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
Invensense MPU-9250
I²C library in Arduino - Wire library

```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
# I2C library in Arduino

```cpp
#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

<table>
<thead>
<tr>
<th>BIT</th>
<th>NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>0:7</td>
<td></td>
</tr>
</tbody>
</table>

Low byte of the temperature sensor output:

\[
\text{TEMP\_degC} = \left[\left(\text{TEMP\_OUT} - \text{RoomTemp\_Offset}\right) / \text{Temp\_Sensitivity}\right] + 21\text{degC}
\]

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!
I\(^2\)C library in Arduino - Wire library

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

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();
}

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

```
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

<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 =     Gyro_Sensitivity * 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
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;
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

```c
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;
```