Adafruit Mini GPS PA1010D UART Guide On BYA

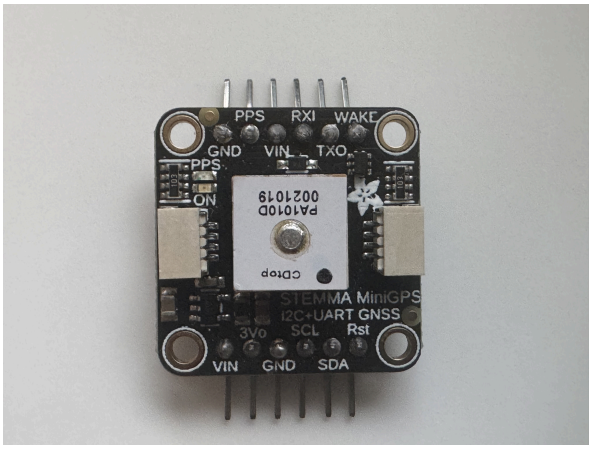
The guide has been tested on
BeagleY-AI (Target): Debian 12.8
VM (host): Debian 12.8
Tested on Linux Kernel 6.1

Preface

You may find other guides that explain how to use this GPS module, but most of them are designed for BeagleBone Green. This guide is specifically designed to show you how to install the Adafruit Mini GPS PA1010D on the **BeagleY-AI** using the **UART header**. Moreover, the method of fetching GPS data here differs slightly from the method introduced in the previous guide. This guide will walk you through all essential processes step by step.

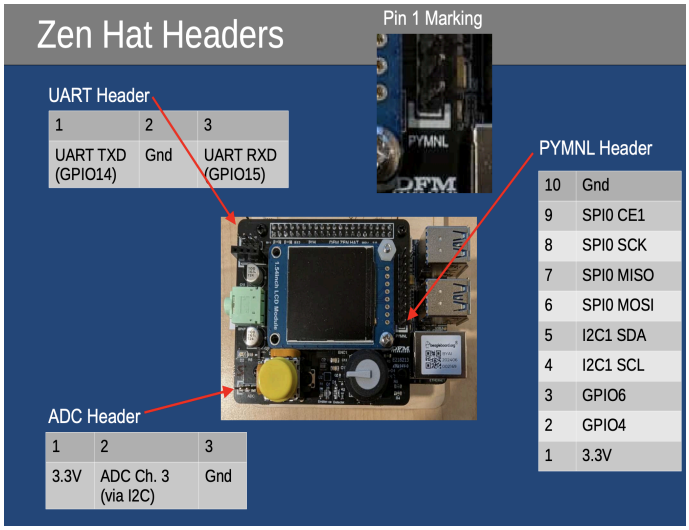
Wire the GPS module with the UART Header

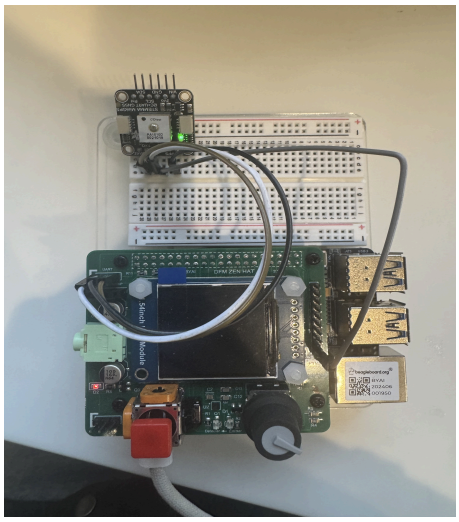
In order to wire up the GPS, you will need 1 jumper and 3 “female/male” jumper.

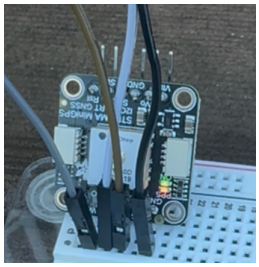


1. Place the module onto the breadboard.
2. **Connect the GND pin** on the module to a **GND pin** on your board (for example, the GND pin on the UART header) using a **jumper**.
3. Use a **male-to-female jumper** to connect the **VIN pin** on the module to a **3.3V or 5V power pin** (either work). For example, you can use **PYMNL pin 1**.
4. Use a **male-to-female jumper** to connect the **RXI pin** on the module to the **TXD pin** on the UART header.
5. Use a **male-to-female jumper** to connect the **TXO pin** on the module to the **RXD pin** on the UART header.

The header info in BeagleY-AI(from the lecture) and the sample GPS module installed image.







If you've wired the module correctly, you'll see a **solid green LED** on the module, indicating that it's powered on. Once the module gets a satellite fix, a **blinking red LED** will appear under the green LED, showing that it's actively receiving GPS location data.

The above content is adapted from another guide that detailed wiring on older Beagle hardware, though the detailed connected pin here has some slight differences.

Load the UART Overlay File

The UART header the GPS module connects with will stream the GPS data to the folder `/dev/ttyAMA0.` To allow us to read the data in that folder, we found that it is necessary to load an overlay file for this UART port based on this guide. Here are the commands we did to enable it:

Check needed overlay file exists in your board:
ls /boot/firmware/overlays/k3-am67a-beagley-ai-uart-ttyama0.dtbo
If it does not exist in your board, follow this guide to add the overlay file.

Edit the config file:
sudo nano /boot/firmware/extlinux/extlinux.conf

Edit the last section to load the Overlay file for ttyAMA0

```
label microSD (default)
    kernel /Image
    append console=ttyS2,115200n8 root=/dev/mmcblk1p3 ro rootfstype=ext4 resume=/dev/mmcblk1p2 rootwait net.ifnames=0 quiet
    fdtdir /
    fdt /ti/k3-am67a-beagley-ai.dtb
    fdtoverlays /overlays/k3-am67a-beagley-ai-uart-ttyama0.dtbo
    initrd /initrd.img
```

Reboot the board and check whether the data is showing in /dev/ttyAMA0.

Reading the GPS Data from the UART port with C Code

If you have completed the wiring and overlay configuration correctly, the GPS module should begin streaming data to `/dev/ttyAMA0`.

To check the output, run the following command in the terminal:

```
cat /dev/ttyAMA0
```

You should see output that looks similar to the following:

```
$GPGSV,3,3,10,09,03,017,,34,,,43
$GLGSV,1,1,0065
$GNRMC,224927.000,A,4915.7960,N,12249.6407,W,28.35,272.35,250325,,,A5C
$GNVTG,272.35,T,,M,28.35,N,52.53,K,A2F
$GNGGA,224928.000,4915.7967,N,12249.6533,W,1,5,1.46,171.9,M,-16.7,M,,7B
$GPGSA,A,3,20,05,31,25,29,,,,,,,,1.74,1.46,0.9300
```

Here are some key message types you might find useful:

- **$GNGGA**: Provides core GPS information, including latitude, and longitude.
- **$GNRMC**: Offers essential navigation data such as current speed, heading (direction), timestamp, and date and latitude, longitude.
- **$GPGSA**: Shows how many satellites are visible for GPS. If this message reports fewer than 3 satellites, the GPS may not have a reliable fix (not able to hook the satellite signal)—try moving to an open area with a clearer view of the sky.

This guide won't cover how to interpret the GPS data values, as that should be a hands-on exercise for anyone working on this kind of project.

Note: If your GPS module isn’t able to lock onto a satellite signal (i.e., the **red LED isn’t blinking**), you’ll likely receive data filled with **empty or invalid values**. This is because the module hasn’t established a GPS fix from hooking a stable satellite signal and therefore can’t provide accurate location information.

```
$GNGGA,000054.876,,,,,0,0,,,M,,M,,*5E
```

$GPGSA,A,1,,,,,,,,,,,,,*1E
$GLGSA,A,1,,,,,,,,,,,,,*02
$GNRMC,000054.876,V,,,,,0.00,0.00,060180,,,N*54

The previous guide already explained how to read UART data using C code, primarily relying on the `termios.h` library, as outlined in an article by Geoffrey Hunter and referenced in that guide. The main difference here is the folder we fetch the data from, and unlike the BeagleBone setup, we don't need to perform extra steps in code to configure the pin.

Sample read UART data code adapted from the previous guide:

```
#include <termios.h>
#include <fcntl.h>
#include <errno.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>

int serial_port;
static bool isInitialized = false;

void GPS_init() {
    serial_port = open("/dev/ttyAMA0", O_RDWR);
    if (serial_port < 0) {
        printf("Error %i from open: %s\n", errno, strerror(errno));
        return;
    }

    struct termios tty;
    if (tcgetattr(serial_port, &tty) != 0) {
        printf("Error %i from tcgetattr: %s\n", errno, strerror(errno));
        return;
    }

    // Configure serial settings
    tty.c_cflag &= ~PARENB;             // No parity
    tty.c_cflag &= ~CSTOPB;             // One stop bit
    tty.c_cflag &= ~CSIZE;
    tty.c_cflag |= CS8;                 // 8 data bits
    tty.c_cflag &= ~CRTSCTS;            // No hardware flow control
    tty.c_cflag |= CREAD | CLOCAL;      // Turn on READ & ignore ctrl lines

    tty.c_lflag |= ICANON;              // Canonical mode
    tty.c_lflag &= ~ECHO;               // Disable echo
    tty.c_lflag &= ~ECHOE;
    tty.c_lflag &= ~ECHONL;
    tty.c_lflag &= ~ISIG;               // Disable interpretation of INTR, QUIT,
```

```
SUSP

    tty.c_iflag &= ~(IXON | IXOFF | IXANY); // Disable software flow control
    tty.c_iflag &= ~(IGNBRK | BRKINT | PARMRK | ISTRIP | INLCR | IGNCR | ICRNL);

    tty.c_oflag &= ~OPOST;              // Raw output
    tty.c_oflag &= ~ONLCR;

    tty.c_cc[VTIME] = 10;               // Wait for up to 1 second
    tty.c_cc[VMIN] = 0;

    // Set baud rate to 9600
    cfsetspeed(&tty, B9600);

    if (tcsetattr(serial_port, TCSANOW, &tty) != 0) {
        printf("Error %i from tcsetattr: %s\n", errno, strerror(errno));
    }
    isInitialized = true;
}

char* GPS_read() {
    static char read_buf[BUFFER_SIZE];
    while (isInitialized) {
        int n = read(serial_port, &read_buf, sizeof(read_buf));
        if (n > 0) {
            read_buf[n] = '\0'; // Properly terminate the string
            // Check if the received message starts with "$GNRMC"
            if (strncmp(read_buf, "$GNRMC", 5) == 0) {
                return read_buf;
            }
        }
    }
    return "";
}
```

Troubleshooting

Here are some interesting problems we found!

Q: Nothing shows up when I run cat /dev/ttyAMA0
A: Make sure you've loaded the overlay file and wired the GPS module correctly.

Q: The red LED on the GPS module is not blinking. It doesn't seem to be able to hook onto the satellite signal.
A: The GPS module's antenna is highly sensitive to obstacles. Ensure that the GPS module has a clear view of the sky and give it some time to acquire satellite signals.

Q: The R5 code seems to crash when I query GPS data via the command "cat /dev/ttyAMA0" on the terminal or using C code.
A: We suspect this issue may occur because the R5 code is using the UART, which could be causing a conflict when trying to access it on the Linux side. We resolve this by disable the UART in the R5 code to prevent any interference with GPS communication. You can disable the UART port in R5 by adding the following code to your overlay file when compiling it with R5.

```
&uart1 {
    status = "disabled";
};
...
```

Acknowledgment And Reference:

As of spring 2025, there are already three other guides that teach how to connect the GPS module to the **BeagleBone Green**. This guide is specifically designed for the BeagleY-AI, but it includes some content that references their work, which may still be useful for your setup.

Adafruit Mini GPS PA1010D UART Guide
GPS-AdaFruit PA1616S Via UART
Grove GPS SIM28 UART Guide For BBG

This discussion forum is helpful for us to set up the UART

Beagley-ai UART1 Port Issues

If you encounter any issues not addressed in our troubleshooting guide, feel free to refer to these resources, as they may provide additional solutions not covered here.