

Ultrasonic Sensor Peer to Peer Detection and Rangefinder User Guide

TABLE OF CONTENTS

1. Scope and Purpose 3

2. Theory of Operation 4

3. System Requirements..... 6

 3.1. Hardware Requirements..... 6

 3.2. Hardware Improvement Recommendations 6

 3.3. Software Requirements 7

4. Building The Social Distancing Example Application 8

5. Programming the DK-CH101 SmartSonic Board 9

6. Running the Application 13

7. Firmware Configuration..... 14

8. Operating Instructions..... 15

 8.1. Sensor Mounting 15

 8.2. Sensor Position 15

Revision History 16

1. SCOPE AND PURPOSE

This user guide details the implementation of Chirp Microsystems’s Ultrasonic Peer to Peer Detection and Rangefinder solution using two CH101 SmartSonic Development Kits.

Augmenting our portfolio of general ultrasonic presence detection solutions, Chirp Microsystems offers a sensor peer to peer detection and rangefinder solution. Using a proprietary algorithm, two Chirp ultrasonic sensors can detect and establish communication between one another. Moreover, the two ultrasonic sensors can determine the associated distance between them with high accuracy.

This two-sensor solution enables a variety of customer use case scenarios, including rangefinding, presence/proximity sensing, object-detection/avoidance, and position-tracking. While there are many use cases for this sensor peer to peer detection and rangefinder solution, this user guide will concentrate on a Social Distancing example. Using Pitch-Catch mode operation, the distance between two users can be measured accurately, and any encroachments within the “safe distance” boundary can be flagged as an alert.

This document gives details on how to use Chirp’s SmartSonic development kit boards with CH101 sensors to measure distance and warn the users if two sensors are within the configurable boundary distance (2.4m maximum). These boards will toggle LED indicators when another user holding a CH101 development kit board approaches within the boundary.

The distance is calculated from the time-of-flight from one sensor to another, and an alert can be issued to a user to maintain a safe distance from another.

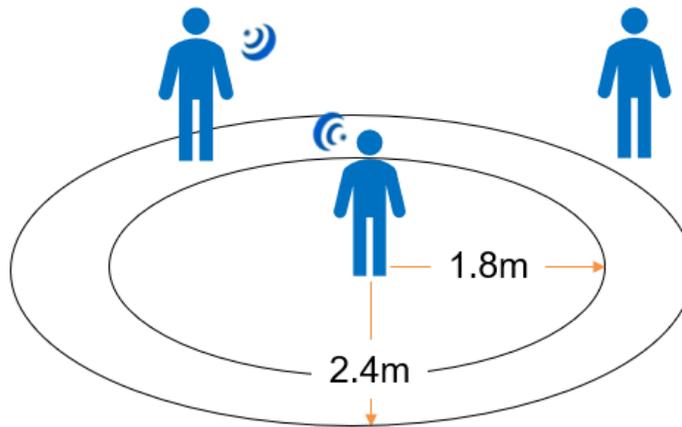


Figure 1. Social distancing detection

2. THEORY OF OPERATION

The Peer to Peer Detection and Rangefinder uses a special mechanism for independent sensors to discover each other. The algorithm executes on the development kit microprocessor.

The ultrasound synchronization of Peer to Peer Detection and Rangefinder uses a “Search” and “Reply” mechanism referred to as “ping-pong” operation. The operation is illustrated in Figure 2 and Figure 3 and operates as follows:

- Each DK-CH101 board operates in the “Pairing” mode continuously. In the Pairing period, the CH101 sensor:
 - Transmits “Search” pulse
 - Waits a reference delay of t_2 milliseconds
 - Executes a random number of receive cycles. In each cycle, the CH101 turns on its receiver for t_1 milliseconds.
- Upon receiving a Search pulse from the 1st CH101 sensor, the 2nd CH101 sensor:
 - Waits a reference delay of t_2 milliseconds
 - Transmits a Reply pulse (“ping”)
 - Waits a reference delay of t_2 milliseconds
 - Executes a receive cycle to receive a return Reply pulse (“pong”) from the 1st CH101 sensor
 - When this return Reply pulse is received, the 2nd CH101 sensor is able to report the range to the 1st sensor
- Upon receiving the Reply pulse from the 2nd CH101 sensor, the 1st CH101 sensor:
 - Is able to report the range to the 2nd CH101 sensor
 - Waits a reference delay of t_2 milliseconds
 - Transmits a return Reply pulse (“pong”)
- After the ping-pong ranging cycle is complete, the nodes will start a new pairing period, but with no Search pulse at the beginning, to avoid the two nodes performing ping-pong ranging forever.

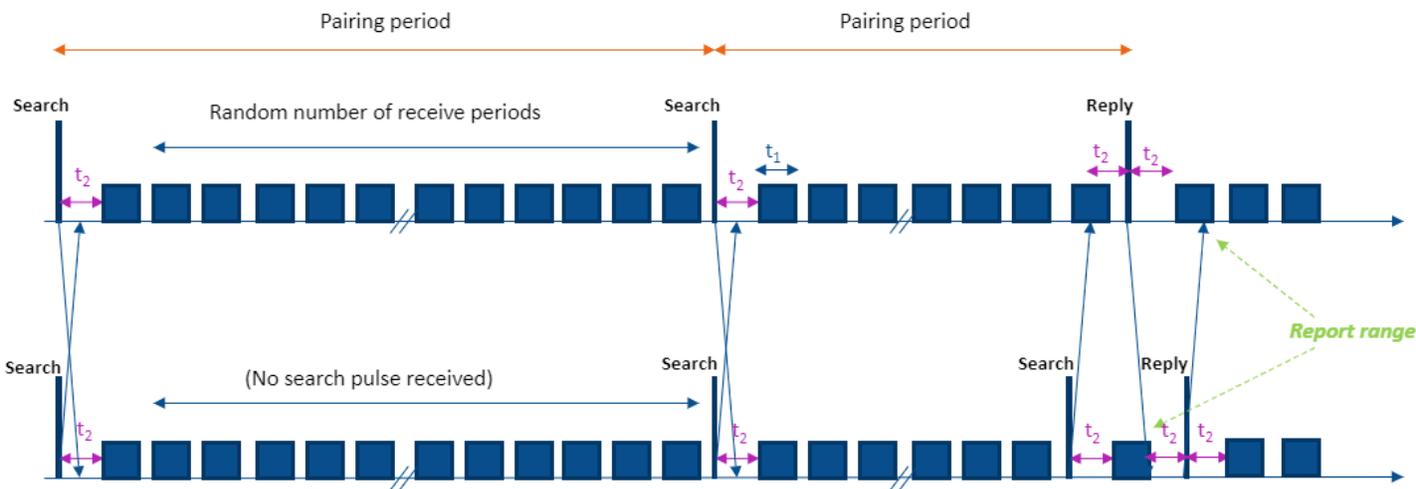


Figure 2. Pairing and ping-pong operation showing the timing for two CH101 sensors (top and bottom).

In the example in Figure 2, no Search pulses are received during the first pairing period due to random chance. In the second pairing period, the 1st CH101’s Search pulse is received by the 2nd CH101, and two Reply pulses are exchanged. When the CH101 receives the Reply pulse, it is able to report the inter-sensor range.

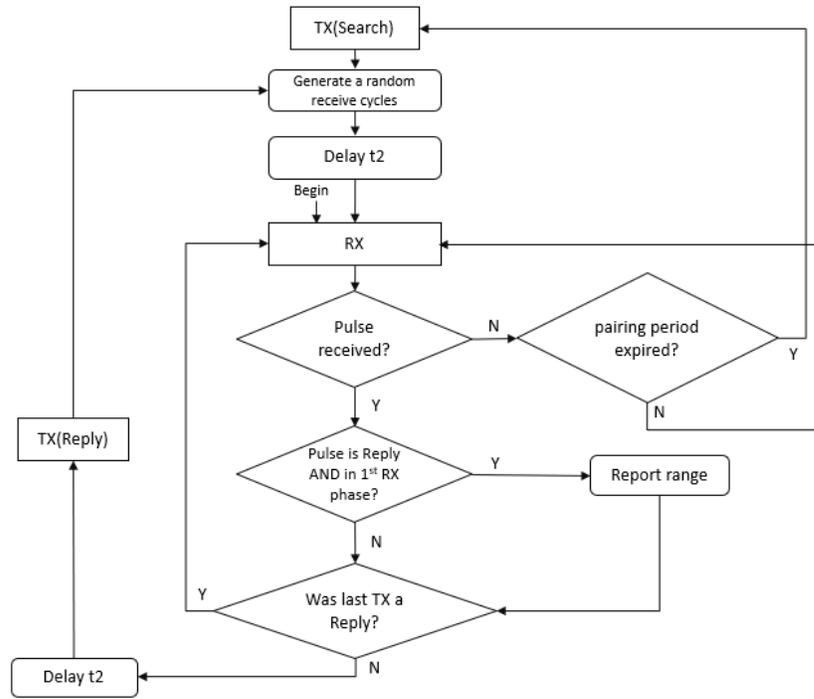


Figure 3. A flowchart that illustrates the pairing and ping-pong operation

3. SYSTEM REQUIREMENTS

3.1. HARDWARE REQUIREMENTS

- Two DK-CH101 SmartSonic evaluation boards



Figure 4. DK-CH101 SmartSonic evaluation board

- Two Micro-USB cables
- Two EV_MOD_CH101-03-01 sensor modules with flex cables
- Two flat plates (recommended)
 - To achieve the best acoustic performance, it is recommended to mount the module in a flat plate measuring 135 mm × 175 mm with a thickness of 1 mm.
 - Please refer to AN-000231 MOD_CH101 Evaluation Module Users Guide for the flat plate design.

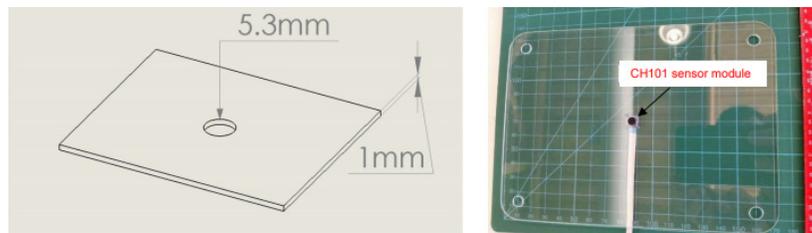


Figure 5. Flat plate measuring 135 mm × 175 mm

3.2. HARDWARE IMPROVEMENT RECOMMENDATIONS

To achieve the 8-foot maximum range, using the MOD_CH101-01-03 with a particle ingress filter and AH-10113-160035 acoustic housing is recommended. For more details, please refer to DS-000387 MOD_CH101-0x-03 Datasheet and AN-000223 Acoustic Horn Glue Procedure.

3.3. SOFTWARE REQUIREMENTS

- **SmartSonic_PeerToPeerRangefinder_vX_X_X.zip** (actual file name will include version number), includes:
 - The Peer to Peer Detection and Rangefinder application source files
 - Chirp SonicLib sensor API and driver files. The Ultrasonic Sensor Peer to Peer Detection and Rangefinder application requires SonicLib v2.1.3 or later.
 - Sensor firmware image files
 - Board support package files for Chirp DK-CH101 SmartSonic board
 - Atmel Studio 7 project files to build the application
- [Atmel Studio 7](#) integrated development environment – download from Microchip.com
- Terminal emulator of your choice (for example PuTTY or TeraTerm)

4. BUILDING THE SOCIAL DISTANCING EXAMPLE APPLICATION

- Open Atmel Studio 7
- Open the Peer to Peer Rangefinder project:
 - Open **File** menu
 - Select **File > Open > Project/Solution...**
 - Select the **atmelstudio/smartsonic-socialdistance-example/ smartsonic-socialdistance-example.atsln** file in the project directory.
 - Click **Open**. The program should locate the project files and display the name of the project.
- The Peer to Peer Rangefinder project files are organized in three sub-directories under **source**:
 - **application** – contains src and inc directories with the application
 - The **application/smartsonic-socialdistance-example/main.c** file contains the entry point for the application along with various routines that demonstrate how to read and manage the Chirp sensor(s). See the comments in that file for detailed information about the operation of the application.
 - The **application/smartsonic-socialdistance-example/inc/app_config.h** file specifies the overall measurement interval of receive cycle **MEASUREMENT_INTERVAL_MS**. The minimum value is 37 ms.
 - The **application/common/src/ultrasound_tag_finder.c** file contains the implementation of the Peer to Peer Detection and Rangefinder algorithm. The reference delay **REF_DELAY_MS** (t_2) described in Section 2 is defined here, with the minimum value 34 ms.
 - **board** – contains support files for the Chirp SmartSonic board.
 - The main board support package routines, as defined in **drivers/chirpmicro/inc/chirp_bsp.h**, are implemented in the **board/HAL/src/chbsp_chirp_samg55.c** file.
 - The **board/config/chirp_board_config.h** file is required by SonicLib. This file contains definitions for the number of possible devices and I²C buses on the board and is used for static allocation of arrays.
 - **drivers/chirpmicro** – contains the SonicLib API and driver files, sensor firmware modules, and other distribution files from Chirp.
 - The **drivers/chirpmicro/inc** directory contains header files that must be included when building applications with SonicLib. In particular, the **drivers/chirpmicro/inc/soniclib.h** file contains the key definitions for the SonicLib API.
- Build the project:
 - Select **Build > Build Solution**

The project should build successfully. The default build configuration is “Debug” so the build output files will be placed in the **atmelstudio/smartsonic-socialdistance-example/Debug** sub-directory. The user can select the **Release** build configuration for fewer output messages.

5. PROGRAMMING THE DK-CH101 SMARTSONIC BOARD

Set up each DK-CH101 board as follows.

- Set Jumper J1 based on the USB cable connection.

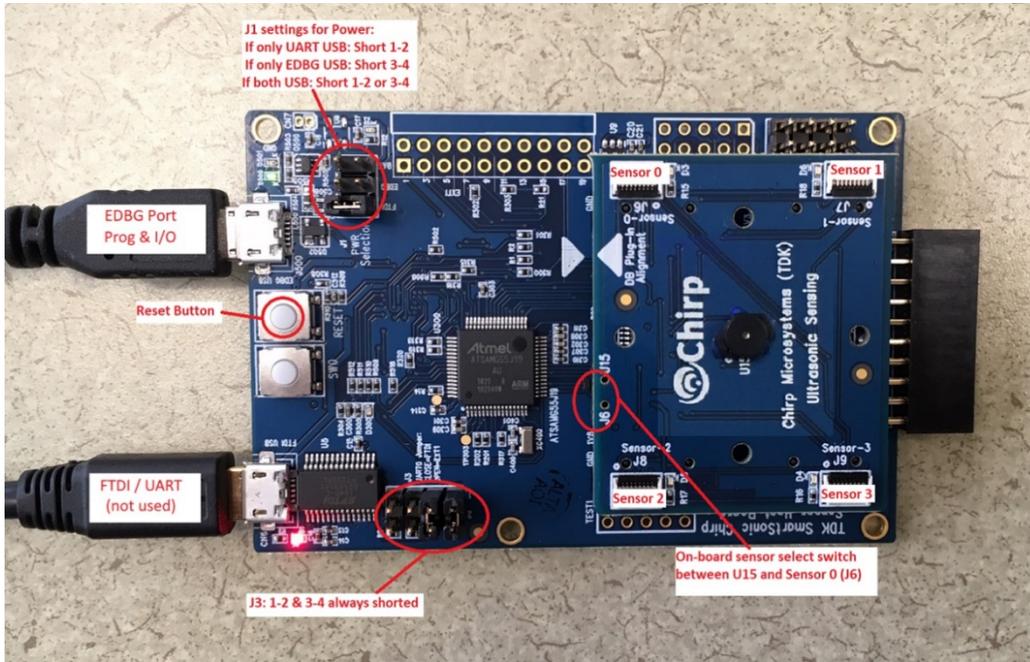


Figure 6. DK-CH101 SmartSonic board

- Connect the DK-CH101 board to a Windows PC with EDBG USB cable.
- Connect one of the MOD_CH101_03_01 sensor modules with a flex cable to J6. Switch the on-board sensor select switch to J6.

- In Atmel Studio 7 select **Tools > Device Programming**.
- The Device Programming screen will appear:

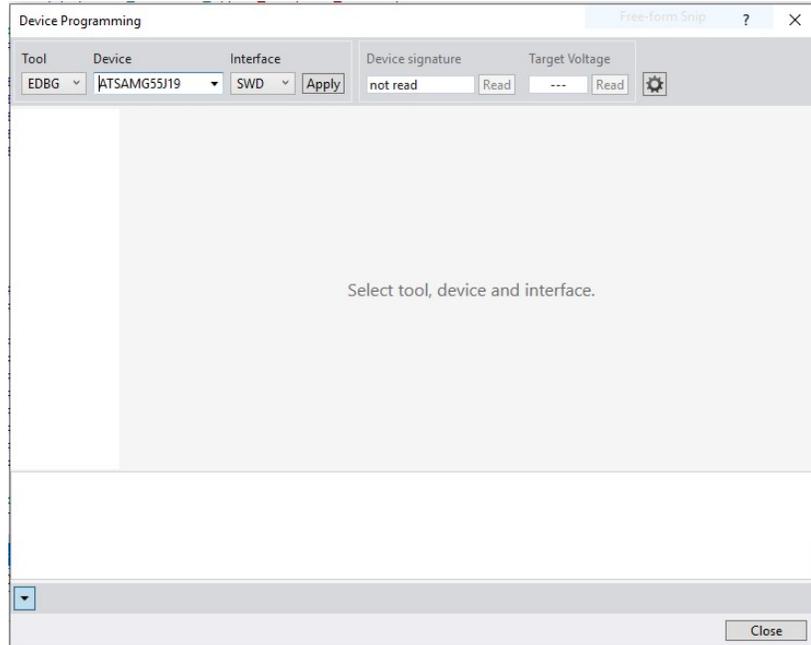


Figure 7. Device programming screen

- Verify that the tool is **EDBG**, device is **ATSAMG55J19**, and interface is **SWD**.
- Select **Apply**.
 - **Note:** Atmel Studio 7 may require you to update the EDBG debug interface firmware on the DK-CH101 board before continuing. Follow the on-screen instructions to update the EDBG firmware.
- The Device Programming screen will prompt to set the programming clock frequency. Leave the clock frequency unchanged (use the default).

- Select **Read** near the **Device signature** field. The device signature bytes should be read and should not generate any error messages. The Device programming menu should look like this:

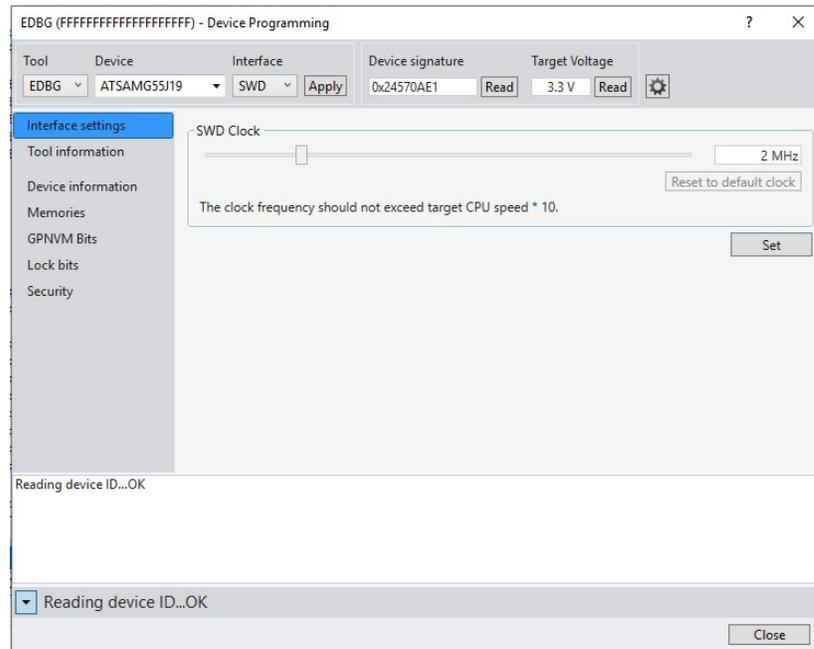


Figure 8. Device signature and target voltage

- Select **Memories** on the Device Programming menu.
- The Device Programming menu will prompt for the name of the file to program:

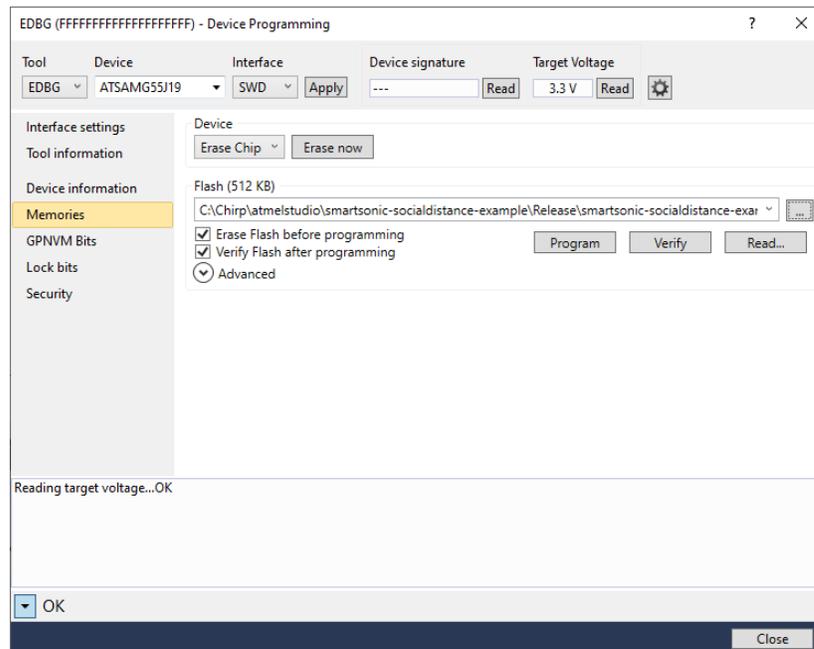


Figure 9. Programming Hex file

- From the Device Programming screen scroll to the project’s Release directory and select the **smartsonic-socialdistance-example.hex** file.
 - **Note:** Alternately, you may use the **smartsonic-socialdistance-example.elf** file, which contains symbolic debug information. (Both files are generated when you build the application. If you are simply running the Peer to Peer Range Finder application and do not need to use the Atmel Studio 7 debugging features, it does not matter which file you use).
- Select **Program**. Your DK-CH101 board is successfully programmed when the Device Programming screen displays the “OK” messages shown below on the bottom left:

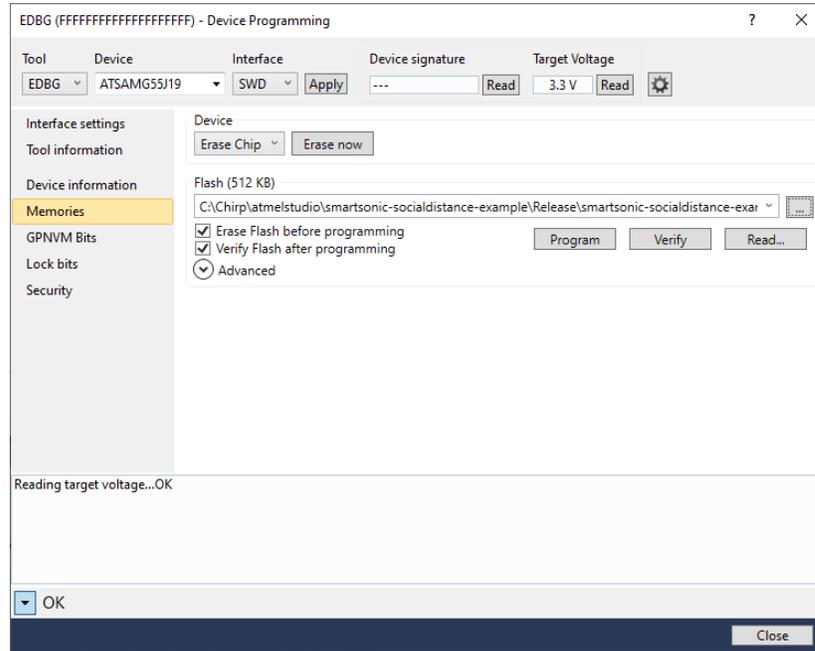


Figure 10. Successful programming

- Repeat the steps from the beginning for the second DK-CH101 board.

6. RUNNING THE APPLICATION

- Start the terminal emulator program and open/configure the COM ports assigned to the DK-CH101 board “EDBG”:
 - **1000000 baud**
 - **8 bits data, no parity, 1 stop bit**
 - **New-line sequence = Line Feed only (no carriage return)**
 - PuTTY: Open **Terminal** configuration. Select **Implicit CR in every LF**.
 - TeraTerm: Open **Setup > Terminal**. Under **New-line** set **Receive** to **LF**.
- Reset the DK-CH101 boards using the board’s reset button (next to the Programming EDBG connector).
- Status messages from the application will appear on the terminal output, followed by summary data from the sensor initialization (device frequency, etc.) and configuration (maximum range, etc.).
- If the two DK-CH101 boards pair successfully, the range (distance) and amplitude data from the sensor device(s) will then be output in a continuous loop. Along with each range measurement, there is the extra information of received pulse length and the number of I/Q samples.

```

COM18 - Tera Term VT
File Edit Setup Control Window Help

# Chirp Social Distance Example Application
#   Compile time: Aug  7 2020 10:34:28
#   Version: 1.0.0#   SonicLib version: 2.1.3

# Initializing sensor(s)... starting group... OK

# Sensor      Type      Freq      RTC Cal      B/W      Firmware
# 0          CH101    171898 Hz  2772@100ms  5316    gppc_13

# Initializing sample timer for 37ms interval... OK
# Configuring sensor(s)...
# Port 0, dco0=4990, dco1=6774, dcoest=160, freq=174428, targ= 175000
# Frequency error above 100Hz, searching for better match to 175000Hz
# dcoest=155, freq=176404
# dcoest=156, freq=176052
# dcoest=157, freq=175714
# *# dcoest=158, freq=175362
# *# dcoest=159, freq=175024
# dcoest=160, freq=174415
# dcoest=161, freq=174076
# dcoest=162, freq=173698
# dcoest=163, freq=173373
# dcoest=164, freq=172967
# dcoest=165, freq=172629
# Final setting dco=159, freq=175024
# Sensor 0:      max_range=2704mm      mode=TRIGGERED_RX_ONLY

# Starting measurements
##### Port 0: Range: 1521 mm Amplitude: 776 pl: 25 344 samples
##### Port 0: Range: 1521 mm Amplitude: 779 pl: 25 344 samples
##### Port 0: Range: 1522 mm Amplitude: 771 pl: 25 344 samples
##### Port 0: Range: 1520 mm Amplitude: 793 pl: 25 344 samples
##### Port 0: Range: 1520 mm Amplitude: 783 pl: 25 344 samples
##### Port 0: Range: 1522 mm Amplitude: 761 pl: 25 344 samples
##### Port 0: Range: 1521 mm Amplitude: 783 pl: 25 344 samples
##### Port 0: Range: 1522 mm Amplitude: 789 pl: 26 344 samples
##### Port 0: Range: 1522 mm Amplitude: 796 pl: 25 344 samples
##### Port 0: Range: 1521 mm Amplitude: 797 pl: 25 344 samples
##### Port 0: Range: 1521 mm Amplitude: 782 pl: 25 344 samples
    
```

Figure 11. Typical application output

7. FIRMWARE CONFIGURATION

The `application/smartsonic-socialdistance-example/inc/app_config.h` header file contains symbolic definitions for parameters that affect the application execution. You may change the following definition to adjust the application's settings.

- **CHIRP_SENSOR_MAX_FREQUENCY_HZ**
 - For optimized pitch-catch operation, transmitting and receiving CH101 should be operated close to the same frequency. The operating frequency should be chosen according to the CH101 part number. The default software release will be set to 178 kHz which is optimal for the CH101-03 sensors available through distributors. Contact Chirp for details.

- **CHIRP_SENSOR_MAX_RANGE_MM**
 - The default value of maximum range is set to 2700 mm to ensure detection at 2400 mm with plates mounted around the sensors. The users may change this value for a shorter range detection.
 - **Note:** The maximum pulse length used in the sensor firmware is 250 cycles. To receive the end of the pulse we need to add the maximum length of the pulse to the maximum range setting. The length of the pulse in millimeters is $\text{Length}_{\text{TX}} / f_{\text{op}} \times c / 2$, whereas f_{op} is the operating frequency of the sensor in kHz and c is the speed of sound (343 m/s).

For example, if the operating frequency of the sensor is 178 kHz, the offset is $250/178 \times 343 / 2 = 241$ mm
 - **Note:** The value of maximum range determines the t_1 described in Section 2, which is approximately $2700(\text{mm}) \times 2 / 343(\text{m/s}) = 15.7$ ms

- **WARNING_RANGE_MM**
 - If the range between tags is shorter than the safe range **WARNING_RANGE_MM**, the LED indicators will be toggled to warn the users. The default is 1830 mm (approximately 6 ft).

- **PAIRING_TIMEOUT_US**
 - If two tags lose pairing for more than this amount of time, the LED indicators will be off.

8. OPERATING INSTRUCTIONS

8.1. SENSOR MOUNTING

As discussed in Section 3.2, an 8-foot maximum range can be achieved with MOD_CH101-01-03 and AH-10113-160035 acoustic housing. To achieve the best acoustic performance with the off the shelf available EV_MOD_CH101-03-01 sensor module, it is recommended to mount the module in a flat plate. The plate should measure at least 135 mm × 175 mm, with a thickness of 1 mm. The photo below shows the connection of a DK-CH101 SmartSonic board with a sensor mounted in such a plate, using a flex cable.

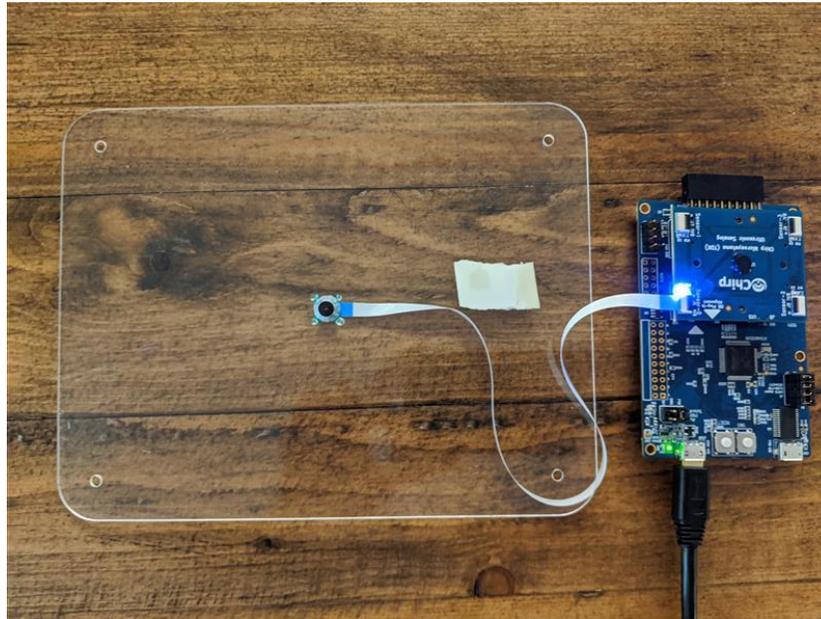


Figure 12. DK-CH101 and the plate mounted around the sensor

8.2. SENSOR POSITION

Placing the sensor housing on a person’s chest area, either the right or left, is recommended. The users can hold the plates with sensor mounted at a height of 1.3m to 1.5m.

The following photo shows two sensors mounted on fixed test stands to simulate a chest-mounted position.



Figure 13. Sensor position example

REVISION HISTORY

REVISION DATE	REVISION	DESCRIPTION
8/10/2020	1.0	Initial Release

This information furnished by Chirp Microsystems, Inc. (“Chirp Microsystems”) is believed to be accurate and reliable. However, no responsibility is assumed by Chirp Microsystems for its use, or for any infringements of patents or other rights of third parties that may result from its use. Specifications are subject to change without notice. Chirp Microsystems reserves the right to make changes to this product, including its circuits and software, in order to improve its design and/or performance, without prior notice. Chirp Microsystems makes no warranties, neither expressed nor implied, regarding the information and specifications contained in this document. Chirp Microsystems assumes no responsibility for any claims or damages arising from information contained in this document, or from the use of products and services detailed therein. This includes, but is not limited to, claims or damages based on the infringement of patents, copyrights, mask work and/or other intellectual property rights.

Certain intellectual property owned by Chirp Microsystems and described in this document is patent protected. No license is granted by implication or otherwise under any patent or patent rights of Chirp Microsystems. This publication supersedes and replaces all information previously supplied. Trademarks that are registered trademarks are the property of their respective companies. Chirp Microsystems sensors should not be used or sold in the development, storage, production or utilization of any conventional or mass-destructive weapons or for any other weapons or life threatening applications, as well as in any other life critical applications such as medical equipment, transportation, aerospace and nuclear instruments, undersea equipment, power plant equipment, disaster prevention and crime prevention equipment.

©2020 Chirp Microsystems. All rights reserved. Chirp Microsystems and the Chirp Microsystems logo are trademarks of Chirp Microsystems, Inc. The TDK logo is a trademark of TDK Corporation. Other company and product names may be trademarks of the respective companies with which they are associated.



©2020 Chirp Microsystems. All rights reserved.