
data_raw_angular_rate);
        angular_rate_mdps[0] =
        lsm6dso32_from_fs2000_to_mdps(data_raw_angular_rate[0]);
        angular_rate_mdps[1] =
        lsm6dso32_from_fs2000_to_mdps(data_raw_angular_rate[1]);
        angular_rate_mdps[2] =
        lsm6dso32_from_fs2000_to_mdps(data_raw_angular_rate[2]);
        printf("Angular rate [mdps]:%4.2f\t%4.2f\t%4.2f\r\n",
               angular_rate_mdps[0], angular_rate_mdps[1],
angular_rate_mdps[2]);
    }

    if (reg.status_reg.tda) {
        /* Read temperature data */
        memset(&lsm6_data_raw_temperature, 0x00, sizeof(int16_t));
        lsm6dso32_temperature_raw_get(&dev_ctx, (uint8_t*)
&lsm6_data_raw_temperature);
        lsm6_temperature_degC = lsm6dso32_from_lsb_to_celsius(
                              lsm6_data_raw_temperature);
        printf("Temperature [degC]:%6.2f\r\n",
lsm6_temperature_degC);
    }
```

5.8 VL53L1X: Proximitry sensor

To work with VL53L1X you have to include VL53L1X_api.h and I2c.h

Variables example:

```
static uint8_t sensorState = 0;
static uint8_t first_range = 1;
static int status;
static uint8_t byteData;
static uint16_t wordData;
static uint16_t Dev = 0;
static VL53L1X_Result_t Results;
```

To initialization:

```
    i2c_init();
    usleep(1000 * 1000);

    status = VL53L1_RdByte(Dev, 0x010F, &byteData);
    printf("VL53L1X Model_ID: %X\n", byteData);
    status += VL53L1_RdByte(Dev, 0x0110, &byteData);
    printf("VL53L1X Module_Type: %X\n", byteData);
```

```
    status += VL53L1_RdByte(Dev, 0x0111, &byteData);
    printf("VL53L1X Revision: %X\n", byteData);
    while (sensorState == 0) {
        status += vl53l1x_boot_state(Dev, &sensorState);
        VL53L1_WaitMs(Dev, 20);
    }
    printf("Chip booted\n");

    status = vl53l1x_sensor_init(Dev);
    /* status += vl53l1x_setInterrupt_polarity(Dev, 0); */
    status += vl53l1x_set_distance_mode(Dev, 2); /* 1=short,
2=long */
    status += vl53l1x_set_timing_budgetIn_ms(Dev, 100);
    status += vl53l1x_set_inter_measurementIn_ms(Dev, 100);
    status += vl53l1x_start_ranging(Dev);
```

To get data and pint:

```
    static uint8_t dataReady = 0;

    while (dataReady == 0) {
        status = vl53l1x_check_for_data_ready(Dev, &dataReady);
        //printf("status = %u, dataReady = %u\r\n", status,
dataReady);
        usleep(20 * 1000);
    }
    dataReady = 0;

    /* Get the data the new way */
    status += vl53l1x_get_result(Dev, &Results);

    printf("Status = %2d, dist = %5d, Ambient = %2d, Signal = %5d,
#ofSpads = %5d\n",
        Results.Status, Results.Distance, Results.Ambient,
                            Results.SigPerSPAD, Results.NumSPADs);

    /* trigger next ranging */
    status += vl53l1x_clear_interrupt(Dev);
    if (first_range) {
        /* very first measurement shall be ignored
            * thus requires twice call
            */
        status += vl53l1x_clear_interrupt(Dev);
        first_range = 0;
    }
    status = vl53l1x_sensor_init(Dev);
    status += vl53l1x_set_distance_mode(Dev, 2); /* 1=short,
2=long */
    status += vl53l1x_set_timing_budgetIn_ms(Dev, 100);
    status += vl53l1x_set_inter_measurementIn_ms(Dev, 100);
    status += vl53l1x_start_ranging(Dev);
```

5.9 APDS-9250: Digital RGB, IR and Ambient Light Sensor

To work with APDS-9250 you have to include Apds9250.h and I2c.h

Variables example:

```
Apds9250 myApds9250;
```

To initialization:

```
    i2c_init();
    if (myApds9250.begin())
    {
        printf("myApds9250.begin() OK\r\n");
    }

    myApds9250.setMode(modeColorSensor);
    myApds9250.setResolution(res18bit);
    myApds9250.setGain(gain1);
    myApds9250.setMeasurementRate(rate100ms);
```

To get data and pint:

```
    uint32_t red = 0;
    uint32_t green = 0;
    uint32_t blue = 0;
    uint32_t ir = 0;
    myApds9250.getAll(&red, &green, &blue, &ir);
    printf("red = %lu\r\ngreen = %lu\r\nblue = %lu\r\nir =
%lu\r\n", red, green, blue, ir);
```

5.10 SGP-40: Air quality sensor

To work with APDS-9250 you have to include sgp40_i2c.h and I2c.h

Variables example:

```
static int16_t error = 0;
static uint16_t serial_number[3];
static uint8_t serial_number_size = 3;
// Parameters for deactivated humidity compensation:
static uint16_t default_rh = 0x8000;
static uint16_t default_t = 0x6666;
```

To initialization:

```
    i2c_init();
```

```
    error = sgp40_get_serial_number(serial_number,
serial_number_size);

    if (error) {
        printf("Error executing sgp40_get_serial_number():
%i\r\n", error);
    } else {
        printf("serial: 0x%04x%04x%04x\r\n", serial_number[0],
serial_number[1],
               serial_number[2]);
        printf("\r\n");
    }
```

To get data and pint:

```
    uint16_t sraw_voc;
    error = sgp40_measure_raw_signal(default_rh, default_t,
&sraw_voc);
    if (error) {
            printf("Error executing sgp40_measure_raw_signal():
%i\n", error);
    } else {
        printf("SRAW VOC: %u\n", sraw_voc);
    }
```

5.11 Low power test

For using low power modes LowPower.h should be included.

After that should be run:

```
    LowPower.begin();
```

And selected one from Low power modes, ex:

```
    LowPower.deepSleep();
```

5.12 SD card

To use SD card to write a raw data should be included Sdcard.h

To initialization:

```
sdcard_init();
```

After that you can read and write SD card block(s) ex:

```
sdcard_write_single_block(0, blockDataWr);
sdcard_read_single_block(0, blockData);
```

5.13 FatFS SD card

To work with FS ff.h and Sdcard.h should be included

To initialization:

```
int result = sdcard_init();
if( f_mount(&USERFatFS,(TCHAR const*)USERPath, 0) != FR_OK) {
    printf("FS mount error\r\n");
}
```

To work with Fat FS refer to

http://elm-chan.org/fsw/ff/00index_e.html

5.14 M24C32 EEPROM

To use EEPROM to write and read data should be included M24c32.h

To initialization:

```
m24c32_init();
```

After that you can read and write data:

```
data = m24c32_read_byte(offset);
m24c32_write_byte(offset, data);
```

5.15 BQ27441DRZR Battery monitor

To use battery monitor should be included Bq27441.h

To initialization:

```
bq27441_init();
```

After that you can get battery data ex:

```
    uint16_t fullcap =
bq27441_g1_get_full_charge_capacity_unfiltered();
```

6. Setup

6.1. Installation - Linux/Ubuntu and Raspbian OS

1. Install Python 3 on your host computer.
2. Install Node.js v14 or higher on your host computer.
   Alternatively, run the following commands:
   curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
   sudo apt-get install -y nodejs
   node -v
   The last command should return the node version, v14 or above.
   Let's verify the node installation directory:
   npm config get prefix
   If it returns */usr/local/*, run the following commands to change npm's default directory:
   mkdir ~/.npm-global
   npm config set prefix '~/.npm-global'
   echo 'export PATH=~/.npm-global/bin:$PATH' >> ~/.profile
3. Install the CLI tools via:
   npm install -g edge-impulse-cli

6.2. Linux console command sequence

Cloning Git project:

```
git clone https://github.com/SensiEDGE/CommonSense.git
```

GCC ARM Install:

```
sudo tar xjf
gcc-arm-none-eabi-9-2019-q4-major-x86_64-linux.tar.bz2 -C
/usr/share/

sudo ln -s
/usr/share/gcc-arm-none-eabi-9-2019-q4-major/bin/arm-none-eabi-gcc
/usr/bin/arm-none-eabi-gcc
sudo ln -s
/usr/share/gcc-arm-none-eabi-9-2019-q4-major/bin/arm-none-eabi-g++
/usr/bin/arm-none-eabi-g++
sudo ln -s
/usr/share/gcc-arm-none-eabi-9-2019-q4-major/bin/arm-none-eabi-gdb
/usr/bin/arm-none-eabi-gdb
sudo ln -s
/usr/share/gcc-arm-none-eabi-9-2019-q4-major/bin/arm-none-eabi-siz
e /usr/bin/arm-none-eabi-size
sudo ln -s
/usr/share/gcc-arm-none-eabi-9-2019-q4-major/bin/arm-none-eabi-obj
copy /usr/bin/arm-none-eabi-objcopy
sudo ln -s
/usr/share/gcc-arm-none-eabi-9-2019-q4-major/bin/arm-none-eabi-ar
/usr/bin/arm-none-eabi-ar
sudo ln -s
/usr/share/gcc-arm-none-eabi-9-2019-q4-major/bin/arm-none-eabi-ld
/usr/bin/arm-none-eabi-ld

sudo apt install libncurses-dev
sudo ln -s /usr/lib/x86_64-linux-gnu/libncurses.so.6
/usr/lib/x86_64-linux-gnu/libncurses.so.5
sudo ln -s /usr/lib/x86_64-linux-gnu/libtinfo.so.6
/usr/lib/x86_64-linux-gnu/libtinfo.so.5
```

Run `sudo apt-get install -y nodejs` to install Node.js 14.x and npm
You may also need development tools to build native addons:

```
  sudo apt-get install gcc g++ make
```

To install the Yarn package manager, run:

```
  curl -sL https://dl.yarnpkg.com/debian/pubkey.gpg | gpg
--dearmor | sudo tee /usr/share/keyrings/yarnkey.gpg >/dev/null
  echo "deb [signed-by=/usr/share/keyrings/yarnkey.gpg]
https://dl.yarnpkg.com/debian stable main" | sudo tee
/etc/apt/sources.list.d/yarn.list
     sudo apt-get update && sudo apt-get install yarn
```

VNC:

```
sudo apt-get install dconf-editor
```

Open dconf-editor >>org>>gnome>>desktop>>remote session >> require-encryption

In dconf-editor we need to disable encryption

Python:

```
pip install pyserial
pip install pyinstaller
```

COM, viewer:
if you have some issues with port connection you can try: `sudo chmod -R 777 /dev/ttyUSB0`

```
sudo apt install minicom
minicom -D /dev/ttyUSB0
```

VS:

```
sudo snap install --classic code
```

7. Ordering information

Part Number : CMSN-1

Com**M**on **S**e**N**se **S**oNy version **1**.0

8. Functional description

The device can work in two modes:

- test;
- measure.

Test is default mode. In this mode the device can receive and send device configuration and sensors data.
In measure mode the device also can get data from sensors by the settings and write data to file to micro SD card.
To start measuring need to do few steps:

1. switch device to measure mode and sensors mode in PC configurator;
2. press devices button to run, after that green LED will shine during 1 s;
3. make a measuring;
4. press devices button to stop, after that red LED will shine during 1 s;
5. switch power to off
6. get data from micro SD card with card reader.

Micro SD card have to be formatted with FAT32 file system.
After the measure finish micro SD card will have a folder with name, witch means time from devices power on in sec in hev format. Inside this wolder will present some files in `*.txt` format with data (every string starts from time `uint64_t` in ns from devices power on):

- `ligth.txt` - data from light sensor: `"%16X, %8X\r\n", time, light`, if sensor in light mode;
- `colors.txt` - data from light sensor:
  `"%016llX, %08X, %08X, %08X, %08X\r\n", time, red, green, blue, ir,`
  if sensor in color mode;
- `acceler.txt` - data from accelerometer:
  `"%016llX, %.02f, %.02f, %.02f\r\n", time, axis_x, axis_y, axis_z;`
- `gyro.txt` - data from gyroscope:
  `"%016llX, %.02f, %.02f, %.02f\r\n", time, axis_x, axis_y, axis_z;`
- `magnet.txt` - data from magnetometer:
  `"%016llX, %.02f, %.02f, %.02f\r\n", time, axis_x, axis_y, axis_z;`
- `pressure.txt` - data from pressure sensor:
  `"%016llX, %.02f\r\n", time, pressure;`
- `humtemp.txt` - data from humidity and temperature sensor:
  `"%016llX, %.02f, %.02f\r\n", hts221_time, hum, temp;`
- `voc.txt` - data from VOC sensor:
  `"%016llX, %d, %d\r\n", time, source, voc,`
  `source`: 0 - default temperature and humidity (50%RH, 25 degC), 1 - temperature and humidity from humidity and temperature sensor;
- `proxim.txt` - data from proximity sensor:
  `"%016llX, %d, %4X\r\n", time, mode_distance, distance,`
  `mode_distance`: 0 - short (limited to 1.3 m), 1 - long (up to 4 m);

Best regards

We wish you an interesting using of the board and discovering new possibilities with this powerful sensor set and MCU