Lab 5 Pre-lab

Below are the key terms that should be useful for this lab and many others.

Key Terms

MISO - Master in, Slave Out. Data line from the Gyro
MOSI - Master Out, Slave in. Data line to our Gyro
SCK - Serial Clock. The clock used to clock the data on the appropriate lines (MISO and MOSi)
CS - Chip Select. Typically, this is an ACTIVE low signal that “enables” the communication between the Processor and the Gyro

HAL - Hardware Abstraction Layer. These are usually predefined functions created by the microcontroller manufacturer, allowing you to configure and interact with the hardware and peripherals.

Tips and Tricks to consider when interfacing with the HAL and Gyro

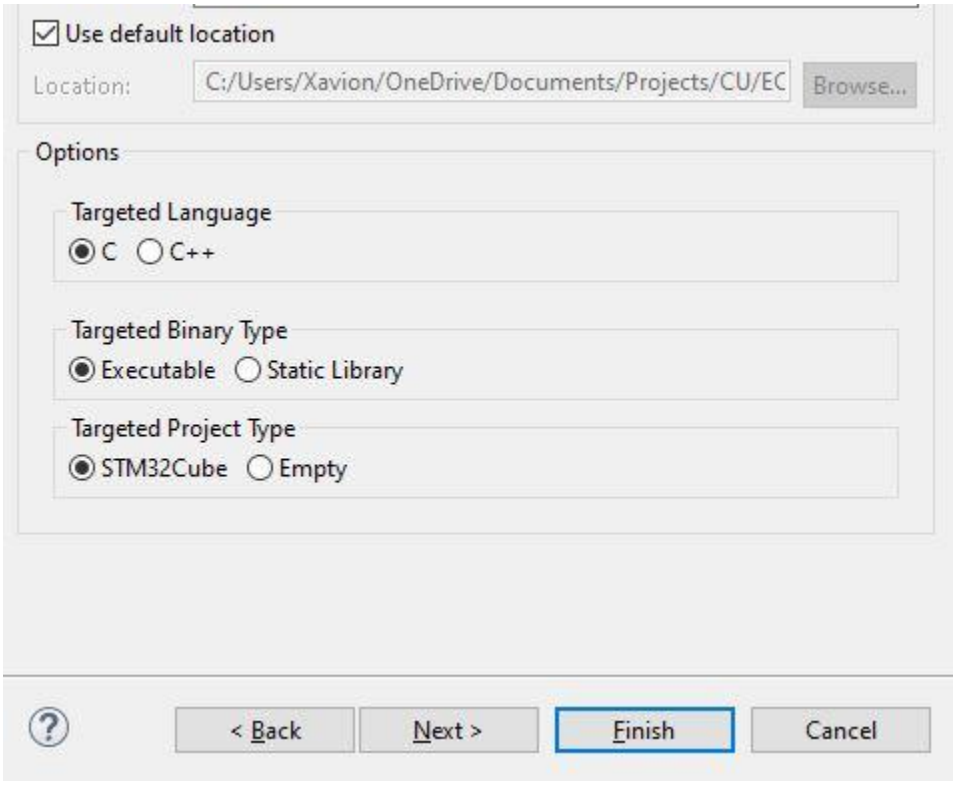
- Be prepared to navigate between multiple files to determine which function does what
- Verify you understand what datatypes are and how to find where they are defined
- Casting will be needed when calling some HAL functions
- When receiving data, you may need to do some arithmetic to read the “correct” data
    - This may be a simple shift due to how the data is transferred from the MISO line to the receive buffer
- Remember, you need to pull the chip select down when trying to communicate with the Gyro

Interacting with the HAL example to send command and receive some data

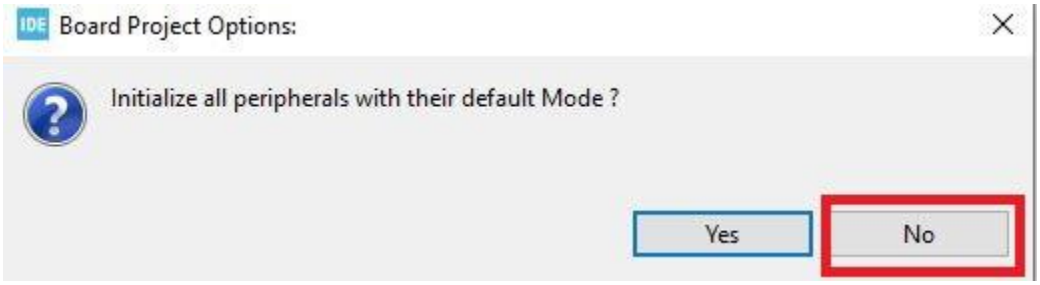
```
void someFunction()
{
    uint8_t commandToSend =  (I3G4250D_READ | I3G4250D_REG_TO_BE_READ);
    uint16_t receivedData = 0x00;

    FunctionResponsibleForEnablingCommunication();
    while(HAL_GPIO_ReadPin(CS_PORT,CS_PIN) != GPIO_PIN_RESET ); // Wait for the communication line to
be low.
    gyroHALStatus = HAL_SPI_TransmitReceive(&hspi5, &commandToSend,(uint8_t*) &receivedData, 2,
TESTING_TIMEOUT);

    if(gyroHALStatus != HAL_OK)
    {
         for(;;); // Infinite Loop to catch errors
    }

    DataReturned = (0xFF00 & receivedData) >> 8;  // Shift the data to get the bytes we want
    printf("The DataReturned is: %d \n", DataReturned);
    FunctionResponsibleForDisablingCommunication();
}
```

How to generate projects with HAL code

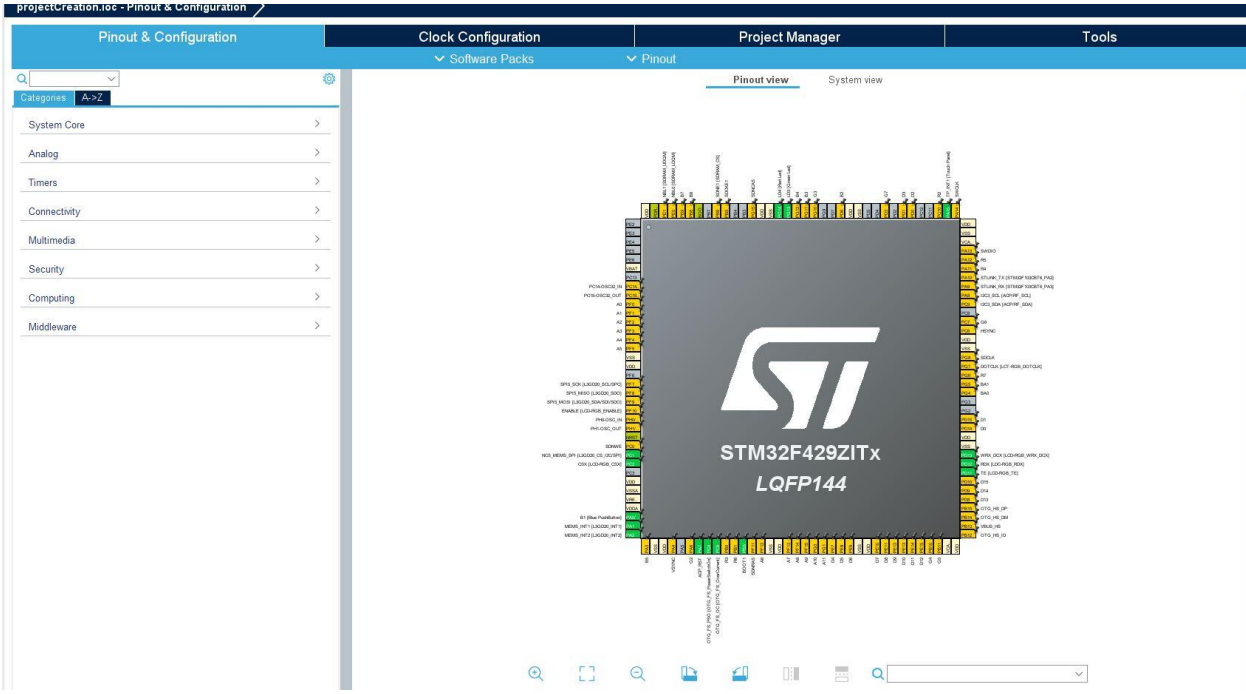
1. From the IDE,
2. Go to “Board Selector,” search and select our microcontroller, and click next
3. In the ‘Setup new project”, name the project appropriately and select “STMCube32” for



4. Click Finish
5. When the popup asks you if you want to “Initialize all peripherals to their default Mode”, **Select NO**
    a. Failure to abide by this will result in a massive amount of code being generated and all peripherals being enabled.

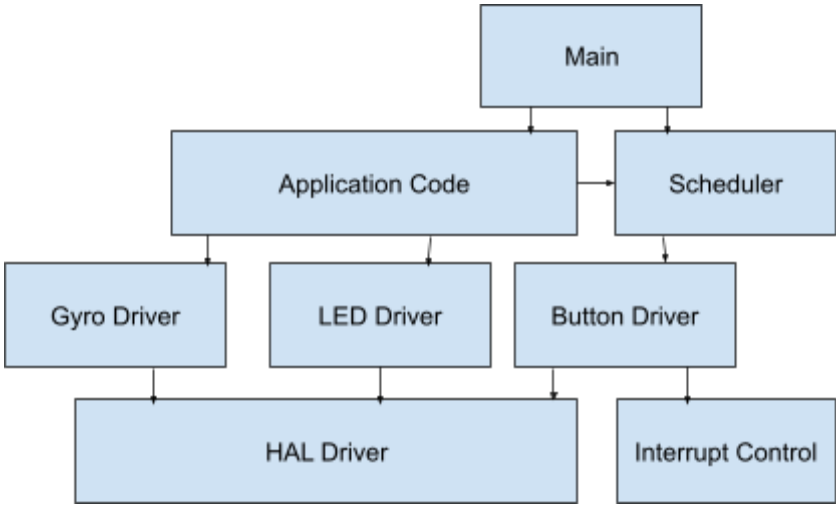


6. This screen should show up:



7. Select which peripherals should be enabled
    a. In the System Core section, ensure the NVIC and GPIO are selected
        i. This is the default, and they should already be selected “Green”
    b. In the Connectivity, make sure you have the correct SPI selected
        i. Refer to the electrical schematic and the document that contains the pinouts to determine which SPI is needed to interface with the GYRO
        ii. You must click the appropriate SPI and change the mode to ‘ Full-Duplex Master’
        iii. We will trigger the “Chip Select” manually, so select ‘Disable” for Hardware NSS Signal
8. Configure the clocks
    a. Using the reference sheet for the Gyro and the properties of SPI, make sure the bus’s clock that is connected to the SPI we have is not going absurdly too fast or too slow
        i. You can technically adjust this value later, but you will need to do some digging to find the code that is responsible for that
    b. You can use change the System Clock Mux to whatever you feel appropriate
        i. HSI is the “default” clock in our microcontroller
9. Go to Project > Generate Code
10. They will ask if you want to go to a C/C++ perspective; select yes.
11. Congratulations! You have generated your project using the HAL
12. Notice how the src and inc directories are in the Core directory

Coding Hierarchy for Lab 5


Main
Application Code
Scheduler
Gyro Driver
LED Driver
Button Driver
HAL Driver
Interrupt Control