Second Milestone presentation

We will move it from Oct 26 to Oct 31 (from Wed to Mon) so that you have a bit more time to work on it.

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

Documentation (which is due on Wed next week)
I²C and IMU 2

Huaishu Peng | UMD CS | Fall 2022
InvenSense MPU-9250

I²C library in Arduino - Wire library

Setup

Reading a register

Updating a register

Address: 0b1101000
(0x68)

```c
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();

    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);
}
```
I²C library in Arduino - Wire library

Setup

**Reading** a register

**Updating** a register

Address: 0b1101000 (0x68)
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
I²C library in Arduino

Setup

Reading a register

Updating a register

4.23 Registers 65 and 66 – Temperature Measurement

Name: TEMP_OUT_H
Serial IF: SyncR
Reset value: 0x00 (if sensor disabled)

<table>
<thead>
<tr>
<th>BIT</th>
<th>NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>[7:0]</td>
<td>D[7:0]</td>
<td>High byte of the temperature sensor output</td>
</tr>
</tbody>
</table>

Name: TEMP_OUT_L
Serial IF: SyncR
Reset value: 0x00 (if sensor disabled)

<table>
<thead>
<tr>
<th>BIT</th>
<th>NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>[7:0]</td>
<td>D[7:0]</td>
<td>Low byte of the temperature sensor output:</td>
</tr>
</tbody>
</table>

\[
TEMP_{\text{degC}} = \left(\frac{\text{TEMP\_OUT}}{\text{RoomTemp\_Offset}}\right) \times 10_{\text{degC}}
\]

Where Temp_{degC} is the temperature in degrees C measured by the temperature sensor. TEMP_OUT is the actual output of the temperature sensor.

3.4.2 A.C. Electrical Characteristics

Typical Operating Circuit of section 4.2. VDD = 2.5V, VDDIO = 2.5V, T_{x} = 25°C, unless otherwise noted.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Conditions</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Ramp Time</td>
<td>Monotonic ramp. Ramp rate is 10% to 90% of the final value</td>
<td>0.1</td>
<td>100</td>
<td></td>
<td>ms</td>
</tr>
<tr>
<td>Operating Range</td>
<td>Ambient</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensitivity</td>
<td>Untrimmed</td>
<td>-40</td>
<td>85</td>
<td></td>
<td>°C</td>
</tr>
</tbody>
</table>

Typical values: 333.87 LSB/°C
I²C library in Arduino

```
#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);
}

```

---

**Setup**

**Reading a register**

**Updating a register**
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

Reading a register

Updating a register

4.7 Register 28 – Accelerometer Configuration

<table>
<thead>
<tr>
<th>BIT</th>
<th>NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>[7]</td>
<td>ax_st_en</td>
<td>X Accel self-test</td>
</tr>
<tr>
<td>[4:3]</td>
<td>ACCEL_FS_SEL[1:0]</td>
<td>Accel Full Scale Select:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±2g (00), ±4g (01), ±8g (10), ±16g (11)</td>
</tr>
<tr>
<td>[2:0]</td>
<td>-</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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

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

  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();
}

Setup

Reading a register

Updating a register

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

### Reading 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

### Updating a register

```cpp
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
```
Understanding Gyro Data

<table>
<thead>
<tr>
<th>BIT</th>
<th>NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>[7:0]</td>
<td>D[7:0]</td>
<td>Low byte of the X-Axis gyroscope output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[GYRO_XOUT = \text{Gyro_Sensitivity} \times \text{X_angular_rate}]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Nominal FS_SEL = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conditions Gyro_Sensitivity = 131 LSB/(°/s)</td>
</tr>
</tbody>
</table>

degree per second
Understanding Gyro Data

```c
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;
```
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

```
# Ardunio code example for calibration

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;
```