

Second Milestone presentation

Nov 6th.

Format:

5 min presentation + 3 min Q& A

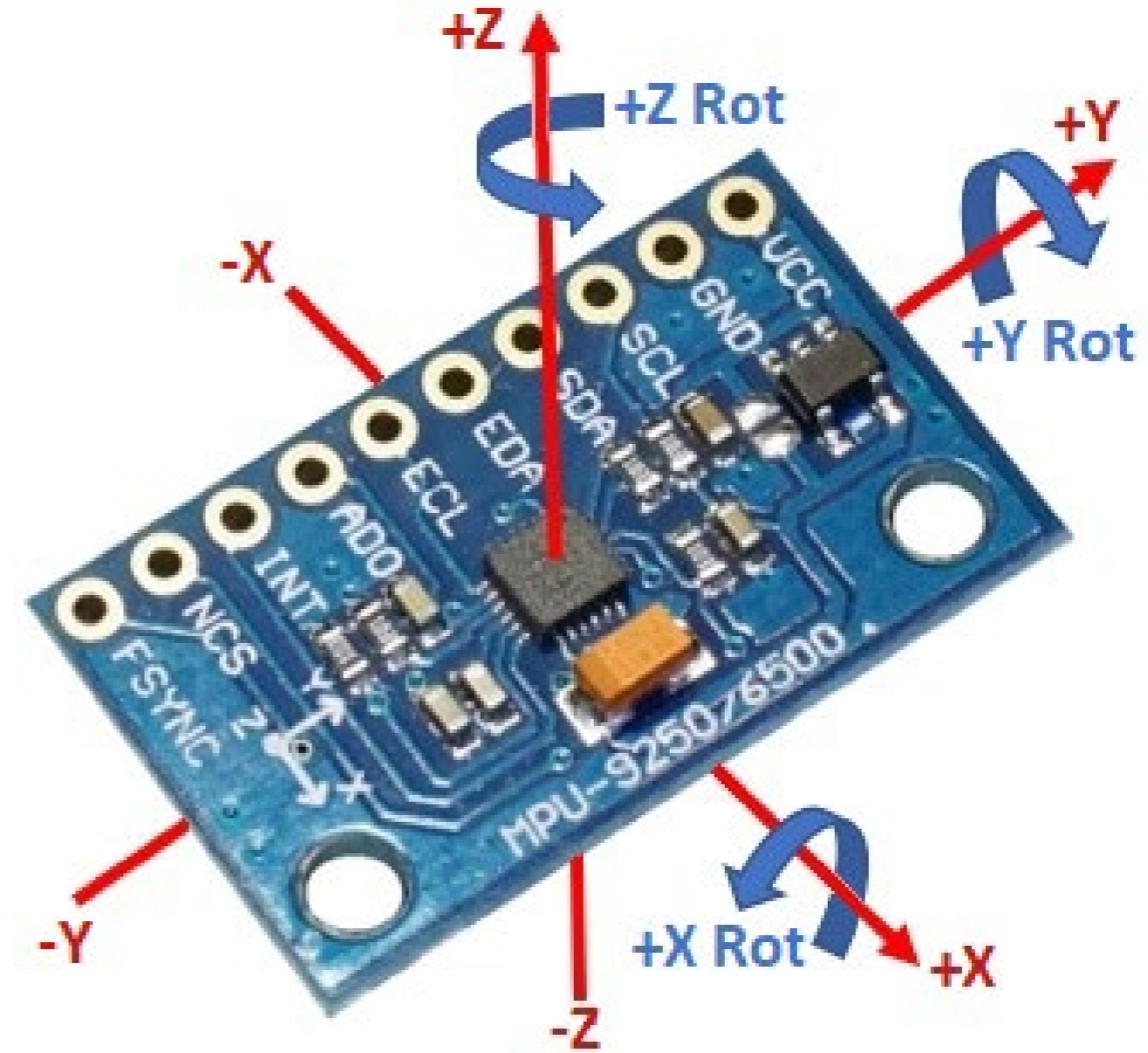
A brief overview of what you've been working on since the first milestone

A live demo to showcase your current progress (achievement and challenges)

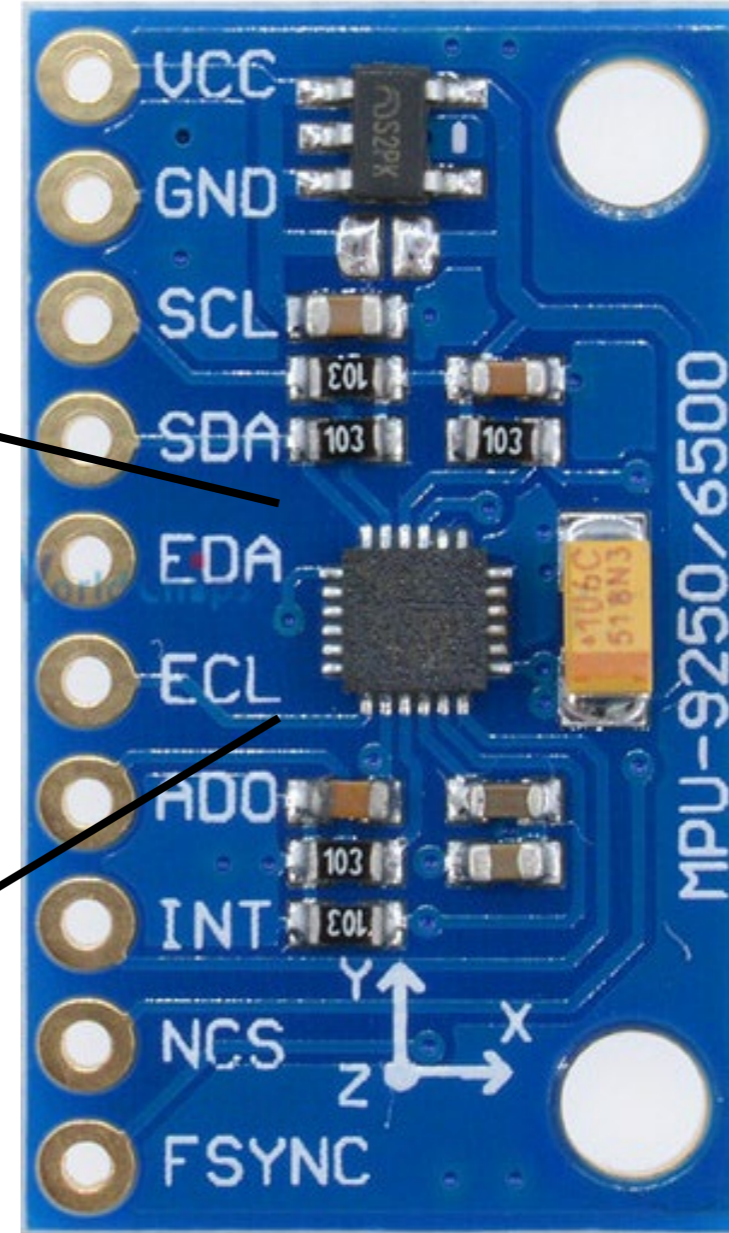
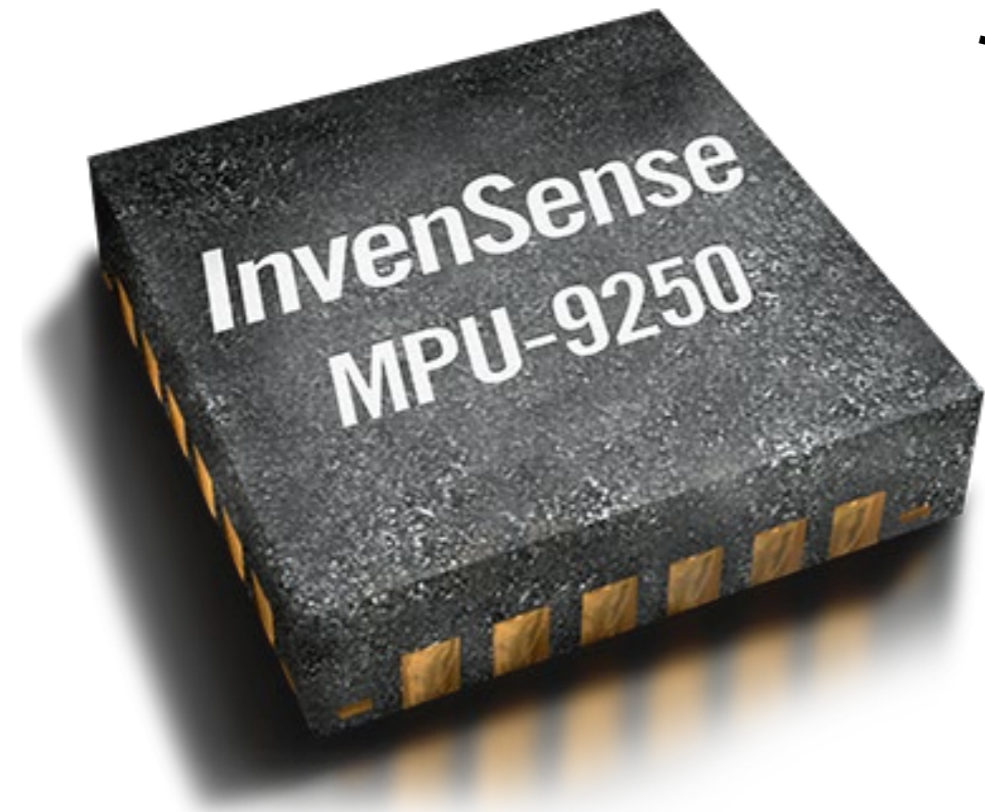
Plan for improvement

I²C and IMU 2

Huaishu Peng | UMD CS | Fall 2023



Accelerometer | Gyro | Magnetometer



Invensense MPU-9250

<https://invensense.tdk.com/products/motion-tracking/9-axis/mpu-9250/>

I²C library in Arduino - Wire library

Setup

```
byte ACCEL_XOUT_H = 0;
byte ACCEL_XOUT_L = 0;
int16_t ACCEL_X_RAW = 0;
float gX;
void loop() {
  // put your main code here, to run repeatedly:
  Wire.beginTransmission(address);
  Wire.write(0x3B);
  Wire.endTransmission();
```

Reading a register

Updating a register

```
Wire.requestFrom(address, 1);
ACCEL_XOUT_H = Wire.read();

Wire.beginTransmission(address);
Wire.write(0x3C);
Wire.endTransmission();
```

```
Wire.requestFrom(address, 1);
ACCEL_XOUT_L = Wire.read();

ACCEL_X_RAW = ACCEL_XOUT_H << 8 | ACCEL_XOUT_L;
gX = ACCEL_X_RAW / 16384.0;
Serial.println(gX);
delay(10);
}
```

Address: 0b1101000
(0x68)



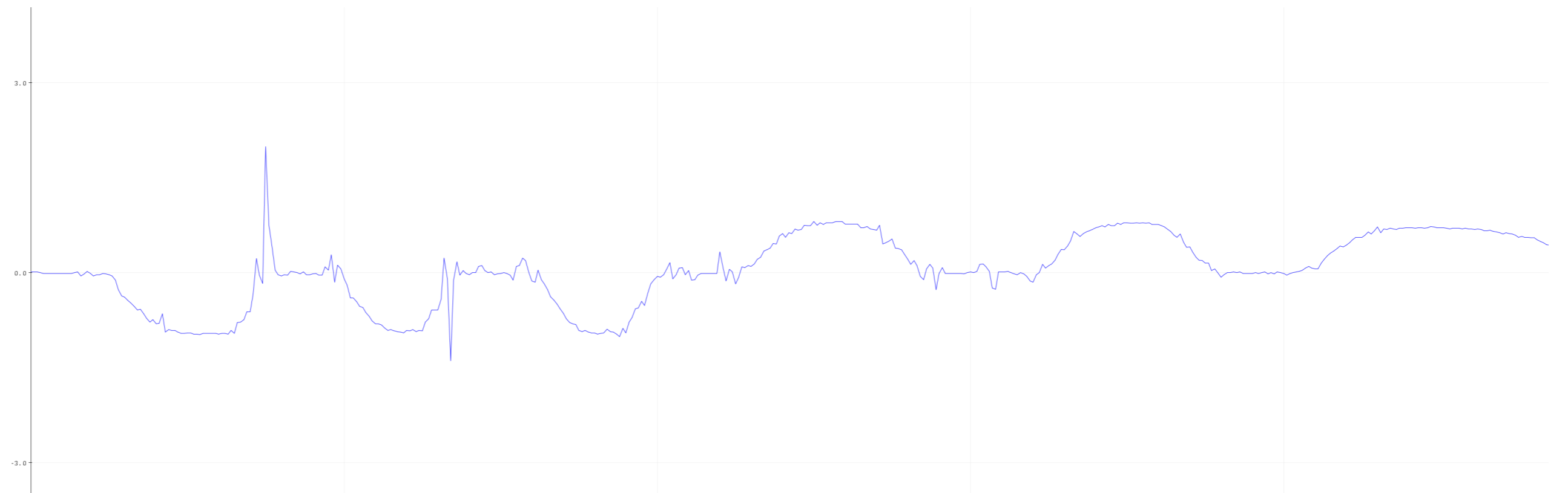
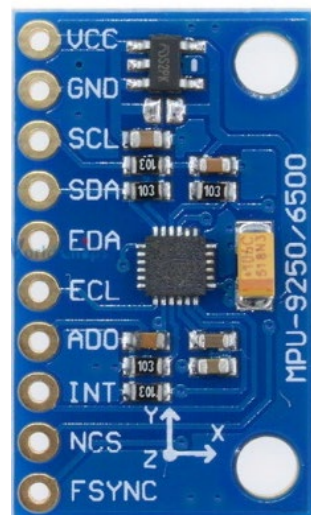
I²C library in Arduino - Wire library

Setup

Reading a register

Updating a register

**Address: 0b1101000
(0x68)**



I²C library in Arduino - Wire library

Practice: Read temperature of the sensor in degrees C

Datasheet:

https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/MPU-9250-Register-Map.pdf

<https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf>

I²C library in Arduino

Setup

Reading a register

Updating a register

REGISTER 53 - I2C_SLV4_DI.....	27
4.18 REGISTER 54 - I ² C MASTER STATUS.....	28
4.19 REGISTER 55 - INT PIN / BYPASS ENABLE CONFIGURATION.....	29
4.20 REGISTER 56 - INTERRUPT ENABLE.....	29
4.21 REGISTER 58 - INTERRUPT STATUS.....	30
4.22 REGISTERS 59 TO 64 - ACCELEROMETER MEASUREMENTS.....	31
4.23 REGISTERS 65 AND 66 - TEMPERATURE MEASUREMENT.....	33
4.24 REGISTERS 67 TO 72 - GYROSCOPE MEASUREMENTS.....	33
4.25 REGISTERS 73 TO 96 - EXTERNAL SENSOR DATA.....	35
4.26 REGISTER 99 - I ² C SLAVE 0 DATA OUT.....	37
4.27 REGISTER 100 - I ² C SLAVE 1 DATA OUT.....	37
4.28 REGISTER 101 - I ² C SLAVE 2 DATA OUT.....	37
4.29 REGISTER 102 - I ² C SLAVE 3 DATA OUT.....	37
4.30 REGISTER 103 - I ² C MASTER DELAY CONTROL.....	38
4.31 REGISTER 104 - SIGNAL PATH RESET.....	39
4.32 REGISTER 105 - ACCELEROMETER INTERRUPT CONTROL.....	39
4.33 REGISTER 106 - USER CONTROL.....	39
4.34 REGISTER 107 - POWER MANAGEMENT 1.....	40
4.35 REGISTER 108 - POWER MANAGEMENT 2.....	41
4.36 REGISTER 114 AND 115 - FIFO COUNT REGISTERS.....	42
4.37 REGISTER 116 - FIFO READ WRITE.....	43
4.38 REGISTER 117 - WHO AM I.....	44
4.39 REGISTERS 119, 120, 122, 123, 125, 126 ACCELEROMETER OFFSET REGISTERS.....	44

4.23 Registers 65 and 66 - **Temperature** Measurement

Name: TEMP_OUT_H

Serial IF: SyncR

Reset value: 0x00 (if sensor disabled)

BIT	NAME	FUNCTION
[7:0]	D[7:0]	High byte of the temperature sensor output

Name: TEMP_OUT_L

Serial IF: SyncR

Reset value: 0x00 (if sensor disabled)

BIT	NAME	FUNCTION
[7:0]	D[7:0]	Low byte of the temperature sensor output: <div style="border: 1px solid black; padding: 5px; margin: 5px 0;">$\text{TEMP_degC} = ((\text{TEMP_OUT} - \text{RoomTemp_Offset}) / \text{Temp_Sensitivity}) + 21\text{degC}$</div> Where Temp_degC is the temperature in degrees C measured by the temperature sensor. TEMP_OUT is the actual output of the temperature sensor.

3.4.2 A.C. Electrical Characteristics

Typical Operating Circuit of section 4.2, VDD = 2.5V, VDDIO = 2.5V, T_A=25°C, unless otherwise noted.

Parameter	Conditions	MIN	TYP	MAX	Units
Supply Ramp Time	Monotonic ramp. Ramp rate is 10% to 90% of the final value	0.1		100	ms
Operating Range	Ambient	-40		85	°C
Sensitivity	Untrimmed		333.87		LSB/°C

I²C library in Arduino

Setup

Reading a register

Updating a register

```
#include <Wire.h>
const int MPU = 0x68;

void setup() {
  Serial.begin(19200);
  Wire.begin(); // Initialize communication
}

void loop() {
```

```
  Wire.beginTransmission(MPU);
  Wire.write(0x41);
  Wire.endTransmission();
  Wire.requestFrom(MPU, 2);
```

```
  int16_t temperature = Wire.read() << 8 | Wire.read();
  Serial.println(temperature / 333.87 + 21);
  delay(20);
}
```

3.4.2 A.C. Electrical Characteristics

Typical Operating Circuit of section 4.2, VDD = 2.5V, VDDIO = 2.5V, T_A=25°C, unless otherwise noted.

Parameter	Conditions	MIN	TYP	MAX	Units
Supply Ramp Time	Monotonic ramp. Ramp rate is 10% to 90% of the final value	0.1		100	ms
Operating Range	Ambient	-40		85	°C
Sensitivity	Untrimmed		333.87		LSB/°C

BIT	NAME	FUNCTION
[7:0]	D[7:0]	Low byte of the temperature sensor output: $\text{TEMP_degC} = ((\text{TEMP_OUT} - \text{RoomTemp_Offset}) / \text{Temp_Sensitivity}) + 21\text{degC}$ Where Temp_degC is the temperature in degrees C measured by the temperature sensor. TEMP_OUT is the actual output of the temperature sensor.

I²C library in Arduino - Wire library

Setup

Reading a register

Updating a register

Let's try to read acceleration data **in the range of +/- 16g**
so that we can **detect strong sudden motions!**

I²C library in Arduino - Wire library

Setup

4.7 Register 28 – Accelerometer Configuration

Serial IF: R/W

Reset value: 0x00

Reading a register

Updating a register

BIT	NAME	FUNCTION
[7]	ax_st_en	X Accel self-test
[6]	ay_st_en	Y Accel self-test
[5]	az_st_en	Z Accel self-test
[4:3]	ACCEL_FS_SEL[1:0]	Accel Full Scale Select: ±2g (00), ±4g (01), ±8g (10), ±16g (11)
[2:0]	-	Reserved

Wire.beginTransmission(addr) //opens communication with the slave device with its addr

Wire.write(data) //prepares to send data to addr

Wire.write(data) //prepares to send data to addr

...

Wire.endTransmission() //sends the data and returns

I²C library in Arduino - Wire library

Setup

```
void setup() {  
  Serial.begin(115200);  
  Wire.begin(); // Initialize communication
```

Reading a register

```
Wire.beginTransmission(MPU);  
Wire.write(0x1C); //Talk to the ACCEL_CONFIG register (1C hex)  
Wire.write(0b00011000); //Set the register bits as 00011000 (+/- 16g full scale range)  
Wire.endTransmission();
```

Updating a register

```
}
```

4.7 Register 28 – Accelerometer Configuration

Serial IF: R/W

Reset value: 0x00

BIT	NAME	FUNCTION
[7]	ax_st_en	X Accel self-test
[6]	ay_st_en	Y Accel self-test
[5]	az_st_en	Z Accel self-test
[4:3]	ACCEL_FS_SEL[1:0]	Accel Full Scale Select: ±2g (00), ±4g (01), ±8g (10), ±16g (11)
[2:0]	-	Reserved

Wire.beginTransmission(addr) //opens communication with the slave device with its addr

Wire.write(data) //prepares to send data to addr

Wire.write(data) //prepares to send data to addr

...

Wire.endTransmission() //sends the data and returns

I²C library in Arduino - Wire library

Setup

Sensitivity Scale Factor	AFS_SEL=0		16,384	LSB/g
	AFS_SEL=1		8,192	LSB/g
	AFS_SEL=2		4,096	LSB/g
	AFS_SEL=3		2,048	LSB/g

Reading a register

Updating a register

```
AccX = (Wire.read() << 8 | Wire.read()) / 2048.0; // X-axis value  
AccY = (Wire.read() << 8 | Wire.read()) / 2048.0; // Y-axis value  
AccZ = (Wire.read() << 8 | Wire.read()) / 2048.0; // Z-axis value
```

Wire.beginTransmission(addr) //opens communication with the slave device with its addr

Wire.write(data) //prepares to send data to addr

Wire.write(data) //prepares to send data to addr

...

Wire.endTransmission() //sends the data and returns

I²C library in Arduino - Wire library

Understanding Gyro Data

BIT	NAME	FUNCTION
[7:0]	D[7:0]	Low byte of the X-Axis gyroscope output GYRO_XOUT = Gyro_Sensitivity * X_angular_rate Nominal FS_SEL = 0 Conditions Gyro_Sensitivity = 131 LSB/(°/s)

degree per second

I²C library in Arduino

Understanding Gyro Data

```
gyroX_Per_S = GYRO_X_RAW/131.0;  
gyroY_Per_S = GYRO_Y_RAW/131.0;  
gyroZ_Per_S = GYRO_Z_RAW/131.0;
```

```
currentTime = millis();  
elapsedTime = (currentTime - previousTime) / 1000;
```

```
gyroAngleX = gyroAngleX + gyroX_Per_S * elapsedTime;  
gyroAngleY = gyroAngleY + gyroY_Per_S * elapsedTime;  
gyroAngleZ = gyroAngleZ + gyroZ_Per_S * elapsedTime;
```

I²C library in Arduino

Dead Reckoning

- Drifting over time because errors accumulated and built upon previous measurements -> data won't be accurate
- Still, we can reduce the error with a simple calibration process
- For example, you can record 10s of raw x y z gyro data to find the average offset -> which you can plug into your final output
- Offset varies based on your device

```
gyroX_Per_S = GYRO_X_RAW/131.0 - caliX;  
gyroY_Per_S = GYRO_Y_RAW/131.0 - caliY;  
gyroZ_Per_S = GYRO_Z_RAW/131.0 - caliZ;
```