PRN:
Name:

EXPERIMENT NO 8

Title: Accelerometer sensor MPU6050 interfaced with NodeMCU Mini.

Components Required:

- NodeMCU (ESP8266)
- Breadboard
- Jumper wires
- USB cable for NodeMCU
- Computer with Arduino IDE installed
- Accelerometer

Theory:

NodeMCU (ESP8266):

NodeMCU is an open-source firmware and development board based on the ESP8266 WiFi module.The ESP8266 is a highly integrated chip designed for the needs of a new connected world. It offers acomplete and self-contained Wi-Fi networking solution, allowing it to either host the application orto offload all Wi-Fi networking functions from another application processor.

Specifications:

microcontroller : Tensilica L106 32-bit microcontroller

Clock Frequency : Up to 80 MHz

Wi-Fi Standards **:** 802.11 b/g/n

Operating voltage: 3.3V (Typical)

Digital I/O Pins : Typically 17 GPIO pins (varies based on module) Typically 1

Analog I/O Pis: analog input pin (varies based on module)

Flash Memory : Typically 512KB to 4MB of integrated flash memory

Operating Temperature : -40°C to +125°C

ESP8266 Wi-Fi module : The ESP8266 Wi-Fi module is a highly integrated chip designed for wireless communication. It features a Tensilica L106 32-bit microcontroller, integrated Wi-Fi transceiver, and on-chip memory. The module supports 802.11 b/g/n Wi-Fi standards and can operate as a station, access point, or both simultaneously.

Features of NodeMCU:

Low Power Consumption: The ESP8266 is designed for low power consumption, making it suitable for battery-powered IoT applications.

PRN:
Name:

Highly Integrated: Despite its compact size, the ESP8266 integrates a microcontroller,Wi-Fi transceiver, and memory, reducing the need for external components.

OTA (Over-The-Air) Updates :OTA updates, allowing firmware to be updated, wirelessly without the need for physical access to the device.

Advanced Networking Features: It supports features such as TCP/IP protocol stack,DNS resolution, DHCP, and UDP for robust network communication.

Wide Range of GPIO Pins: it has variety of GPIO pins, which can be used for digital input/output, analog input, PWM output, and other purposes.

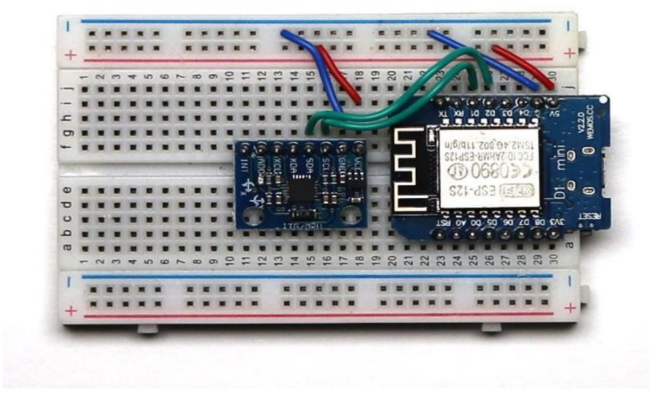
Accelerometer

- Common package for single, dual, and tri axis accelerometers.
- Dual Flat Lead (DFL) plastic package for SMD mounting.
- Excellent bias stability over temperature.
- Excellent noise performance.
- Exceptionally insensitive to shocks and vibrations.
- Automotive qualified according to AEC-Q100 standard Advanced self-diagnostics features.
- Internal temperature sensor

PinOut of MPU-6050:

1) Vcc : Provides power for the module, can be +3V to +5V. Typically +5V is used
2) Ground : Connected to Ground of system
3) Serial Clock (SCL) : Used for providing clock pulse for I2C Communication
4) Serial Data (SDA) : Used for transferring Data through I2C communication
5) Auxiliary Serial Data (XDA) : Can be used to interface other I2C modules with MPU6050. It is optional
6) Auxiliary Serial Clock (XCL) : Can be used to interface other I2C modules with MPU6050. It is optional
7) AD0 : If more than one MPU6050 is used a single MCU, then this pin can be used to vary the address
8) Interrupt (INT) : Interrupt pin to indicate that data is available for MCU to read.

Interfacing diagram:



PRN:
Name:

Upload the Code:

Connect the NodeMCU to your computer using a USB cable. Select the appropriate port from Tools -> Port menu. Upload the code to the NodeMCU by clicking on the upload button (right arrow).

- Wire.requestFrom():

  This function is used by the controller device to request bytes from a peripheral device. The bytes may then be retrieved with the available() and read() functions. The requestFrom() method accepts a boolean argument changing its behavior for compatibility with certain I2C devices.
- I2C_Write()

  The write function does not write directly to the slave device but adds to the I2C buffer. To do so, you need to use the endTransmission function to send the buffered bytes to the slave device.

Program:

```
#include <Wire.h>


// MPU6050 Slave Device Address

const uint8_t MPU6050SlaveAddress = 0x68;


// Select SDA and SCL pins for I2C communication

const uint8_t scl = D6;

const uint8_t sda = D7;


// sensitivity scale factor respective to full scale setting provided in datasheet

const uint16_t AccelScaleFactor = 16384;

const uint16_t GyroScaleFactor = 131;


// MPU6050 few configuration register addresses

const uint8_t MPU6050_REGISTER_SMPLRT_DIV   =  0x19;

const uint8_t MPU6050_REGISTER_USER_CTRL    =  0x6A;
```

PRN:
Name:

```
const uint8_t MPU6050_REGISTER_PWR_MGMT_1   =  0x6B;
const uint8_t MPU6050_REGISTER_PWR_MGMT_2   =  0x6C;
const uint8_t MPU6050_REGISTER_CONFIG       =  0x1A;
const uint8_t MPU6050_REGISTER_GYRO_CONFIG  =  0x1B;
const uint8_t MPU6050_REGISTER_ACCEL_CONFIG =  0x1C;
const uint8_t MPU6050_REGISTER_FIFO_EN      =  0x23;
const uint8_t MPU6050_REGISTER_INT_ENABLE   =  0x38;
const uint8_t MPU6050_REGISTER_ACCEL_XOUT_H =  0x3B;
const uint8_t MPU6050_REGISTER_SIGNAL_PATH_RESET  = 0x68;

int16_t AccelX, AccelY, AccelZ, Temperature, GyroX, GyroY, GyroZ;

void setup() {
  Serial.begin(9600);
  Wire.begin(sda, scl);
  MPU6050_Init();
}

void loop() {
  double Ax, Ay, Az, T, Gx, Gy, Gz;

  Read_RawValue(MPU6050SlaveAddress, MPU6050_REGISTER_ACCEL_XOUT_H);

  //divide each with their sensitivity scale factor
  Ax = (double)AccelX/AccelScaleFactor;
  Ay = (double)AccelY/AccelScaleFactor;
  Az = (double)AccelZ/AccelScaleFactor;
  T = (double)Temperature/340+36.53; //temperature formula
  Gx = (double)GyroX/GyroScaleFactor;
  Gy = (double)GyroY/GyroScaleFactor;
  Gz = (double)GyroZ/GyroScaleFactor;
```

PRN:
Name:

```
 Serial.print("Ax: "); Serial.print(Ax);
 Serial.print(" Ay: "); Serial.print(Ay);
 Serial.print(" Az: "); Serial.print(Az);
 Serial.print(" T: "); Serial.print(T);
 Serial.print(" Gx: "); Serial.print(Gx);
 Serial.print(" Gy: "); Serial.print(Gy);
 Serial.print(" Gz: "); Serial.println(Gz);

 delay(100);
}

void I2C_Write(uint8_t deviceAddress, uint8_t regAddress, uint8_t data){
 Wire.beginTransmission(deviceAddress);
 Wire.write(regAddress);
 Wire.write(data);
 Wire.endTransmission();
}

// read all 14 register
void Read_RawValue(uint8_t deviceAddress, uint8_t regAddress){
 Wire.beginTransmission(deviceAddress);
 Wire.write(regAddress);
 Wire.endTransmission();
 Wire.requestFrom(deviceAddress, (uint8_t)14);
 AccelX = (((int16_t)Wire.read()<<8) | Wire.read());
 AccelY = (((int16_t)Wire.read()<<8) | Wire.read());
 AccelZ = (((int16_t)Wire.read()<<8) | Wire.read());
 Temperature = (((int16_t)Wire.read()<<8) | Wire.read());
 GyroX = (((int16_t)Wire.read()<<8) | Wire.read());
 GyroY = (((int16_t)Wire.read()<<8) | Wire.read());
 GyroZ = (((int16_t)Wire.read()<<8) | Wire.read());
}
```

PRN:
Name:

```
//configure MPU6050
void MPU6050_Init(){
 delay(150);
 I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_SMPLRT_DIV, 0x07);
 I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_PWR_MGMT_1, 0x01);
 I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_PWR_MGMT_2, 0x00);
 I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_CONFIG, 0x00);
 I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_GYRO_CONFIG, 0x00);//set +/-250 degree/second full scale
 I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_ACCEL_CONFIG, 0x00);// set +/- 2g full scale
 I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_FIFO_EN, 0x00);
 I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_INT_ENABLE, 0x01);
 I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_SIGNAL_PATH_RESET, 0x00);
 I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_USER_CTRL, 0x00);
  }
```

Conclusion: In this experiment, you successfully Accelerometer sensor MPU6050 interfaced with NodeMCU Mini.