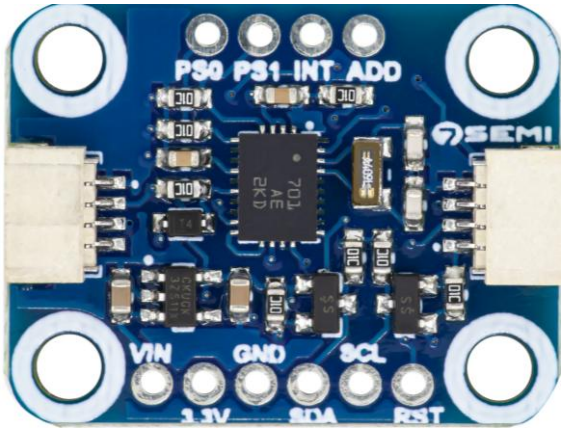
7SEMI

BNO055 9-DOF Absolute Orientation Sensor Breakout I2C Qwiic

Version 1.0

Table of Contents

1.Introduction



The **7Semi BNO055 9-DOF Absolute Orientation Sensor Breakout I2C Qwiic** integrates the **Bosch BNO055 sensor**, providing a robust and accurate solution for motion tracking and orientation sensing. This module combines an accelerometer, gyroscope, and magnetometer with onboard sensor fusion, eliminating the need for external computation..

1.1 Features

- Integrated 9-DOF Sensor Fusion
  - Accelerometer: ±2g, ±4g, ±8g, ±16g
  - Gyroscope: ±125°/s to ±2000°/s
  - Magnetometer: ±1300µT (x-, y-axis), ±2500µT (z-axis)
- Sensor Fusion Outputs
  - Quaternion, Euler Angles, Linear Acceleration, Gravity Vector
- Operating Voltage: 3.3V
- Low Power Modes: Normal, Low Power, Suspend
- Interfaces: I²C (HID-I2C Windows 8 compatible), UART
- Qwiic I²C support for plug-and-play use.

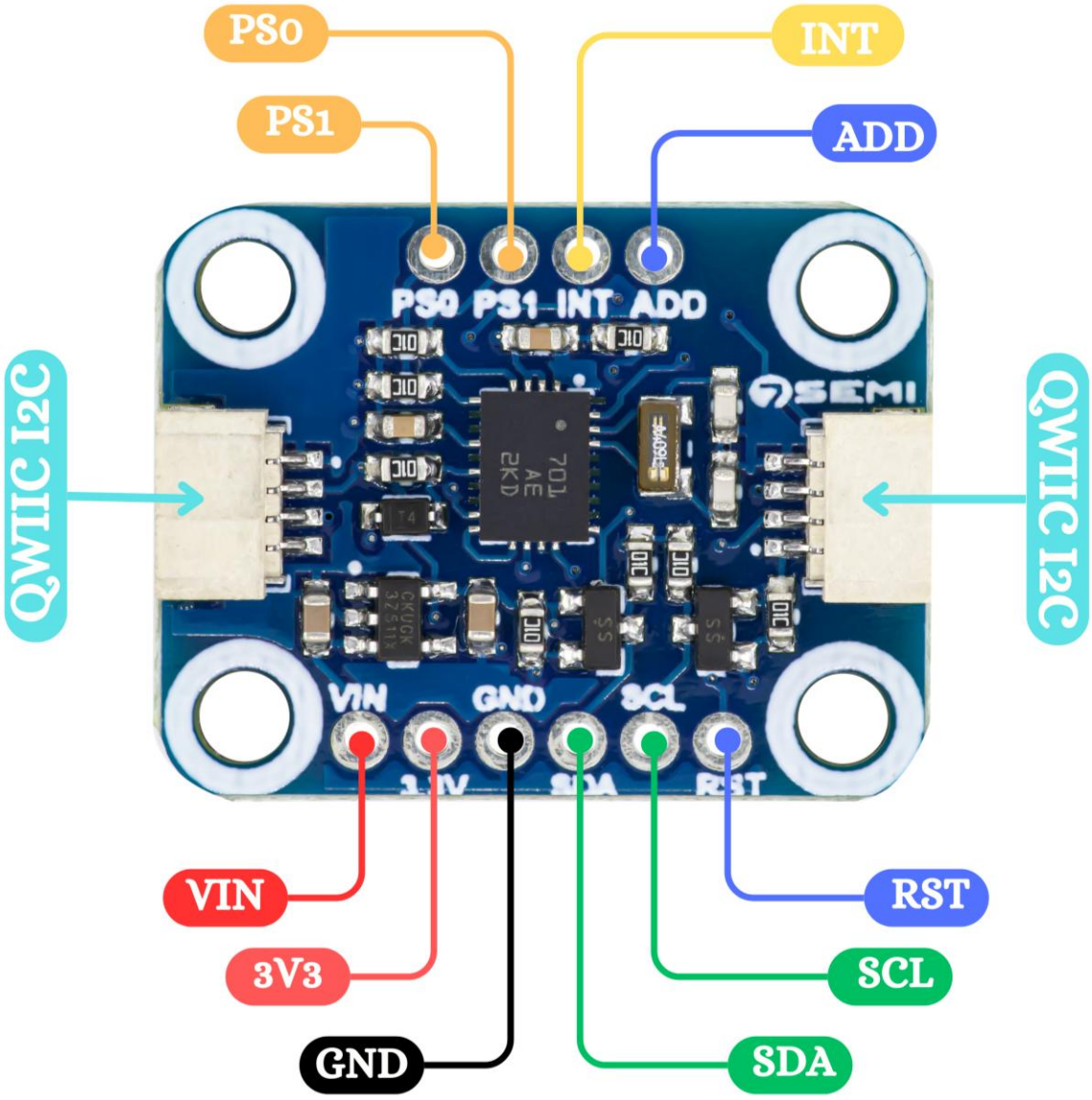
2.Technical Specification

The **Technical Specification** table provides detailed information about **7Semi BNO055 9-DOF Absolute Orientation Sensor Breakout I2C Qwiic**, including its operating voltage, current consumption, and electrical characteristics. This data helps users understand the power requirements, communication parameters, and performance capabilities of the sensor. It ensures compatibility with different microcontrollers and embedded systems while providing guidelines for efficient integration into various applications.

BNO055 Specifications

- Operating Voltage: 2.4V–3.6V
- I²C Address: 0x28 or 0x29
- Communication Interfaces: I²C, UART
- Power Consumption:
  - 12.3mA (Normal Mode)
  - 2.72mA (Low Power Mode)
  - 0.04mA (Suspend Mode)
- Accelerometer Range: ±2g, ±4g, ±8g, ±16g
- Gyroscope Range: ±125°/s to ±2000°/s
- Magnetometer Range: ±1300µT (x-, y-axis), ±2500µT (z-axis)
- Operating Temperature: -40°C to +85°C
- Sensor Breakout Size: 29.82 x 21.86 mm

3.Pinouts

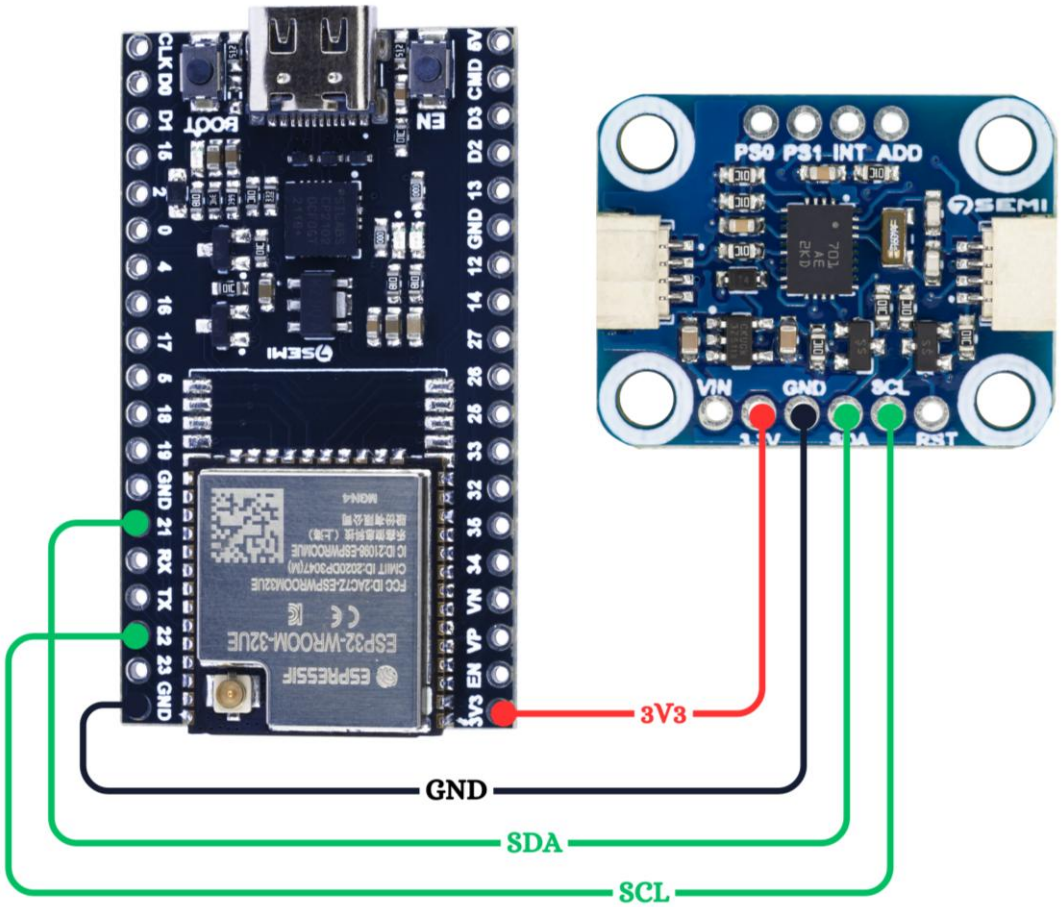

PS0
PS1
INT
ADD
QWIIC I2C
QWIIC I2C
PS0 PS1 INT ADD
VIN
GND
SCL
VIN
3V3
GND
RST
SCL
SDA

Pin	Name	Description
1	VIN	Power Input (3.3V - 5V)
2	3V3	3.3V Regulated Output
3	GND	Ground Connection
4	SDA	I²C Data Line
5	SCL	I²C Clock Line
6	RST	Reset Pin
7	ADD	I²C Address Selection
8	INT	Interrupt Output
9	PS0	Protocol Selection 0
10	PS1	Protocol Selection 1

Connection Guidelines

- Power Supply & Communication: Ensure a stable 3.3V - 5V power input and connect SDA/SCL for I²C.
- I²C Address & Interrupts: Use the ADD pin to change the I²C address and INT pin for motion detection or data-ready signals.

4.Hardware Interface



Connection Explanation :

- Connect ESP32 3.3V to BNO055 3V3 for power.
- Connect ESP32 GND to BNO055 GND for a common ground.
- Connect ESP32 GPIO 21 (SDA) to BNO055 SDA for data communication.
- Connect ESP32 GPIO 22 (SCL) to BNO055 SCL for the clock signal.
- This configuration enables communication between the ESP32 and BNO055 using the I²C protocol.

5.Example code link

We provide example codes to help you get started with the **7Semi BNO055 9-DOF Absolute Orientation Sensor Breakout I2C Qwiic**. These examples demonstrate how to communicate with the sensor and retrieve **absolute orientation, acceleration, gyroscope, and magnetometer data** using the **I²C protocol**. The code is available for two popular platforms: **Arduino and ESP32**.

Code Explanation

```
#include "BNO055_support.h"    //Contains the bridge code between the API and
Arduino

#include <Wire.h>

//The device address is set to BNO055_I2C_ADDR2 in this example. You can
change this in the BNO055.h file in the code segment shown below.

// /* bno055 I2C Address */

#define BNO055_I2C_ADDR1                0x28

// #define BNO055_I2C_ADDR2                 0x29

// #define BNO055_I2C_ADDR                  BNO055_I2C_ADDR2

//Pin assignments as tested on the Arduino Due.

//Vdd,Vddio : 3.3V

//GND : GND

//SDA/SCL : SDA/SCL

//PSO/PS1 : GND/GND (I2C mode)

//This structure contains the details of the BNO055 device that is connected.
(Updated after initialization)

struct bno055_t myBNO;

struct bno055_euler myEulerData; //Structure to hold the Euler data

unsigned long lastTime = 0;

void setup() //This code is executed once
```

```
{

  //Initialize I2C communication

  Wire.begin();

   //Initialization of the BNO055

  BNO_Init(&myBNO); //Assigning the structure to hold information about the
device

   //Configuration to NDoF mode

  bno055_set_operation_mode(OPERATION_MODE_NDOF);

   delay(1);  //Initialize the Serial Port to view information on the Serial
Monitor

  Serial.begin(115200);

}

void loop() //This code is looped forever

{

  if ((millis() - lastTime) >= 100) //To stream at 10Hz without using
additional timers

  {

    lastTime = millis();

    bno055_read_euler_hrp(&myEulerData);      //Update Euler data into the
structure

    Serial.print("Time Stamp: ");       //To read out the Time Stamp

    Serial.println(lastTime);

    Serial.print("Heading(Yaw): ");       //To read out the Heading (Yaw)

    Serial.println(float(myEulerData.h) / 16.00);   //Convert to degrees

    Serial.print("Roll: ");         //To read out the Roll

    Serial.println(float(myEulerData.r) / 16.00);   //Convert to degrees

    Serial.print("Pitch: ");        //To read out the Pitch
```

```
    Serial.println(float(myEulerData.p) / 16.00);    //Convert to degrees

    Serial.println();          //Extra line to differentiate between packets

  }

}
```

1. Header File Inclusions

- `#include "BNO055_support.h"` → This header file acts as a bridge between the **BNO055 API** and the Arduino.
- `#include <Wire.h>` → This library enables **I2C communication** between the Arduino and the BNO055 sensor.

2. BNO055 I2C Address Configuration

- The I2C address of BNO055 can be either:
  - **`0x28` (BNO055_I2C_ADDR1)**
  - **`0x29` (BNO055_I2C_ADDR2)**
- The default address in the code is **BNO055_I2C_ADDR2 (0x29)**.

3. Pin Assignments for Arduino Due

- **Vdd, Vddio** → Connected to **3.3V**
- **GND** → Connected to **GND**
- **SDA/SCL** → Connected to Arduino's **I2C pins**
- **PS0/PS1** → Set to **GND** (which enables **I2C mode**)

4. Structure Definitions

- `struct bno055_t myBNO;` → Holds **device details**.
- `struct bno055_euler myEulerData;` → Holds **Euler angle readings**.

5. `setup()` Function

- Initializes **I2C communication** (`Wire.begin();`).
- Calls **`BNO_Init(&myBNO);`** to initialize the BNO055 sensor.
- Configures the sensor to **NDoF mode** (`bno055_set_operation_mode(OPERATION_MODE_NDOF);`), which enables full **9-axis fusion**.
- Starts **serial communication** at **115200 baud rate** (`Serial.begin(115200);`).

6. `loop()` Function

- Runs **continuously**.
- Uses a **millis() timestamp** to execute the sensor reading every **100 milliseconds** (10Hz frequency).
- Calls `bno055_read_euler_hrp(&myEulerData);` to update the **Euler angles**.
- Prints:
  - **Timestamp**
  - **Yaw (Heading)**
  - **Roll**
  - **Pitch**
- Euler angles are **divided by 16** to convert the raw data into **degrees**.

7. Arduino Example Code

This example is designed for Arduino-compatible boards and demonstrates:

- Initializing the I²C communication with the **BNO055** sensor Board.
- Reading absolute orientation, acceleration, gyroscope, and magnetometer data.
- Printing the sensor data to the Serial Monitor.

8. ESP32 Example Code

This example targets ESP32 boards and showcases:

- Configuring the ESP32 I²C interface to communicate with the **BNO055** sensor Board.
- Reading absolute orientation, acceleration, gyroscope, and magnetometer data.
- Printing the sensor data to the Serial Monitor.

How to Access the Code

- **Download Link for Arduino and ESP32 Example**: Click Here

5.1 Sample Serial Output Arduino

Sample output Image of Arduino:

```
Time Stamp: 58783
Heading(Yaw): 220.75
Roll: 0.62
Pitch: 3.44

Time Stamp: 58883
Heading(Yaw): 220.75
Roll: 0.62
Pitch: 3.37

Time Stamp: 58983
Heading(Yaw): 220.75
Roll: 0.62
Pitch: 3.37
```

6.Mechanical Specification