

CMP-5045B – Embedded Systems

Peripheral Tutorial:

MPU 6050 – 6 Axis Accelerometer & Gyroscope

Created by Joshua Crafton - Team C

# 1 Introduction

This tutorial introduces the use of a micro-processing unit(MPU) that is able to track motion relative to gravity. We will use a buzzer and LED to indicate when a 45 degree rotational angle has been detected.

Accelerometers are typically used in phones to orientate the device, however they can also detect vibrations which can be used to measure seismic activity or machine vibrations.

Gyroscopes can be used as stabilization devices to reduce camera shake or assist robotic systems to keep them upright. The device can measure external rotational force and convert it to a digital system.

# 2 Equipment

1. STM32F746-G discovery board
2. Keil uVision MDK-Lite (v5.18 to v5.36)
3. MPU 6050
4. Buzzer
5. LED

# 3 Objective

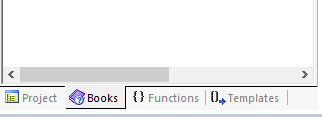
To create a project that can interact with an external MPU, display rotational angles on the GLCD screen and control an external output (buzzer).

# 4 Procedure

1. First, we must create a new project. You should work in a local storage drive. The STM32F746-G discovery board comes with the STM32F746NG device.
2. The RTE manager is configured as shown below. For this tutorial, we will need GLCD, GPIO, I2C drivers and some core systems like HAL and startup drivers:
   * Board Support ⇒ STM32F746G-Discovery;
   * Board Support ⇒ Graphic LCD (API);
   * Board Support ⇒ Drivers ⇒ SDRAM;
   * CMSIS ⇒ Core;
   * CMSIS ⇒ RTOS(API) ⇒ Keil RTX;
   * Device ⇒ Startup;
   * Device ⇒ STM32Cube Framework (API) ⇒ Classic;
   * Device ⇒ STM32Cube HAL ⇒ Common, Cortex, DMA, GPIO, I2C, LTDC, PWR, RCC, SDRAM

Click ‘Resolve’ once you have selected the components, then click ‘OK’.

1. It is important to remember that we have to switch the debugger to ST-LINK instead of the default ULINK, and tick the ‘Reset and Run’ box in the debugger settings.
2. Moving to the books tab at the bottom you can find all relevant information for the STM board and MDK functionalities. Open the User Manual (STM32F746G-Discorvery) file and located page 23 to see the Arduino connectors.



1. The MPU will require 4 pins, two of which are VCC and GND. The last 2 are for I2C communication over the SLC and SDA pin functions. On the discovery board, the pins PB8 and PB9 are connected to these functions respectfully and will need to be initialised. GPIO clock “B” needs to be started and the GPIO struct should be as follows:
   * Mode => GPIO\_MODE\_AF\_OD ;(this is an open drain alternate function)
   * Pull => GPIO\_PULLUP;
   * Speed => GPIO\_SPEED\_FREQ\_VERY\_HIGH;
   * Alternate = GPIO\_AF4\_I2C1 ;

The buzzer and LED can be on any digital pin however this tutorial uses PI3, will require clock “I” and has the following struct:

* + Mode => GPIO\_MODE\_OUTPUT\_PP;
  + Pull => GPIO\_NOPULL;
  + Speed => GPIO\_SPEED\_FREQ\_LOW;
  + Alternate = NULL;

1. These are the includes required for this project.

#include <stdio.h>

#include <math.h>

#include <string.h>

#include <stdlib.h>

#include "stm32f7xx\_hal.h"

#include "stm32f7xx\_hal\_gpio.h"

#include "GLCD\_Config.h"

#include "Board\_GLCD.h"

extern GLCD\_FONT GLCD\_Font\_16x24;

#ifdef \_\_RTX

extern uint32\_t os\_time;

uint32\_t HAL\_GetTick(void) {

return os\_time;

}

#endif

1. Next the system clock, the I2C1 and TIM2 settings need to be Initialised with the code below.

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct;

RCC\_ClkInitTypeDef RCC\_ClkInitStruct;

/\* Enable Power Control clock \*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

/\* The voltage scaling allows optimizing the power

consumption when the device is clocked below the

maximum system frequency. \*/

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

/\* Enable HSE Oscillator and activate PLL

with HSE as source \*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLM = 25;

RCC\_OscInitStruct.PLL.PLLN = 336;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 7;

HAL\_RCC\_OscConfig(&RCC\_OscInitStruct);

/\* Select PLL as system clock source and configure

the HCLK, PCLK1 and PCLK2 clocks dividers \*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_SYSCLK |

RCC\_CLOCKTYPE\_PCLK1 | RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;

HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_5);

}

static void I2C1\_Init(void)

{

\_\_HAL\_RCC\_I2C1\_CLK\_ENABLE();

\_\_HAL\_RCC\_I2C1\_FORCE\_RESET();

HAL\_Delay(2);

\_\_HAL\_RCC\_I2C1\_RELEASE\_RESET();

hi2c1.Instance = I2C1;

hi2c1.Init.Timing = 0x00808CD2;

hi2c1.Init.OwnAddress1 = 0;

hi2c1.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;

hi2c1.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLE;

hi2c1.Init.OwnAddress2 = 0;

hi2c1.Init.OwnAddress2Masks = I2C\_OA2\_NOMASK;

hi2c1.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLE;

hi2c1.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLE;

HAL\_I2C\_Init(&hi2c1) ;

/\*\* Configure Analogue filter\*/

HAL\_I2CEx\_ConfigAnalogFilter(&hi2c1, I2C\_ANALOGFILTER\_ENABLE);

/\*\* Configure Digital filter\*/

HAL\_I2CEx\_ConfigDigitalFilter(&hi2c1, 0);

}

static void TIM2\_Init(void)

{

TIM\_MasterConfigTypeDef sMasterConfig = {0};

htim2.Instance = TIM2;

htim2.Init.Prescaler = 32000;

htim2.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim2.Init.Period = 1;

htim2.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

HAL\_TIM\_Base\_Init(&htim2);

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

HAL\_TIMEx\_MasterConfigSynchronization(&htim2, &sMasterConfig);

}

1. To start using the MPU you will need to define the registries and global variables using the code below.

TIM\_HandleTypeDef htim2;

I2C\_HandleTypeDef hi2c1;

const float M\_PI = 3.14159265358979323846;

#define MPU6050\_ADDR (0x68 << 1) // 0xD0 <- try using this if it doesn’t work

#define SMPLRT\_DIV\_REG 0x19

#define GYRO\_CONFIG\_REG 0x1B

#define ACCEL\_CONFIG\_REG 0x1C

#define ACCEL\_XOUT\_H\_REG 0x3B

#define TEMP\_OUT\_H\_REG 0x41

#define GYRO\_XOUT\_H\_REG 0x43

#define PWR\_MGMT\_1\_REG 0x6B

#define WHO\_AM\_I\_REG 0x75

int16\_t Accel\_X\_RAW = 0;

int16\_t Accel\_Y\_RAW = 0;

int16\_t Accel\_Z\_RAW = 0;

int16\_t Gyro\_X\_RAW = 0;

int16\_t Gyro\_Y\_RAW = 0;

int16\_t Gyro\_Z\_RAW = 0;

float pitch = 0;

float roll = 0;

float yaw = 0;

float Ax, Ay, Az, Gx, Gy, Gz;

1. The MPU can then be initialised with the function below. If the check variable doesn’t return 0x68 then make sure wires are on the correct pins. The MPU board should tell you which pin is which, if not find the datasheet for the MPU. Once this connection is running it doesn’t need to be called again.

void MPU6050\_Init (void)

{

uint8\_t check;

uint8\_t Data;

// check device ID WHO\_AM\_I

HAL\_I2C\_Mem\_Read (&hi2c1, MPU6050\_ADDR,WHO\_AM\_I\_REG,1, &check, 1, 1000);

if (check == 104) // 0x68 will be returned by the sensor if everything goes well

{

// power management register 0X6B we should write all 0's to wake the sensor up

Data = 0;

HAL\_I2C\_Mem\_Write(&hi2c1, MPU6050\_ADDR, PWR\_MGMT\_1\_REG, 1,&Data, 1, 1000);

// Set DATA RATE of 1KHz by writing SMPLRT\_DIV register

Data = 0x07;

HAL\_I2C\_Mem\_Write(&hi2c1, MPU6050\_ADDR, SMPLRT\_DIV\_REG, 1, &Data, 1, 1000);

// Set accelerometer configuration in ACCEL\_CONFIG Register

// XA\_ST=0,YA\_ST=0,ZA\_ST=0, FS\_SEL=0 -> ± 2g

Data = 0x00;

HAL\_I2C\_Mem\_Write(&hi2c1, MPU6050\_ADDR, ACCEL\_CONFIG\_REG, 1, &Data, 1, 1000);

// Set Gyroscopic configuration in GYRO\_CONFIG Register

// XG\_ST=0,YG\_ST=0,ZG\_ST=0, FS\_SEL=0 -> ± 250 °/s

Data = 0x00;

HAL\_I2C\_Mem\_Write(&hi2c1, MPU6050\_ADDR, GYRO\_CONFIG\_REG, 1, &Data, 1, 1000);

}

}

1. Next are the functions that are called in the main() loop. These functions read the raw data from the MPU and places it on an axis. The convertACC() function then takes these values and converts it to an angle that can be displayed between -90 and 90 degrees based on if the device has been rolled left or right.

void MPU6050\_Read\_Accel (void)

{

uint8\_t Rec\_Data[6];

// Read 6 BYTES of data starting from ACCEL\_XOUT\_H register

HAL\_I2C\_Mem\_Read (&hi2c1, MPU6050\_ADDR, ACCEL\_XOUT\_H\_REG, 1, Rec\_Data, 6, 1000);

Accel\_X\_RAW = (int16\_t)(Rec\_Data[0] << 8 | Rec\_Data [1]);

Accel\_Y\_RAW = (int16\_t)(Rec\_Data[2] << 8 | Rec\_Data [3]);

Accel\_Z\_RAW = (int16\_t)(Rec\_Data[4] << 8 | Rec\_Data [5]);

/\*\*\* convert the RAW values into acceleration in 'g'

we have to divide according to the Full scale value set in FS\_SEL

I have configured FS\_SEL = 0. So I am dividing by 16384.0

for more details check ACCEL\_CONFIG Register \*\*\*\*/

Ax = Accel\_X\_RAW/16384.0;

Ay = Accel\_Y\_RAW/16384.0;

Az = Accel\_Z\_RAW/16384.0;

}

void MPU6050\_Read\_Gyro (void)

{

uint8\_t Rec\_Data[6];

// Read 6 BYTES of data starting from GYRO\_XOUT\_H register

HAL\_I2C\_Mem\_Read (&hi2c1, MPU6050\_ADDR, GYRO\_XOUT\_H\_REG, 1, Rec\_Data, 6, 1000);

Gyro\_X\_RAW = (int16\_t)(Rec\_Data[0] << 8 | Rec\_Data [1]);

Gyro\_Y\_RAW = (int16\_t)(Rec\_Data[2] << 8 | Rec\_Data [3]);

Gyro\_Z\_RAW = (int16\_t)(Rec\_Data[4] << 8 | Rec\_Data [5]);

/\*\*\* convert the RAW values into dps (°/s)

we have to divide according to the Full scale value set in FS\_SEL

I have configured FS\_SEL = 0. So I am dividing by 131.0

for more details check GYRO\_CONFIG Register \*\*\*\*/

Gx = Gyro\_X\_RAW/131.0;

Gy = Gyro\_Y\_RAW/131.0;

Gz = Gyro\_Z\_RAW/131.0;

}

void convertAcc (void){ //https://engineering.stackexchange.com/questions/3348/calculating-pitch-yaw-and-roll-from-mag-acc-and-gyro-data

pitch = 180 \* atan (Ax/sqrt(Ay\*Ay + Az\*Az))/M\_PI;

roll = 180 \* atan (Ay/sqrt(Ax\*Ax + Az\*Az))/M\_PI;

yaw = 180 \* atan (Az/sqrt(Ax\*Ax + Az\*Az))/M\_PI;

}

1. In main() use the code below. It will give show many different values as the device moves

int main(void){

char buf[5];

HAL\_Init(); //Init Hardware Abstraction Layer

SystemClock\_Config(); //Config Clocks

SystemCoreClockUpdate();

GPIO\_Init();

I2C1\_Init();

TIM2\_Init();

\_\_HAL\_RCC\_TIM2\_CLK\_ENABLE();

GLCD\_Initialize(); //Init GLCD

GLCD\_ClearScreen();

GLCD\_SetFont(&GLCD\_Font\_16x24);

GLCD\_SetForegroundColor(GLCD\_COLOR\_PURPLE);

MPU6050\_Init();

HAL\_Delay(2000);

for(;;) {

MPU6050\_Read\_Accel();

MPU6050\_Read\_Gyro();

sprintf(buf, "%.2f", Ax);

GLCD\_DrawString((2\*24), (1\*24), buf);

sprintf(buf, "%.2f", Ay);

GLCD\_DrawString((7\*24), (1\*24), buf);

sprintf(buf, "%.2f", Az);

GLCD\_DrawString((12\*24), (1\*24), buf);

sprintf(buf, "%.2f", Gx);

GLCD\_DrawString((2\*24), (4\*24), buf);

sprintf(buf, "%.2f", Gy);

GLCD\_DrawString((7\*24), (4\*24), buf);

sprintf(buf, "%.2f", Gz);

GLCD\_DrawString((12\*24), (4\*24), buf);

convertAcc();

GLCD\_DrawString((2\*24), (2\*24), "pitch");

sprintf(buf, "%.2f", pitch);

GLCD\_DrawString((2\*24), (4\*24), buf);

GLCD\_DrawString((7\*24), (2\*24), "roll");

sprintf(buf, "%.2f", roll);

GLCD\_DrawString((7\*24), (4\*24), buf);

GLCD\_DrawString((12\*24), (2\*24), "yaw");

sprintf(buf, "%.2f", yaw);

GLCD\_DrawString((12\*24), (4\*24), buf);

HAL\_Delay(250);

}

}

# 5 Display angle on a semi-circle

1. For this part you will need a method for drawing diagonal lines that can be found below.

void drawDiagonalLineLow(int x0, int y0, int x1, int y1)

{

int dx, dy, yi, D, x, y;

dx = x1 - x0;

dy = y1 - y0;

yi = 1;

if (dy < 0)

{

yi = -1;

dy = -dy;

}

D = (2 \* dy) - dx;

y = y0;

for (x = x0; x < x1; x++)

{

GLCD\_DrawPixel(x, y);

if (D > 0)

{

y = y + yi;

D = D + (2 \* (dy - dx));

}

else

{

D = D + 2\*dy;

}

}

}

void drawDiagonalLineHigh(int x0, int y0, int x1, int y1)

{

int dx, dy, xi, D, x, y;

dx = x1 - x0;

dy = y1 - y0;

xi = 1;

if (dx < 0)

{

xi = -1;

dx = -dx;

}

D = (2 \* dx) - dy;

x = x0;

for (y = y0; y < y1; y++)

{

GLCD\_DrawPixel(x, y);

if (D > 0)

{

x = x + xi;

D = D + (2 \* (dx - dy));

}

else

{

D = D + 2\*dx;

}

}

}

The next part needs to be beneath the ones above so that they can be called. This function finds out if the line is being draw upwards or downwards and the functions above find the distance between the two points and draw pixels in-between.

void drawDiagonalLine(int x0, int y0, int x1, int y1)

{

if (abs(y1 - y0) < abs(x1 - x0))

{

if (x0 > x1)

{

drawDiagonalLineLow(x1, y1, x0, y0);

}

else

{

drawDiagonalLineLow(x0, y0, x1, y1);

}

}

else

{

if (y0 > y1)

{

drawDiagonalLineHigh(x1, y1, x0, y0);

}

else

{

drawDiagonalLineHigh(x0, y0, x1, y1);

}

}

}

1. Next use these functions to calculate the radian value of the roll angle and find the circumference from the origin and the endpoint of the diagonal line.

float toRadians(float angle){

return angle \* ( M\_PI / 180.0 );

}

void getCircumferenceXY (int x0, int y0, int r, float angle){

float xPos = 0;

float yPos = 0;

float radAngle = toRadians(angle) + (3\*M\_PI)/2;

xPos = r \* (float)cos(radAngle) + x0;

yPos = r \* (float)sin(radAngle) + y0;

circX = (int)xPos;

circY = (int)yPos;

}

1. Add these global variables to you others:
   * int circX = 240;
   * int circY = 182;
2. The drawstrings in main can be removed and replaced with the diagonal line function, make sure you keep Read\_Accel, Read\_Gyro and convertAcc() in the infinite loop.
   * Change foreground colour to the background colour and use the diagonal function to remove the previous line.
   * Get the next X and Y position for the endpoint of the line.
   * Change foreground back to the colour you want.
   * Then draw the diagonal line.

If done correctly the line will follow the angle of lean that the MPU has when facing you.

1. From here you are in a position to add the buzzer so that is makes noise when the degree angle reaches 45 or -45.

# 6 References

Accelerometer Initialisation and Setup:

https://controllerstech.com/how-to-interface-mpu6050-gy-521-with-stm32/

Accelerometer Pitch, Roll and Yaw Calculations:

https://engineering.stackexchange.com/questions/3348/calculating-pitch-yaw-and-roll-from-mag-acc-and-gyro-data