Extra Credit: Tilt-Compensated Electronic Compass

Bryan Blakeslee (bmb8610@rit.edu / bblakeslee4@gmail.com)

Submission Date: 5-15-17

Overview

The purpose of this project was to create an electronic compass out of the STM32 board. The compass was created using an LSM303C accelerometer/magnetometer. The accelerometer was used to detect the roll and pitch of the board, while the magnetometer was used to determine the board's heading. Once the direction was computed, the general compass heading was output to the LCD screen. By adding a jumper on the STM32 board, it was possible to run the board off a battery pack outside.

Analysis/Design

The system consists of an external LSM303C accelerometer/magnetometer connected to the STM32L4 chip over a three-wire serial peripheral interface (SPI) bus. Initial development of the algorithm used an external LSM303C breakout board purchased from SparkFun. (Part Number BOB-13303) The intent was to develop a working algorithm on the external board, due to ease of development and accessibility of the bus lines for a logic analyzer, then reassign pins to the internal LSM303C chip. (This same chip is also found integrated on the STM32 board.) Figure 1 shows the hardware system block diagram.

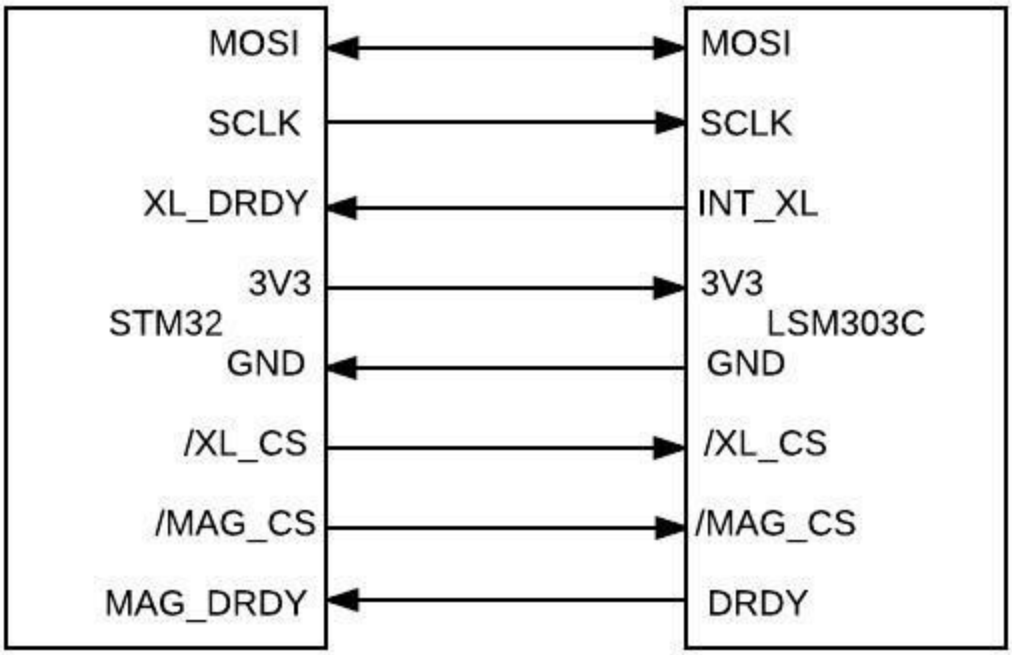

Figure 1: Compass hardware diagram

The pinout is shown in Table 1.

External STM32 Pin	Internal STM32 Pin	LSM303C Pin	Signal
3V3	3V3	VDD	Power
GND	GND	GND	Ground
PE10	PC0	/CS_MAG	Mag Select
PE11	PC2	DRDY	Mag Data Ready
PE12	PE1	INT_XL	Accel Data Ready
PE13	PD1	SCLK	SPI Clock
PE14	PE0	/CS_XL	Accel Select
PE15	PD4	SDA	SPI Data

Table 1: Wiring chart

The software stack consists of several modules, including SPI, accelerometer, magnetometer, compass, and LCD. The software stack diagram is shown in Figure 2.

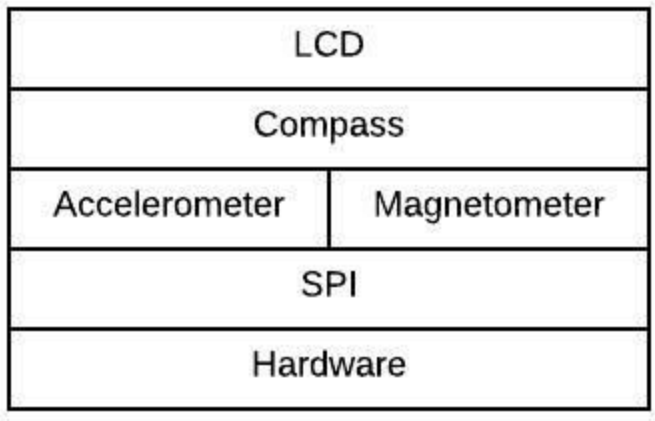

Figure 2: Compass software stack

The first driver to be developed was the SPI module. The SPI initialization function first maps the PE13 and PE15 pins to the SPI peripheral. These pins were chosen, as they had no other significant capacitive loads attached to the lines. It was then configured to run with a 2.5MHz clock, with positive polarity and first edge phase. Positive polarity means that the clock will be considered idle when high, while first edge phase means that data will be transmitted/read on the first clock edge and new data will be shifted onto the bus on the falling edge. The SPI hardware was also set to operate in half duplex mode, which means there is only one bidirectional data line. When transmitting, the most significant bit (MSB) of the data byte was transmitted first. Since multiple devices are being managed by a single SPI bus, the hardware driven chip select line was disabled. (These addressing lines will be managed in software at a higher level in the stack.) As the LSM303C is a slave device, the SPI peripheral was set as a bus master. Finally, the data line was initialized as an output, with a single transmission considered to be 8 bits of data. These parameters were set based on the required operating characteristics of the LSM303C listed in the datasheet.

The remaining two functions in the SPI module were the write and read functions. The write function configures the data line as an output, then enables the SPI peripheral. The device then waits for any other ongoing transmissions to complete, before writing the data to the bus. The function then waits until the SPI peripheral is done transmitting, before disabling it again. This was done to prevent bus errors that were observed to occur when changing the data direction when the SPI peripheral was left enabled. The read function follows similar logic. It sets the data line to an input, enables the SPI peripheral, then immediately disables it. This forces the peripheral to generate a single transmission of 8 bits of read data. Once the transmission is complete, the data is collected from the data register.

The next driver to be implemented was the accelerometer module. The SPI peripheral was used to initialize the accelerometer. In order for this to happen, the address byte contains both the address and the access mode. The access mode is indicated by the value of the MSB, and is a 1 for a read and a 0 for a write. This convention is established by the LSM303C and is also valid for the magnetometer.

First, the chip select line was configured as a push/pull output in high speed mode, with no pullup or pulldown resistors. High speed mode was chosen to ensure that the select line was always raised and lowered as quickly as possible to maximize the efficiency of bus transactions. Next, the data ready line was configured as an input. This allowed the accelerometer to call back to the STM32 to announce that there was new data ready for processing.

The accelerometer operating parameters were configured next. The chip was set to run exclusively in SPI mode, with register address auto-incrementing enabled. Since the 16 bit data registers for the accelerometer are located at consecutive addresses, this allows the high and low bytes to be read with only a single address write, reducing bus activity. The maximum dynamic range was set at +/-4g, which gives some tolerance against sensor saturation due to a sudden mechanical shock. Next, the data ready interrupt was enabled, allowing the LSM303C to call back to the STM32 when new data was ready. The accelerometer data output rate was set at 10Hz, so as to not overwhelm the LCD, which runs at a frame rate of 30Hz, with display updates. The block data update mode was also set, which keeps all three axes updating at the same time.

The remaining functions handled reading data from the accelerometer, checking the accelerometer's status, converting the ADC results to units of milli-g's, and selecting the device for bus communication. The data ready function is used to spin wait on the data ready pin for polled operation. (This could also be implemented as an interrupt; however, polling was chosen due to ease of implementation.) The select and deselect functions enable and disable the accelerometer for SPI communication. The status check function allows for an additional test of the accelerometer's data ready state. The read function returns the data from a single accelerometer axis. In this function, the address is set based on the provided axis, then the data is read using the address auto-increment feature. These two bytes are then assembled into a single 16-bit sample for that axis. Finally, the conversion function multiplies the ADC output by a scale factor to obtain the acceleration value in milli-g's.

Similar logic was used to set up the magnetometer. The same configuration used for the accelerometer for the chip select and data ready lines was applied to pins PE10 and PE11. The magnetometer was first set to operate in SPI mode. The data rate was set to 40Hz, with the XY axes in ultra high performance mode. The data rate was set higher for the magnetometer to avoid an occasional sample skip issue, where the accelerometer would prevent the magnetometer from signalling that it had data ready for reading. The Z axis was also set for ultra high performance mode. This mode was selected for all three axes as power consumption was not an issue, while speed and accuracy were paramount. The full scale range was set to +/-16 Gauss, as this was the only range supported by the device. Block data update was also enabled to keep the three axes in sync. Finally, the data ready interrupt was enabled, allowing the device to call back to the STM32 when data was ready.

The supporting functions of the magnetometer are functionally identical to the accelerometer, with the exception of the data read function. Since the magnetometer does not have a register setting for auto-increment, this information was encoded into the read address. In the event of a multibyte read, the second MSB is set as a 1. Successive reads will return data from the next consecutive registers.

The compass module was implemented next. This module is responsible for computing the roll, pitch, and yaw of the sensor board. It also contains telemetry functions for the raw and computed accelerometer and magnetometer values, which report detailed information to a serial terminal. These functions are only called when the DEBUG mode is enabled.

The roll and pitch functions rely on data from the accelerometer to compute the orientation of the board in 3D space. The equation for the roll computation is shown as Equation 1.

$$Roll = atan2(mgY,\ mgZ)$$

Equation 1: Roll computation

Equation 1 determines the roll by converting the Y and Z vector components into an angle in the YZ plane, which corresponds to roll. The atan2 function was used, as it preserves the quadrant sign, giving a valid answer for all 360 degrees. (The atan function is only valid for the first quadrant.)

Pitch is computed from roll using Equation 2.

$$Z2 = mgY * sin(Roll) + mgZ * cos(Roll)$$
$$Pitch = atan(-mgX / Z2)$$

Equation 2: Pitch computation

The pitch computation requires two equations. The first equation rotates the coordinate frame based on the roll, allowing pitch to be computed, even if the board is tilted. The second equation computes the angle of the pitch (in the same manner as roll), but with respect to the rotated coordinate frame, giving the angle in the XZ plane.

Finally, yaw is computed from the roll, pitch, and magnetometer data. This computation is shown as Equation 3.

$$Y2 = mgZ * sin(Roll) - mgY * cos(Roll)$$
$$Z2 = mgY * sin(Roll) + mgZ * cos(Roll)$$
$$X3 = mgX * cos(Pitch) + Z2 * sin(Pitch)$$
$$Yaw = atan2(Y2, X3)$$

Equation 3: Yaw computation

The first three equations in Equation 3 perform a roll and pitch coordinate frame rotation that bring the reference frame back to the horizontal. From there, the atan2 function is then used to compute the yaw angle of the rotated vectors in the XY plane.

The LCD module was then modified for use with the compass code. The only change made was to the character code table to configure the display of north, south, east, and west indicators. Due to the limited number of segments involved, north was represented as a lowercase “n”, south as an uppercase “S”, east as an uppercase “E”, and west as a backwards uppercase “E”. This allows the indication of eight different directions, north, south, east, west, northeast, southeast, southwest, and northwest. Further details of the LCD driver will not be discussed here. See the LCD report for more information.

The top level main function follows a superloop architecture. After hardware initialization, execution drops into the superloop. First, a spin wait halts execution until new accelerometer data is ready. Once ready, the accelerometer’s status is checked to ensure all three axes have valid data. If they do, then all three axes are read and the data converted to milli-g’s. This information is then used to compute the roll and pitch of the sensor. The same logic is then repeated for the magnetometer. The only additional step is the debiasing of the magnetometer data, which is needed because the magnetometer is much more sensitive to poor calibration. Once corrected, the data was used to compute the yaw, or compass heading, of the sensor. This yaw angle was then passed into a decoding section to determine what compass direction to display on the LCD display.

Special mention should be given to the calibration procedures for the sensors. For the accelerometer, the sensor was oriented with the Z axis pointed towards the ground. A series of 10 data samples were taken and averaged. The inverse of this value was then hard coded as the scaling factor. The same scaling factor was used for all three axes, as the behavior of the accelerometer was symmetric.

For the magnetometer, two factors are needed, a scaling factor and bias offset. Due to the inability to generate a reference magnetic field, the scale factor from the SparkFun

LSM303C driver was used. The bias offset was calculated independently for each axis, as each axis had a different bias level. To compute this factor for the X axis, the +X axis was pointed north and 10 data samples collected. The board was then reoriented with the +X axis pointed south and another 10 data samples collected. The average of each of these sets was then computed to obtain minimum and maximum value. These two values were then averaged together to obtain the bias offset for the X axis. This process was then repeated for the Y and Z axes. This process works because the Earth's magnetic field is relatively constant over short periods of time. In order to increase the accuracy of this calibration, it would be necessary to used a Helmholtz coil array and a reference magnetometer to generate a series of single-axis uniform fields.

Finally, the entire assembly can be configured for mobile operation. This is accomplished by installing a jumper at location JP2, then applying power via a battery pack through the USB port. Once power has been applied, the jumper can be removed.

Test Plan

The test plan chosen for this project was incremental testing. Each module was tested individually, starting with the bottom of the software stack. The first module tested was the SPI driver. This module's output pins were connected to a logic analyzer and a series of arbitrary data bytes were written out. If the bytes decoded successfully, then the write was considered successful. In order to test the read function, the SPI bus was connected to the LSM303C. Arbitrary values were then written into a device register, then read back out. If the read data matched the written data, then the read was considered successful.

To test the accelerometer, the sensor was oriented with the Z axis pointing down. Data from the accelerometer was then read out and converted to units of milli-g's. A correct dataset would show approximately 1000 milli-g on the Z axis and approximately 0 milli-g on all other axes. The board was then reoriented with the X axis pointed down and the same procedure followed. This procedure was repeated for the Y axis.

To test the magnetometer, the GPS coordinates for the Real-Time and Embedded Systems Lab was fed into the National Oceanic and Atmospheric Administration (NOAA) web calculator. (As of this writing, significant portions of the NOAA website return 404 errors, including the magnetic field calculator page.) A set of 10 data points were collected and compared to the reference data. Since the data agreed to within tens of milligauss, the device was considered functional.

To test the compass module, the direction of the compass points were determined. The board was then rotated, while the direction of the +X axis was monitored. When the +X axis was pointed north, the compass should indicate north on the LCD. The same should hold true for northeast, east, southeast, south, southwest, west, and northwest.

Project Results

Overall, the compass project was successful. The direction of the +X axis always corresponded to the correct compass direction. However, there was some extraneous lighting of additional pixels on the LCD display. This was due to some of the LCD control lines being shared with peripherals required to operate other systems of the STM32 board. The only way to mitigate this issue would be to redesign the board so that these lines are no longer shared.

Lessons Learned

An attempt was made to port the compassing program from the off-board LSM303C breakout module to the STM32's onboard LSM303C. This attempt was unsuccessful. When the pinout and SPI peripheral were changed to correspond to the onboard chip using information obtained from the STM32 schematics, the accelerometer would return only zeroes over the bus. However, the magnetometer did return valid data. Troubleshooting indicated that the magnetometer was operating according to the configured parameters. Based on the fact that the accelerometer was behaving as if it were unconfigured (despite the fact that the STM32 was transmitting the correct configuration data), it appears that the issue is localized around the accelerometer's chip select line.

There are two main possibilities for the root cause of the fault. First, the trace on the board may be broken. Second, the trace may run to a pin other than PE0, as shown on the schematic. As for the first cause, an attempt was made to verify the integrity of the trace using a multimeter. Unfortunately, the contact for the LSM303C was located underneath the package, making it impossible to probe. For the second cause, given that the chip select is a GPIO line, it could be wired incorrectly to any number of the nearly 100 GPIO pins on the STM32. Again, given the number of possible connections and that the LSM303C-side contact is inaccessible, it was not possible to determine the correct pin.

Lastly, the math for determining orientation in 3D space is extremely basic. If the board is tilted close to 90 degrees, e.g.: on its edge, then the Z axis component of gravity will become very small and possibly zero. This leads to a divide by zero fault in the roll

computation and potential instability in the following calculations. Should stability be desired, the roll equation should be modified according to Equation 4.

$$Roll = atan2(mgY,\ mgZ + (\alpha * mgX))$$

Equation 4: Stability compensation for roll calculation

The modification in Equation 4 prevents the divide by zero that causes the instability. This modification was not made in the implementation due to time concerns about creating a calibration procedure for the alpha value.