# IMU Exercise: Component Implementation

Now it is time to write code! F´ implements components using C++ and as we saw in the design section, this implementation typically involves filling-in template files allowing us to focus on the specific challenges at-hand.

## Setup

Before proceeding with implementation, we should add our `Imu.cpp` file created in the design step to the `CMakeLists.txt` file in the `Imu` directory. The new `CMakeList.txt` file should look like this:

```
set(SOURCE_FILES
        "${CMAKE_CURRENT_LIST_DIR}/Imu.fpp"
        "${CMAKE_CURRENT_LIST_DIR}/Imu.cpp"
        )
register_fprime_module()
```

We should also ensure that the `#include <SystemReference/Gnu/Imu/Imu.hpp>` import exists in `Imu.cpp` . If you find something like `#include <SystemReference/Gnu/Imu/ImuComponentImpl.hpp>` please change it now.

## Component Implementation

F´ generates a series of handler functions for us to respond to each type of call. We need to fill in our specific logic for each of these empty handlers. Additionally, F´ creates a series of helper functions that may be called to send telemetry, emit events, and call output ports.

To build our implementation we run `fprime-util build` while in the `Imu` folder. Run this now to ensure that the cmake system is configured correctly for the component. Feel free to re-run the `build` command to check for compiler errors or other problems as you build.

Let's take a look at one specific handler. This is the power on command handler:

```
void Imu ::PowerSwitch_cmdHandler(const FwOpcodeType opCode, const U32 cmdSeq,
    power(powerState);
```

```
    this->log_ACTIVITY_HI_PowerStatus(powerState);
    this->cmdResponse_out(opCode, cmdSeq, Fw::CmdResponse::OK);
}
```

The critical functions of this handler are:

1. The `power` helper is called to send the I2C traffic to power on or off the IMU
2. An event announces the change in power state
3. A response is sent to the command dispatcher to inform it the command is complete

**Note:** some helper functions for working with the specifics of the Imu are available in Appendix I and are useful for developers who are less familiar with C++ and working with hardware. Feel-free to use these helpers in your code, or write your own!

A full implementation of the component is available here and the associated header is available here.

## Additional Resources

- F´ User Guide

## Next Steps

- Topology Integration and Testing