# Module 21

## Lab 21: Sensor Integration

# Lab: Sensor Integration

## 21.0 Objectives

The purpose of this lab is to explore sensor integration. You will interface sensors to the robot and develop software, which allows the robot to sample, process and interpret data. See Figure 1.

1. You will use the I2C protocol to interface one or more sensors to the robot, writing low-level communication software.
2. You will use interrupts to sample the sensor(s) at a regular rate.
3. You will study the noise generated in the data acquisition system.
4. You will evaluate digital filters in an attempt to improve **signal to noise** ratio, which is defined as the signal amplitude divided by the noise amplitude.
5. You will develop high-level software to interpret the data and evaluate the performance of the sensor system in your robot application.

**Good to Know**: You have created a sampling data acquisition system in Labs 6 and 10. I.e., the software sampled the line sensor 100 times/second. However, in this lab you will study sampling in a more fundamental way. There are three tasks that most embedded systems perform: collecting data, making decisions, and affecting outputs. In this lab, we will focus on collecting data, but the robot challenge will require sensor integration and decision making..

## 21.1 Getting Started

### 21.1.1 Software Starter Projects

These projects are simple but fully functional data acquisition systems:
**BMI160** (measures acceleration, rotation, direction, 9-axis IMU),
**HDC2080** (measures temperature and humidity),
**OPT3001** (measures light intensity),
**OPT3101** (measures distance in three directions),
**TMP117** (measures temperature)
These projects are the Lab 21 starter projects:
**Lab21_OPT3101** (for doing Lab 21 using OPT3101),
**Lab21_BMI150** (for doing Lab 21 using BMI160/BMM150)

### 21.1.2 Student Resources (in datasheet directory)

MSP432P4xx Technical Reference Manual, eUSCI I2C (SLAU356)
MSP432P401R Datasheet, msp432p401m.pdf (SLAS826)
OPT3101 datasheet
TI BP-BASSENSORSMKII User Guide (SLAU829)
BMI160, BMM150, OPT3001, TMP117, HDC2080 datasheets
BASSENSORSMKII_Conflict_Plan.docx
TI-RSLK MAX Construction Guide (SEKP164A)

### 21.1.3 Reading Materials

Sections 11.4 (I2C), 15.1 (Data Acquisition), (noise), 15.6 (digital signal processing) 17.6 (Fuzzy Logic), and Chapter 21, “Embedded Systems: Introduction to Robotics"

**Good to Know**: The sensors in this lab involve a mixed analog-digital integrated circuit (IC). This means the single IC contains the sensor, an analog circuit front end, an analog to digital converter, digital signal processing, and a communication interface to the microcontroller. The OPT3101 based, wide FOV time of flight distance sensor uses phase measurement to convert analog signals into digital form. The rest of the sensors use a traditional analog to digital voltage converter (ADC). All sensors in this module deploy I2C to communicate between the sensor module and the microcontroller.

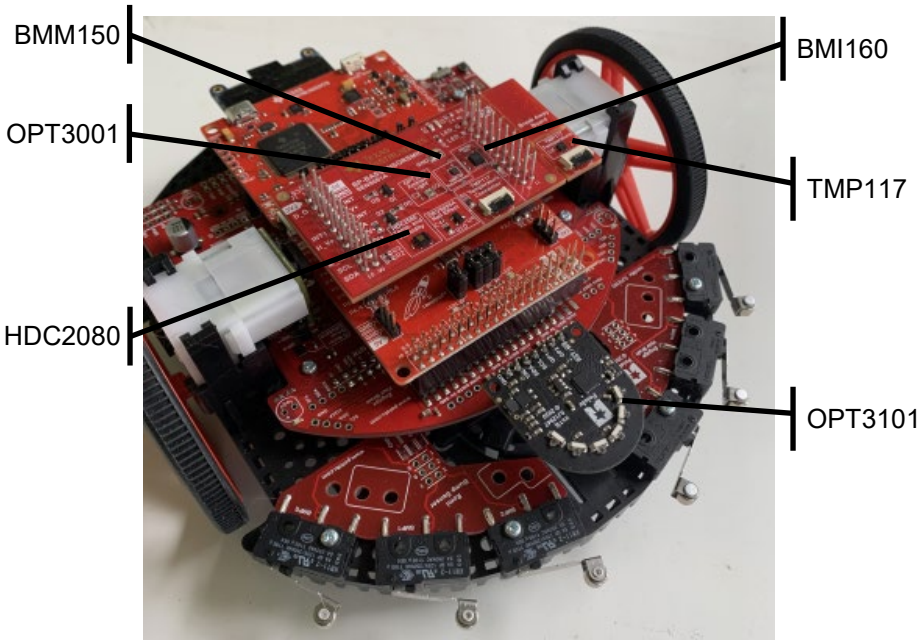


*Figure 1. The OPT3101 measures distance to a wall, the BMI160 measures acceleration and rotation, the BMM150 measures magnetic fields (direction), the TMP117 measures temperature, the OPT3001 measures light intensity, and the HDC2080 measures temperature and humidity.*

# Lab: Sensor Integration

### 21.1.4 Components needed for this lab

Components needed to build the robot are provided in TI-RSLK MAX robot kit (TIRSLK-EVM). For this lab, you will need to purchase either the Pololu OPT3101 distance sensor (Option A) or the TI BP-BASSENSORSMKII BoosterPack (Option B). Batteries also are needed to power your robot.

| Quantity | Description | Manufacturer | Mfg P/N |
|---|---|---|---|
| 1 | TI-RSLK MAX robot kit | TI | TIRSLK-EVM |
| 1 | 3-Channel, OPT3101-Based, Wide FOV Time-of-Flight Distance Sensor for TI-RSLK | Pololu | #3680 |
| 1 | Building Automation Systems Sensors MKII BoosterPack | TI | BP-BASSENSORSMKII |

*Table 1 Parts needed for this lab.*

### 21.1.5 Lab equipment needed

Oscilloscope (2 channels at least 10 MHz sampling), optional
Logic Analyzer (2 channels at least 10 MHz sampling), optional
Soldering iron if the 7-pin OPT3101 socket needs to be soldered to RSLK

## 21.2 System Design Requirements

### 21.2.1 Low-level I2C communication software, both options A and B:

**The first goal** is to write and test four low-level functions that implement I2C communication. These functions are located in the I2CB1.c file.

**I2CB1_Send2** – Sends 2 bytes to the I2C device
**I2CB1_Send3** – Sends 3 bytes to the I2C device
**I2CB1_Send4** – Sends 4 bytes to the I2C device
**I2CB1_Recv1** – Receives 1 byte from the I2C device

The fundamentals about I2C can be found in the many textbooks, the MSP432 datasheet, and in the lectures associated with this module. The low level details of exactly what these four functions should do can be found in the comments of the I2CB1.h and I2CB1.c files.

### 21.2.2 Sensing distance to the wall using the OPT3101 sensor, option A:

**The first goal** is to attach the distance sensor based on TI OPT3101 to the robot and measure raw sensor output as a function of true distance, The OPT3101 device will activate the IR LEDs and receive IR reflected light. It will convert raw phase signals to “distance” giving us sampled data discrete both in amplitude and time. The sensor returns three measurements (left, center, right). The basic sampling rate is 30 Hz, meaning each of left, center, right are updated at 10 Hz. Software plus calibration will allow the robot to measure distance to the wall. The range should be about 10 to 400 mm. The resolution should be about 1 mm at 200 mm.

Consider how the sensor will be used on the robot. For example if the sensor will be used to implement autonomous driving, it needs to measure the left and right distance to the wall defining the race track. Figure 2A shows one possible test scenario for the sensor. Build a portion of the race track and use a ruler to measure truth. Truth is the actual distance from the robot to the wall.

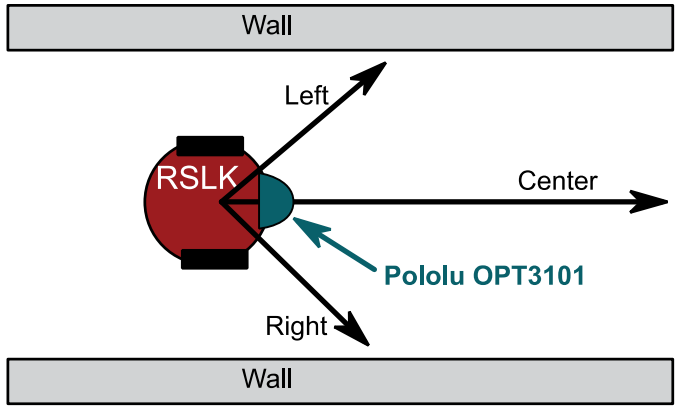


*Figure 2A. Truth is the actual distance from the robot to the wall.*

Pololu defines the front of the sensor module as distance equal to zero, but it doesn’t matter where you define reference as long as you are consistent. In Figure 2A, distance is defined from the center of the robot. Similarly it doesn’t matter what angle you define the distance to the wall, again be consistent. You will use these measurements to drive the robot a constant distance to one wall or an equal distance between the walls. During the robot challenge you will endeavor to keep the robot away from the wall. Due to the nature of the robot challenges, we are not interested in distances beyond 500 mm. Using a physical setup similar to the robot challenge, measure raw sensor output as a function of true distance, similar to Figure 3A. Record both raw distances and amplitudes.

# Lab: Sensor Integration

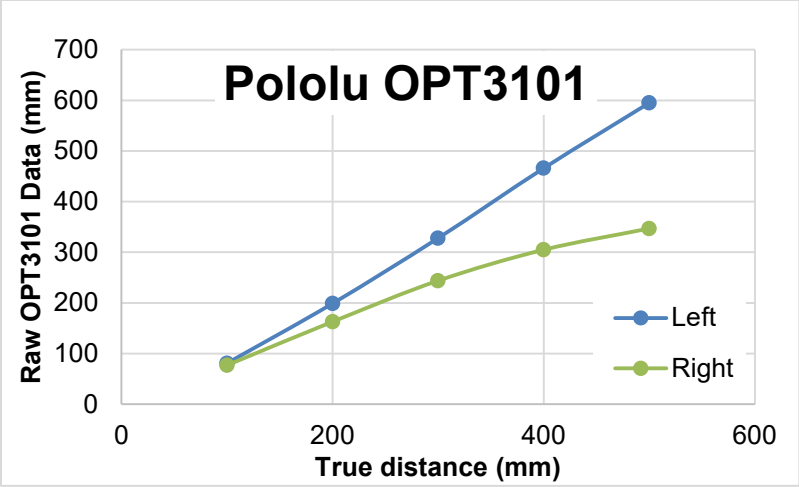


*Figure 3A. True distance was measured from the center of the robot.*

You will write functions that map raw OPT3101 data back into distance, and plot measured distance versus true distance, similar to Figure 4A.

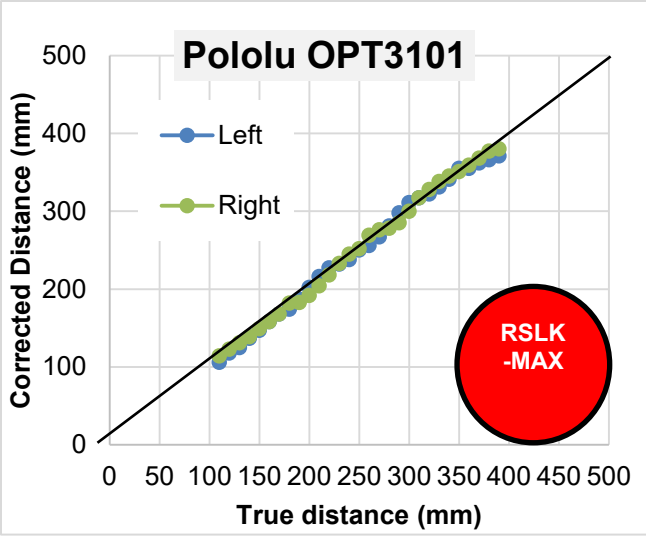


*Figure 4A. Typical response curve after calibration.*

Choose between linear or quadratic functions depending on the shape of the response for your sensor in your situation. The data shown in Figures 3A and 4A used a linear equation for left and a quadratic equation for the right signal.

**Left** = (**A***$n_l$)/2048 + **B**;
**Right** = (($n_r$*(**C***$n_r$ + **D**) + **E**)/32768;

where **A**, **B**, **C**, **D**, and **E** are integer calibration coefficients, which will be empirically determined. Feel free to develop your own method to facilitate the use of the sensor.

**The second goal** of this lab is to measure signal to noise ratio (SNR) and then deploy a digital filter to reduce the noise. If the distance to the object is fixed then variability in repeated samples results from noise. One simple measure of SNR is the average measurement divided by the standard deviation, given multiple measurements on a constant input. The average (X) and standard deviation (S) of N samples are

$$\overline{X} = \frac{1}{N}\sum_i X_i \qquad S^2 = \frac{1}{N-1}\sum_i (X_i - \overline{X})^2$$

The OPT3101 triggers the measurement periodically. This periodic triggering is defined as **sampling**, allowing the system to process the data in both the time and frequency domains. An edge-triggered interrupt occurs when a measurement is complete. Each of the three channels is sampled at 10 Hz. Collecting data periodically allows you to implement a digital filter, which passes some data of some frequencies while rejecting others. The purpose of the digital filter is to improve signal to noise ratio. You will study this low pass filter

y(n) = (x(n)+x(n-1)+…+x(n-N-1)/N

for N = 1 to 256. x(n) is the current sample, x(n-1) is the previous sample, x(n-2) is two samples ago, … and y(n) is the current filter output. You will use the sampled data to study fundamental concepts like the range, resolution, precision, the Oversampling and Nyquist Theorem, aliasing, noise, probability mass function, signal to noise ratio, and the Central Limit Theorem.

An **impulse** digital sequence has one nonzero value and the rest of the points in the sequence are zero, …,0,0,0,1,0,0,0,… The **impulse response** of a filter is the output of the filter given the input is an impulse. If N=4, the impulse response of this filter is …,0,0,0,¼,¼,¼,¼,0,0,0,… This filter is called a **finite impulse response** (FIR) filter, because the impulse response has a finite number of nonzero outputs.

# Lab: Sensor Integration

A **step** digital sequence has an infinite number of zeros, followed by an infinite number of non-zeros of the same value, …,0,0,0,1,1,1,… The **step response** of a filter is the output of the filter given the input is a step function. If N=4, the step response of this filter is …,0,0,0,0.25,0.5,0.75,1,1,1,… In other words if the distance to the wall were to change, this filter will cause a delay in the time the software sees this change in input. You will choose a filter that improves signal to noise ratio without causing too much delay in the step response.

Figure 5A shows a typical probability mass function. The OPT3101 was sampled 1000 times with the distance fixed at 200 mm. The N=1 curve shows the distribution with no filter (raw data from OPT3101). All curves have the same average, but notice the standard deviation reduces with the size of the digital filter. We assume the noise in one sample is independent from noise in another sample. The Central Limit Theorem states as N is increased, the shape of this noise distribution will approach a Gaussian shape (normally distributed).

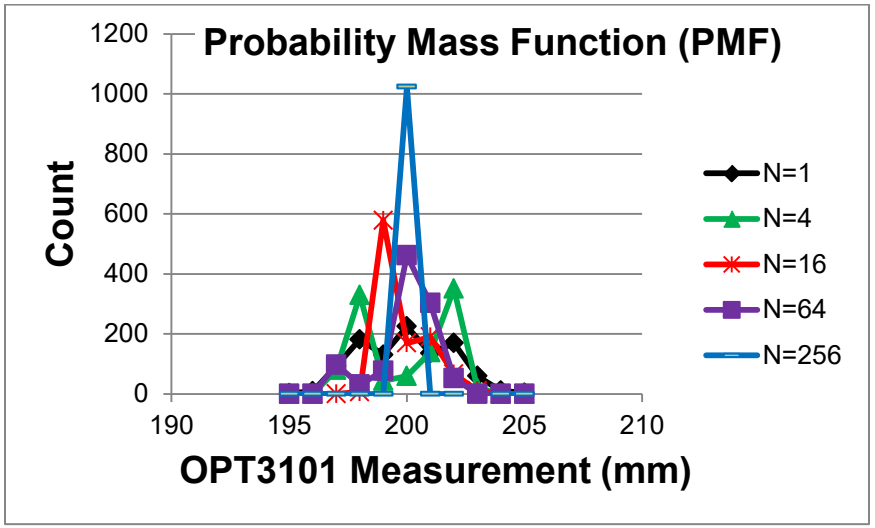


*Figure 5A. Typical PMF data, counting the number of times a particular output was observed when measured 1000 times with a constant input.*

**Good to Know**: While the SNR improves with the size of the digital filter, the digital filter will introduce a **group delay** or lag. The step response of the digital filter can be used to estimate lag. Basically if the input distance were to change in a step manner, how quickly will the output of the filter respond to the change? In general, the group delay for an averaging filter is $0.63*N/f_s$. For a sampling rate of 10 Hz and N=16, the lag will be 1 sec.

**The third goal** is to evaluate the accuracy of the distance measurement. **Accuracy** is defined as the difference between truth and measured. Let $x_t$ be the true distance from the robot reference point to the wall, as measured with a ruler. The instrument accuracy is the absolute error referenced to the National Institute of Standards and Technology (NIST) of the entire system including transducer, electronics, and software. Let $x_m$ be the values as measured by the instrument. We define average accuracy of full scale in percent as

$$\frac{100}{n}\sum_{i=0}^{n}\frac{|x_{ti}-x_{mi}|}{x_{t\max}}$$

The system **resolution** is the smallest input signal difference, $\Delta x$ that can be detected by the entire system including transducer, electronics, and software. The resolution of the system is limited by noise processes in the transducer itself, noise processes in the electronics, and the number of bits in the conversion from analog to digital. For this lab, resolution will be limited by noise in the distance sensor. In particular we will take the standard deviation S as a simple indicator of resolution.

**The fourth goal** is to evaluate the field of view (FOV) of the sensor, see Figure 6A. Basically, the FOV is the area each sensor can reliably detect an object. A common mistake beginning robot engineers make is to consider this sensor similar to a camera or a human eye. This sensor returns three numbers: right, center, and left, representing the distances of the closest object in the FOV of each measurement. One simple approach to evaluating FOV is to look at the amplitude parameter, which is a measure of the signal strength of the captured return light. You will find while collecting data similar to Figure 3A, there is an amplitude above which the measurement is reliable and below which the data is not reliable. For this data, the amplitude threshold was 1000. To approximate FOV, keep the robot fixed and move an object around in front of the sensors making note of the places the amplitude is above 1000.

A more formal definition of FOV is to move the object in a semicircle pattern at constant distance but variable angle in front of the sensor. Record the maximum amplitude, and determine the range of angles the magnitude is above ½ of this maximum.

Go back and modify the mapping functions so amplitude is also considered. If the amplitude is below some threshold, set the distance measurement to its maximum usable value.

# Lab: Sensor Integration

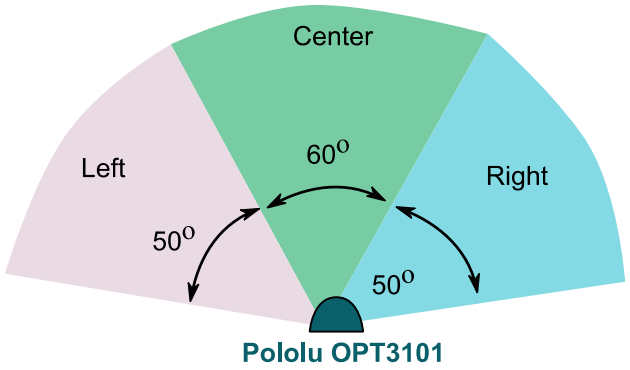


*Figure 6A. The Pololu OPT3101 has a wide field of view.*

If your four I2C functions are operational, you can use the Lab21_OPT3101 project to study the behavior of the data acquisition system. Conversely, if your four I2C functions do not work, you can use the OPT3101 project to complete the remaining tasks of this lab.

**21.2.3 Using the TI Building Automation Systems Sensors MKII BoosterPack to measure parameters with the 9-axis IMU, option B:**

**The first goal** is to attach the BP-BASSENSORSMKII BoosterPack to the robot. There are some pin conflicts between the BoosterPack and the RSLK robot. In this option, we will use the 9-axis IMU with the BMI160 and BMM150 ICs. The 9-axis IMU can be safely used with the robot as long as we do not activate INT2.

**Good to Know**: Each of the sensors on the BP-BASSENSORSMKII has pin conflicts with the RSLK robot. Look at the file BASSENSORSMKII_Conflict_Plan.docx …Please refer to the source code for the TMP117, OPT3001, and HDC2080 before using these sensors on the robot.

Consider how the IMU will be used on the robot. For example, you might use the accelerometer to detect collisions. You might use the magnetometer to determine absolute angle. The specific goal of this section is to sample the IMU at a regular rate and write software that takes the raw measurements and returns a sensor parameter the robot can use.

One way to detect collisions is to measure **jerk**, which is the derivative of acceleration, $\mathbf{j} = d\mathbf{a}/d\mathbf{t}$. Assume the IMU is sampled every $\Delta t$. Let $a_x$ and $a_y$ be the x and y axis measures of acceleration respectively.

$j_x = (a_x(n) - a_x(n-1))/\Delta t$

$j_y = (a_y(n) - a_y(n-1))/\Delta t$

Collision if $(j_x^2 + j_y^2) >$ threshold

You may or may not need to know the direction of impact. One simple approach to determine direction is to look for the largest jerk

Front impact if $j_y >$ +Constant

Rear impact if $j_y <$ -Constant

Right impact if $j_x >$ +Constant

Left impact if $j_x <$ -Constant

If you need to know more precisely you could use trigonometry

Angle of impact is $\Phi = \arctan(j_y / j_x)$

Adjust $\Phi$ depending on the signs of $j_x$ and $j_y$

The presence of the DC magnets in the RSLK motors will have a significant effect on the magnetometer. Luckily the DC component of the magnetic field from the motors is constant and can be calibrated out. The data in Figure 2B shows the calibration can be perform by adding a fixed constant to the data.

```
mx = mag_data.x+Kx;
my = mag_data.y+Ky;
```

where Kx and Ky are calibration constants.

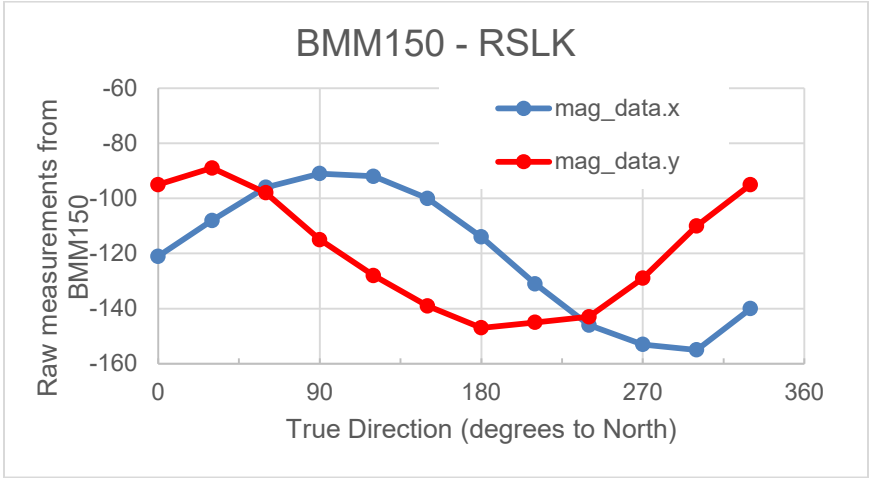


*Figure 2B. True distance was measured with a compass, raw measurements where obtained from the BMM150 while the robot was stationary.*

Depending on how your robot will use the IMU, you may or may not need to calculate compass direction. For example, if you want to know whether or not the

robot is facing North (direction=0), you could execute a simple test to see if mx is close to 0, and my is large positive.

```
mx = mag_data.x+120;
my = mag_data.y+120;
if((mx > -4)&&(mx < 4)&&(my > 10)){
  Motor_Stop(); // pointing North
  return 1; // success
}
```

However, if you want to measure direction you will need to use **arctan(my/mx)** and to adjust the direction depending on the sign of mx and my, as shown in Figure 3B.

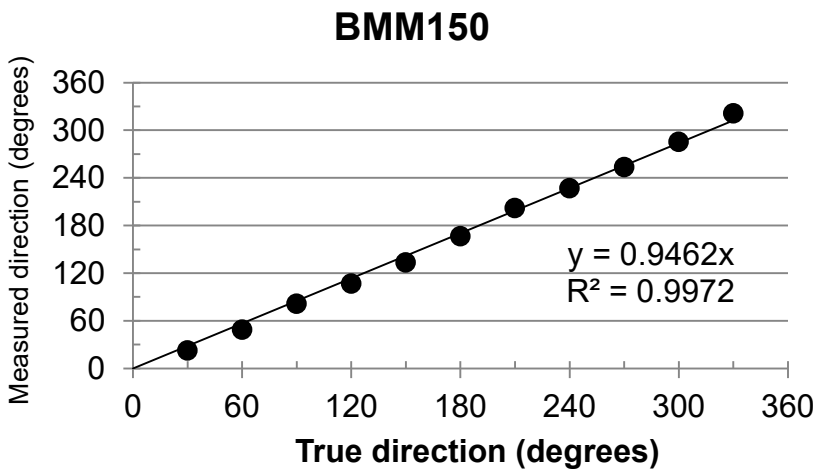


*Figure 3B. Calibrated data showing RSLK can accurately measure direction*

**The second goal** of this lab is to measure signal to noise ratio (SNR) and then deploy a digital filter to reduce the noise. If the input is fixed then variability in repeated samples results from noise. One simple measure of SNR is the average measurement divided by the standard deviation, given multiple measurements on a constant input. Perform this SNR measurement in a situation similar to how the sensors will be used on the robot.

If needed add a digital filter is to improve signal to noise ratio. You will study this low pass filter

$$y(n) = (x(n)+x(n-1)+\ldots+x(n-N-1)/N$$

for N = 1 to 256. x(n) is the current sample, x(n-1) is the previous sample, x(n-2) is two sample ago, … and y(n) is the current filter output. You will use the sampled data to study fundamental concepts like the range, resolution, precision, the Oversampling and Nyquist Theorem, aliasing, noise, probability mass function, signal to noise ratio, and the Central Limit Theorem.

Figure 4B shows y-axis acceleration measured while the robot is rolling at a constant speed, which should have had no acceleration. The plot also shows three digital filters of length 4, 8, and 16. Notice the improvement in SNR as the filter length is increased.

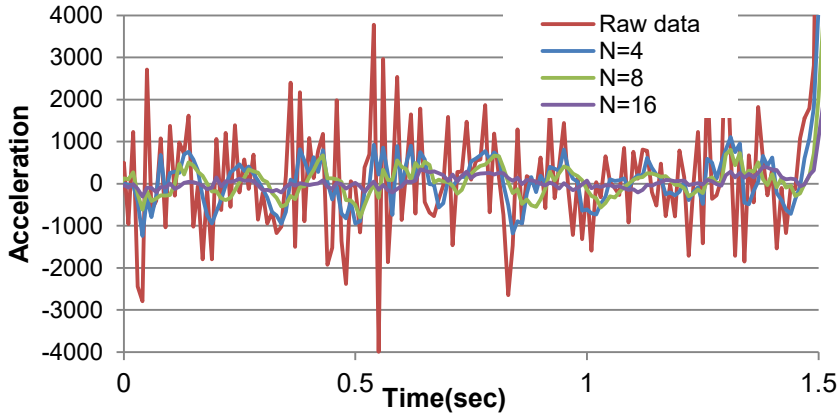


*Figure 4B. Measured acceleration while moving at a constant speed. Some of the variation is mechanical vibrations bouncing on the hard floor.*

**The third goal** is to evaluate the accuracy of the measurement. Accuracy is relevant for measuring compass angle with the magnetometer. **Accuracy** is defined as the difference between truth and measured. The system **resolution** is the smallest input signal difference, $\Delta x$ that can be detected by the entire system including transducer, electronics, and software. The resolution of the system is limited by noise processes in the transducer itself, noise processes in the electronics, and the number of bits in the conversion from analog to digital. For this lab, resolution will be limited by noise in the sensor. In particular we will take the standard deviation S as a simple indicator of resolution.

If we are using the accelerometer to detect collisions, we will want to measure the **response time**. Make simultaneous recordings of the bump sensor (true

measure of collision) and the jerk calculations from your software. Response time is the lapsed time between actual collision and the time the acceleration sensor/software detect the collision.

Figure 5B shows data measured and calculated data during a collision. The plot shows raw measurement, digital filters of length 4 and 8, and jerk. Notice the delay caused by using the digital filter. The jerk was calculated using this approximation of derivative

$$jerk = x(n) + 2x(n-1) - 2x(n-2) - x(n-3)$$

Feel free to use whatever means is appropriate for your use of the robot.

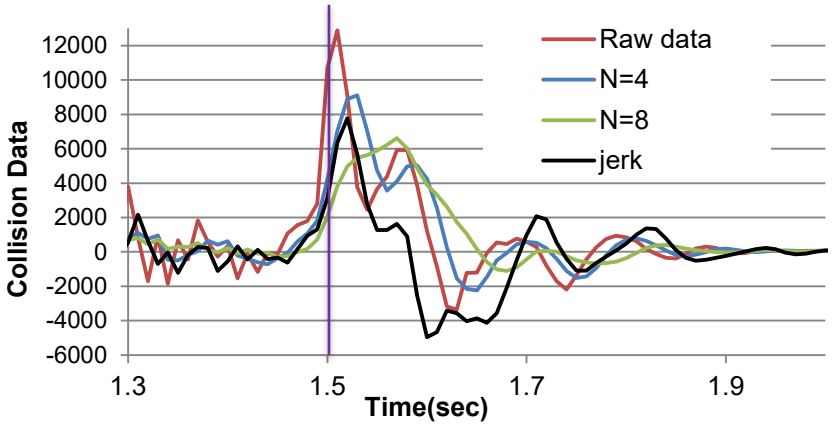


*Figure 5B. Measured and calculated data collected during a collision. The collision occurred at 1.5 sec. This data shows a 20-ms response time between collision and maximum jerk.*

**Good to Know**: The magnitude of acceleration and jerk during a collision will be a function of robot speed.

**21.2.4 Sensor Integration:**
There are many sensors available for the RSLK. The goal is to combine three or more sensors to solve a single task. In other labs, we introduced

- Line sensor to detect black tape on floor (Labs 6, 7, 10)
- Bump sensors to detect collisions (Labs 9, 10, 13, 14)
- IR distance sensor to measure distance to wall (Lab 15)
- Microphone to measure sound (Lab 15)
- Tachometer to measure wheel rotational speed (Lab 16)

In this lab, we presented

- OPT3101 sensor to measure distance to wall
- BMI160 to measure acceleration
- BMI160 to measure rotational speed
- BMM150 to measure magnetic field
- OPT3001 to measure light intensity (remove bumpers)
- TMP117 to measure temperature (remove bumpers)
- HDC2080 to measure temperature and humidity (remove bumpers)

Refer to Activity 4 5 and 6 for example tasks that will require multiple sensors to complete. Feel free to modify or create your own task.

## 21.3 Experiment set-up

**21.3.1 Sensing distance to the wall using the OPT3101 sensor, option A:**
Connect the 3-Channel, Pololu OPT3101-Based Distance Sensor to the TI-RSLK robot, see Figure 1. Refer to the Pololu #3680 website for information on connecting it to RSLK. For instructions on how to take the robot apart, refer to Section III: Disassembly guide within the RSLK MAX Construction Guide (SEKP164). Once the robot is apart, solder the OPT3101 header onto the RSLK-MAX PCB, and then rebuild the robot. Run the **OPT3101** project in the debugger and visualize the values in the **Distances** array. Move an object in front of the three sensors to verify the sensor is operation. The data should be monotonic with the true distance.

**21.3.2 Measuring parameters with the 9-axis IMU, option B:**
Connect the BP-BASSENSORSMKII BoosterPack to the TI-RSLK robot simply by plugging it onto the MSP432 LaunchPad as shown in Figure 1. Run the **BMI160** project in the debugger and visualize the acceleration (Ax,Ay,Az), rotation (Rx,Ry,Rz), and magnetic field (Mx,My,Mz). Acceleration vector due to gravity should be observable. Moving the robot in the yaw, pitch and roll direction should affect rotation data. The magnetic data Mx and My should be affected by compass direction. .

## 21.4 System Development Plan

**21.4.1 Low-level I2C communication software, both options A and B:**
If you have an actual oscilloscope, you should look at the SDA and SCL signals between the MSP432 and the I2C device you are using in this module. Use either of the **OPT3101** project or **BMI160** project to visualize communication between the processor (master) and the sensor (slave). Compare your scope traces with the appropriate lecture PPTX and identify start, address, R/W, data, and stop fields.

# Lab: Sensor Integration

Some full featured logic analyzers will decode and interpret I2C data. If you have both, use both because the information is not the same. The scope shows the actual waveform with the slow rising edge that occurs because I2C has passive pull up to high. The logic analyzer can show start, ack, nak, and stop codes.

As you develop and test the following four functions, compare expected waveforms with measured waveforms.

**I2CB1_Send2** – Sends 2 bytes to the I2C device
**I2CB1_Send3** – Sends 3 bytes to the I2C device
**I2CB1_Send4** – Sends 4 bytes to the I2C device
**I2CB1_Recv1** – Receives 1 byte from the I2C device

**Note**: You will not be able to complete this lab without reading the MSP432 Technical Reference Manual (slau356h.pdf). Look at the chapter on I2C (eUSCI – I2C Mode), and go line by line through the existing I2CB1_Init I2CB1_Send I2CB1_Recv and I2CB1_Send1 functions. These four functions work, but you need to understand each line, by looking up each of the registers it accesses. Once you understand each line, you will be able to write the necessary four functions.

**21.4.2 Develop and test sensor integration system, both options A and B:**
The steps during testing should focus on exactly how the sensors will be used on the robot, as described in section 21.2.2. You can display calculated results on the LCD, send them to the PC via the UART or dump them in a buffer. The procedural steps include:

- Collect calibration data
- Write functions to extract important parameters
- Measure signal to noise ratio
- Study the experimental behavior of digital filtering
- Collect noise data and produce a PMF of the noise process
- Study the theortetical behavior of digital filtering
- Determine accuracy, resolution, FOV, and response time as needed

Assuming there were no noise and the input were constant, you would expect all the samples to be equal. The fact that the samples are not the same is the result of **noise**. Without noise, the variance and standard deviations would be zero. The **coefficient of variation** (CV) is the standard deviation divided by the mean (μ),

$$CV = \sigma/\mu$$

1/CV is a simple estimate of the signal to noise ratio (SNR). With the input fixed at a typical value, we will approximate the precision in bits of the system as

$$\log_2(\mu/\sigma)$$

This project also implements a simple digital filter. This filter calculates the average of the last N samples.

$$y(n) = (x(n)+x(n-1)+\ldots+x(n-N-1)/N$$

Averaging is a simple method to improve signal to noise ratio. Collect 1/CV (SNR) and $\log_2(\mu/\sigma)$ data as a function of N = 1, 2, 4, …, and 512. What do you observe?

Next review the Central Limit Theorem (CLT) from your Probability and Statistics class. Look up the assumptions to see if the CLT applies to these measurements. The CLT states that as independent random variables are added, their sum tends toward a Normal or Gaussian distribution. In this experiment you should find, as N is increased the PMF goes from having multiple peaks to having just one peak (see Figure 5A).

If you wish to learn more about the simple averaging filter, open the spreadsheet **FIR_Digital_LowPassFilter.xls** located in the **inc** folder. You can change the sampling rate (fs) and filter size (N) and visualize the frequency and step responses. The two parameters you can adjust are highlighted in yellow. If the sampling rate is 2000 Hz, and the size N is 64, then the filter has a cutoff frequency (fc) of 16 Hz, see Figure 7.

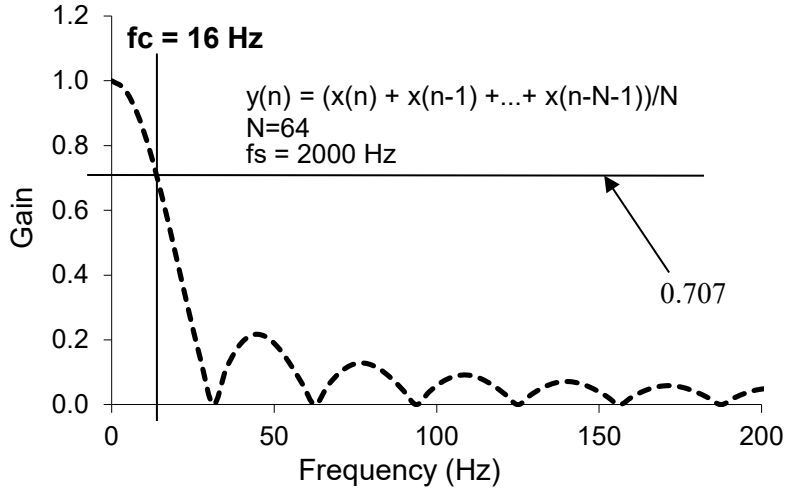


*Figure 7. Frequency response of averaging filter with N=64.*

# Lab: Sensor Integration

**21.4.3 Sensor Integration:**
There are many ways to combine sensors to solve a single tasks. If you were to have two measurements of the same parameter, such as temperature from the TMP117 and temperature from the HDC2080, you could combine them using a weighted average

$$T = (k1*T1 + k2*T2)/(k1+k2)$$

Statistics suggests using $\sigma^{-2}$ as the weighting. A smaller standard deviation means we have more confidence in the signal.

Another natural way to combine sensors uses the finite state machine approach presented in Lab 7. Let S1 S2 and S3 be three different sensor readings, and let c1 c2 c3 be corresponding constants. You could define state transition arrows using syntax like

- Change state if (S1 > c1) AND (S2 < c2)
- Change state if If (S1 == c1) OR (S3 >= c3)

Notice the units of $S_i$ must match the units of the corresponding constant $c_i$. The outputs of the state would be PWM duty cycle outputs to the two motors.

A third way to combine sensors uses Fuzzy Logic. See chapters 17 and 21 in the book for more information about Fuzzy Logic.

## 21.5 Troubleshooting

***Sensors don't work:***

- Verify the operation of the sensors with the corresponding project: BMI160, HDC2080, OPT3001, OPT3101, or TMP117. These are simple but fully functional systems
- Check the wiring and use a scope or logic analyzer to observe the SDA and SCL signals

***Software crashes:***

- Add instrumentation profiling software execution (set/clear unused GPIO pins) and observe signals on scope or logic analyzer.
- Add software dumps to collect strategic information as it executes

***Statistical calculations are incorrect:***

- Look at the collected data in the arrays. Bad data causes bad statistics.
- You can reduce the sizes of arrays, and perform the statistical calculations by hand to check the software.

## 21.6 Things to think about

In this section, we list thought questions to consider after completing this lab. These questions are meant to test your understanding of the concepts in this lab.

- In I2C what causes the signal to go low? What causes it to go high?
- What is the limiting factor in this system that restricts resolution and accuracy of the sensor measurement?
- What is the tradeoff when choosing the size digital filter? What is the advantage of a large filter? What is the advantage of a small filter?
- Why are interrupts required in this lab? i.e., what do interrupts enable us to do?
- How is the mailbox used in the lab? What does Semaphore=0 mean? What does Semaphore =1 mean?
- What would it mean if the Semaphore were already 1 at the time the ISR is trying to set it to 1 again?
- Consider detecting collisions with bump switches and the accelerometer. What advantage does the bump switches have? What advantage does the accelerometer have? Would there be a situation that uses both?

## 21.7 Additional challenges

In this section, we list additional activities you could do to further explore the concepts of this module. You could extend the system or propose something completely different. For example,

- You could use the distance sensor to drive the robot autonomously around a track, like Figure 2A. Refer to Module 17 for more information on control systems.
- Read the data sheet on the BMI160 concerning shock events, and write software to detect collisions directly in the sensor. Configure the BMI160 to change INT1 on a collision, and configure edge-triggered interrupts on this pin. Stop the motor on collision. Compare the latency of this hardware approach to the software approach.
- You could use the accelerometer to determine if the robot is moving up or down a hill.
- You could use the temperature, humidity, and/or light sensor to provide functionality for the robot. Be careful to review pin conflicts requiring the removal of the bump switches.

- If you performed Lab 11 (LCD), then you could output data to the LCD, making it easier to debug, calibrate, and test.

## 21.8 Which modules are next?

This was our first of many uses of interrupts in this course. The following modules will build on this module:

Module 16) Interface tachometers to the microcontroller and use input capture to measure wheel velocity.
Module 17) Combine modules 12, 13 and 21 to develop closed loop motor controllers. In this module you will be able to drive the robot a constant distance from a wall, or have one robot follow another robot.

## 21.9 Things you should have learned

In this section, we review the important concepts you should have learned in this module how to:

- Use periodic interrupts to implement sampling
- Use the I2C to interface sensors to the microcontroller.
- Noise is difficult and important problem to solve; noise is often the limiting factor for resolution and not the number of bits in the digital sample
- Software can efficiently and effectively implement filtering
- Software can effectively handle a nonlinear transducer
- Accuracy depends on two processes: resolution and calibration. Resolution depends on noise, and calibration depends on the stability of the transducer.
- Sensor integration with 3 or more sensors to complete the robot challenge.

## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265