9-DOF IMU - ICM20948 - Trēo™ Module

Module Features

- TDK/InvenSense ICM20948
- RoHS Compliant
- Software Library
- NightShade Trēo™ Compatible
- Breakout Headers

ICM20948 Features

(from TDK/InvenSense)

- 3-Axis Accelerometer
  - FSR of ±2g, ±4g, ±8g, or ±16g
- 3-Axis Gyroscope
  - FSR of ±250dps, ±500 dps, ±1000dps, or ±2000dps
- 3-Axis Compass
  - FSR of ±4900μT
- On-Chip 16-bit ADCs and Programmable Filters
- Digital-output temperature sensor
- MEMS structure hermetically sealed and bonded at wafer level

Applications

- Flight Stabilization
- Robotics

Description

The ICM20948 Trēo™ Module is a 9-DOF IMU module that that features TDK/InvenSense's ICM20948 9-DOF IMU. It can measure acceleration, rotation, and magnetic field in all 3 axes. These measurements can be used to track all movement and the compass can be used to compensate for sensor drift. This module is a part of the NightShade Treo system, patent pending.

Trēo™ Compatibility

Electrical

Communication	I2C
Max Current, 3.3V	4mA
Max Current, 5V	0mA

Mechanical

- 25mm x 25mm Outline
- 20mm x 20mm Hole Pattern
- M2.5 Mounting Holes

Table of Contents

1 Summary

The ICM20948 contains an accelerometer, gyroscope, and magnetometer. It is first initialized with the begin() method. Then data can be measured with the getData() method and retrieved with the axis specific methods. (e.g. readAccelX, readGyroX, readCompassX, etc.) The measurement parameters can be varied using the other methods available in this library.

2 What is Trēo™?

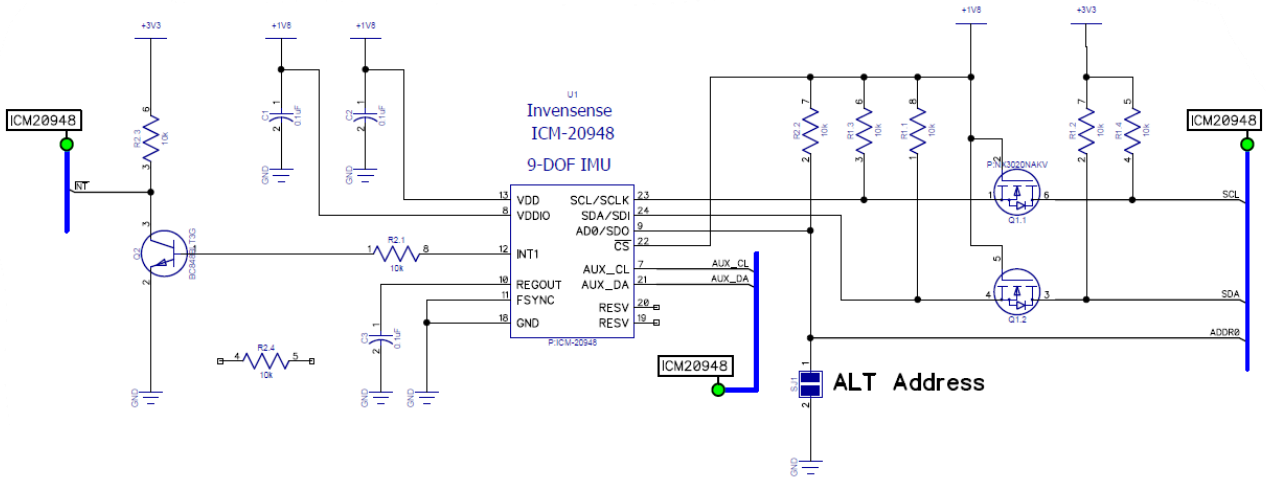
NightShade Trēo is a system of electronic modules that have standardized mechanical, electrical, and software interfaces. It provides you with a way to quickly develop electronic systems around microprocessor development boards. The grid attachment system, common connector/cabling, and extensive cross-platform software library allow you more time to focus on your application. Trēo is supported with detailed documentation and CAD models for each device.

Learn more about Trēo here.

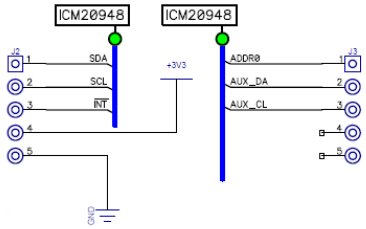
4 Electrical Characteristics

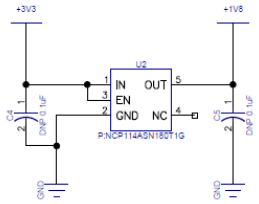
	Minimum	Nominal	Maximum
Voltages			
$V_{i/o}$ (SDA, SCL, INT)	-0.3V	-	3.6V
$V_{3.3V}$	3.1V	3.3V	3.5V
Accelerometer			
Sample Rate	Single	-	4500Hz
Range	-16g	-	+16g
Precision	488µg/LSB	-	61µg/LSB
Gyroscope			
Sample Rate	Single	-	9000Hz
Range	-2000dps	-	+2000dps
Precision	0.0610dps/LSB	-	0.0076dps/LSB
Magnetometer			
Sample Rate	Single	-	100Hz
Range	-4900µT	-	+4900µT
Precision		0.150 µT/LSB	
I2C Slave Address			
SJ1 Open (Default)		0x69	
SJ1 Closed (Soldered)		0x68	
Operating Temperature	-25°C	-	+85°C

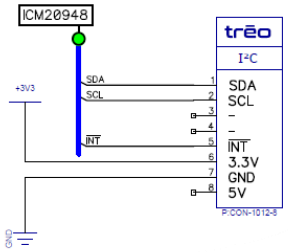
5 Electrical Schematic



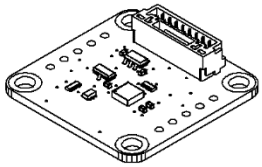
Breakout Headers

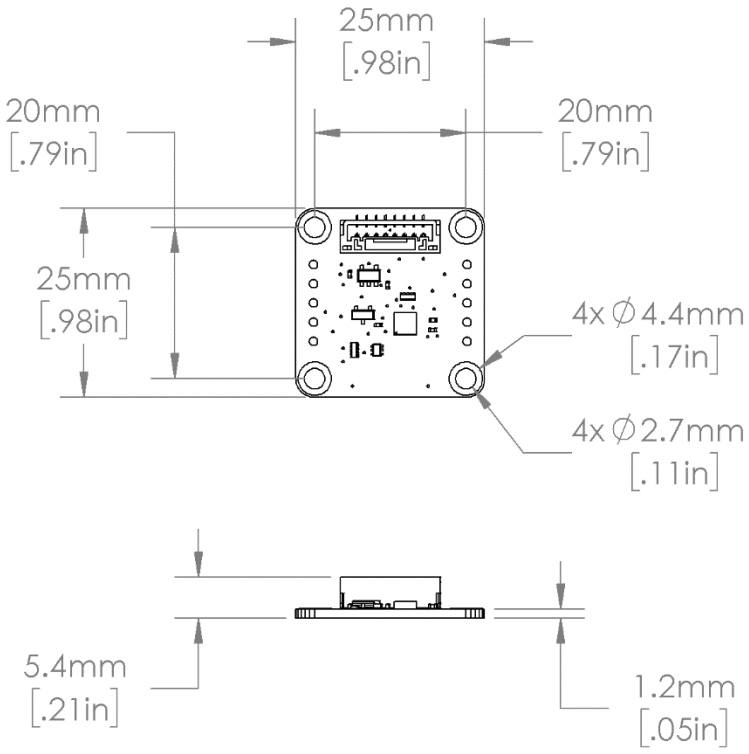






6 Mechanical Outline





7 Example Arduino Program

```
/*********************************************************
  ICM20948_9DoF_IMU - NightShade_Treo by NightShade Electronics

  This sketch demonstrates the functionality of the
  NightShade Trēo ICM20948 9 DoF IMU module. (NSE-1124-1)
  It reads the accelerometer, gyro, and magnetometer data
  from the sensor and prints it over Serial at 115200
  baudrate.

  Created by Aaron D. Liebold
  on February 15, 2021

  Links:
  NightShade Trēo System: https://nightshade.net/treo
  Product Page: https://nightshade.net/product/treo-9-dof-imu-icm20948/

  Distributed under the MIT license
  Copyright (C) 2021 NightShade Electronics
  https://opensource.org/licenses/MIT
*********************************************************/

// Include NightShade Treo Library
#include <NightShade_Treo.h>

#include <math.h>

// Declare Objects
NightShade_Treo_ICM20948 sensor(1);

void setup() {
  Serial.begin(115200);
  sensor.begin();
}

void loop() {
  sensor.getData(1, 1);
  float accelX = (float) sensor.readAccelX() / sensor.accelSensitivity(); // G
  float accelY = (float) sensor.readAccelY() / sensor.accelSensitivity(); // G
  float accelZ = (float) sensor.readAccelZ() / sensor.accelSensitivity(); // G
  float gyroX = (float) sensor.readGyroX() / sensor.gyroSensitivity(); // dps
  float gyroY = (float) sensor.readGyroY() / sensor.gyroSensitivity(); // dps
  float gyroZ = (float) sensor.readGyroZ() / sensor.gyroSensitivity(); // dps
  float magX = (float) (sensor.readCompassX() * 15) / 100; // uT
  float magY = (float) (sensor.readCompassY() * 15) / 100; // uT
  float magZ = (float) (sensor.readCompassZ() * 15) / 100; // uT

  // Calculate Azimuth at X+ (2-axis)
  float azimuth =  atan2(-1*magY, magX) * 180 / M_PI;
  if (azimuth < 0) azimuth += 360.0; // Keep azimuth positive (0 - 360deg)
```

```
  // Print Results
  Serial.print("Accel X: ");
  Serial.print(accelX);
  Serial.print("g \tY: ");
  Serial.print(accelY);
  Serial.print("g \tZ: ");
  Serial.print(accelZ);
  Serial.print("g \tGyro X: ");
  Serial.print(gyroX);
  Serial.print("dps \tY: ");
  Serial.print(gyroY);
  Serial.print("dps \tZ: ");
  Serial.print(gyroZ);
  Serial.print("dps \tCompass Heading: ");
  Serial.print(azimuth);
  Serial.print("deg\n");

  delay(500);
}
```

8 Library Overview (C++ & Python)

C++ Class

NightShade_Treo_ICM20948 <classObject>();

Python Module

<classObject> = NightShade_Treo_ICM20948()

8.1 Constructors

NightShade_Treo_ICM20948(int port, uint8_t slaveAddress, uint32_t clockSpeed)

Creates a ICM20948 object.

Arguments:

port	Integer of the I2C port used (e.g. 0 = "/dev/i2c_0")
slaveAddress	7-bit slave address
clockSpeed	Desired clock speed for the bus

Returns:

Nothing

NightShade_Treo_ ICM20948(int port)

Creates a ICM20948 object assuming the default slave address and clock speed.

Arguments:

port	Integer of the I2C port used. (e.g. 0 = "/dev/i2c_0")

Returns:

Nothing

8.2 Methods

begin()

Initializes the ICM20948.

Arguments:

None.

Returns:

Error	0 = Success

readDeviceId()

Returns the device ID code.

Arguments:

None	

Returns:

Error	0 = Success

enableSleep(int enable)

Enables sleep mode.

Arguments:

enable	true/false

Returns:

Error	0 = Success

clockSelect(uint8_t clkSource)

Sets the clock source.

Arguments:

clkSource	0: Internal 20MHz oscillator 1-5: Auto-selects PLL (if ready) or internal oscillator 6: Internal 20MHz oscillator 7: Stops the clock and keeps timing generator in reset

Returns:

Error	0 = Success

enableSensors(int enableAccel, int enableGyro, int enableTemp);

Enables or disables certain sensors.

Arguments:

enableAccel	true/false
enableGyro	true/false
enableTemp	true/false

Returns:

Error	0 = Success

enableLowPower(int enable)

Puts the IMU into full sleep mode.

Arguments:

enable	true/false

Returns:

Error	0 = Success

setupSensors(uint8_t accelSensitivity, uint8_t accelFilter, uint8_t gyroSensitivity, uint8_t gyroFilter)

Sets the sensitivity/full-scale range (FSR) of the accelerometer and gyroscope and their respective low-pass filter setting.

Arguments:

accelSensitivity	0: ±2g	(16,384 LSB/g)
	1: ±4g	(8,192 LSB/g)
	2: ±8g	(4,096 LSB/g)
	3: ±16g	(2,048 LSB/g)

accelFilter	0: BW: 1209Hz	ODR: 4500Hz (LPF Disabled)
	1: BW: 246.0Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)
	2: BW: 111.4Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)
	3: BW: 50.4Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)
	4: BW: 23.9Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)
	5: BW: 11.5Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)
	6: BW: 5.7Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)
	7: BW: 473.0Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)

gyroSensitivity:	0: ±250dps (131 LSB/dps)	
	1: ±500dps (65.5 LSB/dps)	
	2: ±1000dps (32.8 LSB/dps)	
	3: ±2000dps (16.4 LSB/dps)	

gyroFilter	0: BW: 12106HZ	ODR: 9000Hz (LPF Disabled)
	1: BW: 151.8Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)
	2: BW: 119.5Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)
	3: BW: 51.2Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)
	4: BW: 23.9Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)
	5: BW: 11.6Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)
	6: BW: 5.7Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)
	7: BW: 361.4Hz	ODR: 1125Hz/(1+ACCEL_SMPLRT_DIV)

readIntPinCfg()

Reads the interrupt pin configuration byte.

Arguments:

None

Returns:

INT_PIN_CFG byte

getData(int getAccelGyroTempData, int getCompassData)

Retrieves data from the IC and stores it in a local software buffer. The compass data is a separate data communication and the arguments allow the user to control which data is retrieved. This allows the user to match how much time is consumed reading the device with their application needs.

Arguments:

getAccelGyroTempData	true/false
getCompassData	true/false

Returns:

Error	0 = Success

readAccelX()

Retrieves the X-axis accelerometer value from the local software buffer. This data is acquired with the getData() method.

Arguments:

None

Returns:

X-Axis accelerometer value (int16_t)

readAccelY()

Retrieves the Y-axis accelerometer value from the local software buffer. This data is acquired with the getData() method.

Arguments:

None

Returns:

Y-Axis accelerometer value (int16_t)

readAccelZ()

Retrieves the Z-axis accelerometer value from the local software buffer. This data is acquired with the getData() method.

Arguments:

None

Returns:

Z-Axis accelerometer value (int16_t)

readGyroX()

Retrieves the X-axis gyroscope value from the local software buffer. This data is acquired with the getData() method.

Arguments:
None

Returns:
X-Axis gyroscope value (int16_t)

readGyroY()

Retrieves the Y-axis gyroscope value from the local software buffer. This data is acquired with the getData() method.

Arguments:
None

Returns:
Y-Axis gyroscope value (int16_t)

readGyroZ()

Retrieves the Z-axis gyroscope value from the local software buffer. This data is acquired with the getData() method.

Arguments:
None

Returns:
Z-Axis gyroscope value (int16_t)

readCompassX()

Retrieves the X-axis magnetometer value from the local software buffer. This data is acquired with the getData() method.

Arguments:
None

Returns:
X-Axis magnetometer value (int16_t)

readCompassY()

Retrieves the Y-axis magnetometer value from the local software buffer. This data is acquired with the getData() method.

Arguments:
None

Returns:
Y- Axis magnetometer value (int16_t)

readCompassZ()

Retrieves the Z-axis magnetometer value from the local software buffer. This data is acquired with the getData() method.

Arguments:
None

Returns:
Z- Axis magnetometer value (int16_t)

readTemperature()

Retrieves the temperature data from the local software buffer. This data is acquired with the getData() method.

Arguments:
None.

Returns:
Temperature value in 0.01°C increments (int16_t)

accelSensitivity()

Returns the current value of 1G in LSB, based on the current sensitivity setting. The sensitivity is set with the setSensitivity() method.

Arguments:
None

Returns:
LSB/G (float)

gyroSensitivity()

Returns the current value of 1 degree per second in LSB, based on the current sensitivity setting. The sensitivity is set with the setSensitivity() method.

Arguments:
None

Returns:
LSB/DPS (float)

magReadDeviceId()

Returns the device ID code of the magnetometer.

Arguments:
None

Returns:
Magnetometer device ID (uint8_t)

magDataReady()

Returns true when new magnetometer data is ready.

Arguments:
None

Returns:
Magnetometer ready flag (true/false)

magSetMeasurementMode(uint8_t mode)

Sets the sample rate/mode of the magnetometer.

Arguments:

mode	0: Off
	1: Single Measurement
	2: Continuous 10Hz
	4: Continuous 20Hz
	6: Continuous 50Hz
	8: Continuous 100Hz
	16: Self-test Mode

Returns:

Error	0 = Success

magReset(uint8_t enable)

Resets the magnetometer.

Arguments:
None

Returns:

Error	0 = Success