# TI-RSLK MAX

Texas Instruments Robotics System Learning Kit

Texas Instruments

# Module 15

## Lab 15: Data Acquisition Systems

# Lab: Data Acquisition Systems

## 15.0 Objectives

The purpose of this lab is to explore data acquisition systems, which are a combination of sensors, analog circuits, the ADC, and signal processing software, allowing the robot to collect data. See Figures 1A and 1B.

1. You will use the ADC to input data into the microcontroller.
2. You will use periodic interrupts to sample the ADC 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 evaluate the performance of the data acquisition system.

**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/sec. 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 decision making and outputs.

## 15.1 Getting Started

### 15.1.1 Software Starter Projects

Look at these three projects:
**ADCSWTrigger** (busy-wait ADC, simple digital filter, periodic interrupt sampling),
**Lab13_Timers** (your solution to Lab 13),
**Lab15_ADC** or **Lab15_ADC_Sound** (starter projects for this lab)

### 15.1.2 Student Resources (in datasheet directory)

MSP432P4xx Technical Reference Manual, ADC14 (SLAU356)
MSP432P401R Datasheet, msp432p401m.pdf (SLAS826)
GP2Y0A21YK0F_IR_Distance_Sensor.pdf, datasheet
CMA-4544PF-W.pdf, microphone datasheet
tlv9004.pdf, op amp datasheet
tpa731.pdf, audio amplifier datasheet
http://www.ti.com/lit/an/spra657/spra657.pdf

### 15.1.3 Reading Materials

Chapter 15, “Embedded Systems: Introduction to Robotics"

**Good to Know**: Analog to digital conversion is one of the most basic operations a microcontroller performs. ADC sampling requires us to make approximations in both the amplitude and time dimensions. The ADC on the MSP432 has 14 bits. However, noise is usually the limiting factor on most data acquisition systems, and not the number of bits in the ADC. Noise comes from external electromagnetic fields, from the sensor, or within the analog circuit interface.

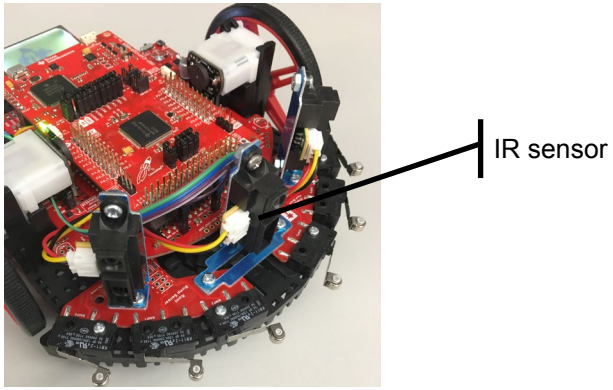


*Figure 1A. IR distance sensors positioned at the front of the robot (option A).*

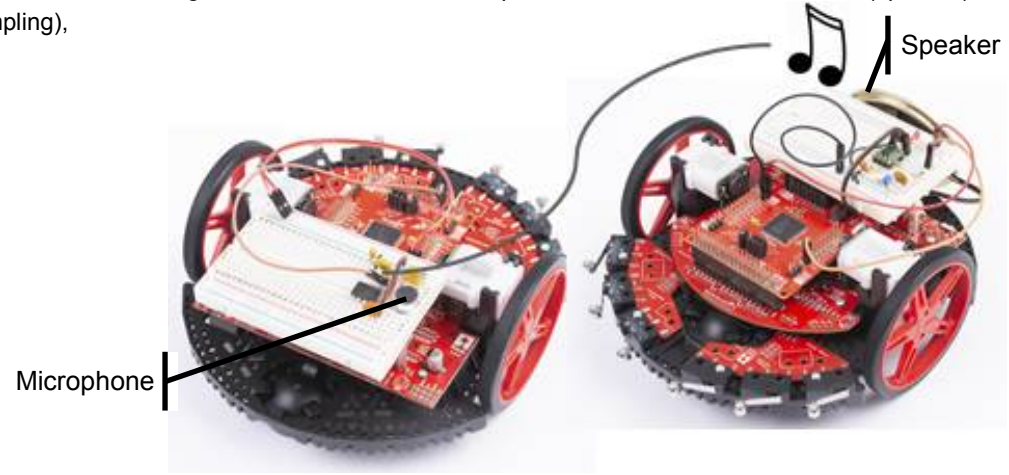


*Figure 1B. Sound out of one robot and into the other (option B).*

# Lab: Data Acquisition Systems

### 15.1.4 Components needed for this lab

All components needed to build the robot are provided in TI-RSLK MAX robot kit (TIRSLK-EVM), this lab does require additional components to be purchased for this portion of the lab. Batteries also are needed to power your robot.

| Quantity | Description | Manufacturer | Mfg P/N |
|---|---|---|---|
| 1 | TI-RSLK MAX robot kit | TI | TIRSLK-EVM |
| 1 | Sharp Distance Sensor Kit | Pololu | #3677 |

*Table 1A Parts needed for the IR distance version this lab (option A).*

| Quantity | Description | Manufacturer | Mfg P/N |
|---|---|---|---|
| 2 | TI-RSLK MAX robot kit | TI | TIRSLK-EVM |
| 1 | microphone | Digikey | #102-1721-ND |
| 1 | TLV9004 or TLC2272CP | TI TI | On robot board TLC2272CP |
| 1 | TPA731 | TI | TPA731D |
| 3 | 1k resistors | Yageo | CFR-12JB-52-1K |
| 2 | 10k resistors | Yageo | CFR-12JB-52-10K |
| 1 | 20k resistor | Yageo | CFR-12JB-52-20K |
| 2 | 22k resistors | Yageo | CFR-12JB-52-22K |
| 1 | 200k resistor | Yageo | CFR-12JB-52-200K |
| 3 | 100nF ceramic | Kemet | C320C104M5U5TA |
| 1 | 220nF ceramic | Kemet | C320C224M5U5TA |
| 1 | 1uF ceramic | TDK | FK18X5R1C105K |
| 1 | 2.2uF tantalum | AVX | TAP225K016SCS |
| 1 | 4.7uF tantalum | AVX | TAP475K016SCS |
| 1 | SOIC to DIP | Sparkfun | BOB-13655 |
| 1 | 8 by 1 male header | Sparkfun | PRT-00116 |

*Table 1B Parts needed for the audio version of this lab (option B).*

### 15.1.5 Lab equipment needed

Oscilloscope (one or two channels at least 10 kHz sampling)
Logic Analyzer (4 channels at least 10 kHz sampling
Optional: Spectrum Analyzer

## 15.2 System Design Requirements

### 15.2.1 Sensing distance to the wall using the distance sensor: Requirements (option A):

**The first goal** is to attach three GP2Y0A21YK0F IR distance sensors to the robot, and then interface the outputs of the sensors to inputs of the ADC converter on your MSP432 Launchpad. You will then convert raw ADC signal to

# Lab: Data Acquisition Systems

"distance" from the analog domain to the digital domain. Software plus calibration will allow the robot to measure distance to the wall. The range should be about 70 to 800 mm. The resolution should be about 1 mm at 200 mm.

Figure 2 shows the measured relationship between output of the IR sensor and the distance to the wall (use a block wood and a ruler). Notice the non-monotonic behavior of the sensor. For example, if the system records a sensor value of 2 V, it could mean 33 mm or 130 mm. During the robot challenge you will endeavor to keep the robot away from the wall, so you will assume the sensor distance is greater than 70 mm. Due to the nature of the robot challenges, we are not interested in distances beyond 800 mm.

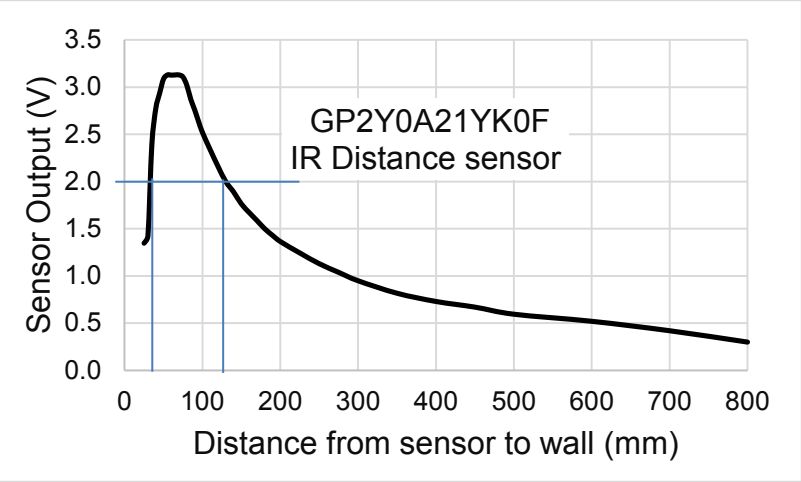


*Figure 2A. Typical sensor output as a function of distance (option A).*

You will develop a function that converts raw ADC samples into distance measured from the wall. Let **n** be a 14-bit sample from the ADC (0 to 16383), and **X** be the distance in mm from the sensor to the wall. The basic form of this nonlinear transfer relation is hyperbolic,

$$\mathbf{X} = A/(\mathbf{n} + B)$$

where A and B are calibration coefficients to be empirically determined.

The above equation defines distance from the sensor to the wall. Your system will however define all distances in mm from a common spot on the robot ($\mathbf{D_r}$, $\mathbf{D_c}$, $\mathbf{D_l}$), as illustrated in Figure 3. This common reference will allow the robot to calculate angle to the wall using geometry and two distance measurements. To handle this change of reference, we will introduce a third calibration coefficient. Define $\mathbf{n_r}$, $\mathbf{n_c}$, and $\mathbf{n_l}$ as the three ADC inputs. Let $A_r$, $A_c$, $A_l$, $B_r$, $B_c$, $B_l$, $C_r$, $C_c$, and $C_l$, be calibration coefficients.

$$\mathbf{D_r} = A_r/(\mathbf{n_r} + B_r) + C_r$$
$$\mathbf{D_c} = A_c/(\mathbf{n_c} + B_c) + C_c$$
$$\mathbf{D_l} = A_l/(\mathbf{n_l} + B_l) + C_l$$

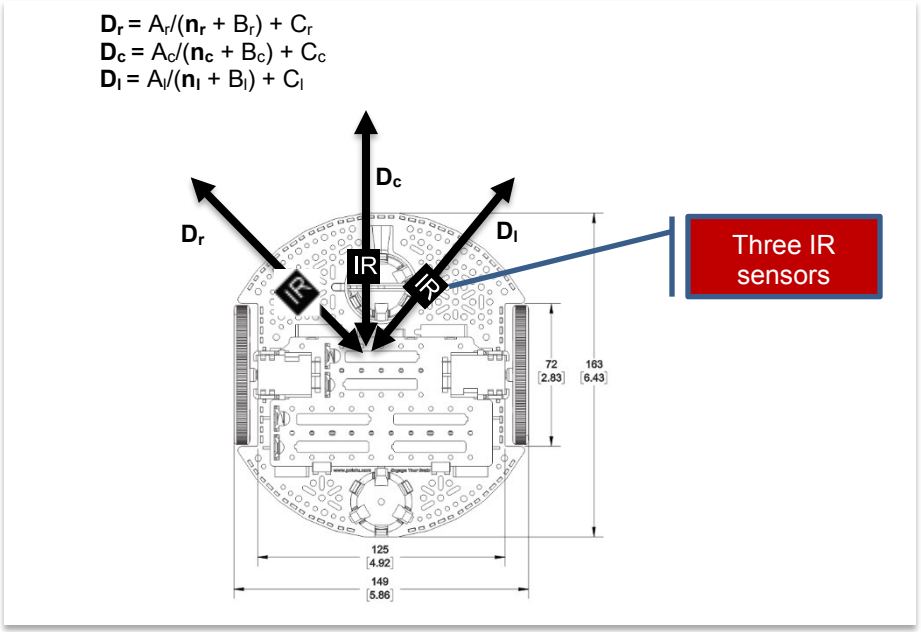


*Figure 3A. Distance defined from a central point on the robot (option B).*

The maximum measurement distance for the sensor is 800 mm, so if the ADC value is less than a certain threshold, your function should return 800+C. The C prototypes for your functions are

```
int32_t LeftConvert(int32_t nl);       // returns left distance in mm
int32_t CenterConvert(int32_t nc);     // returns center distance in mm
int32_t RightConvert(int32_t nr);      // returns right distance in mm
```

**The second goal** of this lab is to use periodic interrupts to **sample** the ADC. Triggering the ADC periodically is defined as **sampling**, allowing the system to process the data in both the time and frequency domains. For example, 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

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

for N = 1 to 512. 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.

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 on nonzero outputs.

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 than improves signal to noise ratio without causing too much delay in the step response.

**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

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

*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 ADC. For this lab, resolution will be limited by noise in the IR distance sensor.*

**15.2.2 Robot to robot audio communication channel**

**Requirements (option B):**

**The first goal** is to adapt the PWM output from Lab 9 to create two to five different sine-wave tones. The shape (sine wave) and amplitude (as large as possible) of each tone will be the same. The difference will be the frequency or pitch of the sound. Choose sine-wave frequencies within the range of about 200 to about 1000 Hz, so that the sound can be processes with low-cost speakers and microphones. You can use MSP432 LaunchPad switches or bump switches to activate the sounds. You will need two or more robots for this lab option. The first robot will be the master that generates the commands. Each tone represents a different command (e.g., forward, backward, left, or right). Figure 2B shows typical PWM outputs and the corresponding outputs of an analog low pass filter, creating the sine wave. For each wave, the digital PWM output has a fixed frequency of 20 kHz, but the duty cycle varies sinusoidally. For these examples, the 1175 Hz sine wave has 18 PWM pulses per period, and the 440 Hz sine wave has 45 PWM pulses per period.

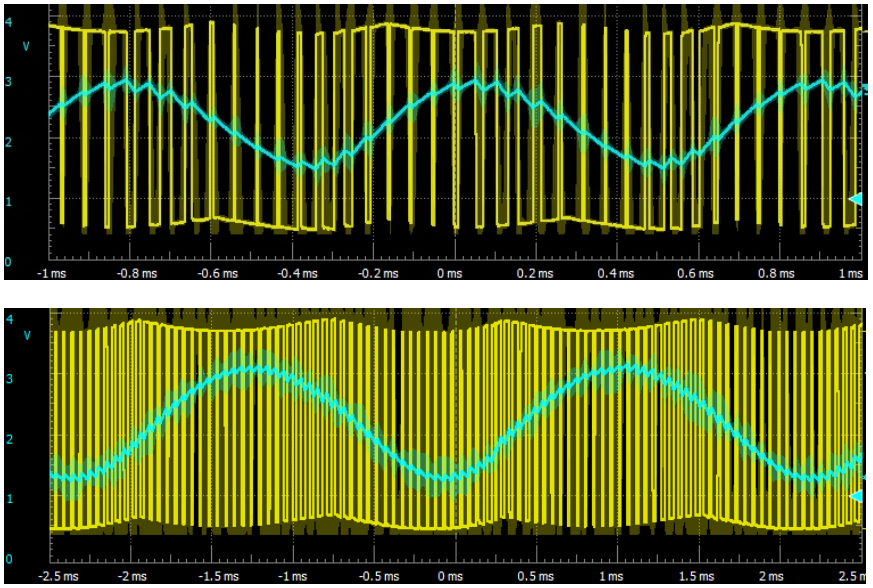


*Figure 2B. Typical outputs from the PWM DAC (option B).*

**The second goal** is to interface a microphone on the other robots. For this you will need one master and one or more slave, which work as team. These slave robots will accept commands from its master. Choose the gain and frequency response of the microphone analog circuit so the various tones from the master

are reliably passed. Figure 3B shows outputs of the microphone circuit measured on the slave for different tones originating from the master.

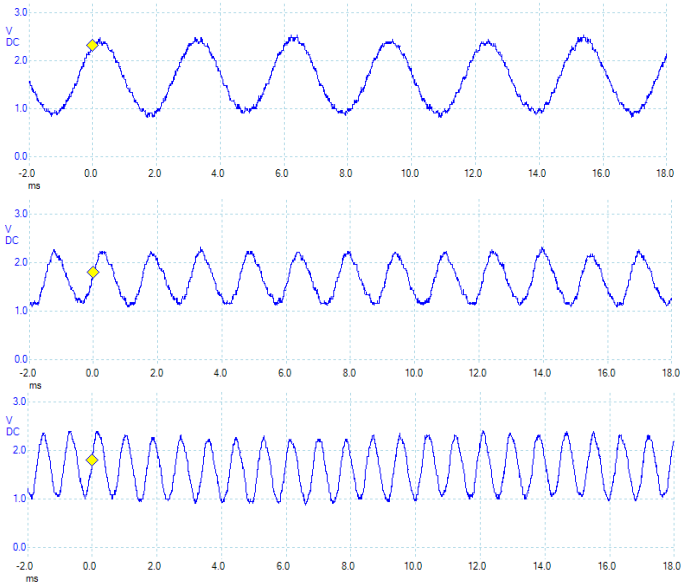


*Figure 3B. Typical outputs of the microphone circuit (option B).*

**The third goal** of this lab is to use periodic interrupts to **sample** the ADC. The output of the microphone circuit will be connected to the ADC input on the MSP432. Triggering the ADC periodically is defined as **sampling**, allowing the system to process the data in both the time and frequency domains. You must sample faster than the highest frequency tone used for the communication channel. Nyquist Theorem dictates the sampling frequency must be strictly larger than twice the maximum signal frequency. However, if the largest signal frequency is about 1000 Hz, we suggest sampling at 10,000 Hz.

**The fourth goal** of this lab is to study the signal to noise ratio (SNR) of the data acquisition system. The basic theory of this nonlinear filter is real sound signals are correlated to themselves, whereas noise is not. The filter calculates the autocorrelation of the input with itself shifted by two samples (200μs). If the signal shows high correlation, the gain is set to one (passes the signal). If the signal shows low correlation, the gain is close to zero (rejects the noise). You will tune this noise-reject nonlinear filter by adjusting parameters to improve SNR.

**The fifth goal** of this lab is to develop a signal processing that detects and classifies incoming commands. You will evaluate the performance of the communication channel by:

- Counting the number of commands properly received
- Counting the number of commands misclassified
- Counting the number of false positives (noise classified as signal)
- Measuring the time from the start of a command to its recognition

Use the commands to perform actions on the slave robot. E.g., forward, backward, left, or right.

## 15.3 Experiment set-up

Note: You must use the 0 to 3.3V ADC range and not use the precision internal voltage reference at 2.5 V.

### 15.3.1 Sensing distance to the wall using the distance sensor

You will attach three IR distance sensors near the front of the robot. See the construction guide for attaching the IR sensors.. Recall that the sensor is confused for distances from 0 to 70 mm (Figure 2A). It is preferable to recess the sensors away from the edge of the robot as much as possible. The robot in Figure 1 has the sensors recessed 25 mm from the point at which the bump sensors (used in Module 10) will activate. Therefore, this robot will report incorrect distances when 25 to 70 mm from a wall. The exact placement and angle will be readjusted as part of the robot challenge you attempt. For this lab, it is not critical exactly where the sensors are placed, see Figure 3A.

The interface circuit in Figure 4A shows one of the three analog low pass filters that exist on the TI-RSLK chassis board, refer to the schematics for the TI-RSLK chassis boards which could be used for the IR distance sensors. The inputs are labeled FINA FINB and FINC and the outputs are labeled FOUTA FOUTB and FOUTC. Each IR distance sensor requires four connections to be made, as illustrated in Figure 5A:

- Sensor power should be connected to +5V
- Sensor ground should be connected to ground
- Sensor output should be connected to a low pass filter input
- Low filter output should be connected to DISTL DISTC or DISTR

On the TI-RSLK chassis board, these signals are connected to ADC input pins

- DISTL is connected to P9.1 (ADC channel 16)
- DISTC is connected to P6.1 (ADC channel 14)
- DISTR is connected to P9.0 (ADC channel 17)

Any ADC input pins could have been used, but the issue with a complex design is assigning pins for special purposes like timer PWM, edge-triggered input,

UART, SPI and ADC. The three pins listed above do not conflict with the other labs in these modules. The sensor is very noisy, and this circuit will improve signal to noise ratio (SNR).

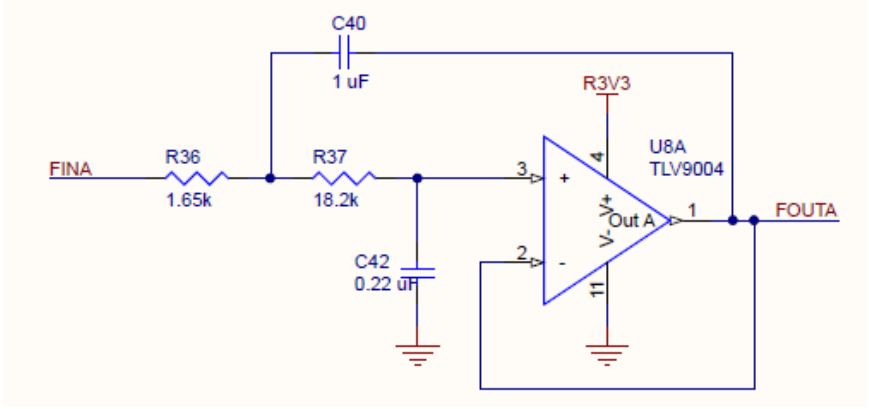


*Figure 4A. One possible interface circuit for the IR sensors (option A).*

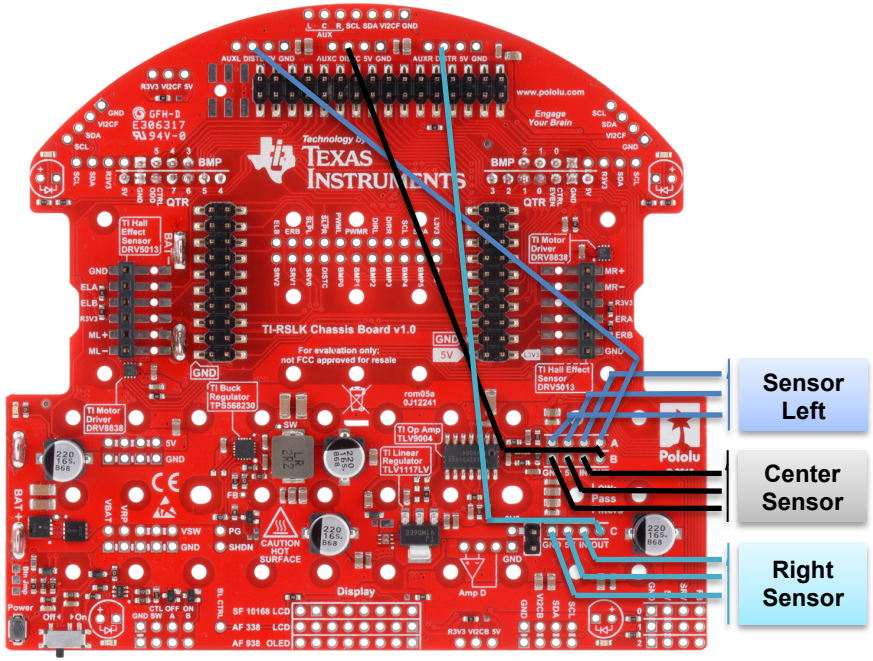


*Figure 5A. Wiring up the three IR sensors with filter (option A).*

**Warning**: Please ensure the +5V jumper on the MSP432 LaunchPad is disconnected or removed. Not removing this jumper will cause permanent damage to the LaunchPad and the TI-RSLK chassis board.

As an option, you could connect the sensor output directly to the ADC inputs. Even though the sensor is powered with 5 V, the output will only range from 0 to 3.1 V, see Figure 2A. Therefore, it is safe to connect this signal to the MSP432 powered at 3.3 V. **This approach will be simpler to wire but noisier.**

### 15.3.2 Robot to robot audio communication channel

One possible speaker amplifier circuit is shown in Figure 4B. The 1/(2πR7*C4) LPF cutoff is chosen to pass the audio frequencies 200 to 1000 Hz, but reject the 20 kHz PWM wave. The TPA731 audio amp can drive a standard 8-ohm speaker. C5 and C6 are high pass filters for the amplifier allowing the speaker output voltages to switch all the way from 0 to +5V. If you wish to use surface mount chips like the TPA731, you will need a SOIC to DIP adapter board.

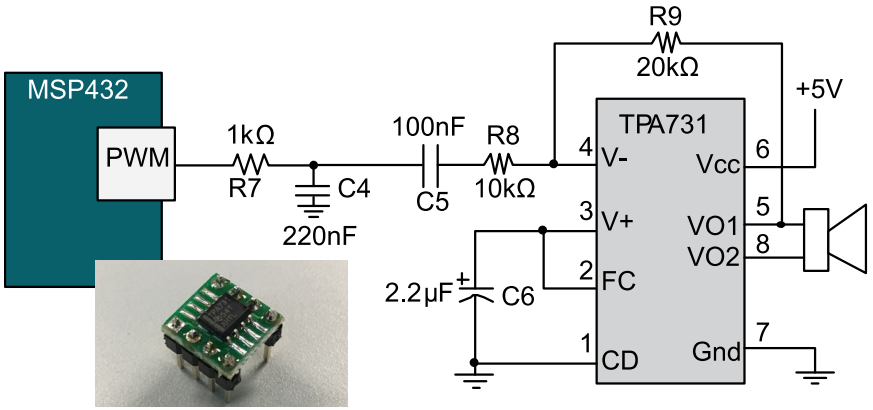


*Figure 4B. One possible speaker output circuit (option B).*

Second possible microphone input circuit is shown in Figure 5B. You can use the TLV9004 op amp on the interface board, or any rail to rail +3.3V op amp on the solderless breadboard, e.g., TLC2272. Adjust the gain and cutoff frequencies of this circuit to pass the specific tones generated by your master. See Figure 3.B.

# Lab: Data Acquisition Systems

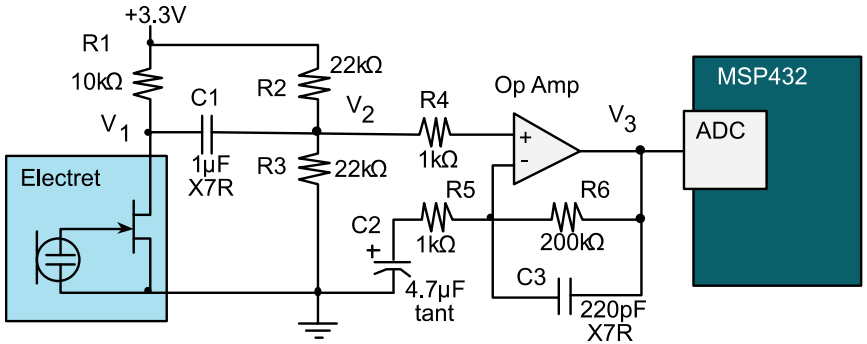


*Figure 5B. Second possible microphone input circuit (option B).*

## 15.4 System Development Plan (option A)

### 15.4.1 Explore the ADC, discover the Central Limit Theorem (both options)

In this section we will explore how the ADC works using TI's Launchpad. Use the 3.3V supply and two resistors to create a voltage **of about 3 V (any value from 2.5 to 3.1V)** to P4.7. P4.7 is ADC channel 6. Build, debug and run the project **ADCSWTrigger**. This project samples P4.7 at 1000 Hz using SysTick interrupts, implements a simple averaging digital filter, and performs statistical analysis on the collected data. In particular, it calculates a PMF (probability distribution of noise), mean (μ), range (max-min), variance ($\sigma^2$) and standard deviation (σ). If you run a terminal emulator like PuTTY or TExaSdisplay, you can see the output of this statistical analysis.

Note: Alternatively, if you have interfaced the LCD to your robot, then outputting these parameters to the LCD will simplify testing.

Place a voltmeter on the P4.7 input and observe the true voltage. Comparing the true voltage with the ADC digital output allows you to see how the ADC operates. Assuming there were no noise and the input were constant, you would expect all the ADC 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 voltage around 3 V, 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. In this project the value N is defined in the variable Size, varying from N = 1 to 512. By pressing and releasing either of the LaunchPad switches, the software cycles through these ten values of N. With the input voltage at about 3V, 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 lecture slides). This behavior will be even more pronounced when studying the noise from the IR distance sensor.

### 15.4.2 Study the digital filters (optional) (both options)

If you wish to learn more about the simple averaging filter, open the spreadsheet **FIR_Digital_LowPassFilter.xls** (you should have this in your folder when you opened up the zip file). 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 6A.

# Lab: Data Acquisition Systems

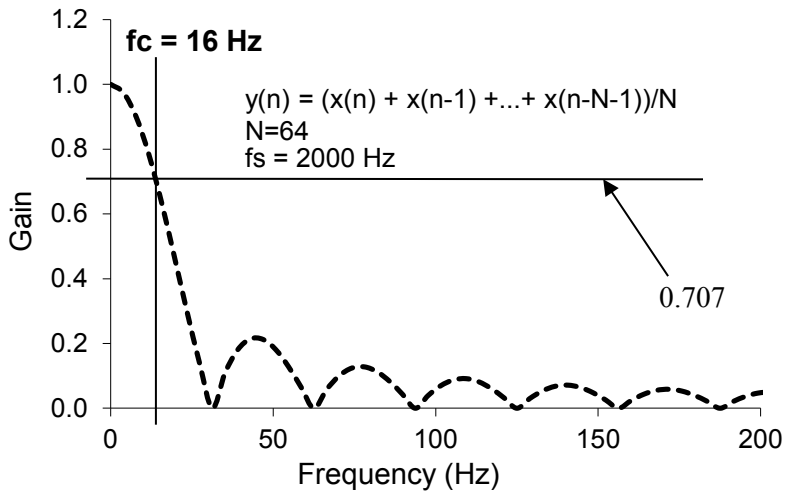


*Figure 6A. Frequency response of averaging filter with N=64.*

**Note**: You will not be able to complete this lab without reading the MSP432 data sheet. Look at the chapter on ADC14, and go line by line through the existing ADC0_InitSWTriggerCh6 and ADC_In6 functions. These two 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 convert it from sampling channel 6 to sampling a different channel.

**15.4.3 Low-level ADC software driver (option A)**

The first software step is to write two functions that sample the analog signal from the center IR distance sensor. The prototypes for these functions are

void ADC0_InitSWTriggerCh12(void); // initialize P4.1, channel A12
uint32_t ADC_In12(void); // sample P4.1, channel A12

Basically you will convert the initialization and sample function for channel 6 (P4.7) to channel 12 (P4.1), leaving all the design decisions the same, configured for the following:

- Single channel
- Software start
- Busy wait synchronization
- 14-bit, unsigned binary
- 3.3V V(R+), analog input range is 0 to 3.3V
- ADC14MEM0 address

To test these functions you can run Program15_1. The ISR samples channel 12, runs an averaging digital low pass filter, and passes the data through a mailbox (variable and semaphore) to the main thread.

```
void Program15_1_ISR(void){  // runs at 2000 Hz
  uint32_t RawADC;
  P1->OUT ^= 0x01;           // profile
  P1->OUT ^= 0x01;           // profile
  RawADC = ADC_In12();  // sample P4.1/channel 12
  ADCvalue = LPF_Calc(RawADC);
  ADCflag = 1;               // semaphore
  P1->OUT ^= 0x01;           // profile
}
```

**15.4.4 PWM DAC output software (option B)**

Build and test the analog circuits required to output sound from the PWM DAC (Figure 4B). You can use the solderless breadboard that comes with your TI-RSLK MAX robot kit. Write software that inputs from LaunchPad switches or bump switches and outputs sounds of different frequencies depending on which switch is pressed. Create two to five different sounds, one for each command.

**15.4.5 Low-level ADC software driver (option B)**

The first software step is to write two functions that sample the analog signal from the microphone. The prototypes for these functions are

void ADC0_InitSWTriggerCh23(void); // initialize P8.2, channel A23
uint32_t ADC_In23(void); // sample P8.2, channel A23

Basically you will convert the initialization and sample function for channel 6 (P4.7) to channel 23 (P8.2), leaving all the design decisions the same, configured for the following:

# Lab: Data Acquisition Systems

- Single channel
- Software start
- Busy wait synchronization
- 14-bit, unsigned binary
- 3.3V V(R+), analog input range is 0 to 3.3V
- ADC14MEM0 address

To test these functions you can run Program15_1. The ISR samples channel 23, removes the DC offset, and passes the data through a mailbox (variable and semaphore) to the main thread.

```
#define DC 8192
void Program15_1_ISR(void){  // runs at 10 kHz
  P1->OUT ^= 0x01;           // profile
  P1->OUT ^= 0x01;           // profile
  ADCvalue = ADC_In23()-DC;  // sample P8.2/channel 23
  ADCflag = 1;               // semaphore
  P1OUT ^= 0x01;             // profile
}
```

Adjust the **DC** constant so the ADC average is about 0. When connected to the microphone circuit.

**15.4.6 Signal to Noise Ratio of IR distance sensor (option A)**

To analyze sensor noise, you can use **Program15_1**, which is similar to the project **ADCSWTrigger**, replacing all the channel 6 accesses to channel 12. Run **Program15_1** and determine the SNR for various sizes of the averaging filter, from **N = 32 to 512.**

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

While deciding the size of the averaging, also consider the step response. In other words, let the input go from 0 to 1, and calculate the output of filter as a function of time, assuming a sampling rate of 2000 Hz. Calculate the time constant of the filter, defined as the time it takes the output to achieve 63% ($1-e^{-1}$) of the new value given a change in input. The time constant of the FIR filter is 20 ms, see Figure 7A. Select a value of N for your robot.

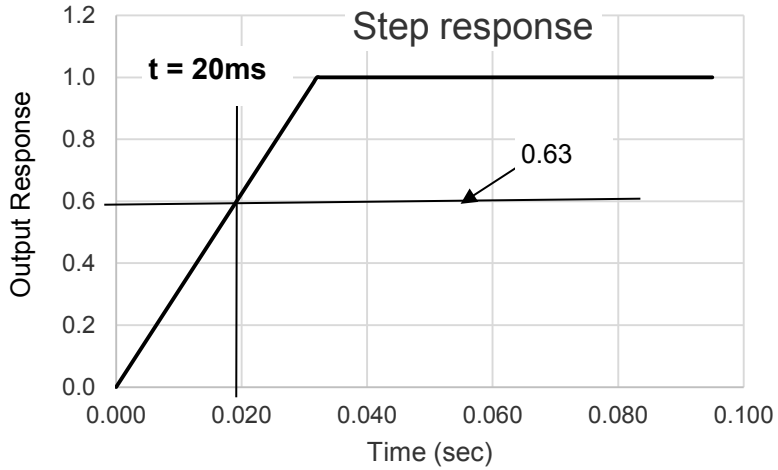


*Figure 7A. Step response of averaging filter with N= 64.*

**Note**: Since the time constant of the motor is about 100ms, we wish to have the time constant of the filter to be smaller than 100ms.

If you have access to a spectrum analyzer, it is very interesting to observe the noise in the frequency domain. Figure 8A illustrates there is significant noise throughout the frequencies, with peaks at multiples of 1 kHz. The output of a spectrum analyzer is given in decibels full scale. The spectrum analyzer in Figure 7 has a full scale of 5V, so

$$dB_{FS} = 20 \log_{10}(V/5)$$

The noise at 1 kHz, -35 dB, is the equivalent of about 90 mV. On the MSP432, a 90 mV noise will reduce the precision of the system from 14 bits possible with the ADC to $\log_2(3.3V/0.09V)$, which is about 5 bits. The purpose of the digital filter is to remove some of the noise, and improve precision.

# Lab: Data Acquisition Systems

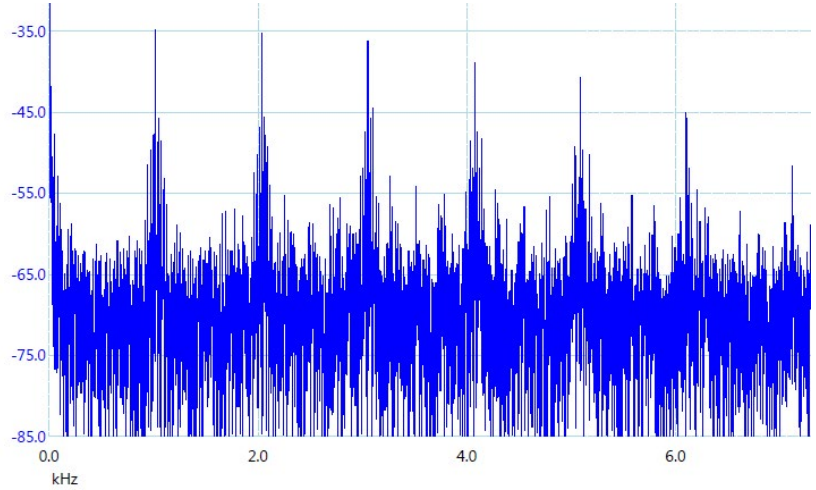


*Figure 8A. Frequency spectrum of the output of the distance sensor (without analog filter).*

**15.4.7 Signal to Noise Ratio of microphone input (option B)**

To analyze SNR for the microphone input, you will use **Program15_1** and **Program15_2** both of which sample channel 23 at 10 kHz. The system will be tested for *signal* using one of the tones created by the master robot. The system will be tested for *noise* with the room as quiet as possible. **Program15_1** has no digital filtering and **Program15_2** has an adaptive digital filter designed specifically for removing noise from electret microphones. The two programs collect NN samples and then perform statistics on the collected data. When measuring noise (silence), look at the shape of the histogram in an attempt to classify the type of noise. For SNR divide the **Sigma** with signal by the **Sigma** with noise (silence). Adjust following parameters in an attempt to improve SNR.

- Choose **M** so **Rxx2** goes to **M** with signal and to 0 with silence
- Choose **K** to define how fast it responds

**15.4.8 Three-channel ADC software driver (option A)**

The next software step is to write two functions that sample all three analog signals from the sensors. You will need to make three copies of the digital low pass filter, so all channels are filtered. The prototypes for your functions are

**void ADC0_InitSWTriggerCh17_12_16 (void);**
**void ADC_In17_12_16(uint32_t *ch17, uint32_t *ch12, uint32_t *ch16);**

Basically you will convert the initialization and function for channels 6 and 7 to channel 12 (P4.1), leaving most the design decisions the same

- Three channel sampling
- Software start
- Busy wait synchronization
- 14-bit, unsigned binary
- 3.3V V(R+), analog input range is 0 to 3.3V
- ADC14MEM2, ADC14MEM3, ADC14MEM4 addresses

**To test these functions you can run Program15_2. The ISR samples the three channels, runs three averaging digital low pass filters, and passes the data through a mailbox (variables and semaphore) to another thread.**

**Note**: Again, you must read the MSP432 data sheet when looking at the existing ADC0_InitSWTriggerCh67 and ADC_In67 functions. These two 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 convert it from sampling channels 6 and 7 to sampling channels 17, 12, and 16.

```
volatile uint32_t nr,nc,nl;
void Program15_2_ISR(void){  // runs at 2000 Hz
  uint32_t raw17,raw12,raw16;
  P1OUT ^= 0x01;          // profile
  P1OUT ^= 0x01;          // profile
  ADC_In17_12_16(&raw17,&raw12,&raw16);  // sample
  nr = LPF_Calc(raw17);  // right is channel 17 P9.0
  nc = LPF_Calc2(raw12); // center is channel 12, P4.1
  nl = LPF_Calc3(raw16); // left is channel 16, P9.1
  ADCflag = 1;           // semaphore
  P1OUT ^= 0x01;          // profile
}
int Program15_2(void){ // example program 15.2
  uint32_t raw17,raw12,raw16; int32_t n; uint32_t s;
  Clock_Init48MHz();
  ADCflag = 0; s = 256; // replace with your choice
  ADC0_InitSWTriggerCh17_12_16();   // initialize
  ADC_In17_12_16(&raw17,&raw12,&raw16);  // sample
  LPF_Init(raw17,s);      // P9.0/channel 17
  LPF_Init2(raw12,s);     // P4.1/channel 12
  LPF_Init3(raw16,s);     // P9.1/channel 16
  UART0_Init();           // initialize UART0
  LaunchPad_Init();
  TimerA1_Init(&Program15_2_ISR,250);  // 2000 Hz
```

# Lab: Data Acquisition Systems

```
  UART0_OutString("Program 15.2 \n");
  EnableInterrupts();
  while(1){
   for(n=0; n<2000; n++){
     while(ADCflag == 0){};
     ADCflag = 0; // show every 2000th point
   }
   UART0_OutUDec5(nl);UART0_OutUDec5(LeftConvert(nl));
   UART0_OutUDec5(nc);UART0_OutUDec5(CenterConvert(nc));
   UART0_OutUDec5(nr);UART0_OutUDec5(RightConvert(nr));
   UART0_OutChar('\n'); // once a second
  }
}
```

Connect a scope to P1.0 and measure the time between interrupts and the time within the ISR. Figure 8 shows this system (Size=256) executes 25 µs each time in the ISR. At 2 kHz, the ADC sampling and digital filtering consume 25/500 =0.05 (5%) of the processor time.

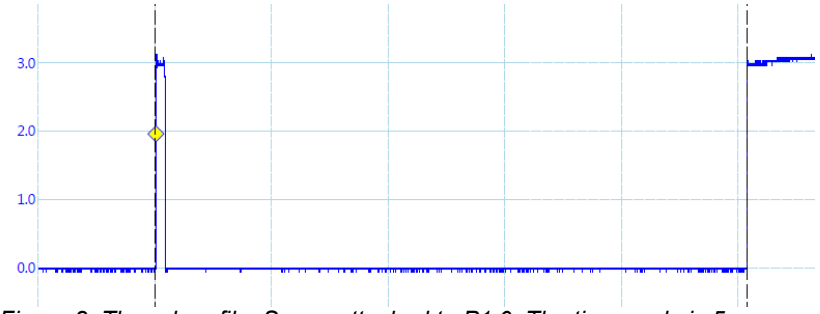


*Figure 8. Thread profile. Scope attached to P1.0. The time scale is 5 µs per division.*

**15.4.9 Calibration and accuracy (option A)**

Implement the three conversion functions, and use **Program15_2** first to calibrate and then to test these functions. Define exactly where on the robot you wish to set the reference point, see Figure 3. Use a ruler to measure the true distances ($\mathbf{D_r}$, $\mathbf{D_c}$, $\mathbf{D_l}$), and use **Program 15_2** to display the raw ADC values ($\mathbf{n_r}$, $\mathbf{n_c}$, and $\mathbf{n_l}$). Collect about 10 points for each sensor and fit the data to find the calibration coefficients for each sensor. Limit your calibration for distances between than 70 mm and 800 mm from the sensor.

Enter the coefficients into your software, and repeat the process collecting another set of 10 measurements for each sensor. Calculate the average accuracy of full scale in percent for each sensor.

**15.4.10 Command recognition (option B)**

Develop software in the slave robot that processes the **Sound[NN]** array in an attempt to classify which command has been received. The classification should return yes or no. If yes, it should also return which command has been received. First you will test it for false positives (FP), false negatives (FN), misclassifications (MIS), and true positives (TP). I.e., does it receive commands when none have been sent. FPs will naturally happen with background sound (people talking), and FNs will happen as the distance from the master to slave increases causing the amplitude of the recorded sound to decrease. Adjust the output sound and input classification until there are very few misclassifications. Calculate specificity, sensitivity, and positive predictive value:

- Sensitivity = TP / (TP + FN)
- Specificity = TN / (TN + FP)
- PPV = TP / (TP + FP)

Toggle a GPIO pin in the master robot whenever it begins a new command (i.e., it goes from a no command to a yes command).Toggle a GPIO pin in the slave robot whenever it recognizes a new command (i.e., it goes from a no command to a yes command). Using a dual trace scope, estimate the time from when the master starts a new command until the time the slave recognizes it. Knowing this delay will be useful if you attempt to use this audio communication channel in any robot challenges. Have the commands move the robot.

**15.4.11 Discovering the Nyquist Theorem (option A)**

The **Nyquist Theorem** states that if the signal is sampled with a frequency of $f_s$, then the digital samples only contain frequency components from 0 to ½ $f_s$. Conversely, if the analog signal does contain frequency components larger than ½ $f_s$, then there will be an aliasing error during the sampling process. Aliasing is when the digital signal appears to have a different frequency than the original analog signal.

Although the ADC is sampled at 2000 Hz, since **Program15_2** outputs once a second, the data observed on the terminal program can be considered to have been sampled at 1 Hz.

1) Observing one sensor output, oscillate the wall (block of wood) 100 to 200 mm from the sensor with a period of 4 to 10 seconds. Notice data tracks the signal in such a manner that you could reconstruct both the frequency and amplitude of the oscillations.

2) Very carefully attempt to oscillate the wall at a constant amplitude but with a frequency of 0.5 Hz (period of 2 sec). At this frequency the sampled data will oscillate 100,200,100,200,100… This is the Nyquist frequency (½ $f_s$) at which the system transitions from operational region (able to recover amplitude and frequency) to a region at which the digital data cannot be used to recover the amplitude and frequency of the oscillations.

3) Finally, oscillate the wall at a constant amplitude but with a frequency much faster than 0.5 Hz. Data existing at frequencies above ½ $f_s$ will be aliased (frequency folded into 0 to 0.5Hz), and the digital data cannot be used to recover the amplitude and frequency of the oscillations. The problem with aliasing is that high frequency noise will appear as low frequency signals. Therefore we must remove high amplitude signals at or above ½ $\mathbf{f_s}$ using an analog low pass filter.

### 15.4.12 Discovering the Nyquist Theorem (option B)

The **Nyquist Theorem** states that if the signal is sampled with a frequency of $f_s$, then the digital samples only contain frequency components from 0 to ½ $f_s$. Conversely, if the analog signal does contain frequency components larger than ½ $f_s$, then there will be an aliasing error during the sampling process. Aliasing is when the digital signal appears to have a different frequency than the original analog signal.

Set the master to continuously create a typical sound. Let **f** be the frequency of the tone being sent by the master. Running **Program15_2** in the slave, the ADC is sampled at 10 kHz. For the operational system, $f_s$ is much greater than 2**f**. Push Button 1 on the slave, so **Program15_2** prints out the entire **Sound[NN]** array. Capture this data into a spreadsheet (ctrl-C in TExaSdisplay and ctrl-V into Excel). Set the sampling so $f_s$ is approximately 2**f**, and collect/capture another data set. Set the sampling rate $f_s$ is less than 2**f**, and collect/capture a third data set. Plot the three sets of data on the same time axis.

## 15.5 Troubleshooting

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

- Using a voltmeter observe power and ground at each place in the circuit that requires connection. Don't check it one place, verify all power and ground connections. One loose wire will prevent the circuit from operating.
- Check the wiring and use a scope to checks along each stage of the circuit,

***ADC doesn't work:***

- Run the **ADCSWTrigger**. This project should properly sample channel 6
- Review the data sheet and double check each register accessed by your software

***ISR takes more than 25us to execute:***

- Notice that the option A implementation of the filter requires about 9us for one channel and 25 us for three channels. This is because each filter calculation requires one subtraction, one addition and one division.
- Make sure there are no loops in the ISR (except for the busy-wait in the ADC sampling).

***Statistical calculations are incorrect:***

- Look at the collected data in the arrays. Bad statistics may be a result of bad data.
- You can reduce the statistical array to N=8, and perform the statistical calculated by hand to check the software.
- Information in Sum2 can overflow if the data is very noisy.

## 15.6 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.

- What is the mathematic relationship between ADC input voltage and digital output number?
- What is the limiting factor in this system that restricts resolution and accuracy of the distance measurement?
- Why did the function **ADC_In17_12_16** use call by reference parameter passing?
- What happens as the size of the averaging filter is doubled?
- 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 ADCflag=0 mean? What does ADCflag=1 mean?
- What would it mean if the ADCflag were already 1 at the time the ISR is trying to set it to 1 again?

## 15.7 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,

- If you add three analog filters between the sensor output and ADC input (fc=100 Hz) you can greatly reduce the noise and could also reduce the sampling frequency and size of the digital filter.
- A median filter is an alternate digital filter than could be added to improve signal to noise ratio.
- If you performed Lab 11 (LCD), then you could output data to the LCD, making it easier to debug, calibrate, and test.

## 15.8 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 16 to develop closed loop motor controllers. In this module you will be able to spin the motors at a constant speed.

## 15.9 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 ADC to convert from analog to digital domain
- Noise is difficult and important problem to solve; with a constant voltage connected the LaunchPad, noise will limit the 14-bit ADC to 10 or 11 bits; noise is often the limiting factor for resolution and not the number of bits in the ADC
- Software can efficiently and effectively implement filtering
- Software can effectively handle a nonlinear (hyperbolic) transducer
- Accuracy depends on two processes: resolution and calibration. Resolution depends on noise, and calibration depends on the stability of the transducer.

# ti.com/rslk

## 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