# the Koopa Slingers

Generated by Doxygen 1.10.0

# Chapter 1

# MAIN PAGE

## 1.1 Goals

- Understand the use of wireless protocols like Bluetooth
- Understand electronics considerations and budget management using bill of material
- Use SolidWorks to CAD the external components
- Use Fusion 360 to draw the hardware schematics and draw the PCB
- Use CubeIDE to code in C and understand the different functions of each hardware pin.

## 1.2 Requirements

This project requirements were made in California Polytechnic San Luis Obispo and provided by Professor Charlie Revfrem:

### 1.2.1 The Electronics Requirements:

- A custom PCB designed around an STM32F411 MCU (or similar) programmed in either C, C++ (or Rust, with permission).
- Two or more actuators, such as motors, driving the machine, actuated by suitable electronics, such as motor drivers.
- Two or more unique sensors.
- Some sort of closed-loop control loop or similarly complex algorithm.
- A wireless controller allowing you to command the bot hands-free or to be used as a wireless e-stop. The controller and receiver will be provided to students for use during ME 507.

### 1.2.2 The Manufacturing Requirements:

- 3D Printing: All custom non-flat parts should likely be manufactured using FDM 3D printing.
- PCB Fabrication: Flat parts can be made to order out of PCB material (fiberglass) along with the PCB controlling your robot. This option provides high precision and rigidity.
- Laser-cutting: Flat parts can be made quickly and accurately on the laser cutter if the parts are of a suitable material; many materials are not safe to cut with a laser.
- Water-jetting: Flat parts not suitable for the laser cutter can be made accurately and quickly using the water-jet.

### 1.2.3 The Safety Requirements:

- If you use a battery, you need to communicate with your instructor before plugging it in or charging it in the lab.
- Your robot must have an emergency stop feature that can be triggered by the provided radio transmitter. For example, releasing the trigger on the transmitter can act as a “dead man’s switch”. The emergency stop must also activate if communication is lost with the wireless controller.
- You may also want to include a way to safely grab your robot if it is mobile in nature, just in case your robot gains sentience and goes after one of the operators.
- Your robot may not injure a human being or, through inaction, allow a human being to come to harm.
- Your robot must obey the orders given it by human beings except where such orders would conflict with the First Law.
- Your robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

## 1.3 Hardware

If you want to learn more about the hardware, click here.

## 1.4 Software

If you want to learn more about the software, click here.

## 1.5 REPOSITORY REFERENCE

All code that will be referenced in this portfolio relate to the project is accessible through `https://github.↩ com/jlam94/ME-507-Portfolio`.

However, you may find it more useful to read through the website exploring the source code.

## 1.6 CONTACT INFO

|  |  |
|---|---|
| **Author:** | Jonathan Lam |
| **Email:** | jonathanlam0806@gmail.com |
| **LinkedIn:** | www.linkedin.com/in/jonathanlam2000 |
| **Phone Number:** | (909) 760-9129 |
| **Major:** | Mechanical Engineer at Cal Poly San Luis Obispo |
| **Date:** | May 18, 2024 |

# Chapter 2

# Hardware Documentation

## 2.1 Introduction

Detailed description of the hardware components used in the project.

## 2.2 Components

- List of hardware components
- Schematics and diagrams

## 2.3 Design

- Design considerations
- Design process

# Chapter 3

# Software Documentation

## 3.1 Introduction

Detailed description of the software components used in the project.

## 3.2 Architecture

- Software architecture overview
- Key modules and their interactions

## 3.3 Code

- Code examples
- Key functions and their descriptions

# Chapter 4

# Topic Index

## 4.1 Topics

Here is a list of all topics with brief descriptions:

# Chapter 5

# Data Structure Index

## 5.1 Data Structures

Here are the data structures with brief descriptions:

# Chapter 6

# File Index

## 6.1 File List

Here is a list of all files with brief descriptions:

# Chapter 7

# Topic Documentation

## 7.1 CMSIS

**Topics**

- Stm32f4xx_system

### 7.1.1 Detailed Description

### 7.1.2 Stm32f4xx_system

**Topics**

- STM32F4xx_System_Private_Includes
- STM32F4xx_System_Private_TypesDefinitions
- STM32F4xx_System_Private_Defines
- STM32F4xx_System_Private_Macros
- STM32F4xx_System_Private_Variables
- STM32F4xx_System_Private_FunctionPrototypes
- STM32F4xx_System_Private_Functions

#### 7.1.2.1 Detailed Description

#### 7.1.2.2 STM32F4xx_System_Private_Includes

**Macros**

- #define HSE_VALUE ((uint32_t)25000000)
- #define HSI_VALUE ((uint32_t)16000000)

#### 7.1.2.2.1 Detailed Description

#### 7.1.2.2.2 Macro Definition Documentation

##### 7.1.2.2.2.1 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)25000000)
```

Default value of the External oscillator in Hz

##### 7.1.2.2.2.2 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)16000000)
```

Value of the Internal oscillator in Hz

### 7.1.2.3 STM32F4xx_System_Private_TypesDefinitions

### 7.1.2.4 STM32F4xx_System_Private_Defines

### 7.1.2.5 STM32F4xx_System_Private_Macros

### 7.1.2.6 STM32F4xx_System_Private_Variables

**Variables**

- uint32_t SystemCoreClock = 16000000
- const uint8_t AHBPrescTable [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t APBPrescTable [8] = {0, 0, 0, 0, 1, 2, 3, 4}

#### 7.1.2.6.1 Detailed Description

#### 7.1.2.6.2 Variable Documentation

##### 7.1.2.6.2.1 AHBPrescTable

```
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
```

##### 7.1.2.6.2.2 APBPrescTable

```
const uint8_t APBPrescTable[8] = {0, 0, 0, 0, 1, 2, 3, 4}
```

##### 7.1.2.6.2.3 SystemCoreClock

```
uint32_t SystemCoreClock = 16000000
```

#### 7.1.2.7 STM32F4xx_System_Private_FunctionPrototypes

#### 7.1.2.8 STM32F4xx_System_Private_Functions

**Functions**

- void SystemInit (void)

  *Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.*
- void SystemCoreClockUpdate (void)

  *Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.*

##### 7.1.2.8.1 Detailed Description

##### 7.1.2.8.2 Function Documentation

###### 7.1.2.8.2.1 SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
            void  )
```

Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

**Note**

Each time the core clock (HCLK) changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.

- The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

- If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(∗)
- If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(∗∗)
- If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(∗∗) or HSI_VALUE(∗) multiplied/divided by the PLL factors.

(∗) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h file (default value 16 MHz) but the real value may vary depending on the variations in voltage and temperature.

(∗∗) HSE_VALUE is a constant defined in stm32f4xx_hal_conf.h file (its value depends on the application requirements), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

- The result of this function could be not correct when using fractional value for HSE crystal.

**Parameters**

|  |  |
|---|---|
| *None* |  |

**Return values**

|  |  |
|---|---|
| *None* |  |

#### 7.1.2.8.2.2 SystemInit()

```
void SystemInit (
             void  )
```

Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.

**Parameters**

|  |  |
|---|---|
| *None* |  |

**Return values**

|  |  |
|---|---|
| *None* |  |

# Chapter 8

# Data Structure Documentation

## 8.1 MPU6050 Struct Reference

Struct representing a mpu6050 imu sensor.

```
#include <mpu6050.h>
```

### Data Fields

- I2C_HandleTypeDef * hi2c
- HAL_StatusTypeDef status
- uint16_t addr
- uint16_t dt
- int16_t gX
- int16_t gY
- int16_t gZ
- int16_t gX_offset
- int16_t gY_offset
- int16_t gZ_offset

### 8.1.1 Detailed Description

Struct representing a mpu6050 imu sensor.

### 8.1.2 Field Documentation

#### 8.1.2.1 addr

```
uint16_t addr
```

I2C address

#### 8.1.2.2 dt

```
uint16_t dt
```

Delta time

#### 8.1.2.3 gX

```
int16_t gX
```

Gyroscope X axis data

#### 8.1.2.4 gX_offset

```
int16_t gX_offset
```

Gyroscope X axis offset

#### 8.1.2.5 gY

```
int16_t gY
```

Gyroscope Y axis data

#### 8.1.2.6 gY_offset

```
int16_t gY_offset
```

Gyroscope Y axis offset

#### 8.1.2.7 gZ

```
int16_t gZ
```

Gyroscope Z axis data

#### 8.1.2.8 gZ_offset

```
int16_t gZ_offset
```

Gyroscope Z axis offset

#### 8.1.2.9 hi2c

```
I2C_HandleTypeDef* hi2c
```

I2C handle

#### 8.1.2.10 status

```
HAL_StatusTypeDef status
```

HAL status

The documentation for this struct was generated from the following file:

- Ball_Launcher_Controller/Core/Inc/mpu6050.h

# Chapter 9

# File Documentation

## 9.1 Ball_Launcher_Controller/Core/Inc/main.h File Reference

Header for main.c file. This file contains the common defines of the application.

```
#include "stm32f4xx_hal.h"
```

#### Functions

- void Error_Handler (void)

  *This function is executed in case of error occurrence.*

### 9.1.1 Detailed Description

Header for main.c file. This file contains the common defines of the application.

**Attention**

### 9.1.2 Function Documentation

#### 9.1.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

**Return values**

|  |  |
|---|---|
| *None* |  |

## 9.2 Ball_Launcher_Controller/Core/Inc/mpu6050.h File Reference

Header for mpu6050.c file. This file contains the common defines of the application.

```
#include <stdio.h>
#include <stdint.h>
#include "stm32f4xx_hal.h"
```

**Data Structures**

- struct MPU6050

    *Struct representing a mpu6050 imu sensor.*

**Functions**

- uint16_t mpu6050_init (MPU6050 ∗imux, I2C_HandleTypeDef ∗hi2c)

    *Initializes the MPU6050 sensor.*
- void mpu6050_calibrate (MPU6050 ∗imux)

    *Calibrates the MPU6050 sensor.*
- void mpu6050_update (MPU6050 ∗imux)

    *Updates the MPU6050 sensor data.*
- int16_t mpu6050_get_gX (MPU6050 ∗imux)

    *Gets the calibrated X-axis gyroscope data.*
- int16_t mpu6050_get_gY (MPU6050 ∗imux)

    *Gets the calibrated Y-axis gyroscope data.*
- int16_t mpu6050_get_gZ (MPU6050 ∗imux)

    *Gets the calibrated Z-axis gyroscope data.*
- int32_t mpu6050_get_X (MPU6050 ∗imux)

    *Gets the X-axis accelerometer data.*
- int32_t mpu6050_get_Y (MPU6050 ∗imux)

    *Gets the Y-axis accelerometer data.*
- int32_t mpu6050_get_Z (MPU6050 ∗imux)

    *Gets the Z-axis accelerometer data.*

### 9.2.1 Detailed Description

Header for mpu6050.c file. This file contains the common defines of the application.

### 9.2.2 Function Documentation

#### 9.2.2.1 mpu6050_calibrate()

```
void mpu6050_calibrate (
            MPU6050 * imux )
```

Calibrates the MPU6050 sensor.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |

#### 9.2.2.2 mpu6050_get_gX()

```
int16_t mpu6050_get_gX (
            MPU6050 * imux )
```

Gets the calibrated X-axis gyroscope data.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |

**Returns**

int16_t Calibrated X-axis gyroscope data.

#### 9.2.2.3 mpu6050_get_gY()

```
int16_t mpu6050_get_gY (
            MPU6050 * imux )
```

Gets the calibrated Y-axis gyroscope data.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |

**Returns**

int16_t Calibrated Y-axis gyroscope data.

#### 9.2.2.4 mpu6050_get_gZ()

```
int16_t mpu6050_get_gZ (
            MPU6050 * imux )
```

Gets the calibrated Z-axis gyroscope data.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |

**Returns**

int16_t Calibrated Z-axis gyroscope data.

#### 9.2.2.5 mpu6050_get_X()

```
int32_t mpu6050_get_X (
            MPU6050 * imux )
```

Gets the X-axis accelerometer data.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |

**Returns**

int32_t X-axis accelerometer data.

#### 9.2.2.6 mpu6050_get_Y()

```
int32_t mpu6050_get_Y (
            MPU6050 * imux )
```

Gets the Y-axis accelerometer data.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |

**Returns**

int32_t Y-axis accelerometer data.

#### 9.2.2.7 mpu6050_get_Z()

```
int32_t mpu6050_get_Z (
            MPU6050 * imux )
```

Gets the Z-axis accelerometer data.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |

**Returns**

int32_t Z-axis accelerometer data.

#### 9.2.2.8 mpu6050_init()

```
uint16_t mpu6050_init (
            MPU6050 * imux,
            I2C_HandleTypeDef * hi2c )
```

Initializes the MPU6050 sensor.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |
| *hi2c* | Pointer to the I2C handle structure. |

**Returns**

uint16_t Returns 1 if initialization is successful, otherwise 0.

#### 9.2.2.9 mpu6050_update()

```
void mpu6050_update (
            MPU6050 * imux )
```

Updates the MPU6050 sensor data.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |
| *dt* | Time interval in milliseconds. |

## 9.3 Ball_Launcher_Controller/Core/Inc/stm32f4xx_hal_conf.h File Reference

```
#include "stm32f4xx_hal_rcc.h"
#include "stm32f4xx_hal_gpio.h"
#include "stm32f4xx_hal_exti.h"
#include "stm32f4xx_hal_dma.h"
#include "stm32f4xx_hal_cortex.h"
#include "stm32f4xx_hal_flash.h"
#include "stm32f4xx_hal_i2c.h"
#include "stm32f4xx_hal_pwr.h"
#include "stm32f4xx_hal_uart.h"
```

## Macros

- #define HAL_MODULE_ENABLED

    *This is the list of modules to be used in the HAL driver.*
- #define HAL_I2C_MODULE_ENABLED
- #define HAL_UART_MODULE_ENABLED
- #define HAL_GPIO_MODULE_ENABLED
- #define HAL_EXTI_MODULE_ENABLED
- #define HAL_DMA_MODULE_ENABLED
- #define HAL_RCC_MODULE_ENABLED
- #define HAL_FLASH_MODULE_ENABLED
- #define HAL_PWR_MODULE_ENABLED
- #define HAL_CORTEX_MODULE_ENABLED
- #define HSE_VALUE 25000000U

    *Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).*
- #define HSE_STARTUP_TIMEOUT 100U
- #define HSI_VALUE ((uint32_t)16000000U)

    *Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).*
- #define LSI_VALUE 32000U

    *Internal Low Speed oscillator (LSI) value.*
- #define LSE_VALUE 32768U

    *External Low Speed oscillator (LSE) value.*
- #define LSE_STARTUP_TIMEOUT 5000U
- #define EXTERNAL_CLOCK_VALUE 12288000U

    *External clock source for I2S peripheral This value is used by the I2S HAL module to compute the I2S clock source frequency, this source is inserted directly through I2S_CKIN pad.*
- #define VDD_VALUE 3300U

    *This is the HAL system configuration section.*
- #define TICK_INT_PRIORITY 15U
- #define USE_RTOS 0U
- #define PREFETCH_ENABLE 1U
- #define INSTRUCTION_CACHE_ENABLE 1U
- #define DATA_CACHE_ENABLE 1U
- #define USE_HAL_ADC_REGISTER_CALLBACKS 0U /∗ ADC register callback disabled ∗/
- #define USE_HAL_CAN_REGISTER_CALLBACKS 0U /∗ CAN register callback disabled ∗/
- #define USE_HAL_CEC_REGISTER_CALLBACKS 0U /∗ CEC register callback disabled ∗/
- #define USE_HAL_CRYP_REGISTER_CALLBACKS 0U /∗ CRYP register callback disabled ∗/
- #define USE_HAL_DAC_REGISTER_CALLBACKS 0U /∗ DAC register callback disabled ∗/
- #define USE_HAL_DCMI_REGISTER_CALLBACKS 0U /∗ DCMI register callback disabled ∗/
- #define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /∗ DFSDM register callback disabled ∗/
- #define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /∗ DMA2D register callback disabled ∗/
- #define USE_HAL_DSI_REGISTER_CALLBACKS 0U /∗ DSI register callback disabled ∗/
- #define USE_HAL_ETH_REGISTER_CALLBACKS 0U /∗ ETH register callback disabled ∗/
- #define USE_HAL_HASH_REGISTER_CALLBACKS 0U /∗ HASH register callback disabled ∗/
- #define USE_HAL_HCD_REGISTER_CALLBACKS 0U /∗ HCD register callback disabled ∗/
- #define USE_HAL_I2C_REGISTER_CALLBACKS 0U /∗ I2C register callback disabled ∗/
- #define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /∗ FMPI2C register callback disabled ∗/
- #define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /∗ FMPSMBUS register callback disabled ∗/
- #define USE_HAL_I2S_REGISTER_CALLBACKS 0U /∗ I2S register callback disabled ∗/
- #define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /∗ IRDA register callback disabled ∗/
- #define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /∗ LPTIM register callback disabled ∗/
- #define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /∗ LTDC register callback disabled ∗/

- #define USE_HAL_MMC_REGISTER_CALLBACKS 0U /∗ MMC register callback disabled ∗/
- #define USE_HAL_NAND_REGISTER_CALLBACKS 0U /∗ NAND register callback disabled ∗/
- #define USE_HAL_NOR_REGISTER_CALLBACKS 0U /∗ NOR register callback disabled ∗/
- #define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /∗ PCCARD register callback disabled ∗/
- #define USE_HAL_PCD_REGISTER_CALLBACKS 0U /∗ PCD register callback disabled ∗/
- #define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /∗ QSPI register callback disabled ∗/
- #define USE_HAL_RNG_REGISTER_CALLBACKS 0U /∗ RNG register callback disabled ∗/
- #define USE_HAL_RTC_REGISTER_CALLBACKS 0U /∗ RTC register callback disabled ∗/
- #define USE_HAL_SAI_REGISTER_CALLBACKS 0U /∗ SAI register callback disabled ∗/
- #define USE_HAL_SD_REGISTER_CALLBACKS 0U /∗ SD register callback disabled ∗/
- #define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /∗ SMARTCARD register callback disabled ∗/
- #define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /∗ SDRAM register callback disabled ∗/
- #define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /∗ SRAM register callback disabled ∗/
- #define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /∗ SPDIFRX register callback disabled ∗/
- #define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /∗ SMBUS register callback disabled ∗/
- #define USE_HAL_SPI_REGISTER_CALLBACKS 0U /∗ SPI register callback disabled ∗/
- #define USE_HAL_TIM_REGISTER_CALLBACKS 0U /∗ TIM register callback disabled ∗/
- #define USE_HAL_UART_REGISTER_CALLBACKS 0U /∗ UART register callback disabled ∗/
- #define USE_HAL_USART_REGISTER_CALLBACKS 0U /∗ USART register callback disabled ∗/
- #define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /∗ WWDG register callback disabled ∗/
- #define MAC_ADDR0 2U

    *Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.*
- #define MAC_ADDR1 0U
- #define MAC_ADDR2 0U
- #define MAC_ADDR3 0U
- #define MAC_ADDR4 0U
- #define MAC_ADDR5 0U
- #define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /∗ buffer size for receive ∗/
- #define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /∗ buffer size for transmit ∗/
- #define ETH_RXBUFNB 4U /∗ 4 Rx buffers of size ETH_RX_BUF_SIZE ∗/
- #define ETH_TXBUFNB 4U /∗ 4 Tx buffers of size ETH_TX_BUF_SIZE ∗/
- #define DP83848_PHY_ADDRESS
- #define PHY_RESET_DELAY 0x000000FFU
- #define PHY_CONFIG_DELAY 0x00000FFFU
- #define PHY_READ_TO 0x0000FFFFU
- #define PHY_WRITE_TO 0x0000FFFFU
- #define PHY_BCR ((uint16_t)0x0000U)
- #define PHY_BSR ((uint16_t)0x0001U)
- #define PHY_RESET ((uint16_t)0x8000U)
- #define PHY_LOOPBACK ((uint16_t)0x4000U)
- #define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U)
- #define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U)
- #define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U)
- #define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U)
- #define PHY_AUTONEGOTIATION ((uint16_t)0x1000U)
- #define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U)
- #define PHY_POWERDOWN ((uint16_t)0x0800U)
- #define PHY_ISOLATE ((uint16_t)0x0400U)
- #define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U)
- #define PHY_LINKED_STATUS ((uint16_t)0x0004U)
- #define PHY_JABBER_DETECTION ((uint16_t)0x0002U)
- #define PHY_SR ((uint16_t))
- #define PHY_SPEED_STATUS ((uint16_t))
- #define PHY_DUPLEX_STATUS ((uint16_t))
- #define USE_SPI_CRC 0U
- #define assert_param(expr) ((void)0U)

    *Include module's header file.*

## 9.3.1 Macro Definition Documentation

### 9.3.1.1 assert_param

```
#define assert_param(
              expr ) ((void)0U)
```

Include module's header file.

### 9.3.1.2 DATA_CACHE_ENABLE

```
#define DATA_CACHE_ENABLE 1U
```

### 9.3.1.3 DP83848_PHY_ADDRESS

```
#define DP83848_PHY_ADDRESS
```

### 9.3.1.4 ETH_RX_BUF_SIZE

```
#define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for receive */
```

### 9.3.1.5 ETH_RXBUFNB

```
#define ETH_RXBUFNB 4U /* 4 Rx buffers of size ETH_RX_BUF_SIZE */
```

### 9.3.1.6 ETH_TX_BUF_SIZE

```
#define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for transmit */
```

### 9.3.1.7 ETH_TXBUFNB

```
#define ETH_TXBUFNB 4U /* 4 Tx buffers of size ETH_TX_BUF_SIZE */
```

### 9.3.1.8 EXTERNAL_CLOCK_VALUE

```
#define EXTERNAL_CLOCK_VALUE 12288000U
```

External clock source for I2S peripheral This value is used by the I2S HAL module to compute the I2S clock source frequency, this source is inserted directly through I2S_CKIN pad.

Value of the External audio frequency in Hz

#### 9.3.1.9 HAL_CORTEX_MODULE_ENABLED

```
#define HAL_CORTEX_MODULE_ENABLED
```

#### 9.3.1.10 HAL_DMA_MODULE_ENABLED

```
#define HAL_DMA_MODULE_ENABLED
```

#### 9.3.1.11 HAL_EXTI_MODULE_ENABLED

```
#define HAL_EXTI_MODULE_ENABLED
```

#### 9.3.1.12 HAL_FLASH_MODULE_ENABLED

```
#define HAL_FLASH_MODULE_ENABLED
```

#### 9.3.1.13 HAL_GPIO_MODULE_ENABLED

```
#define HAL_GPIO_MODULE_ENABLED
```

#### 9.3.1.14 HAL_I2C_MODULE_ENABLED

```
#define HAL_I2C_MODULE_ENABLED
```

#### 9.3.1.15 HAL_MODULE_ENABLED

```
#define HAL_MODULE_ENABLED
```

This is the list of modules to be used in the HAL driver.

#### 9.3.1.16 HAL_PWR_MODULE_ENABLED

```
#define HAL_PWR_MODULE_ENABLED
```

#### 9.3.1.17 HAL_RCC_MODULE_ENABLED

```
#define HAL_RCC_MODULE_ENABLED
```

#### 9.3.1.18 HAL_UART_MODULE_ENABLED

```
#define HAL_UART_MODULE_ENABLED
```

#### 9.3.1.19 HSE_STARTUP_TIMEOUT

```
#define HSE_STARTUP_TIMEOUT 100U
```

Time out for HSE start up, in ms

#### 9.3.1.20 HSE_VALUE

```
#define HSE_VALUE 25000000U
```

Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).

Value of the External oscillator in Hz

#### 9.3.1.21 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)16000000U)
```

Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).

Value of the Internal oscillator in Hz

#### 9.3.1.22 INSTRUCTION_CACHE_ENABLE

```
#define INSTRUCTION_CACHE_ENABLE 1U
```

#### 9.3.1.23 LSE_STARTUP_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT 5000U
```

Time out for LSE start up, in ms

#### 9.3.1.24 LSE_VALUE

```
#define LSE_VALUE 32768U
```

External Low Speed oscillator (LSE) value.

< Value of the Internal Low Speed oscillator in Hz The real value may vary depending on the variations in voltage and temperature. Value of the External Low Speed oscillator in Hz

#### 9.3.1.25 LSI_VALUE

```
#define LSI_VALUE 32000U
```

Internal Low Speed oscillator (LSI) value.

LSI Typical Value in Hz

#### 9.3.1.26 MAC_ADDR0

```
#define MAC_ADDR0 2U
```

Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.

#### 9.3.1.27 MAC_ADDR1

```
#define MAC_ADDR1 0U
```

#### 9.3.1.28 MAC_ADDR2

```
#define MAC_ADDR2 0U
```

#### 9.3.1.29 MAC_ADDR3

```
#define MAC_ADDR3 0U
```

#### 9.3.1.30 MAC_ADDR4

```
#define MAC_ADDR4 0U
```

#### 9.3.1.31 MAC_ADDR5

```
#define MAC_ADDR5 0U
```

#### 9.3.1.32 PHY_AUTONEGO_COMPLETE

```
#define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U)
```

Auto-Negotiation process completed

#### 9.3.1.33 PHY_AUTONEGOTIATION

```
#define PHY_AUTONEGOTIATION ((uint16_t)0x1000U)
```

Enable auto-negotiation function

#### 9.3.1.34 PHY_BCR

```
#define PHY_BCR ((uint16_t)0x0000U)
```

Transceiver Basic Control Register

#### 9.3.1.35 PHY_BSR

```
#define PHY_BSR ((uint16_t)0x0001U)
```

Transceiver Basic Status Register

#### 9.3.1.36 PHY_CONFIG_DELAY

```
#define PHY_CONFIG_DELAY 0x00000FFFU
```

#### 9.3.1.37 PHY_DUPLEX_STATUS

```
#define PHY_DUPLEX_STATUS ((uint16_t))
```

PHY Duplex mask

#### 9.3.1.38 PHY_FULLDUPLEX_100M

```
#define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U)
```

Set the full-duplex mode at 100 Mb/s

#### 9.3.1.39 PHY_FULLDUPLEX_10M

```
#define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U)
```

Set the full-duplex mode at 10 Mb/s

#### 9.3.1.40 PHY_HALFDUPLEX_100M

```
#define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U)
```

Set the half-duplex mode at 100 Mb/s

#### 9.3.1.41 PHY_HALFDUPLEX_10M

```
#define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U)
```

Set the half-duplex mode at 10 Mb/s

#### 9.3.1.42 PHY_ISOLATE

```
#define PHY_ISOLATE ((uint16_t)0x0400U)
```

Isolate PHY from MII

#### 9.3.1.43 PHY_JABBER_DETECTION

```
#define PHY_JABBER_DETECTION ((uint16_t)0x0002U)
```

Jabber condition detected

#### 9.3.1.44 PHY_LINKED_STATUS

```
#define PHY_LINKED_STATUS ((uint16_t)0x0004U)
```

Valid link established

#### 9.3.1.45 PHY_LOOPBACK

```
#define PHY_LOOPBACK ((uint16_t)0x4000U)
```

Select loop-back mode

#### 9.3.1.46 PHY_POWERDOWN

```
#define PHY_POWERDOWN ((uint16_t)0x0800U)
```

Select the power down mode

#### 9.3.1.47 PHY_READ_TO

```
#define PHY_READ_TO 0x0000FFFFU
```

#### 9.3.1.48 PHY_RESET

```
#define PHY_RESET ((uint16_t)0x8000U)
```

PHY Reset

#### 9.3.1.49 PHY_RESET_DELAY

```
#define PHY_RESET_DELAY 0x000000FFU
```

#### 9.3.1.50 PHY_RESTART_AUTONEGOTIATION

```
#define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U)
```

Restart auto-negotiation function

#### 9.3.1.51 PHY_SPEED_STATUS

```
#define PHY_SPEED_STATUS ((uint16_t))
```

PHY Speed mask

#### 9.3.1.52 PHY_SR

```
#define PHY_SR ((uint16_t))
```

PHY status register Offset

#### 9.3.1.53 PHY_WRITE_TO

```
#define PHY_WRITE_TO 0x0000FFFFU
```

#### 9.3.1.54 PREFETCH_ENABLE

```
#define PREFETCH_ENABLE 1U
```

#### 9.3.1.55 TICK_INT_PRIORITY

```
#define TICK_INT_PRIORITY 15U
```

tick interrupt priority

#### 9.3.1.56 USE_HAL_ADC_REGISTER_CALLBACKS

```
#define USE_HAL_ADC_REGISTER_CALLBACKS 0U /* ADC register callback disabled */
```

#### 9.3.1.57 USE_HAL_CAN_REGISTER_CALLBACKS

```
#define USE_HAL_CAN_REGISTER_CALLBACKS 0U /* CAN register callback disabled */
```

#### 9.3.1.58 USE_HAL_CEC_REGISTER_CALLBACKS

```
#define USE_HAL_CEC_REGISTER_CALLBACKS 0U /* CEC register callback disabled */
```

#### 9.3.1.59 USE_HAL_CRYP_REGISTER_CALLBACKS

```
#define USE_HAL_CRYP_REGISTER_CALLBACKS 0U /* CRYP register callback disabled */
```

#### 9.3.1.60 USE_HAL_DAC_REGISTER_CALLBACKS

```
#define USE_HAL_DAC_REGISTER_CALLBACKS 0U /* DAC register callback disabled */
```

#### 9.3.1.61 USE_HAL_DCMI_REGISTER_CALLBACKS

```
#define USE_HAL_DCMI_REGISTER_CALLBACKS 0U /* DCMI register callback disabled */
```

#### 9.3.1.62 USE_HAL_DFSDM_REGISTER_CALLBACKS

```
#define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /* DFSDM register callback disabled */
```

#### 9.3.1.63 USE_HAL_DMA2D_REGISTER_CALLBACKS

```
#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */
```

#### 9.3.1.64 USE_HAL_DSI_REGISTER_CALLBACKS

```
#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */
```

#### 9.3.1.65 USE_HAL_ETH_REGISTER_CALLBACKS

```
#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */
```

#### 9.3.1.66 USE_HAL_FMPI2C_REGISTER_CALLBACKS

```
#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */
```

#### 9.3.1.67 USE_HAL_FMPSMBUS_REGISTER_CALLBACKS

```
#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */
```

#### 9.3.1.68 USE_HAL_HASH_REGISTER_CALLBACKS

```
#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */
```

#### 9.3.1.69 USE_HAL_HCD_REGISTER_CALLBACKS

```
#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */
```

#### 9.3.1.70 USE_HAL_I2C_REGISTER_CALLBACKS

```
#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */
```

#### 9.3.1.71 USE_HAL_I2S_REGISTER_CALLBACKS

```
#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */
```

#### 9.3.1.72 USE_HAL_IRDA_REGISTER_CALLBACKS

```
#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */
```

#### 9.3.1.73 USE_HAL_LPTIM_REGISTER_CALLBACKS

```
#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */
```

#### 9.3.1.74 USE_HAL_LTDC_REGISTER_CALLBACKS

```
#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */
```

#### 9.3.1.75 USE_HAL_MMC_REGISTER_CALLBACKS

```
#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */
```

#### 9.3.1.76 USE_HAL_NAND_REGISTER_CALLBACKS

```
#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */
```

#### 9.3.1.77 USE_HAL_NOR_REGISTER_CALLBACKS

```
#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */
```

#### 9.3.1.78 USE_HAL_PCCARD_REGISTER_CALLBACKS

```
#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */
```

#### 9.3.1.79 USE_HAL_PCD_REGISTER_CALLBACKS

```
#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */
```

#### 9.3.1.80 USE_HAL_QSPI_REGISTER_CALLBACKS

```
#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */
```

#### 9.3.1.81 USE_HAL_RNG_REGISTER_CALLBACKS

```
#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */
```

#### 9.3.1.82 USE_HAL_RTC_REGISTER_CALLBACKS

```
#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */
```

#### 9.3.1.83 USE_HAL_SAI_REGISTER_CALLBACKS

```
#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */
```

#### 9.3.1.84 USE_HAL_SD_REGISTER_CALLBACKS

```
#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */
```

#### 9.3.1.85 USE_HAL_SDRAM_REGISTER_CALLBACKS

```
#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */
```

#### 9.3.1.86 USE_HAL_SMARTCARD_REGISTER_CALLBACKS

```
#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */
```

#### 9.3.1.87 USE_HAL_SMBUS_REGISTER_CALLBACKS

```
#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */
```

#### 9.3.1.88 USE_HAL_SPDIFRX_REGISTER_CALLBACKS

```
#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */
```

#### 9.3.1.89 USE_HAL_SPI_REGISTER_CALLBACKS

```
#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */
```

#### 9.3.1.90 USE_HAL_SRAM_REGISTER_CALLBACKS

```
#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */
```

#### 9.3.1.91 USE_HAL_TIM_REGISTER_CALLBACKS

```
#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */
```

#### 9.3.1.92 USE_HAL_UART_REGISTER_CALLBACKS

```
#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */
```

#### 9.3.1.93 USE_HAL_USART_REGISTER_CALLBACKS

```
#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */
```

#### 9.3.1.94 USE_HAL_WWDG_REGISTER_CALLBACKS

```
#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */
```

#### 9.3.1.95 USE_RTOS

```
#define USE_RTOS 0U
```

#### 9.3.1.96 USE_SPI_CRC

```
#define USE_SPI_CRC 0U
```

#### 9.3.1.97 VDD_VALUE

```
#define VDD_VALUE 3300U
```

This is the HAL system configuration section.

Value of VDD in mv

## 9.4 Ball_Launcher_Controller/Core/Inc/stm32f4xx_it.h File Reference

This file contains the headers of the interrupt handlers.

### Functions

- void NMI_Handler (void)

  *This function handles Non maskable interrupt.*
- void HardFault_Handler (void)

  *This function handles Hard fault interrupt.*
- void MemManage_Handler (void)

  *This function handles Memory management fault.*
- void BusFault_Handler (void)

  *This function handles Pre-fetch fault, memory access fault.*
- void UsageFault_Handler (void)

  *This function handles Undefined instruction or illegal state.*
- void SVC_Handler (void)

  *This function handles System service call via SWI instruction.*
- void DebugMon_Handler (void)

  *This function handles Debug monitor.*
- void PendSV_Handler (void)

  *This function handles Pendable request for system service.*
- void SysTick_Handler (void)

  *This function handles System tick timer.*

### 9.4.1 Detailed Description

This file contains the headers of the interrupt handlers.

**Attention**

### 9.4.2 Function Documentation

#### 9.4.2.1 BusFault_Handler()

```
void BusFault_Handler (
             void  )
```

This function handles Pre-fetch fault, memory access fault.

#### 9.4.2.2 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

This function handles Debug monitor.

#### 9.4.2.3 HardFault_Handler()

```
void HardFault_Handler (
             void  )
```

This function handles Hard fault interrupt.

#### 9.4.2.4 MemManage_Handler()

```
void MemManage_Handler (
             void  )
```

This function handles Memory management fault.

#### 9.4.2.5 NMI_Handler()

```
void NMI_Handler (
             void  )
```

This function handles Non maskable interrupt.

#### 9.4.2.6 PendSV_Handler()

```
void PendSV_Handler (
             void )
```

This function handles Pendable request for system service.

#### 9.4.2.7 SVC_Handler()

```
void SVC_Handler (
             void )
```

This function handles System service call via SWI instruction.

#### 9.4.2.8 SysTick_Handler()

```
void SysTick_Handler (
             void )
```

This function handles System tick timer.

#### 9.4.2.9 UsageFault_Handler()

```
void UsageFault_Handler (
             void )
```

This function handles Undefined instruction or illegal state.

## 9.5 Ball_Launcher_Controller/Core/Src/main.c File Reference

Main program body.

```
#include "main.h"
#include "mpu6050.h"
#include <stdio.h>
#include <string.h>
```

### Functions

- void SystemClock_Config (void)

  *System Clock Configuration.*
- int main (void)
- void Error_Handler (void)

  *This function is executed in case of error occurrence.*

**Variables**

- I2C_HandleTypeDef hi2c1
- UART_HandleTypeDef huart1
- UART_HandleTypeDef huart2
- int32_t gX = 0
- int32_t gY = 0
- int32_t gZ = 0
- uint16_t stat = 0
- uint16_t shot = 0
- uint16_t move = 0
- char buffer [100]

## 9.5.1 Detailed Description

Main program body.

**Attention**

## 9.5.2 Function Documentation

### 9.5.2.1 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

**Return values**

| | |
|---|---|
| *None* | |

### 9.5.2.2 main()

```
int main (
             void  )
```

### 9.5.2.3 SystemClock_Config()

```
void SystemClock_Config (
             void  )
```

System Clock Configuration.

**Return values**

| | |
|---|---|
| *None* | |

Configure the main internal regulator output voltage

Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.

Initializes the CPU, AHB and APB buses clocks

## 9.5.3 Variable Documentation

### 9.5.3.1 buffer

```
char buffer[100]
```

### 9.5.3.2 gX

```
int32_t gX = 0
```

### 9.5.3.3 gY

```
int32_t gY = 0
```

### 9.5.3.4 gZ

```
int32_t gZ = 0
```

### 9.5.3.5 hi2c1

```
I2C_HandleTypeDef hi2c1
```

### 9.5.3.6 huart1

```
UART_HandleTypeDef huart1
```

### 9.5.3.7 huart2

```
UART_HandleTypeDef huart2
```

### 9.5.3.8 move

```
uint16_t move = 0
```

#### 9.5.3.9 shot

```
uint16_t shot = 0
```

#### 9.5.3.10 stat

```
uint16_t stat = 0
```

## 9.6 Ball_Launcher_Controller/Core/Src/mpu6050.c File Reference

Implementation of mpu6050 sensor functions.

```
#include "mpu6050.h"
```

**Functions**

- uint16_t mpu6050_init (MPU6050 ∗imux, I2C_HandleTypeDef ∗hi2c)

  *Initializes the MPU6050 sensor.*
- void mpu6050_calibrate (MPU6050 ∗imux)

  *Calibrates the MPU6050 sensor.*
- void mpu6050_update (MPU6050 ∗imux)

  *Updates the MPU6050 sensor data.*
- int16_t mpu6050_get_gX (MPU6050 ∗imux)

  *Gets the calibrated X-axis gyroscope data.*
- int16_t mpu6050_get_gY (MPU6050 ∗imux)

  *Gets the calibrated Y-axis gyroscope data.*
- int16_t mpu6050_get_gZ (MPU6050 ∗imux)

  *Gets the calibrated Z-axis gyroscope data.*

### 9.6.1 Detailed Description

Implementation of mpu6050 sensor functions.

Created on: May 3, 2024 Author: vvinh

### 9.6.2 Function Documentation

#### 9.6.2.1 mpu6050_calibrate()

```
void mpu6050_calibrate (
            MPU6050 * imux )
```

Calibrates the MPU6050 sensor.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |

### 9.6.2.2 mpu6050_get_gX()

```
int16_t mpu6050_get_gX (
            MPU6050 * imux )
```

Gets the calibrated X-axis gyroscope data.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |

**Returns**

int16_t Calibrated X-axis gyroscope data.

### 9.6.2.3 mpu6050_get_gY()

```
int16_t mpu6050_get_gY (
            MPU6050 * imux )
```

Gets the calibrated Y-axis gyroscope data.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |

**Returns**

int16_t Calibrated Y-axis gyroscope data.

### 9.6.2.4 mpu6050_get_gZ()

```
int16_t mpu6050_get_gZ (
            MPU6050 * imux )
```

Gets the calibrated Z-axis gyroscope data.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |

**Returns**

int16_t Calibrated Z-axis gyroscope data.

#### 9.6.2.5 mpu6050_init()

```
uint16_t mpu6050_init (
             MPU6050 * imux,
             I2C_HandleTypeDef * hi2c )
```

Initializes the MPU6050 sensor.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |
| *hi2c* | Pointer to the I2C handle structure. |

**Returns**

uint16_t Returns 1 if initialization is successful, otherwise 0.

#### 9.6.2.6 mpu6050_update()

```
void mpu6050_update (
             MPU6050 * imux )
```

Updates the MPU6050 sensor data.

**Parameters**

| | |
|---|---|
| *imux* | Pointer to the MPU6050 structure. |
| *dt* | Time interval in milliseconds. |

## 9.7 Ball_Launcher_Controller/Core/Src/stm32f4xx_hal_msp.c File Reference

This file provides code for the MSP Initialization and de-Initialization codes.

```
#include "main.h"
```

### Functions

- void HAL_MspInit (void)
- void HAL_I2C_MspInit (I2C_HandleTypeDef ∗hi2c)

  *I2C MSP Initialization This function configures the hardware resources used in this example.*

- void HAL_I2C_MspDeInit (I2C_HandleTypeDef ∗hi2c)

  *I2C MSP De-Initialization This function freeze the hardware resources used in this example.*
- void HAL_UART_MspInit (UART_HandleTypeDef ∗huart)

  *UART MSP Initialization This function configures the hardware resources used in this example.*
- void HAL_UART_MspDeInit (UART_HandleTypeDef ∗huart)

  *UART MSP De-Initialization This function freeze the hardware resources used in this example.*

### 9.7.1 Detailed Description

This file provides code for the MSP Initialization and de-Initialization codes.

**Attention**

### 9.7.2 Function Documentation

#### 9.7.2.1 HAL_I2C_MspDeInit()

```
void HAL_I2C_MspDeInit (
             I2C_HandleTypeDef ∗ hi2c )
```

I2C MSP De-Initialization This function freeze the hardware resources used in this example.

**Parameters**

| | |
|---|---|
| *hi2c* | I2C handle pointer |

**Return values**

| | |
|---|---|
| *None* | |

I2C1 GPIO Configuration PB6 ------> I2C1_SCL PB7 ------> I2C1_SDA

#### 9.7.2.2 HAL_I2C_MspInit()

```
void HAL_I2C_MspInit (
             I2C_HandleTypeDef ∗ hi2c )
```

I2C MSP Initialization This function configures the hardware resources used in this example.

**Parameters**

| | |
|---|---|
| *hi2c* | I2C handle pointer |

**Return values**

| | |
|---|---|
| *None* | |

I2C1 GPIO Configuration PB6 ------> I2C1_SCL PB7 ------> I2C1_SDA

#### 9.7.2.3 HAL_MspInit()

```
void HAL_MspInit (
            void  )
```

Initializes the Global MSP.

#### 9.7.2.4 HAL_UART_MspDeInit()

```
void HAL_UART_MspDeInit (
            UART_HandleTypeDef * huart )
```

UART MSP De-Initialization This function freeze the hardware resources used in this example.

**Parameters**

| | |
|---|---|
| *huart* | UART handle pointer |

**Return values**

| | |
|---|---|
| *None* | |

USART1 GPIO Configuration PA9 ------> USART1_TX PA10 ------> USART1_RX

USART2 GPIO Configuration PA2 ------> USART2_TX PA3 ------> USART2_RX

#### 9.7.2.5 HAL_UART_MspInit()

```
void HAL_UART_MspInit (
            UART_HandleTypeDef * huart )
```

UART MSP Initialization This function configures the hardware resources used in this example.

**Parameters**

| | |
|---|---|
| *huart* | UART handle pointer |

**Return values**

| | |
|---|---|
| *None* | |

USART1 GPIO Configuration PA9 ------> USART1_TX PA10 ------> USART1_RX

USART2 GPIO Configuration PA2 ------> USART2_TX PA3 ------> USART2_RX

## 9.8 Ball_Launcher_Controller/Core/Src/stm32f4xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f4xx_it.h"
```

### Functions

- void NMI_Handler (void)

  *This function handles Non maskable interrupt.*
- void HardFault_Handler (void)

  *This function handles Hard fault interrupt.*
- void MemManage_Handler (void)

  *This function handles Memory management fault.*
- void BusFault_Handler (void)

  *This function handles Pre-fetch fault, memory access fault.*
- void UsageFault_Handler (void)

  *This function handles Undefined instruction or illegal state.*
- void SVC_Handler (void)

  *This function handles System service call via SWI instruction.*
- void DebugMon_Handler (void)

  *This function handles Debug monitor.*
- void PendSV_Handler (void)

  *This function handles Pendable request for system service.*
- void SysTick_Handler (void)

  *This function handles System tick timer.*

### 9.8.1 Detailed Description

Interrupt Service Routines.

**Attention**

## 9.8.2 Function Documentation

### 9.8.2.1 BusFault_Handler()

```
void BusFault_Handler (
             void  )
```

This function handles Pre-fetch fault, memory access fault.

### 9.8.2.2 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

This function handles Debug monitor.

### 9.8.2.3 HardFault_Handler()

```
void HardFault_Handler (
             void  )
```

This function handles Hard fault interrupt.

### 9.8.2.4 MemManage_Handler()

```
void MemManage_Handler (
             void  )
```

This function handles Memory management fault.

### 9.8.2.5 NMI_Handler()

```
void NMI_Handler (
             void  )
```

This function handles Non maskable interrupt.

### 9.8.2.6 PendSV_Handler()

```
void PendSV_Handler (
             void  )
```

This function handles Pendable request for system service.

#### 9.8.2.7 SVC_Handler()

```
void SVC_Handler (
             void  )
```

This function handles System service call via SWI instruction.

#### 9.8.2.8 SysTick_Handler()

```
void SysTick_Handler (
             void  )
```

This function handles System tick timer.

#### 9.8.2.9 UsageFault_Handler()

```
void UsageFault_Handler (
             void  )
```

This function handles Undefined instruction or illegal state.

## 9.9 Ball_Launcher_Controller/Core/Src/syscalls.c File Reference

STM32CubeIDE Minimal System calls file.

```
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
```

### Functions

- int __io_putchar (int ch) __attribute__((weak))
- int __io_getchar (void)
- void initialise_monitor_handles ()
- int _getpid (void)
- int _kill (int pid, int sig)
- void _exit (int status)
- __attribute__ ((weak))
- int _close (int file)
- int _fstat (int file, struct stat ∗st)
- int _isatty (int file)
- int _lseek (int file, int ptr, int dir)
- int _open (char ∗path, int flags,...)
- int _wait (int ∗status)
- int _unlink (char ∗name)
- int _times (struct tms ∗buf)
- int _stat (char ∗file, struct stat ∗st)
- int _link (char ∗old, char ∗new)
- int _fork (void)
- int _execve (char ∗name, char ∗∗argv, char ∗∗env)

**Variables**

- char ∗∗ environ = __env

## 9.9.1 Detailed Description

STM32CubeIDE Minimal System calls file.

**Author**

Auto-generated by STM32CubeIDE

```
For more information about which c-functions
need which of these lowlevel functions
please consult the Newlib libc-manual
```

**Attention**

## 9.9.2 Function Documentation

### 9.9.2.1 __attribute__()

```
__attribute__ (
                (weak) )
```

### 9.9.2.2 __io_getchar()

```
int __io_getchar (
                void )  [extern]
```

### 9.9.2.3 __io_putchar()

```
int __io_putchar (
                int ch )  [extern]
```

### 9.9.2.4 _close()

```
int _close (
                int file )
```

#### 9.9.2.5 _execve()

```
int _execve (
            char * name,
            char ** argv,
            char ** env )
```

#### 9.9.2.6 _exit()

```
void _exit (
            int status )
```

#### 9.9.2.7 _fork()

```
int _fork (
            void  )
```

#### 9.9.2.8 _fstat()

```
int _fstat (
            int file,
            struct stat * st )
```

#### 9.9.2.9 _getpid()

```
int _getpid (
            void  )
```

#### 9.9.2.10 _isatty()

```
int _isatty (
            int file )
```

#### 9.9.2.11 _kill()

```
int _kill (
            int pid,
            int sig )
```

#### 9.9.2.12 _link()

```
int _link (
            char * old,
            char * new )
```

#### 9.9.2.13 _lseek()

```
int _lseek (
             int file,
             int ptr,
             int dir )
```

#### 9.9.2.14 _open()

```
int _open (
             char * path,
             int flags,
              ... )
```

#### 9.9.2.15 _stat()

```
int _stat (
             char * file,
             struct stat * st )
```

#### 9.9.2.16 _times()

```
int _times (
             struct tms * buf )
```

#### 9.9.2.17 _unlink()

```
int _unlink (
             char * name )
```

#### 9.9.2.18 _wait()

```
int _wait (
             int * status )
```

#### 9.9.2.19 initialise_monitor_handles()

```
void initialise_monitor_handles ( )
```

## 9.9.3 Variable Documentation

#### 9.9.3.1 environ

```
char** environ = __env
```

## 9.10 Ball_Launcher_Controller/Core/Src/sysmem.c File Reference

STM32CubeIDE System Memory calls file.

```
#include <errno.h>
#include <stdint.h>
```

### Functions

- void ∗ _sbrk (ptrdiff_t incr)

  *_sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library*

### 9.10.1 Detailed Description

STM32CubeIDE System Memory calls file.

**Author**

Generated by STM32CubeIDE

```
For more information about which C functions
need which of these lowlevel functions
please consult the newlib libc manual
```

**Attention**

### 9.10.2 Function Documentation

#### 9.10.2.1 _sbrk()

```
void * _sbrk (
            ptrdiff_t incr )
```

_sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

```
* ############################################################################
* #  .data  #  .bss  #       newlib heap       #          MSP stack          #
* #         #        #                         # Reserved by _Min_Stack_Size #
* ############################################################################
* ^-- RAM start      ^-- _end                             _estack, RAM end --^
*
```

This implementation starts allocating at the '_end' linker symbol The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack The implementation considers '_estack' linker symbol to be RAM end NOTE: If the MSP stack, at any point during execution, grows larger than the reserved size, please increase the '_Min_Stack_Size'.

**Parameters**

| | |
|---|---|
| *incr* | Memory size |

**Returns**

Pointer to allocated memory

## 9.11 Ball_Launcher_Controller/Core/Src/system_stm32f4xx.c File Reference

CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.

```
#include "stm32f4xx.h"
```

### Macros

- #define HSE_VALUE ((uint32_t)25000000)
- #define HSI_VALUE ((uint32_t)16000000)

### Functions

- void SystemInit (void)

  *Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.*
- void SystemCoreClockUpdate (void)

  *Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.*

### Variables

- uint32_t SystemCoreClock = 16000000
- const uint8_t AHBPrescTable [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t APBPrescTable [8] = {0, 0, 0, 0, 1, 2, 3, 4}

### 9.11.1 Detailed Description

CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.

**Author**

MCD Application Team

This file provides two functions and one global variable to be called from user application:

- SystemInit(): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32f4xx.s" file.
- SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
- SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.

**Attention**

## 9.12 pages/hardware.c File Reference

Documentation for Hardware.

### 9.12.1 Detailed Description

Documentation for Hardware.

## 9.13 pages/mainpage.c File Reference

Main Page Documentation for Project.

### 9.13.1 Detailed Description

Main Page Documentation for Project.

## 9.14 pages/software.c File Reference

Documentation for Software.

### 9.14.1 Detailed Description

Documentation for Software.

# Index