Using Arduino Shield Boards on BL652
BL652 Series

Application Note

INTRODUCTION

Using an Arduino shield connector, it is possible to easily connect a 9-Axes Motion Shield to one BL652 development board and a LCD display with keypad shield to another BL652 development board and use it for application menu navigation/displaying application data received wirelessly via Bluetooth Low Energy without needing a connected PC. This application note guides you through setting up a demonstration system to send sensor data from one BL652 module to another and display it on an LCD screen.

REQUIREMENTS

- Two DVK-BL652 development kits
- PC with two spare USB ports (using a USB hub, if appropriate)
- LCD display and keypad shield (in our example, we use the SainSmart 16x2 LCD Display and Keypad Shield)
- 9-axes motion shield (in our example, we use the Arduino 9 Axes Motion Shield)
- UwTerminalX – available for Windows, Linux, and Mac: https://github.com/LairdCP/UwTerminalX/releases
- 2x 6-pin through header (for LCD Keypad shield)
- 2x 8-pin through header (for LCD Keypad shield)
- Soldering iron
- 6x AAA cell batteries (for use without USB cable)

Note: For the purposes of this document, we assume you have familiarised yourself with compiling/loading smartBASIC applications.

PRE-REQUISITES

Before using the add-on boards with the DVK-BL652 development kits, some modifications must be made to the 9-axes sensor shield using a soldering iron: remove the resistor on the D4 pin and place a solder bridge on the D3 and D2 pins of the shield (Figure 1).
FIRST BL652 SETUP (9-AXES MOTION SENSOR SERVER)

This BL652 is referred to as the server. This runs a smartBASIC application which advertises a custom service for reading the motion sensor data and is a peripheral device.

1. Place jumpers on the SIO_26 and SIO_27 headers bridging the pins to SCL_ARD and SDA_ARD to enable the I2C Arduino header pin interface (Figure 2).
2. Plug the 9-axes motion shield into the development kit header pins (Figure 3).

3. Next, connect the first BL652 development kit to your PC via the USB micro cable to the USB1 port on the left side of the development board (Figure 4). The power LED illuminates when the board is receiving power.
4. Open UwTerminalX. In the Config tab, set the parameters and COM port associated with your first development board.

![Port Settings]

Figure 5: Setting connection parameters

5. Click OK to advance to the Terminal tab.
6. Use UwTerminalX to return the BL652 to factory defaults using the command `at&f*` as shown (Figure 6). If you are using a new development board with the sample application, you may need to remove the autorun jumper on J12 and press the reset button to exit out of the sample application, and then issue the `at&f*` command to erase the file system and all non-volatile data.

![UwTerminalX (v1.01)]

Figure 6: Factory default

7. Load the Accel Server application – use the right-click menu to select XCompile + Load (Figure 7).

![XCompile]

Figure 7: Select XCompile + Load

8. Select the `$autormap.shield.accel.server.sb` file which was downloaded and extracted from the zipped file from Github or saved from the Github raw file page: https://raw.githubusercontent.com/LairdCP/BL652-Applications/master/Applications/$autormap.shield.accel.server.sb.

9. Wait for the Accel Server program to load; this should take approximately ten seconds.

The Accel Server application can now be run by typing `at+run "$autormap$"` followed by return.
HARDWARE SETUP (LCD SHIELD BL652)

This BL652 is referred to as the client. This runs a smartBASIC application which connects to the other BL652 and reads the sensor data which is then displayed on the LCD screen. This module operates in central mode. The application references specific pins, so it’s important to wire the shield as explained here. Otherwise, you’ll need to modify the application to use the pins you’ve wired.

1. Place a jumper between the 5V and A0 pins on J17 (Figure 8).

![Figure 8: Jumper between 5V and A0 (J17)](image)

2. Solder two 5-pin through headers and two 8-pin through headers to the LCD Display shield as shown in Figure 9.

![Figure 9: Soldered pins on LCD Display shield](image)

(Pins emerge through bottom of board)

These pins mate directly with the DVK-BL652 development board as shown in Figure 10.
3. Connect the second BL652 development kit to your PC via the USB micro cable to the USB1 port on the left side of the development board (Figure 11). The power LED illuminates when the board is receiving power.

4. Open another instance of UwTerminalX. In the Config tab, set the parameters and COM port associated with your second development board (Figure 12).
5. Click OK to advance to the Terminal tab.
6. Use UwTerminalX to return the BL652 to factory defaults using the command `at&f*` as shown (Figure 13). If you are using a new development board with the sample application, you may need to remove the autorun jumper on J12 and press the reset button to exit out of the sample application, and then issue the `at&f*` command to erase the file system and all non-volatile data.

![Figure 13: Factory default](image)

7. Load the Accel Client application – use the right-click menu to select XCompile + Load (Figure 14).

![Figure 14: Select XCompile + Load](image)

8. Select the `$autorun$.shield.accel.client.sb` file which was downloaded and extracted from the Zip file from Github or saved from the Github raw file page: https://raw.githubusercontent.com/LairdCP/BL652-Applications/master/Applications/$autorun$.shield.accel.client.sb.

9. Wait for the Accel Client program to load; this should take approximately 10 seconds. The Accel Client application can now be run by typing `at+run "$autorun$"` followed by return.

RUNNING THE DEMO

When the Accel Client application has been launched on the second DVK-BL652, status messages are displayed on the LCD screen. The application connects to the first server it finds named Laird Accel; the program can be controlled by using the buttons on the LCD shield (Figure 15):

- Select button – Enables notifications
- Left button – Disables notifications
- Reset button – Resets the BL652
- Other buttons – Display which button was pressed on the LCD
The display updates according to the application’s current status. At startup, the application searches for the server and connects (Figure 16). Once connected, if the correct service, characteristics, and descriptor is found, the Descriptor Found displays (Figure 17); the identity of which button is pressed also displays (Figure 18).

Once connected, press and hold the Select button to enable notifications (Figure 19) which will then be displayed on the LCD screen (Figure 20). Pressing the left button disables notifications (Figure 21). The readings show the X and Y positions on the top line of the LCD and the Z position on the bottom line of the LCD.

**RUNNING THE DEMO ON BATTERIES**

It’s possible to run the demo applications on both DVK-BL652 boards without USB connections to the computer by using six*AAA cell batteries (three per development board). Remove the USB cable from the development boards, flip them over, and insert the batteries according to the diagram shown. Some jumpers will need moving/fitting and switches moved as so (Figure 22) starting with the 9-Axes Motion Shield BL652 development board first:

- **J12 n_AutoRun** – Jumper moved to pins 2-3 (always enables autorun)
- **J27** – Jumper on pins (this holds the FTDI serial chip in reset, reducing the power consumption)
- **SW5** – Left position (3V3)
- **SW6** – Left position (3V3)
- **SW4** – Bottom position (DC_5V)
Once SW4 is moved to the bottom position, the power LED should illuminate showing that the module has power. If it does not illuminate, check the switch and jumper positions, ensure that the batteries are inserted correctly and that they have sufficient remaining power to power the development board.

After turning on the client BL652 it should operate as it did previously by connecting to the BL652 server and allowing notifications to be enabled and disabled using the buttons. If the display is not readable, the trim pot on the LCD shield might need adjusting with a small flathead screwdriver (Figure 23).

The development boards can be powered off at any time by moving the SW4 switch to the up position (USB_5V).

**ERROR HANDLING/DISPLAY**

By default, errors encountered whilst running the application are printed. In other sample applications, the output is printed to the UART; since there is no UART communication by default in this sample application, errors are instead printed to the LCD screen. The top line shows the four-character hex error code and the bottom line shows the line on which it occurred (Figure 24); this displays for two minutes.

In UwTerminalX v1.06 onwards, you can look up error codes and view their descriptions via the error code lookup utility. This can be used by opening UwTerminalX, accepting the terms, and clicking the Error Codes button (Figure 25). The error code can be typed and, upon pressing Enter, the error description displays at the
bottom of the window (Figure 26). If an error code lookup fails either due to the error code database not being found (Figure 27) or because the error code is not listed, then the error code database should be updated to the latest version. To update the error code database, click the Update tab in UwTerminalX and check if there is an update available by pressing the Check for Updates button in the left Error Codes box; if an update is available, it can be installed by using the Update error codes button (Figure 28). Once updated, the error code can be looked up from the error code dialogue without restarting UwTerminalX.

Figure 25: Location of error code utility

Figure 26: Looking up an error code

Figure 27: Error when error code database is not found

Figure 28: Updating the error code database
HOW THE APPLICATION WORKS

The server application acts as a BLE peripheral device with a GATT table characteristic for notifying clients the values of the accelerometer data, which is read from the 9-Axes Motion Shield via I²C. As there is no official Bluetooth SIG profile for this application, a 128-bit base UUID is used for the custom service and a sibling UUID is used for the characteristics (16-bit UUIDs are reserved by Bluetooth SIG for official profiles).

The client application comprises of a central mode application for connecting to the peripheral device and retrieving the motion data. The application itself supports outputting data to the UART or the LCD shield and supports being controlled by the LCD shield buttons or the development board buttons. By default, the output is limited to the LCD and control limit to the LCD shield buttons; this can be changed by altering the conditional compilation flag in the source of the application (the ‘#set $cmpif’ line). The LCD is put into a 4-bit operating mode instead of the usual 8-bit operating mode to reduce the number of I/O pins required to drive it. The buttons on the LCD shield are connected in a resistor ladder configuration to an analog-in pin; a potential divider function in the smartBASIC code calculates a range of values for each of the buttons based on the voltage at the pin at startup (as it differs if being run on full batteries, half-empty batteries, or USB) so it is important that the buttons are not held before the application has begun execution. The analogue in pin is checked on a recurring timer every 750 milliseconds and only performs an action if the button that is pressed is not the same as the previous button press. Upon starting, the BL652 starts scanning for devices and connects to the first device it detects with the name Laird Accel. It then begins searching for the service and descriptor and disconnects if either are not found.

REVISION HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Notes</th>
<th>Approver</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>15 Sep 2016</td>
<td>Initial Release</td>
<td>Jonathan Kaye</td>
</tr>
</tbody>
</table>

© Copyright 2016 Laird. All Rights Reserved. Patent pending. Any information furnished by Laird and its agents is believed to be accurate and reliable. All specifications are subject to change without notice. Responsibility for the use and application of Laird materials or products rests with the end user since Laird and its agents cannot be aware of all potential uses. Laird makes no warranties as to non-infringement nor as to the fitness, merchantability, or sustainability of any Laird materials or products for any specific or general uses. Laird, Laird Technologies, Inc., or any of its affiliates or agents shall not be liable for incidental or consequential damages of any kind. All Laird products are sold pursuant to the Laird Terms and Conditions of Sale in effect from time to time, a copy of which will be furnished upon request. When used as a tradename herein, Laird means Laird PLC or one or more subsidiaries of Laird PLC. Laird™, Laird Technologies™, corresponding logos, and other marks are trademarks or registered trademarks of Laird. Other marks may be the property of third parties. Nothing herein provides a license under any Laird or any third party intellectual property right.