Gyroscope (BNO085) UART-RVC Guide

by David Choi, HiFen Kong and Eric Kempton

Introduction

This guide shows how to wire the gyroscope to the Beaglebone for the UART-RVC mode. It shows how to configure the Beaglebone to receive data from the gyroscope and how to configure the connection. Furthermore, it shows how to read and convert the data as needed.

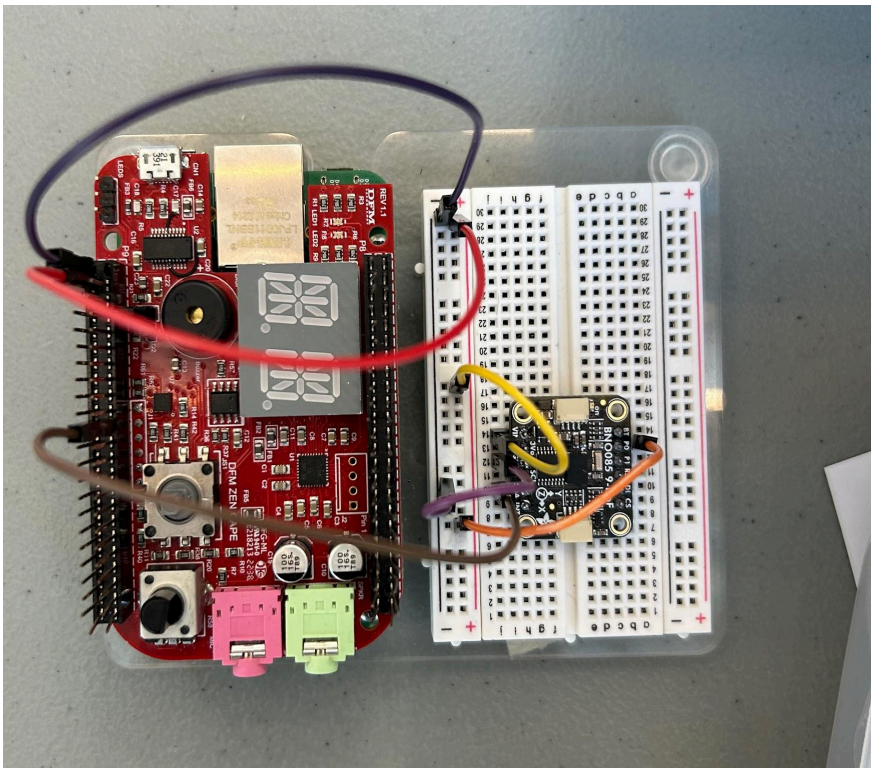
Table of Contents

1. **Wiring the Gyroscope to Beaglebone for UART-RVC** **2**
2. **Configuring the Beaglebone** **2**
3. **Configuring the Connection** **3**
4. **Reading and Converting the Data** **4**
5. **Troubleshooting** **5**
6. **References** **6**

Formatting:
Target (board) commands start with **(bbg)$**:
(bbg)$ echo "Hello!"

Wiring the Gyroscope to Beaglebone for UART-RVC

Place the gyroscope on the breadboard with the text facing the same direction as the red Zen cape.



Wire the BNO085 **VIN** to **P9.03 (DC 3.3V)**.

Wire the BNO085 **P0** to **P9.03 (DC 3.3V)**. This is to enable UART-RVC mode.

Wire the BNO085 **GND** to **P9.01 (GND)**.

Wire the BNO085 **RX** to **P9.26 (UART1 RXD)**. P9.22 (UART2 RXD) could also be used. However, it shares a pin with the buzzer so only one can be controlled.

Source: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-9-dof-orientation-imu-fusion-breakout-bno085.pdf

Configuring the Beaglebone

After wiring the gyroscope to the Beaglebone, connect to your virtual machine. Once powered on, a green light should light up on the gyroscope. (If P9.22 is used, the buzzer should buzz if connected properly). To configure the Beaglebone to receive data from the gyroscope, run the following command:

(bbg)$ config-pin P9_26 uart

or if P9.22 was used:

(bbg)$ config-pin P9_22 uart

To check if it is configured properly, print the data from the UART's device node with the following command:

(bbg)$ cat /dev/ttyS1

or if P9.22 was used:

(bbg)$ cat /dev/ttyS2

If values are printed to the screen, the gyroscope has been properly connected and configured.

Configuring the Connection

To configure the connection between the gyroscope and Beaglebone we must first open the serial port device and then configure the serial port. This can be done with the following C code:

```
// C library headers
#include <stdio.h>
#include <string.h>

// Linux headers
#include <fcntl.h> // Contains file controls like O_RDWR
#include <errno.h> // Error integer and strerror() function
#include <termios.h> // Contains POSIX terminal control definitions
#include <unistd.h> // write(), read(), close()

#define UART1 "/dev/ttyS1" // Or /dev/ttyS2 if P9.22 was used

int main (void) {
    static int uart_fd;
    struct termios tio;
        uart_fd = open(UART1, O_RDWR);
        if (uart_fd == -1)
        {
            perror("Error opening UART");
            exit(EXIT_FAILURE);
        }
    tcgetattr(uart_fd, &tio);
    cfsetispeed(&tio, B115200); // Set baud rate to 115200
    cfsetospeed(&tio, B115200); // Set baud rate to 115200
    tio.c_cflag |= (CLOCAL | CREAD);
```

```
    tio.c_cflag |= CREAD | CLOCAL;
    tio.c_lflag &= ~(ICANON | ECHO | ECHOE | ECHONL | ISIG);
    tio.c_oflag &= ~OPOST;
    tio.c_oflag &= ~ONLCR;
    tio.c_iflag &= ~(IXON | IXOFF | IXANY);
    tio.c_iflag &= ~(IGNBRK | BRKINT | PARMRK | ISTRIP | INLCR | IGNCR | ICRNL);
    tio.c_cc[VTIME] = 0;
    tio.c_cc[VMIN] = UART_FD_NUM_BYTES; // 19 bytes
    tio.c_cflag &= ~CSIZE;              // clear bit data
    tio.c_cflag |= CS8;                 // 8-bit data
    tio.c_cflag &= ~PARENB;             // No parity
    tio.c_cflag &= ~CSTOPB;             // 1 stop bit
    tcsetattr(uart_fd, TCSANOW, &tio);

    close(uart_fd); // Closing the connection
    return 0;
}
```

Source: https://blog.mbedded.ninja/programming/operating-systems/linux/linux-serial-ports-using-c-cpp/

Reading and Converting the Data

Once the serial port has been configured and opened, we can start to read and convert the data. This can be done with the following example code:

```
   while (1)
    {
        static double value;
        char buffer[19]; // Since each packet is 19 bytes
        memset(&buffer, '\0', sizeof(buffer));
        ssize_t bytes_read = read(uart_fd, buffer, 19);
        if (bytes_read > 0)
        {
            uint16_t roll_raw = (buffer[8] << 8) | buffer[7]; // Byte 8,9: Roll LSB,
MSB
            int16_t roll = (int16_t)roll_raw;
            value = (double)roll / 100; // Converts int to float
        }
        usleep(10000); // 10 ms for 100Hz
    }
```

This example code reads the data sent from the gyroscope at 100Hz (the rate BNO085 transmits data) and sets the variable value to the roll value converted into degrees. Use the following packet info to get specific values needed:

```
Byte 1,2  : Header
Byte 3    : Index
Byte 4,5  : Yaw LSB, MSB
Byte 6,7  : Pitch LSB, MSB
Byte 8,9  : Roll LSB, MSB
Byte 10,11: X-axis Accel LSB, MSB
Byte 12,13: Y-axis Accel LSB, MSB
Byte 14,15: Z-axis Accel LSB, MSB
Byte 16   : Motion Intent
Byte 17   : Motion Request
Byte 18   : Reserved
Byte 19   : Checksum
```

Source: https://www.ceva-ip.com/wp-content/uploads/2019/10/BNO080_085-Datasheet.pdf

Troubleshooting

- **If the green light is not turning on:** Make sure all the wires are in place.
- **If nothing is being printed from /dev/ttyS1:** Make sure all the wires are in place and check that BNO085 RX is connected to P9.26 (UART1 RXD).
  Or
  If nothing is being printed from /dev/ttyS2: Make sure all the wires are in place, check that BNO085 RX is connected to P9.22 (UART2 RXD), and listen for the buzzer.
- **If the data seems delayed:** Make sure you are reading at 100Hz, any delay in reading will make it seem like the data is delayed.
- **If you want to read the data slower than 100Hz:** Since the gyroscope constantly sends data at 100Hz once the connection is open, the data will start to stack up until it is read. Therefore, there might be a delay in processing the data unless it is read at the same rate. To prevent this, we can open the connection, read the data once, and then close the connection. Completely, stopping the data from stacking up. This can be done with the following code:

```
while (1)
 {
     // Open new connection
     struct termios tio;
     uart_fd = open(UART1, O_RDWR);
     if (uart_fd == -1)
     {
         perror("Error opening UART");
         exit(EXIT_FAILURE);
     }
     // Insert configuration here
     memset(&buffer, '\0', sizeof(buffer));
     ssize_t bytes_read = read(uart_fd, buffer, UART_FD_NUM_BYTES);
     if (bytes_read > 0)
     {
         uint16_t roll_raw = (buffer[8] << 8) | buffer[7];
         int16_t roll = (int16_t)roll_raw;
         value = (double)roll / INT_TO_FLOAT_CONVERTER;
     }
     // Close current connection
     close(uart_fd);
     usleep(LOOP_DELAY_IN_US); // Can be any rate < 100Hz
 }
```

This example code reads the gyroscope packet at a rate lower than 100Hz and converts it into roll data.

References

1. https://cdn-learn.adafruit.com/downloads/pdf/adafruit-9-dof-orientation-imu-fusion-breakout-bno085.pdf
2. https://www.ceva-ip.com/wp-content/uploads/2019/10/BNO080_085-Datasheet.pdf
3. https://blog.mbedded.ninja/programming/operating-systems/linux/linux-serial-ports-using-c-cpp/