Report Air Mouse - Fingers

Group 11

Cirillo Benedetto	0622701741	b.cirillo6@studenti.unisa.it
D'Amato Stefano	0622701723	s.damato16@studenti.unisa.it
Scovotto Luigi	0622701702	l.scovotto1@studenti.unisa.it
Tortora Francesco	0622701700	f.tortora21@studenti.unisa.it

Supervisors: Prof. Francesco Moscato
Prof. Vincenzo Carletti

Università degli Studi di Salerno
DIEM

Salerno, 4th July 2022

Contents

1

Intro

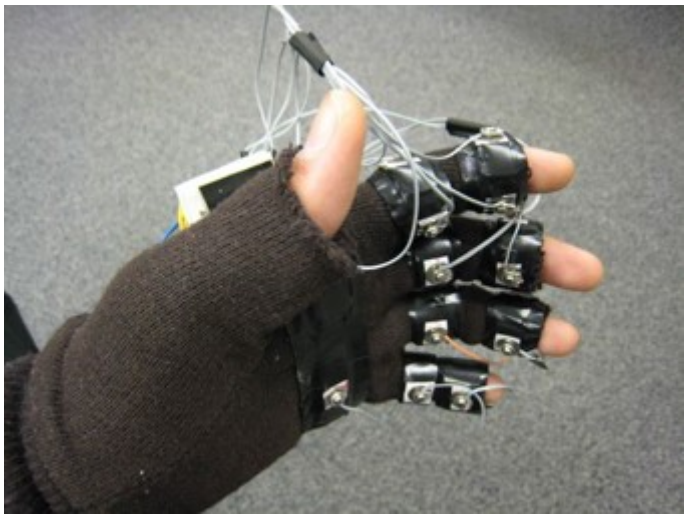
1.1 Purpose of the document

In this report we present our solution for the Air Mouse with fingers application, final project for Embedded System course. We attach a video that shows the functioning of the Air Mouse

(https://drive.google.com/file/d/17unxxmc1Oj8zn8KL4WNMXdvz1g1tJ5tz/view?usp=shar

1.2 Topic

The project consists in the realization of a mouse controlled by the movements of the fingers, through the use of two accelerometers gyroscopes.



1.3 Functioning

Our work was aimed to implement most of the functionalities executed by a simple mouse, through the use of the fingers: each action of the mouse corresponds to a movement or a combination of movements of index and middle fingers of the right hand, on which are mounted two accelerometer gyroscopes.

When starting the system, an automatic calibration phase begins: the two fingers needs to be stationary in a position for a few seconds; this position will be interpreted as the "home landmark", and every time the fingers will be in this position, the pointer stops. When this phase stops the pointer can be moved according to some rules:

- tilt the index to the left → move the pointer to the left;
- tilt the index to the right → move the pointer to the right;
- move the index up → move the pointer up;
- move the index down → move the pointer down.

Furthermore, the following movements correspond to various clicks of the mouse or combinations of the keyboard:

- one rapid movement downward of the middle → left clcik;
- two rapid movements downward of the middle → double left click;
- one rapid movement upward of the middle → right click;
- three rapid movements downward of the middle → zoom in;
- four rapid movements downward of the middle → zoom out;
- a rapid movement of 90° horizontally of both fingers → Alt+Tab.

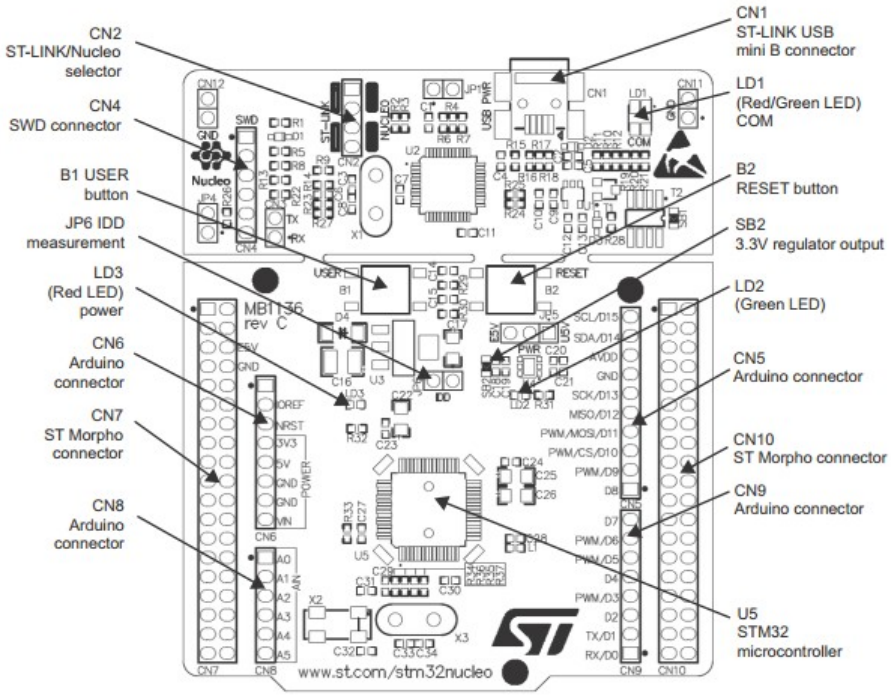
2

Hardware Architecture

2.1 Board

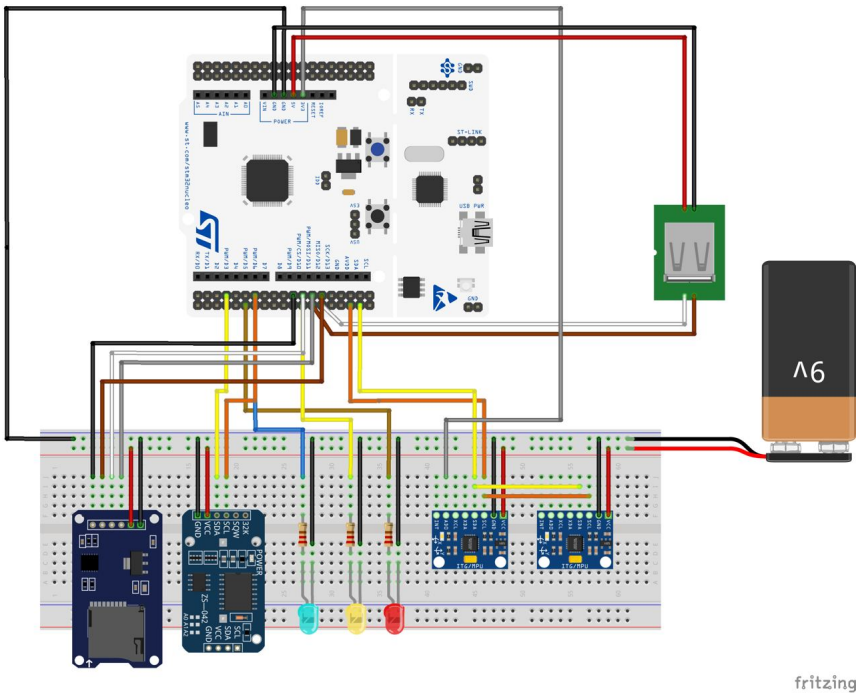
For this project we used the STM32F401RET6 Nuceo64 Board with the following technical characteristics:

- Arm® Cortex®-M4 core at 84MHz;
- 512 Kbytes of Flash memory, 96 Kbytes of SRAM;
- two extension types:
  - Arduino™Uno V3 connectivity;
  - ST morpho extension pin header footprints for full access to all STM32 I/Os;
- embedded ST-LINK/V2-1 debugger/programmer;
- Arm® Mbed Enabled™.

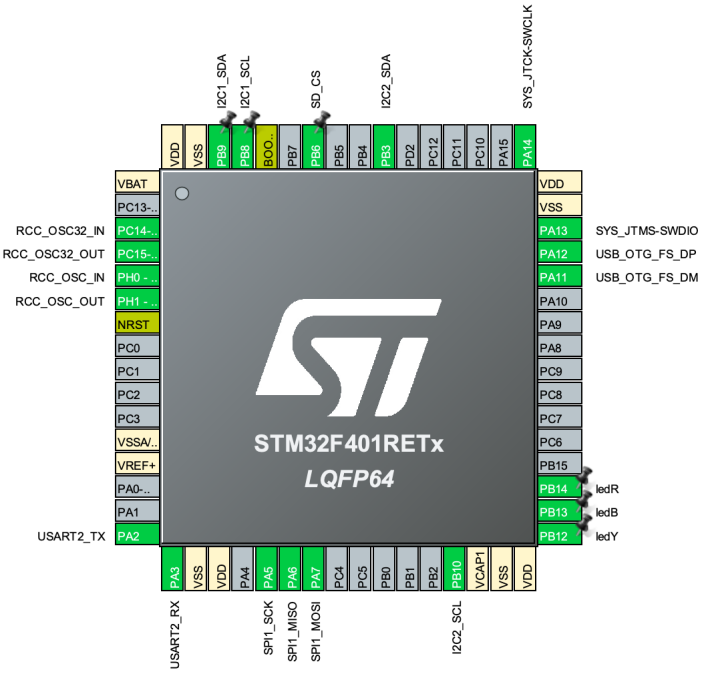


2.2 Electrical scheme

In the following figure is represented a mockup of the electrical scheme realized. In our project the system is powered by a 5V battery (instead of 9V).



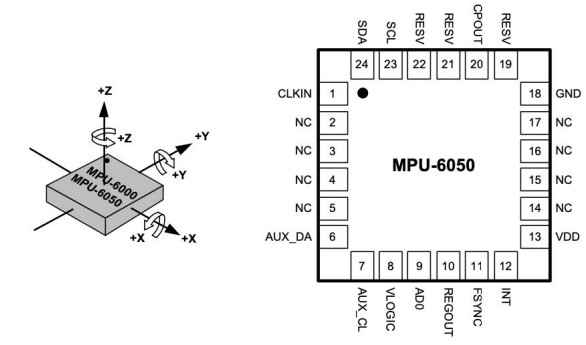
2.3 Pin Table



Port	Pin	Functionality	Description
A	2	USART2_TX	Send serial data
A	3	USART2_RX	Receive serial data
A	5	SPI1_SCK	Clock for the SD card
A	6	SPI1_MISO	Communication with SD
A	7	SPI1_MOSI	Communication with SD
A	11	USB_OTG_FS_DM	HID communication
A	12	USB_OTG_FS_DP	HID communication
A	13	SYS_JTMS-SWDIO	Serial Wire Data I/O
A	14	SYS_JTCK-SWCLK	Serial Wire Clock
B	3	I2C2_SDA	Serial Data transfer for RTC
B	6	SD_CS	Chip select for SD (SPI)
B	8	I2C1_SCL	Serial Clock for for MPUs
B	9	I2C1_SDA	Serial Data transfer for MPUs
B	10	I2C2_SCL	Serial Clock for RTC
B	12	ledY	Yellow led
B	13	ledB	Blue led
B	14	ledR	Red led
C	14	RCC_OSC32_IN	Resonator crystal ceramic input
C	15	RCC_OSC32_OUT	Resonator crystal ceramic output
H	0	RCC_OSC_IN	Resonator crystal ceramic input
H	1	RCC_OSC_OUT	Resonator crystal ceramic output

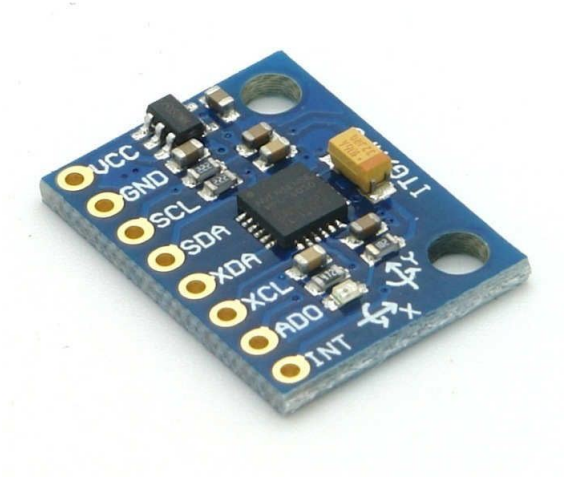
2.4 Description of sensors used

2.4.1 MPU6050



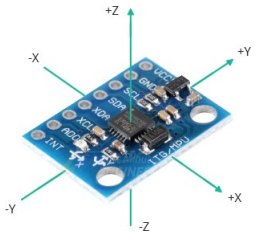
Orientation of Axes of Sensitivity and Polarity of Rotation

QFN Package
24-pin, 4mm x 4mm x 0.9mm

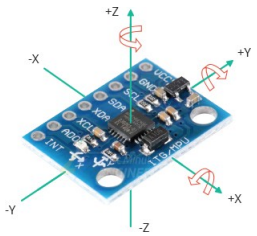


The MPU-6050 is an integrated 6-axis MotionTracking device that combines a 3-axis gyroscope, 3-axis accelerometer. Communication with all registers of the device is performed using I2C. The VDD voltage it operates is 5V, provided by an external power supply.

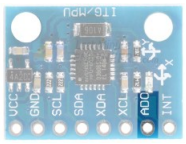
Measuring Acceleration The MPU6050 can measure acceleration using its on-chip accelerometer with four programmable full scale ranges of ±2g, ±4g, ±8g and ±16g.



Measuring Rotation The MPU6050 can measure angular rotation using its on-chip gyroscope with four programmable full scale ranges of ±250°/s, ±500°/s, ±1000°/s and ±2000°/s.



The I2C Interface It supports two separate I2C addresses: 0x68HEX and 0x69HEX.

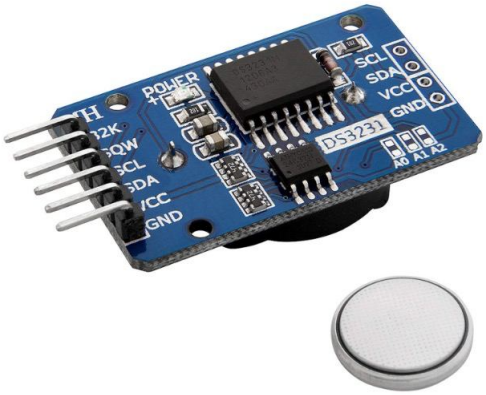


The AD0 pin determines the I2C address of the module. This pin has a built-in 4.7K pull-down resistor. Therefore, when you leave the AD0 pin unconnected, the default I2C address is 0x68HEX and when you connect it to 3.3V, the line is pulled HIGH and the I2C address becomes 0x69HEX.

Application in the project Two MPU-6050 have been used in our project to implement all the functionality of the mouse described above. They are welded on a glove in correspondence of the index and middle fingers. We changed the address of the middle one; this allows the two MPU6050s to be used on the same I2C bus or to avoid address conflicts with another device of the same type on the bus.

2.4.2 DS3231

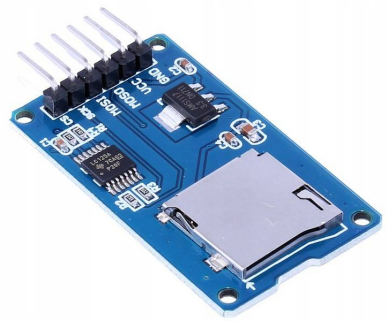
The DS3231 is a low-cost, extremely accurate I2C real-time clock (RTC). The device incorporates a battery input, and maintains accurate timekeeping when main power to the device is interrupted. The integration of the crystal resonator enhances the long-term accuracy of the device.



Application in the project This is used as high speed clock of the board and for getting the current timestamp and print it in the log file.

2.4.3 MicroSD card Adapter

In order to track actions preformed by the Air Mouse, it is necessary to insert a micro SD card module to store all sensors and RTC module data. In particular, information are stored in the "log.txt" file. Because it's a text file, it doesn't get very large, even with continued use of the Air Mouse and therefore a lot of writing.



Application in the project We used a 2GB micro SD. FatFS is the filesystem module library used for open, close and write on the log file. For writing the event log, the SD card used is formatted with the FAT32 file system and the allocation unit size is 4096 bytes. The resulting txt file will contain the relative coordinates provided by the accelerometer and the value of the gyroscope at that particular point in that time, the timestamp of the event just entered and its type. Possible events that can be detected are: movement, left-click, right-click, and double-click, zoom in, zoom out, Alt+Tab.

Example:

```
03-07-22 16:06:27
MOVE TO: (-0.72, 3.31) - accel1: (-0.055,-0.024,-0.060);
   gyro1: (-0.571,0.257,-0.616); accel2:
   (-0.168,0.077,-0.105);gyro2: (-0.599,0.552,-0.382);
03-07-22 16:06:27
ALT-TAB - accel1: (-0.055,-0.024,-0.060);gyro1:
   (-0.571,0.257,-0.616); accel2: (-0.168,0.077,-0.105);
   gyro2: (-0.599,0.552,-0.382);
```

2.5 Description of actuators used

2.5.1 Led

To receive a visual feedback of the events that occur during the execution, there are three led, all on when starting the system:

- blue led → turns off when SD card is detected;
- yellow led → turns off when the accelerometers/gyroscopes are detected;
- red led → turns off when the calibration phase stops.

Furthermore there are some combinations that report errors about the SD:

- • failure in mounting the SD;
- • failure in opening the log file;
- • • failure in writing on the log file.

3

Software Architecture

Our software is developed in C and in the first step we realized the libraries for the communication with each single hardware module, and then the application logic, that involves the definition of the tasks, in the main, according to a bottom-up approach.

3.1 Project configuration

In the following are listed the main changes on the project configuration:

- **clock tree setup:** High Speed Source has been changed to "BYPASS Clock Source" provided by DS3231, with a clock rate set to its maximum value (84MHz).
- **FATFS:** enabled to save logs on a text file. Since FatFs module is the Filesystem Layer independent of platforms and storage media, it is completely separated from the physical devices, such as memory card, hard disk and any type of storage device. The storage device control module is not any part of FatFs module and it needs to be provided by implementer. FatFs controls the storage devices via a simple media access interface shown below. Storage device controls:
  - `disk_status` - Get device status;
  - `disk_initialize` - Initialises the device;
  - `disk_read` - Read data;
  - `disk_write` - Writes data;
  - `disk_ioctl` - Controls device dependent functions.

  FatFs provides various filesystem functions for the applications as shown below.
  - `f_open` - Open Create a file;
  - `f_close` - Close an open file;
  - `f_sync` - Flush cached data;

  - `f_printf` - Write a formatted string.

- **FREERTOS:** CMSIS_V2 has been choosed as OS to allow the execution of two or more concurrent tasks. Actions that cannot be executed at the same time are grouped in the same task as follows:
  - `moveMouse`, handles the movements of the pointer;
  - `rightLeftALt`, handles instant actions, i.e. right click, left click, Alt+Tab;
  - `zoom`, handles zoom in and zoom out;
  - `resetMouse`, resets the mouse to default values;
  - `readAll`, reads data from the two MPU6050.

  They all have real time priority (osPriorityRealTime).

- **USB_OTG_FS AND USB_DEVICE :** to use the mouse directly with the PC, without using drivers or software that convert the messages sent by the Air Mouse, device mode is selected in USB_OTG_FS so that the STM32 is seen by the PC as a device, HID (Human Interface Device) mode is selected in USB_DE so that the PC sees the board as an HID, specifically as a mouse.

- **USART :** USART is activated for debugging the software, in the final release of the project it is therefore not used, but it was of fundamental use during the development phase.

- **I2C :** I2C is activated because there are several devices in the system using this protocol, I2C1 is activated for the two MPU6050 devices, I2C2 for the DS3231.

- **SPI :** SPI is activated to enable the communication with the micro SD reader, which communicates precisely via this protocol, in Full-Duplex Master mode; we also increased the pre-scaler to 128.

3.2 DS3231

This library is used for the communication with the the RTC sensors. It contains a data structure for storing the date and time (seconds - minutes - hours - day

of week - day of month - month - year). It provides the function for:

- updating the current date and time from RTC;
- setting the date and time;
- getting the current temperature;
- getting the current date and time from RTC;
- sending the time in uart.

3.3 MPU-6050

This library is used for the communication with the the gyroscope and accelerometers, i.e. the MPU-6050 devices. It contains a data structure for all the information about a single accelerometer-gyroscope MPU6050 and functions for:

- getting both accelerometers values (`MPU6050_read_accel`);
- getting both gyroscopes values (`MPU6050_read_gyro`);
- getting both gyroscopes and accelerometers value (`MPU6050_read`);
- sending to uart gyroscope and accelerometer values of a single MPU6050 (`MPU6050_to_uart_single`);
- sending to uart gyroscopes and accelerometers values of both MPU6050 (`MPU6050_to_uart`);
- setting the home position for the gyroscopes and accelerometers sensor (`MPU6050_calibration`);
- calculating the accelerometers and gyroscopes values angle on x,y and z axis (`MPU6050_convert`);
- initializing the two MPU6050 devices (`MPU6050_init`);

Trough these function is possible to read their values, with standard accuracy fixed to +/- 2g for the accelerometer values and +/-250 °/s for the gyroscope,

to compute the Pitch and Roll angles allowing the management of the movements of the mouse cursor.

3.4 MicroSD Card Adapter

This library, compared to the others, is more high-level because it exploits function calls of the FATFS 3.1 software module provided by the ST environment, in order to write the various events to the log file on the MicroSD. The main functionalities provided by this library are as follows:

- function that initialize the sd, mounting the volume (`sd_init`);
- function for reading the file "log.txt" from SD (`sd_read`);
- function for writing the file "log.txt" into SD (`sd_write`);
- function for unmounting the volume (`sd_unmount`);

Each time the `sd_write` is called, before writing the SD card in mounted and after the operation it is unmounted. This was done to allow writing on all the four sections of the SD card.

3.5 Dynamic_Queue

This library is used for the creation and the management of a dynamic queue, that is used to memorize the sequence of actions, according to FIFO logic. When the task that writes the actions in the log file is scheduled, the queue is emptied.

3.6 Mouse

This library is used to communicate with the computer trough USB interface so that the STM board is seen as a human interface device (HID) (like Keyboards and mouse). It keeps a data structure containing four fields for:

- `mouse_x` for movements along x axis;

- `mouse_y` for movements along y axis;
- `wheel` for movements of the wheel;
- `button` that assumes a different value for each type of click (right, left, middle or a combination of these).

This module provides the necessary functions for:

- initializing/resetting the mouse (`mouse_reset`);
- resetting the click button (`mouse_reset_btn`);
- doing the left click (`mouse_left_click`);
- doing the right click (`mouse_right_click`);
- making the movement of the mouse (`mouse_move`);
- scrolling on a page (`mouse_scroll`);
- sending the middle click (`mouse_macro1`);
- sending the middle and right click simultaneously (`mouse_macro2`);
- sending the middle and left click simultaneously (`mouse_macro3`);

All these functions use a function provided by the USBD_HID software module. The `USBD_HID_SendReport` function is used to send HID reports from the device to the computer.

3.7 Main

It contains the application logic of the whole program. It initializes all the components, and after the calibration phase described before it turns off the three leds. This module contains also the entry function used by the tasks, and the instantiation of each task in new threads.

4

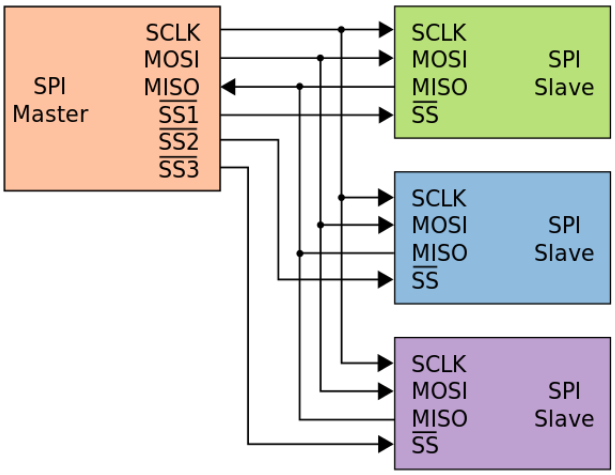
Software Interface

The interfaces used are as follows:

- SPI;
- I2C;
- UART;
- USB;

4.1 SPI

It is a standard communication bus where transmission takes place between a device called a master and one or more slaves. The master controls the bus, issues the clock signal, and decides when to start and stop communication.



The slave device that exploits this interface is the MicroSD reader.

For SPI, the configuration on the Board is the default configuration:

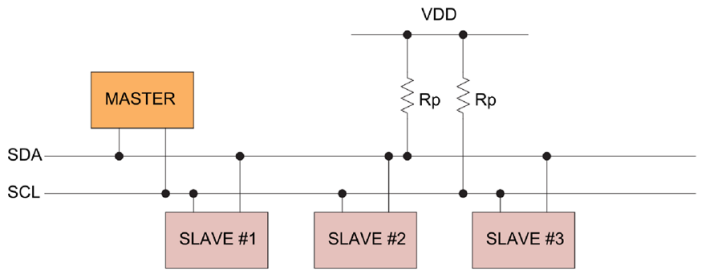
- Mode: Full-Duplex Master
- Frame Format: Motorola
- First Bit: MSB First
- Clock Prescaler: 128

The SPI bus specifies four logic signals:

- SCLK: Serial Clock (output from master)
- MOSI: Master Out Slave In (data output from master)
- MISO: Master In Slave Out (data output from slave)
- CS /SS: Chip/Slave Select (often active low, output from master to indicate that data is being sent)

4.2 I2C

The Inter-Integrated Circuit (I2C) interface is a full-duplex synchronous protocol that is designed to support a very large number of devices compared to SPI. On I2C there is a single Open-Collector BUS on which data travels, which allows many heterogeneous devices to communicate and gives greater flexibility on their interfacing (up to 1008 slaves are supported). The communication logic is masterslave, and it is also possible for there to be multiple masters communicating on the shared line, so competition between them must be managed.



We used two I2C interfaces (I2C1 and I2C2) because the MPU-6050 sensors have a different memory address (section 2.4.1), 0x68 and 0x69 respectively, while the DS3231 device shares the memory address, 0x68, with the first MPU-6050 device, so for this reason:

- the MPU-6050 sensors were connected on I2C1
- the DS3231 device on I2C2

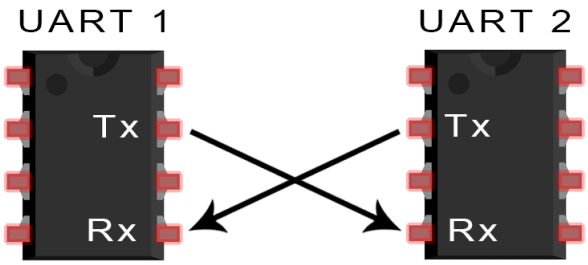
For both I2C, the configuration on the Board is the default configuration:

- Speed Mode: Standard

- Clock Speed: 100000 Hz
- Primary Address Length: 7 bits
- Primary Slave Address: 0

4.3 UART

The Universal Asynchronous Receiver-Transmitter (UART) interface transmits serial data asynchronously using two input-output lines (one for transmit and one for receive), with no shared clock between the two communicating devices. The goal of the UART interface is to perform a translation of bit streams from a parallel format to a serial one and vice versa.

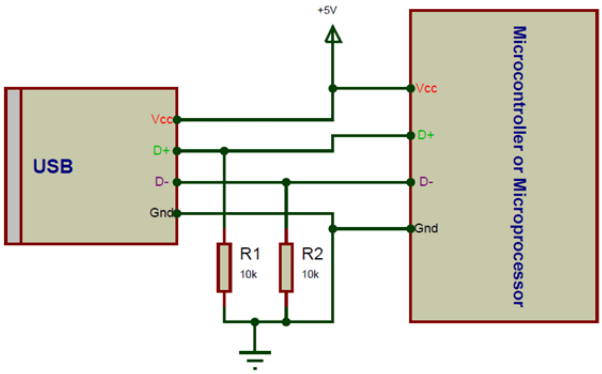


The configuration on the Board is the default configuration:

- Baud Rate: 9600 bits
- Word Length: 8 bit
- Stop Bit: 1
- Data Direction: Receive and Transmit
- Oversampling: 16 samples
- UART Global Interrupt: Enable

4.4 USB

The Universal Serial Bus (USB) is an interface used to connect peripheral devices.



The configuration on the Board is:

- Mode: Device Only
- Full Speed: 12 Mbits

The mode "Device Only" is used to enable USB_DEVICE mode so that the computer sees the board not as a host but as an external device. For this purpose, we have configured the usb device class as Human Interface Device Class (HID).

4.5 Supporting Software - HydraMouse

For some actions we would perform on the Air Mouse, i.e. zoom in, zoom out and Alt+Tab, there aren't the corresponding button values in the library we used, as explained (here section3.6).

So we used a software, HydraMouse, in order to interprete some buttons values provided by the library as these commands. In particular:

- middle click → Alt+Tab;
- middle click and right click → 'Ctrl' + '+' , that is 'zoom in' in some applications;
- middle click and left click → 'Ctrl' + '-' , that is 'zoom out' in some applications.

5

Software Protocol

In this chapter will be shown the functioning of each protocol we designed to communicate with the hardware modules.

5.1 MPU6050

For the initialization of MPU6050s, the `MPU6050_init` function is used. With this function, it is checked whether the devices are ready, the sampling rate and the accuracy of the accelerometers and gyroscopes is set. In addition, the `MPU6050_calibration` function is also called up, which takes care of determining the accelerometer and gyroscope values when the hand is stationary, in order to find out what the home landmark is. The function `MPU6050_read` is used to read the values from the devices. This function uses `MPU6050_read_accel` and `MPU6050_read_gyro`, which read the memory sector of the device related to the desired parameter and axis. Finally, there is the `MPU6050_convert` function which is used to calculate roll, pitch and yaw.

5.2 SD

The sd library has as its main functionality reading and writing to sd. The `sd_write` function can be used to write text to the log file (log.txt). This function calls the `sd_init` function which takes care of the volume mount, opens and writes the file and finally calls the `sd_unmount` function which takes care of the de-mount. The `sd_read` function is used to read the file.

5.3 DS3231

The library for the DS3231 clock firstly provides two functions for converting from binary to decimal and vice versa. The function `set_time` is used to set the date and time within the clock. The `get_time` function is used to take the date and time from the clock and save the values in the struct. Once the struct has been updated, the `time_to_string` function is used to save the current date and time value in the string passed by reference. The `time_to_uart` function is used to pass the current date and time into uart. In addition, `get_temp` and `force_temp_conv` are used to take the temperature and perform the conversion.