Evelta L86 GNSS Arduino Logger Shield

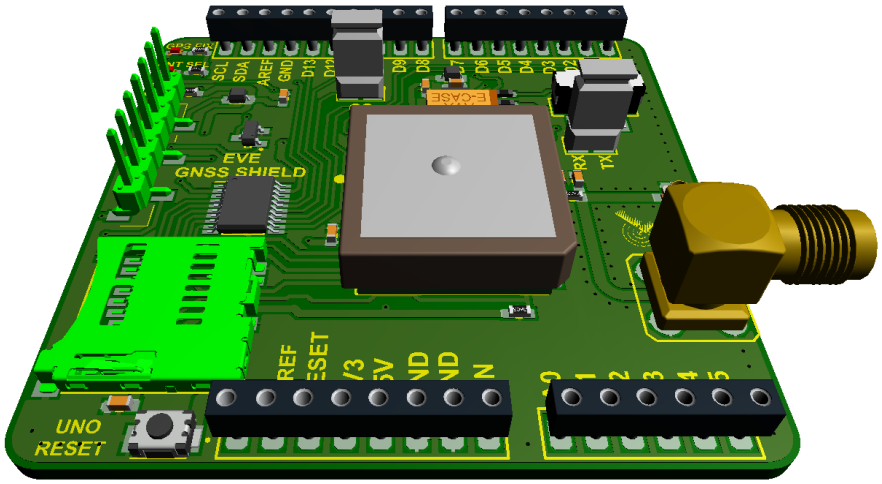
User Manual

Overview

The Arduino shield is based on Quectel's L86 an ultra-compact GNSS module that integrates a 18.4x18.4x4.0mm patch antenna along with an external antenna connector. It uses MediaTek's new generation GNSS MT3333 chip. Through advanced AGPS (EASY) orbit prediction technology and power saving mode (AlwaysLocate technology), the L86 module achieves high performance and fully meets industry standards. EASY technology enables L86 to automatically calculate and predict orbit information for up to three days, and store this information in the internal RAM memory, which can achieve low-power fast positioning even under weak indoor signals. The use of AlwaysLocate technology enables the L86 to automatically adjust the positioning time according to different environmental conditions and operating modes, which greatly reduces the power consumption of the module while ensuring positioning accuracy.

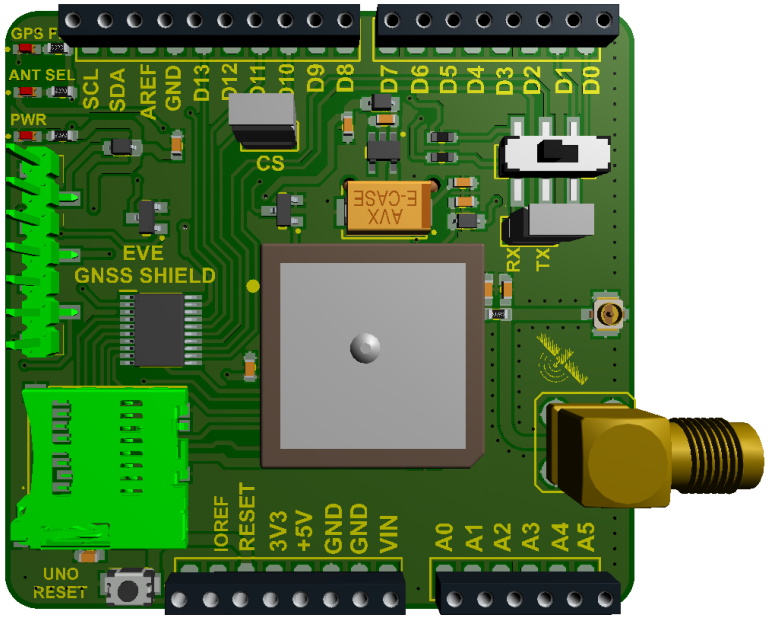
The shield supports the intelligent detection of antenna plug-in and short-circuit protection. It also supports automatic switching between the built-in patch antenna and the external active antenna, and the positioning is continued during the switching process.

L86 GNSS Arduino Shield

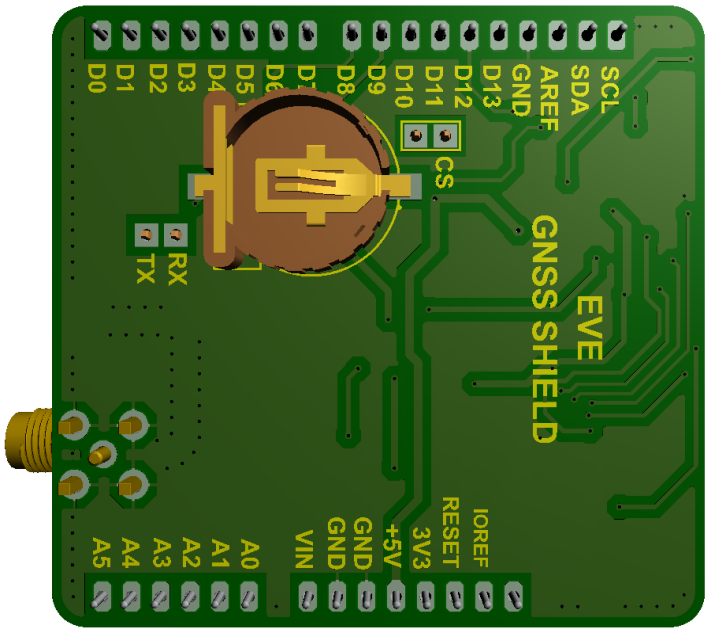


Features

- Support multiple satellite systems: GPS, GLONASS, Galileo and QZSS
- Internal patch antenna: 18.4mm x 18.4mm x 4.0mm
- Support internal and external antenna automatic switching
- Support short circuit protection and antenna detection
- Built-in low-noise amplifier to improve receiving high sensitivity
- Support self-assisted AGPS (EASY TM technology, no external memory required)
- Very low current consumption: 26mA @tracking mode
- Multiple power saving modes: standby Mode, Backup Mode, Periodic Mode, AlwaysLocate™ Mode
- LOCUS technology, support automatic log information recording and storage
- High sensitivity: -167dBm@tracking mode, -149dBm@acquisition mode
- Number of channels: 99 capture channels, 33 tracking channels
- Support Balloon mode, positioning altitude up to 80km
- Support DGPS, SBAS (WAAS/EGNOS/MSAS/GAGAN)
- Multi-frequency active interference cancellation technology to enhance anti-interference ability
- CR1220 GPS/GNSS Back-up Battery holder at the bottom side of the board
- External Active GNSS Antenna Connector
- LEDs: Power, Status, Netlight

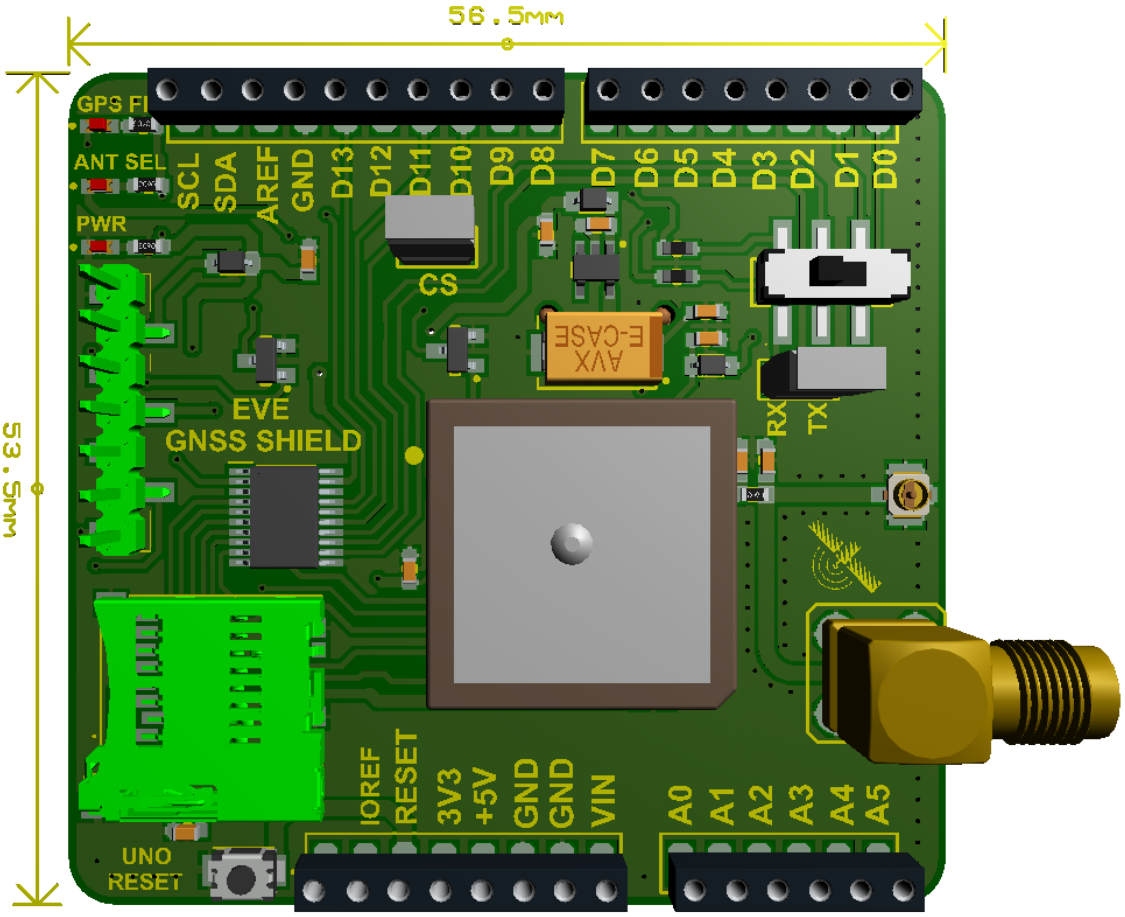


Front



Back

Dimensions



Arduino Code for GPS Read

```
#include <SoftwareSerial.h> // Include the SoftwareSerial library
#define ARDUINO_GPS_RX 9 // Arduino RX pin connected to GPS TX
#define ARDUINO_GPS_TX 8 // Arduino TX pin connected to GPS RX
#define GPS_BAUD_RATE 9600 // The GPS Shield module defaults to 9600
baud
// Create a SoftwareSerial object called gps:
SoftwareSerial gpsPort(ARDUINO_GPS_TX, ARDUINO_GPS_RX);

// This is the hardware serial port on pins 0/1.
#define SerialMonitor Serial

void setup()
{
  gpsPort.begin(GPS_BAUD_RATE);
  SerialMonitor.begin(9600);
}

void loop()
{
  if (gpsPort.available()) // If GPS data is available
    SerialMonitor.write(gpsPort.read()); // Read it and print to
SerialMonitor
  if (SerialMonitor.available()) // If SerialMonitor data is
available
    gpsPort.write(SerialMonitor.read()); // Read it and send to GPS
}
```

Arduino Code for SD Card Read/Write

```
/*
  SD card read/write

  This example shows how to read and write data to and from an SD
card file
  The circuit:
   SD card attached to SPI bus as follows:
 ** MOSI - pin 11
 ** MISO - pin 12
 ** CLK - pin 13
 ** CS - pin 10
  This example code is in the public domain.

*/

#include <SPI.h>
#include <SD.h>

File myFile;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for the serial port to connect. Needed for native USB
port only
  }


  Serial.print("Initializing SD card...");

  if (!SD.begin(10)) {
    Serial.println("initialization failed!");
    while (1);
```

```
  }
  Serial.println("initialization done.");

  // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
  myFile = SD.open("test.txt", FILE_WRITE);

  // if the file opened okay, write to it:
  if (myFile) {
    Serial.print("Writing to test.txt...");
    myFile.println("testing 1, 2, 3.");
    // close the file:
    myFile.close();
    Serial.println("done.");
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }

  // re-open the file for reading:
  myFile = SD.open("test.txt");
  if (myFile) {
    Serial.println("test.txt:");

    // read from the file until there's nothing else in it:
    while (myFile.available()) {
      Serial.write(myFile.read());
    }
    // close the file:
    myFile.close();
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
}
```

```
void loop() {
  // nothing happens after setup
}
```