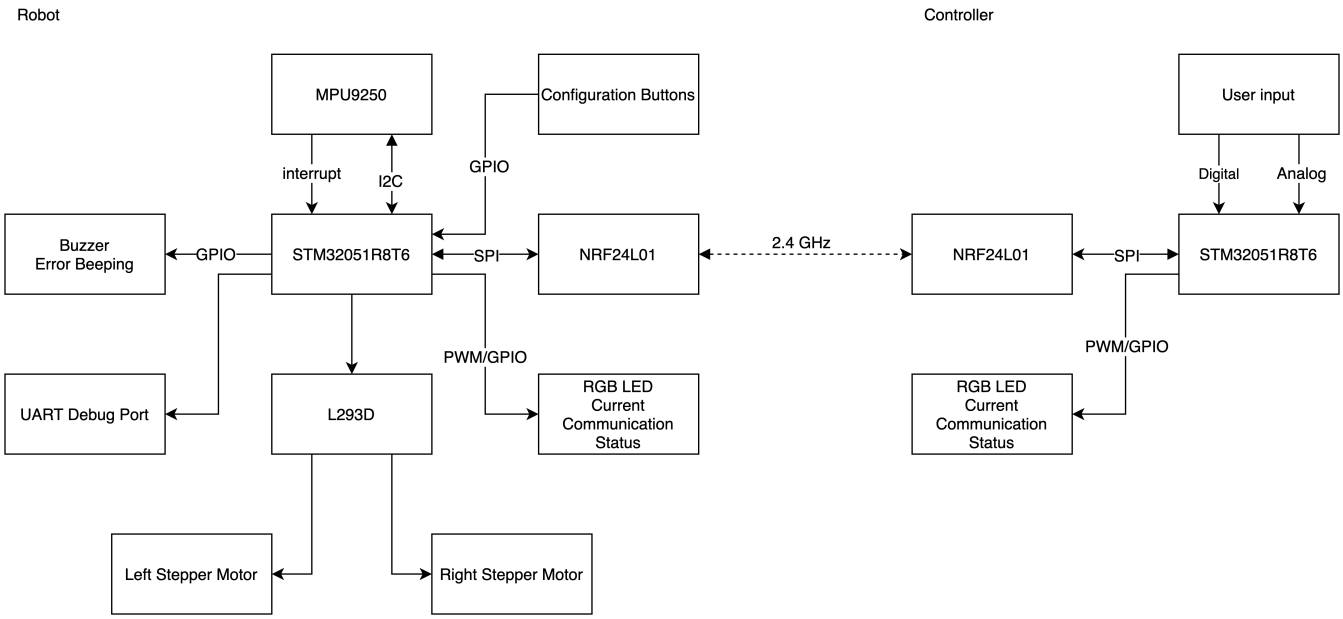
Adjustic

This is the git repository for the Spring 2020 ECE 362 mini project.

A self-balanced two wheel robot using:

- STM32F051R8T6
- MPU9250 Nine-Axis acclerator, gyroscope, and compass
- Stepper Motor driver
  - L293D Four channel motor driver
  - or A4988 Stepper motor driver
- Stepper motors
  - 28BYJ-48 5V 4 Phase DC Gear Stepper motor 15 N·cm
- NRF24L01 2.4GHz Transceiver

Project Architecture



Objective

1. ☐ Two wheel self-balanced within +3/-3 deg
   1. ☐ PID controller
   2. ☐ Angle/angular speed sensor interface
   3. ☐ Driving chassis interface
      1. ☐ Generic stepper motor interface
      2. ☐ L293D interface
   4. ☐ Logging library
      1. Logging level
         - `NONE`: Nothing gets logged
         - `ERROR`: Only errors get logged
         - `WARNING`: Errors and warning get logged
         - `DEBUG`: More debugging related messages get logged

            - `INFO`: Log everything
        2. ☐ Error message and signal interface (buzzer, Leds)
        3. ☐ UART serial debug message output
            1. Only included in debug version of code
2. ☐ Remote control via another STM32051R8T6 along with NRF24L01
    1. ☐ Transimitter/Receiver generic communication interface
        1. wrapper for various kind of SPI transimission device
    2. ☐ Communication packet struct definition
    3. ☐ NRF24L01 interface
    4. ☐ User input panel
        1. ☐ Digital button input
        2. ☐ Analog joystick input

Materials to Learn

- DMA
    - For automatically loading data from sensors and peripherals to memory
    - Textbook chapter 19
- I2C Protocol
    - For communication with acclerator and gyroscope (MPU 9250)
    - Textbook chapter 22.2
- Stepper Motor control
    - Textbook chapter 16
- PID Controller
    - wiki page
- SPI Protocol
    - For wireless controller
    - Textbook chapter 22.3
- git operation

Collaboration Process Explanation

Terminology

1. `dev` branch: hosts new features to be added to the project
2. `master` branch: stable software release
3. `pull request`: change by other collaborator to the branch
    1. ref
4. Collaborator: developers working in this project

Development process

1. All developing work will be conducted on `dev` branch
    1. Modules/Features developing should be kept as local branches prior to integrate them to `dev` branch
    2. Collaborators are expected to **test their modules/features** prior to submit a pull-request
2. After **throughout** testing
    1. Collaborator can submit a pull-request to the `master` branch

2. Other personnel should examine the submitted code throughly prior to approve the pull-request (code review)

Coding Standard

Below are the coding standards for this project. Pull-request not following these will be rejected and advised to change.

General coding

- Naming
  - Variables and function naming must follow camelCase naming format
    - `void getChar(int tmp);`
    - `int isError = 0;`
  - Constants need be capitalized and connected via underline _
    - e.g. `MAX_BUFF_SIZE`
  - Variable or constant naming must be meaningful
    - **NO** single letter naming unless as temporary variable in loop
      - ❌ `int a = 0;`
      - ✅ `for (int i = 0; i < 10; i++);`
    - Naming variable as `tmp` are allowed if the variable serves only as
      - place to temperarily hold exchanged value like in swapping two integer
      - value to be discarded / not used in current naming scope
    - Preferably variable with the same or similar purpose should have close name
      - Example

        ```
        int sensorXAccel;
        int sensorYAccel;
        ```

      - However, consider the following two questions prior to create this type of variables
        - Can I use array to represent them?
        - Can I define a struct to hold them?
- Branching
  - Space follow branching keywords (`if`, `while`, or `for`, etc.)
  - Left curly parantheses on the same line of the branching keywordds
  - Example

    ```
    if (flag == 0) {
      // do something
    }
    ```

- Miscellaneous

    - Space around every equal sign or comparison symbol
        - `int i = 1;` rather than `int i=0;`
        - `if (a > 0)` rather than `if (a>0)`

Debug, Release, and Logging

- Debug configuration code block
    - In eclipse, the IDE will pass a c `#define DEBUG` directive in debug configuration when build for debug and not for release
    - Therefore, it is wise to use this to limit some of the logging code to debug mode only using the following structure

```
// Debug block
#ifdef

// Code only gets to source code in debug build setting

#else

// Optional branch which will get in to source code in release
build setting

#endif
```

- Function implementation
    - Use logging library `assert` to validate inputs
    - and output error message (should be handled by logging library automatically)

Comment specification

An eclipse snippet template has been exported and can be downloaded here. You will need to import it to your eclipse workbench to use the keywords to invoke them.

Header comment (keyword: `header`)

```
/*************************************************************
*
* @file:      FILE_NAME
* @author:    YOUR_NAME
* @email:     YOUR_EMAIL
* @version:   VERISON_NUMBER (e.g. v1.0.0)
* @date:      DATE
* @brief:     Short explanation of the file
*
*************************************************************/
```

Function docstring (keyword: `docstring`)

```
/*************************************************************
*
* Function: FUNCTION_NAME
* @brief:   what does the fuction do
* @param:   function parameter name, type, and short explanation
* @return:  function return name, type, and short explanation
*
*************************************************************/
```

Block comment (keywordd: `comment`)

```
/*************************************************************
* COMMENTS
*************************************************************/
```

TODO

- [x] Project proposal
- [ ] Robot component and subsystem architecture design
    - Protocols
    - Interrupts selection
    - Communication Packet design
        - specify the distance the robot need to move
        - or the speed
        - speed should have higher priority
        - UART command should have higher priority then SPI
    - Library design / Program Diagram
        - Stepper control library
            - Generic?
                - for other chip as well
            - angular speed control
            - angle control
            - use configuration struct to parameterize the function
            - maintain a global config struct for the functions to use?
        - Generical wrapper sensor library
            - general interface for other accelerator, gyroscope, or compass
        - MPU 9250 interface Library
    - Detailed Program Flowchart/outline
    - Rough mechanical design
        - Rough specification for motors and wheel
        - Basic structural design for robot
- [ ] Develop schedule and task assignment