Connecting an I2C GPS Module to the RM1xx

RM1xx Series

Application Note *v1.0*

INTRODUCTION

In this application note, we connect a GPS module to the RM1xx and use a *smart*BASIC script to process the GPS data. We have chosen to use the NEO-6 GPS receiver with an I2C-GPS NAV module. Most GPS receivers like this one communicate via UART. Since the RM1xx device has only a single UART, we use the I2C-GPS NAV module to communicate with the GPS receiver using I2C. This module parses the UART output from the GPS receiver and stores the data to be retrieved by the RM1xx using the I2C bus.

REQUIREMENTS

- Arduino IDE (for programming the I2C-GPS NAV module)
- I2C GPS NAV Source Code
- DVK-RM186 or DVK-RM191 development board
- FTDI USB to RS232 cable (more information may be found at FTDI's website)
- UwTerminalX, available from Laird (v1.03 or later recommended)
- *smart*BASIC sample applications *gps.sb* and *regdef.gps.net6.i2c.sb*, available at: https://github.com/LairdCP/RM1xx-Applications

INITIAL SETUP

Programming the I2C-GPS NAV Module

The GPS receiver comes preprogrammed and configured. The I2C-GPS NAV module must be programmed using the Arduino IDE and the following source code:

https://code.google.com/p/i2c-gps-nav/

You must modify the config.h file to specify the format of data coming from the GPS receiver, the UART baud rate, and I2C address for the slave device. In this case, we are using NMEA data coming from the GPS receiver at 9600 baud and the I2C address of 0x20. Do this by changing the values of `#define I2C_ADDRESS` and `#define GPS_SERIAL_SPEED` as shown in Figure 1.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0600

```
I2C_GPS_NAV | LeadFilter.cpp | LeadFilter.h | PICtrl.cpp | PICtrl.h | PIDCtrl.cpp | PIDCtrl.h | WireMW.cpp | WireMW.h | config.h | registe

//////////////////////////////////////////////////////////////////////////////
// i2C comm definitions
//
#define I2C_ADDRESS        0x20                    //7 bit address 0x40 write, 0x41 read

/* GPS Lead filter - predicts gps position based on the x/y speed. helps overcome the gps lag. */
#define GPS_LEAD_FILTER

/* Serial speed of the GPS */
#define GPS_SERIAL_SPEED 9600

/* GPS protocol
 * NMEA  - Standard NMEA protocol GGA, GSA and RMC  sentences are needed
 * UBLOX - U-Blox binary protocol, use the ublox config file (u-blox-config.ublox.txt) from the source tree
 * MTK   - MTK binary protocol with auto setup, load (AXN1.51_2722_3329_384.1151100.5.bin) firmware to the GPS module (MTK3329 chi
 * With MTK and UBLOX you don't have to use GPS_FILTERING in multiwii code !!!
 *
 */

#define NMEA
//#define UBLOX
//#define MTK
```

Figure 1: Modified Config.h file

Connect the FTDI USB to RS232 cable from the USB port of your PC to the I2C-GPS NAV module (wiring shown in Figure 2). Then select the Arduino Pro Mini in the Arduino IDE and upload the I2C-GPS NAV source code to the I2C-GPS NAV module.

Figure 2: FTDI USB-to-RS232 cable

Optionally, the RS232 cable can be used to verify the settings on the GPS receiver as well. Connect the Tx, Rx, Vcc and Gnd lines as shown in Figure 3.

Figure 3: Connected Tx, Rx, Vcc, and Gnd lines

Hardware Setup

In order to use the GPS with the RM1xx Development kit, you must connect the GPS to the I2C-GPS NAV module and then connect the I2C-GPS NAV module to the development kit board as shown in Figure 4 and Figure 5.

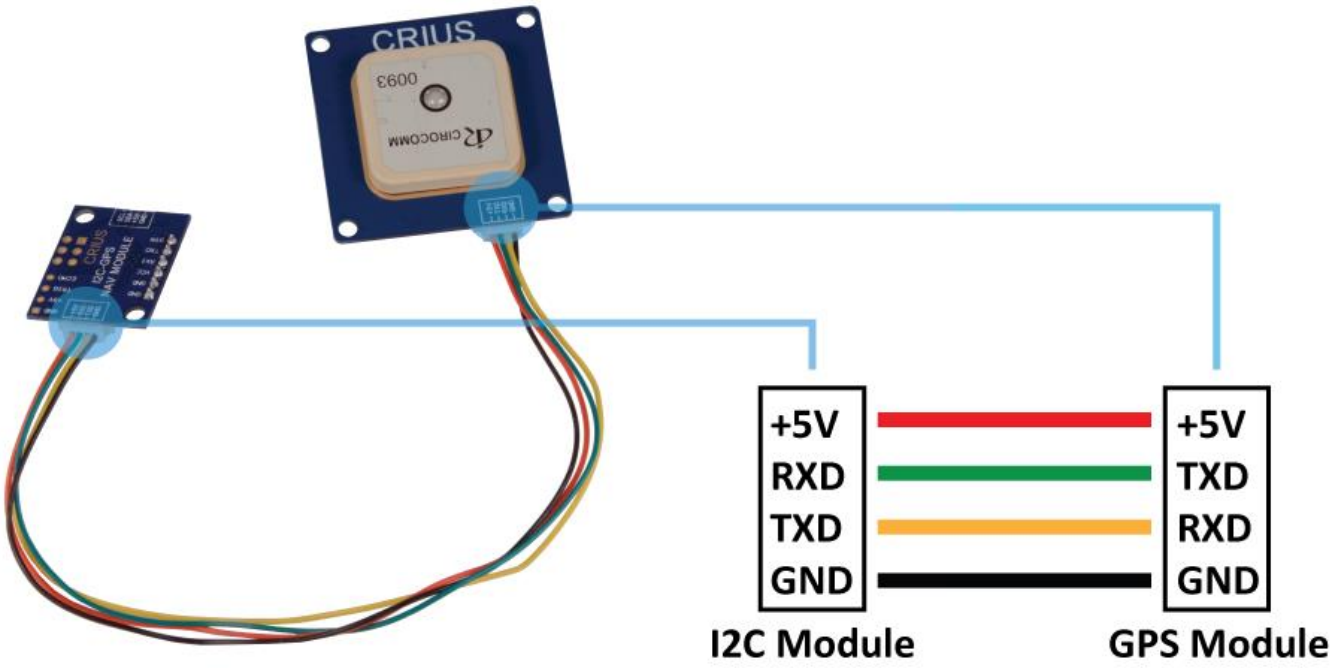

Figure 4: Connecting the GPS to the I2C-GPS NAV module

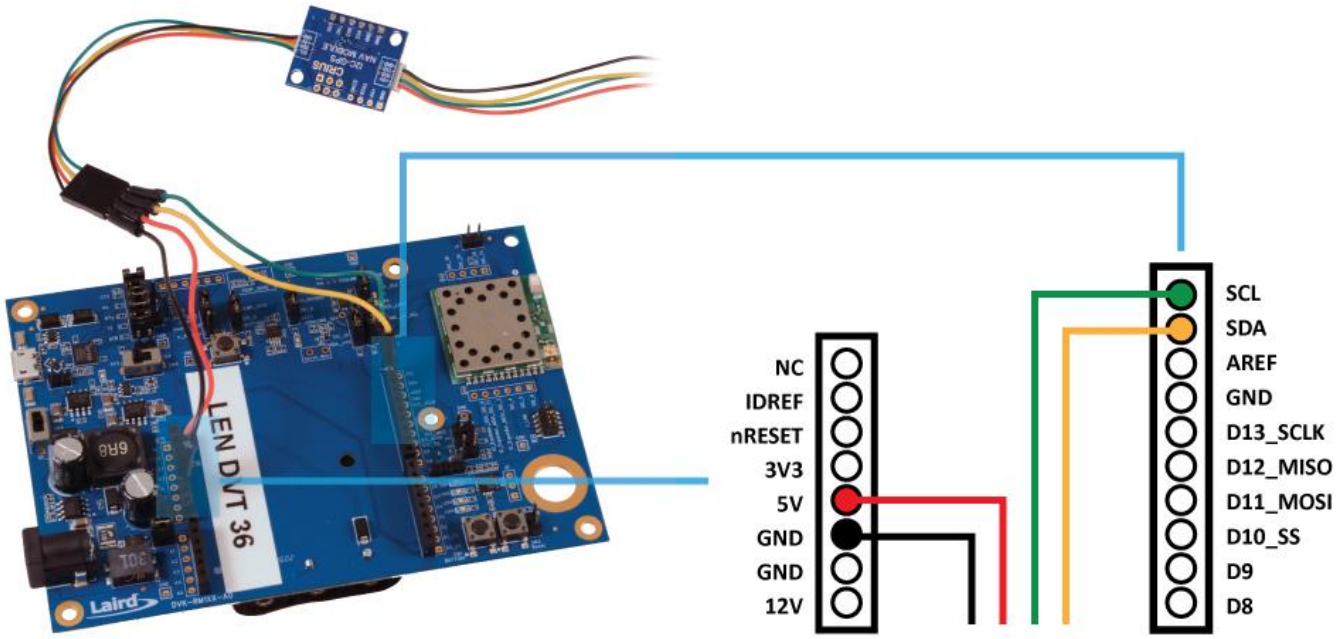

Figure 5: Connecting the I2C-GPS NAV module to the BT900 Development Kit

MONITORING GPS DATA

To monitor the serial data, we use the sample RM1xx I2C-GPS *smart*BASIC script included in the appendix of this application note. Once the script is running on the RM1xx, the application continues to search for GPS signals and acquires GPS data in an automated loop.

To do so, complete the following steps:

1. Download *gps.sb* and *regdef.gps.net6.i2c.sb* from https://github.com/LairdCP/RM1xx-Applications to a directory on your PC. They must be saved in the same directory.
2. Connect the RM1xx development board to your PC via the included USB micro cable.
3. Power your development board.
4. Launch UwTerminalX.
5. On the Update tab within the UwTerminalX pane, click **Check for Updates** to ensure you're using the latest version of UwTerminalX with support for the RM1xx Series.
6. In the Config tab from the Device drop-down menu, select either **RM186** or **RM191** based on your setup.
7. Select the correct port to which your development board is connected.
8. Click **OK** to proceed to the Terminal tab.
9. Hit **Enter** on your keyboard. If you see the return *00*, you are successfully connected.
10. Right-click in the terminal window; in the context menu, click **XCompile + Load.**
11. In the file selector window, select **gps.sb** and click **Open.**
12. When the terminal displays *00*, the compiler is successfully finished.
13. Type **at+dir** and press **Enter**. You should see *gps* in the file list.
14. To run the GPS script, type **gps** and press **Enter.**

Output from the GPS receiver should look as is shown in Figure 6:

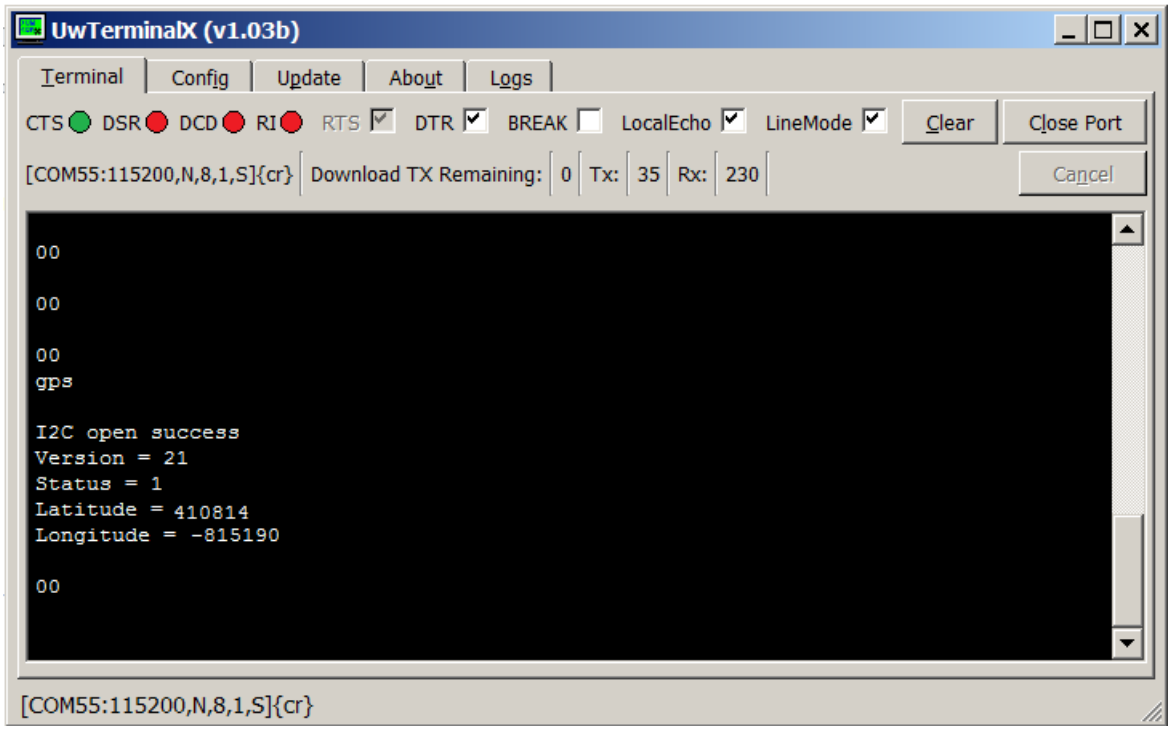

Figure 6: Output of GPS data

REVISION HISTORY

Version	Date	Notes	Approver
1.0	20 May 2016	Initial Release	Tim Carney