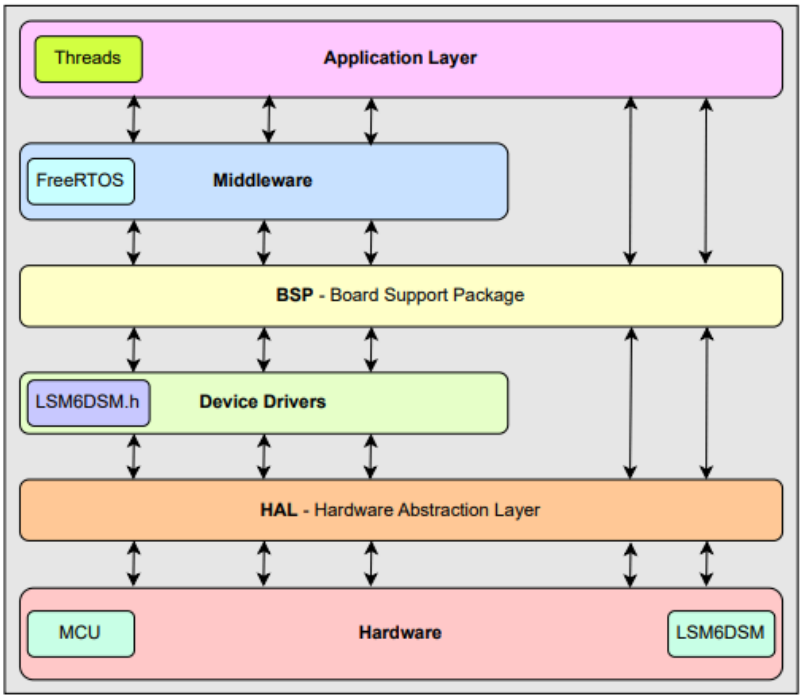
Breakdown of Tasks:

1. LSM6DSM Driver Integration



- The ST LSM6DSM driver publicly available on Github was integrated to the project via a BSP layer in ***bsp_accGyr.h***

  The BSP layer allows abstraction of specifics of the sensor IC so that hardware changes can be carried out without affecting Application Layer code.

- Time dependencies were accounted for using software timers from FreeRTOS in *thread_i2c.c* to regulate the sample rate from the sensor.

  The complementary filter implemented in *thread_main.c* also takes cares of time dependencies by keeping track of timestamps.

2. System Integration

- The BSP layer implemented for the Accelerometer and Gyroscope sensor in ***bsp_accGyr.h*** features platform dependent functions to handle I2C communication which use the HAL.

```
//---------------------------------------------------------------------------
// Function Definitions
static int32_t BSP_AccGyr_WriteReg(void *handle, uint8_t reg, const uint8_t *bufp,
                                   uint16_t len);
static int32_t BSP_AccGyr_ReadReg(void *handle, uint8_t reg, uint8_t *bufp,
                                  uint16_t len);
```

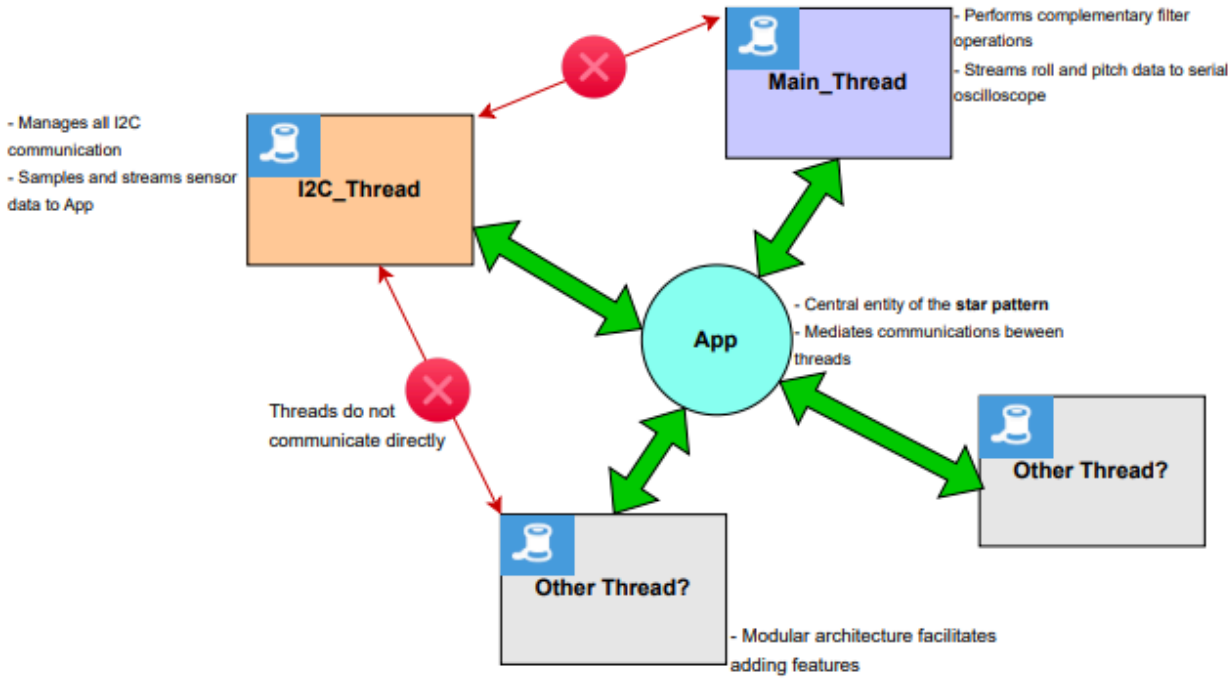
- Weakly implemented functions from ***bsp_accGyr.h*** are to be implemented by application code, enabling application dependent implementations supporting bare-metal codebases as well as RTOS.

```
/*
 * @brief  platform specific delay (platform/application dependent)
 *
 * @param  ms        delay in ms
 *
 */
__weak void BSP_AccGyr_Delay(uint32_t ms)
{
    // Implement in user application according to needs
    HAL_Delay(ms);
}

/*
 * @brief  platform specific bus TX RX wait(platform/application dependent)
 *
 * @param  ms        timeout in ms
 *
 */
__weak _Bool BSP_WaitForRxTx(uint32_t ms)
{
    UNUSED(ms);
    // Implement in user application if needed

    return true;
}
```

These weakly defined functions from the BSP were implemented by the *thread_i2c.c* file to provide the Accelerometer and Gyroscope BSP with capabilities to operate in a FreeRTOS context.

```
//---------------------------------------------------------------------------
// BSP Functions
void BSP_AccGyr_Delay(uint32_t ms)
{
    uint32_t delayTicks = pdMS_TO_TICKS(ms);
    vTaskDelay(delayTicks);
}

_Bool BSP_WaitForRxTx(uint32_t ms)
{
    return waitForNotify(ms);
}
```

- The above implementation for BSP was integrated into a framework featuring a Star Pattern, where an App entity handles communication between threads; Making it simpler to debug and to add new features.



- The testing strategy could be categorized into first unit testing for the sensor driver and BSP, then it would be important to perform integration testing where the functionality of the parts of the framework that interact with the newly added component/feature would be tested.
  Finally, end-to-end testing would be performed to ensure the system as a whole was not affected by the new component added, paying careful attention to malfunction that could indicate indirect dependencies like timings, memory consumption, and raise condition issues do not come up as a result of the changes.

- Regression testing is critical for old features/components where a new dependency has been added/removed by the new changes. This means that it would be of the highest importance to perform the previously existent test suites for those features/components first.
  End-to-end regression testing, stress testing and long-term testing would be ideally performed as well, making sure that new test cases are added to the existing test suite as it is executed.

3. Limitations and Challenges
   - Given the nature of the complementary filter implementation, the alpha value is fixed, which means that when the speakers are experiencing extreme and constant

acceleration changes the accelerometer data is taken into account in the exact same manner as when the speaker would be near to rest -when the accelerometer data would be more relevant since gravity would be easily measured.

- Rounding errors are to be expected due to truncation and conversions using floating point.

- A better filter model could be implemented since the *RCfilter.h* implementation is one of the most basic and low performing ones. A Kalman Filter would probably yield better results given that the alpha value would be better matched.

- Roll(x) and Pitch(y) estimates get noisy when Yaw(z) angles change.