UNIVERSITY OF CAPE TOWN

EEE3088F

SOFTWARE AND DOCUMENTATION ASSIGNMENT

MAY, 2022

GROUP 18

JONATHAN APPS (APPJON001)

SHAMEERA CASSIM (CSSSHA020)

JAHAR PERSAD (PRSJAH002)

SECTION 1: SOFTWARE

1.1. Code Structure

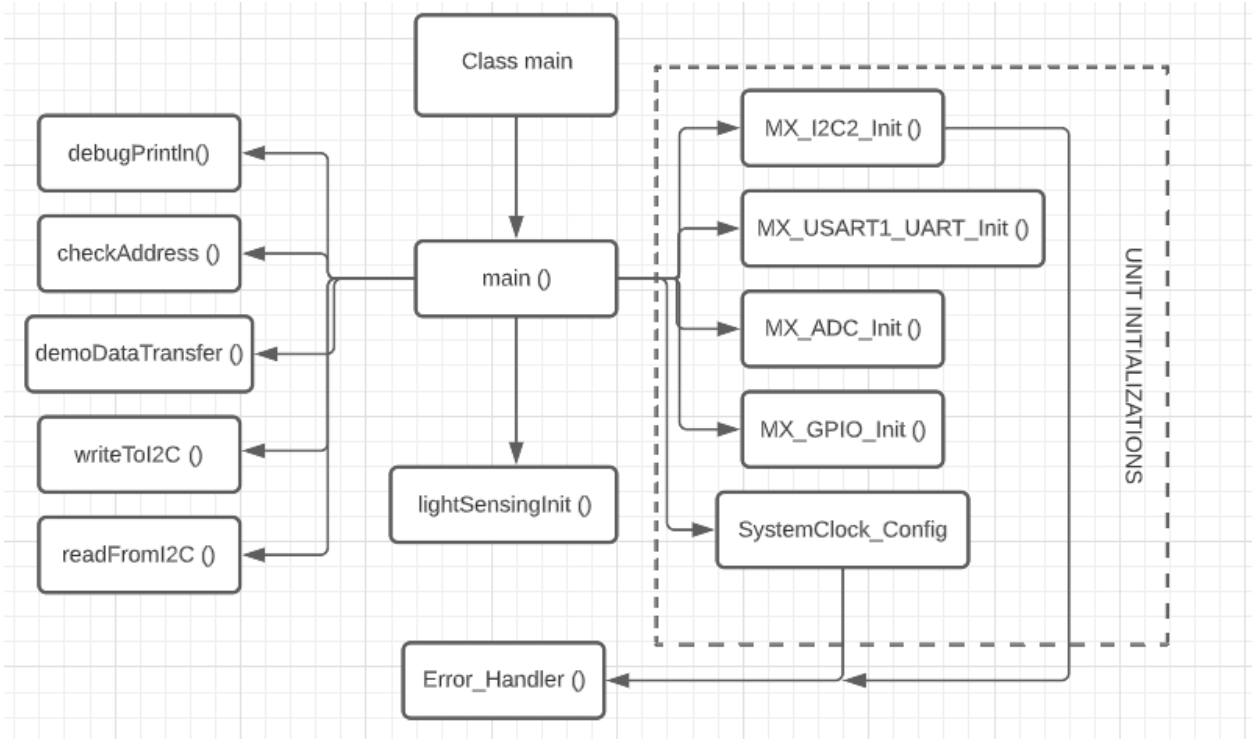

Figure 1: Code Basic Structure

The basic structure of the code for the HAT is outlined in the block diagram above. A simple set of functions initialize the different modules needed in the code (labeled “unit initializations” in the diagram above). The main method calls the functions when necessary, as it runs the main logic of the program. The lightSensinginit() function is a one that can be run to display the light sensing functionality of the LDRs on board the HAT. All the other functions deal with data transfer from the HAT to the STM and vice versa.

1.2. Code Reference

Link: https://gitlab.com/g5168/eee3088f-group-18-project/-/blob/main/README.md

---

Group 18 Code Reference Document

The following functions are used in the firmware for Group 18's HAT. Please note that the bulk of this code is taken from the Coding Help slides by Jane Wyngaard [1]:

Function name	**debugPrintln**
Function description	General purpose Function to send a char array over the UART and to automatically send a new line character after it
Parameters	**uart_handle**: pointer to a UART_HandleTypeDef structure that contains the configuration information for the UART being used **_out[ ]** : array of characters to be sent
Return values	None
Function name	**demoDataTransfer**
Function description	Demonstration of sending data over UART to laptop. "Hello, this is STMF0 Discovery board" should be printed to the laptop and the blue LED should be flashed once this is completed
Parameters	None
Return values	None
Function name	**checkAddress**
Function description	This function checks that the correct address has been chosen
Parameters	**huart**: pointer to a UART_HandleTypeDef structure that contains the configuration information for the UART being used All other parameters are used to call the HAL_I2C_IsDeviceReady function. Thus, the following is copied from the HAL reference sheet[2]: **hi2c**: Pointer to a I2C_HandleTypeDef structure that contains the configuration information for the specified I2C. **DevAddress**: Target device address: The device 7 bits address value in datasheet must be shift at right before call interface **Trials**: Number of trials **Timeout**: Timeout duration
Return values	None
Function name	**writeToI2C**

Function description	This function is used to read data from the EEPROM
Parameters	**EEPROM_DEVICE_ADDR**: pointer to the address of the EEPROM device on the I2C bus **madd**: starting memory address for the location of memory to read from **Data**: starting value of data to write to memory
Return values	None
Function name	**readToI2C**
Function description	This function is used to write data to the EEPROM
Parameters	**EEPROM_DEVICE_ADDR**: pointer to the address of the EEPROM device on the I2C bus **madd**: starting memory address for the location of memory to read from **Result**: address of stored value to read back from memory in
Return values	None
Function name	**lightSensingInit**
Function description	Blink 2 LEDs based on the output of the LDRs from the HAT
Parameters	None
Return values	None

[1] J. Wyngaard, *Coding Help*. 2022, pp. 1-16.

[2] STMicroelectronics, *Description of STM32F0 HAL and low-layer drivers*. STMicroelectronics, 2022, pp. 217-218.

SECTION 2: LEARNING DOCUMENTATION

2.1. PCB Getting Started README

Link: https://gitlab.com/g5168/eee3088f-group-18-project/-/blob/main/README.md

EEE3088F Group 18 Sensing HAT (Version 1.0)

This project is a group project for the course EEE3088F at UCT. The purpose of the HAT is to detect light and move a motor or another device in the direction of the light.

Instructions For Use (For Version 1.0)

- Hardware Requirements -

To use this HAT, a mini-USB connector is required to interface the HAT with the STM (Or whatever other microcontroller is in use).The USB must be capable of data transmission. That being said, an STM32F0 discovery board, or another board containing a microcontroller with read/write/ADC/memory capabilities must be used if the full functionality of the HAT is to be claimed. The USB must be capable of data transmission.

A 5V Li-Ion battery and battery holder are necessary for the powering of the HAT, unless some other power source is to be used. Jumper cables are necessary for the current design, as it does not conform to the general standard of board measurements (the wires are needed to connect the HAT to the board in use).

- Software Requirements -

The coding of this board can be done via the STM Cube IDE (preferred) or Code Blocks. Other coding platforms can work as well. The terminal must be configured to drive the USB ports that connect to the discovery board or the HAT.

Programs can be uploaded to the STM or to the HAT itself, depending on which method is preferred.

The EEPROM on the HAT must be coded via the HAT's USB port, as this sends data to and from the FTDI and the EEPROM on board the HAT together.

Currently the HAT has not been revised, and secondary versions have not yet been created. Because of this, the recommended approach to coding is to do so via the STM or the board in use (and not via the HAT).

The temperature sensor which sends data to and from the HAT's EEPROM was not populated in version 1.0, and as such the EEPROM and the FTDI on board the HAT are not needed or useful currently.

Thus, any code that is written should be uploaded directly to the board in use, after which wiring can interface the board with the HAT.

- Initialization Instructions -

To initialize the hardware, connect the battery to the HAT to power the STM board (if batteries are not being used, find another way to connect the HAT to power). Connect each pin on the STM to the corresponding pin on the HAT via jumper wires.

To initialize the software, connect to the board via a USB, and compile and upload the desired code to the board via an appropriate IDE. How to upload code to the board will vary from IDE to IDE. Check the IDE's online documentation for further details.

- Running Your First Program on the HAT -

To run your first program on the HAT, refer to the self-explanatory code in the "main.c" file in the following directory:

"Firmware/GROUP18 Microcontroller Interfacing/Core/src/main.c"

This file can be run to toggle two user connected LEDs based on the strength of light sensed by the LDRs on the HAT. To set up the hardware for this basic program, two LEDs should be connected in series with 1k resistors which will be connected to GPIOA and GPIOB on the STM (GPIO pin 10 and pin 12 respectively).

Alternatively, the LEDs on the STM can be flashed based on the state of the LDRs if the STM LEDs are set to "HIGH" in the lightSensingInit() function (this would need to be added). Inside of the while loop in the code, the first statement is a call to initialize the light sensing sequence, and so everything is set up for ease of use.

Have fun!!

2.2. CONTRIBUTING How-To Guide

Contribute To Group 18's HAT

DESIGNERS:

Shameera Cassim (CSSSHA020@myuct.ac.za)

Jonathan Apps (APPJON001@myuct.ac.za)

Jahar Persad (PRSJAH002@myuct.ac.za)

All contributions to this project are welcome, however, the guide below must be read and adhered to.

- How To Contribute

  First, email one of the designers (Shameera Cassim, Jonathan Apps or Jahar Persad), detailing what is wished to change/contribute. Once approval has been received from one of the designers, submit a pull request. A branch will be created in GitLab where contributions and changes can be worked on,once the changes/contributions are complete, submit a merge request and the contributions will be evaluated and either used or discarded.

- Finding Issues (Repository, Documentation, PCB)

  If issues are found in the GitLab repository, be it to do with the PCB, Documentation or even just the way the repository is structured, please raise an issue on GitLab or email one of the designers so that they can take a look at the problem.

- Extending Repository, Contributing to the PCB/Code

  When contributing to the repository, please ensure that all contributions are in well-written English, that is in a well set out document format.

  When contributing to the PCB, onlyKiCAD software may be used, alongside KiCADs standard library. If there is a footprint or anything that is missing from the KiCAD library, please seek approval by emailing one of the designers, detailing what you would like to use and from which library you will use it from.

  When contributing to the code please ensure that it is well commented and follows the same format that the original code was written in. The code compiler/editer that must be used is STM32CubeIDE.

- Lastly, ensure that all changes are well documented and easy to understand.

The GitLab repository link:
https://gitlab.com/g5168/eee3088f-group-18-project/-/blob/main/Documentation/CONTRIBUTING_HOW-TO_GUIDE.pdf