Arduino NANO 33 Made Easy BLE, Sense and IoT

by drmpf

Quick Start

This tutorial shows you how to use the free pfodDesignerV3 V3.0.3774+ Android app to create a general purpose Bluetooth Low Energy (BLE) and WiFi connection for Arduino NANO 33 boards without doing any programming. There are three (3) Arduino NANO 33 boards, the NANO 33 BLE and NANO 33 BLE Sense, which connect by BLE only and the NANO 33 IoT which can connect via BLE or WiFi.

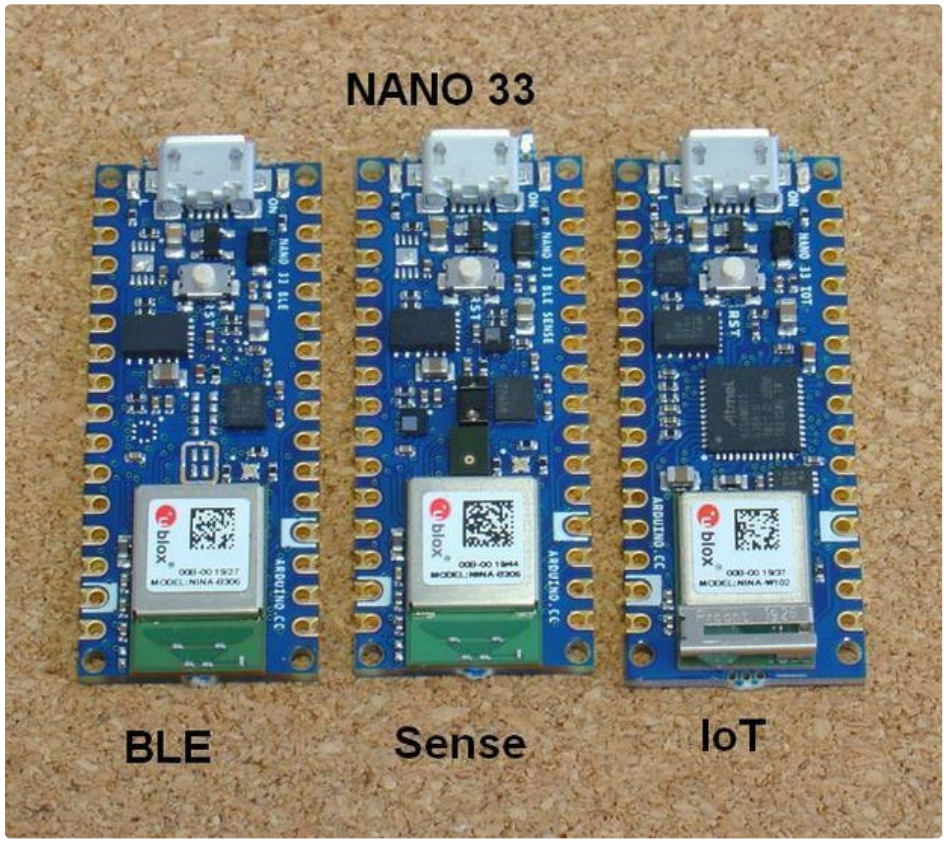
Connecting using pfodApp is the most flexible way to connect via BLE (or WiFi). See
Using pfodApp to connect to the NANO 33 BLE (Step 4) and
Using pfodApp to connect to the NANO 33 IoT via BLE (Step 7) and
Using pfodApp to connect to the NANO 33 IoT via WiFi (Step 9) below.

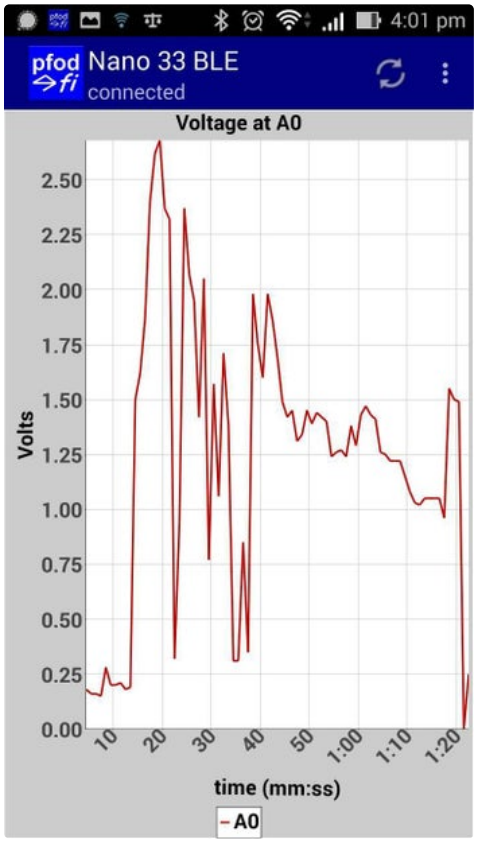
However simple sketches are also provided to send user defined command words via Telnet, for WiFi, or via the free Nordic nRF UART 2.0 for BLE. See
Using the Nordic nRF UART 2.0 app to connect to NANO 33 BLE (Step 5) and
Using the Nordic nRF UART 2.0 app to connect to the NANO 33 IoT via BLE (Step 8) and
Using a Telnet terminal program to connect to the NANO 33 IoT via WiFi (Step 10) below

This tutorial is also available on-line at Arduino NANO 33 Made Easy

Supplies:

Arduino NANO 33 -- either BLE or Sense or IoT
optionally pfodDesignerV3 and pfodApp





Step 1: Introduction

There are a number of problems with BLE. ***See this page for BLE problems and solutions and there are some BLE trouble shooting tips***. The learning curve is steep and the specification has hundreds of specialise connect services each of which requires its own mobile application to connect to. This tutorial shows you how to generate Arduino code for a general purpose Nordic UART BLE connection over which you can send and receive a stream of commands and data to a general purpose BLE UART mobile application. The free pfodDesignerV3 Android application is used to generate the Arduino code. The output is designed to connect to the paid pfodApp Android application which can display menus, send command, log data and show charts. No Android programming is required. The Arduino code has complete control over what is displayed by pfodApp.

For the NANO 33 IoT you can also connect via WiFi. Again the pfodDesignerV3 generates all the Arduino code and by default is designed to connect to pfodApp with optional 128bit security.

However you do not need to use pfodApp, you can connect to the generated code using the free Nordic nRF UART 2.0 or a Telnet programs (for the WiFi connection). Sketches are included which provide command words to control the boards.

The free pfodDesigner V3.0.3774+ will generate Arduino code for a wide range of boards and connection types including Serial connections, Bluetooth Low Energy (BLE), WiFi, SMS, Radio/LoRa, Bluetooth Classic and Ethernet. For examples Arduino code for of a wide range of BLE boards see Bluetooth Low Energy (BLE) made simple with pfodApp. Here we will be using a BLE connection for NANO 33 BLE, Sense and IoT and a WiFi connection for the IoT.

Each of the NANO 33 boards has extra sensor components that differ between the three (3) boards. The pfodDesignerV3 generates code to read/write the digital outputs and perform analogReads and analogWrites. In the examples below we will turn the board LED on and off and read the voltage at A0 and log and plot it. Once that sketch is running you can add each board's specialised sensor libraries and sent their data in place of the analogRead(A0).

pfodDesigner generates simple menus and charts, however you can also program custom graphical interfaces in your Arduino sketch. Above is an example of slider control adjusting a guage. All the code for this control is in your Arduino sketch. No Android programming necessary. See Custom Arduino Controls for more examples.

How pfodApp is optimised for short BLE style messages

Bluetooth Low Energy (BLE) or Bluetooth V4 is a completely different version of Bluetooth. BLE has been optimised for very low power consumption. pfodApp is a general purpose Android app whose screens, menus, buttons, sliders and plots are completely defined by the device you connect to.

BLE only sends 20 bytes in each message. Fortunately the pfod Specification was designed around very small messages. Almost all of pfod's command are less then 20 bytes. The usual exception is the initial main menu message which specifies what text, menus, buttons, etc. pfodApp should display to the user, but the size of this message is completely controlled by you and you can use sub-menus to reduce the size of the main menu.

The pfod specification also has a number of features to reduce the message size. While the BLE device must respond to every command the pfodApp sends, the response can be as simple as {} (an empty response). If you need to update the menu the user is viewing in response to a command or due to a re-request, you need only send back the changes in the existing menu, rather then resending the entire menu. These features keep the almost all messages to less than 20 bytes.

pfodApp caches menus across re-connections so that the whole menu only needs to be sent once. Thereafter short menu updates can be sent.

Step 2: Creating the Custom Android Menus and Generating the Code

Before looking at each of these BLE modules, pfodDesignerV3 will first be used to create a custom menu to turn a Led on and off and plot the voltage read at A0. pfodDesignerV3 can then generate code tailored to the particular hardware you select.

You can skip over this step and come back to it later if you like. The section on each module, below, includes the completed code sketch for this example menu generated for that module and also includes sketches that do not need pfodApp

The free pfodDesignerV3 is used to create the menu and show you an accurate preview of how the menu will look on your mobile. The pfodDesignerV3 allows you to create menus and sub-menus with buttons and sliders optionally connected to I/O pins and generate the sketch code for you (see the pfodDesigner example tutorials) but the pfodDesignerV3 does not cover all the features pfodApp supports. See the pfodSpecification.pdf for a complete list including data logging and plotting, multi- and single- selections screens, sliders, text input, etc.

Create the Custom menu to turn the Arduino LED on and off and Plot A0

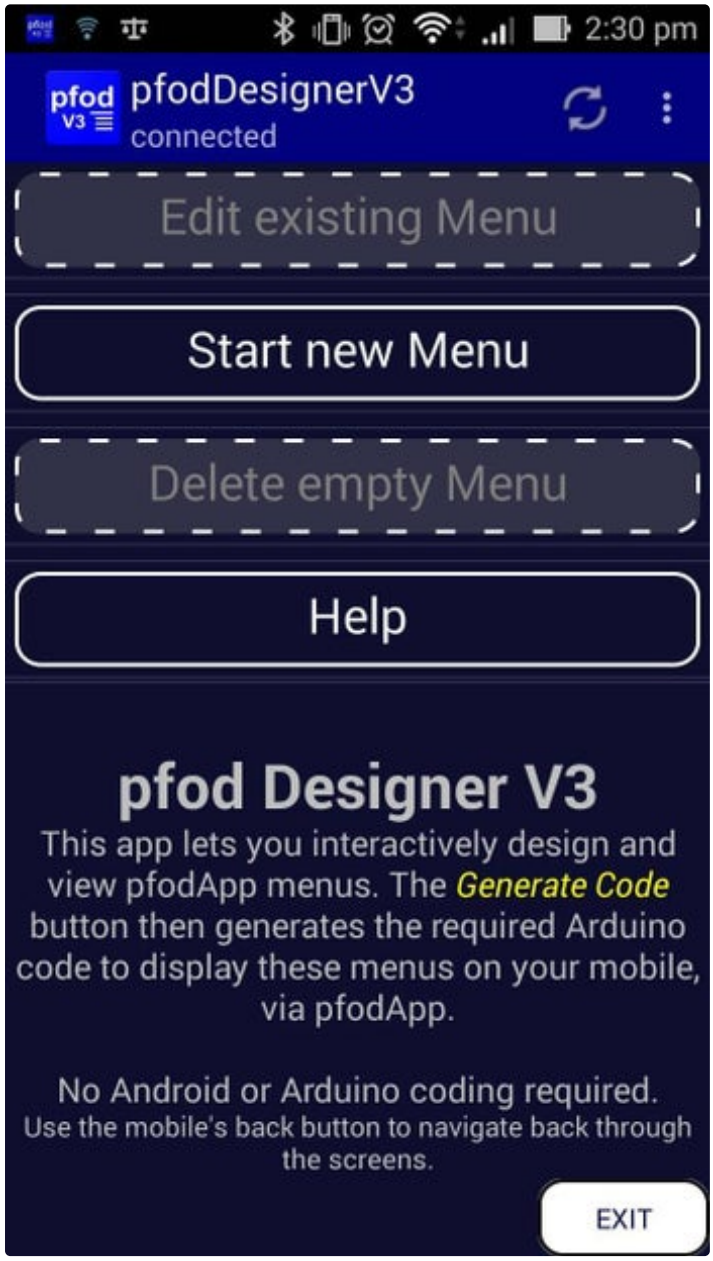
Start a new menu and select as a target Bluetooth Low Energy (BLE) and then select NANO 33 BLE (and Sense). pfodDesignerV3.0.3770+ has support for NANO 33 boards.

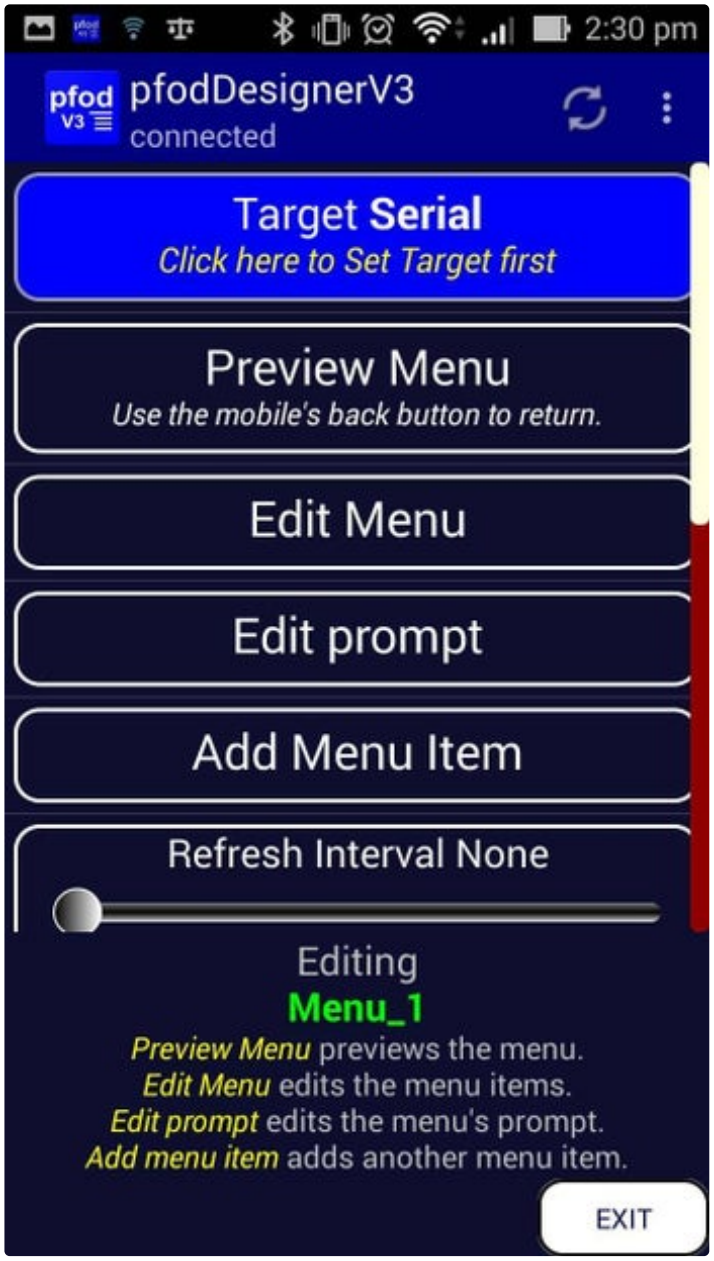
Then follow the tutorial Design a Custom menu to turn the Arduino Led on and off for step by step instructions for creating a LED on/off menu using pfodDesignerV3.

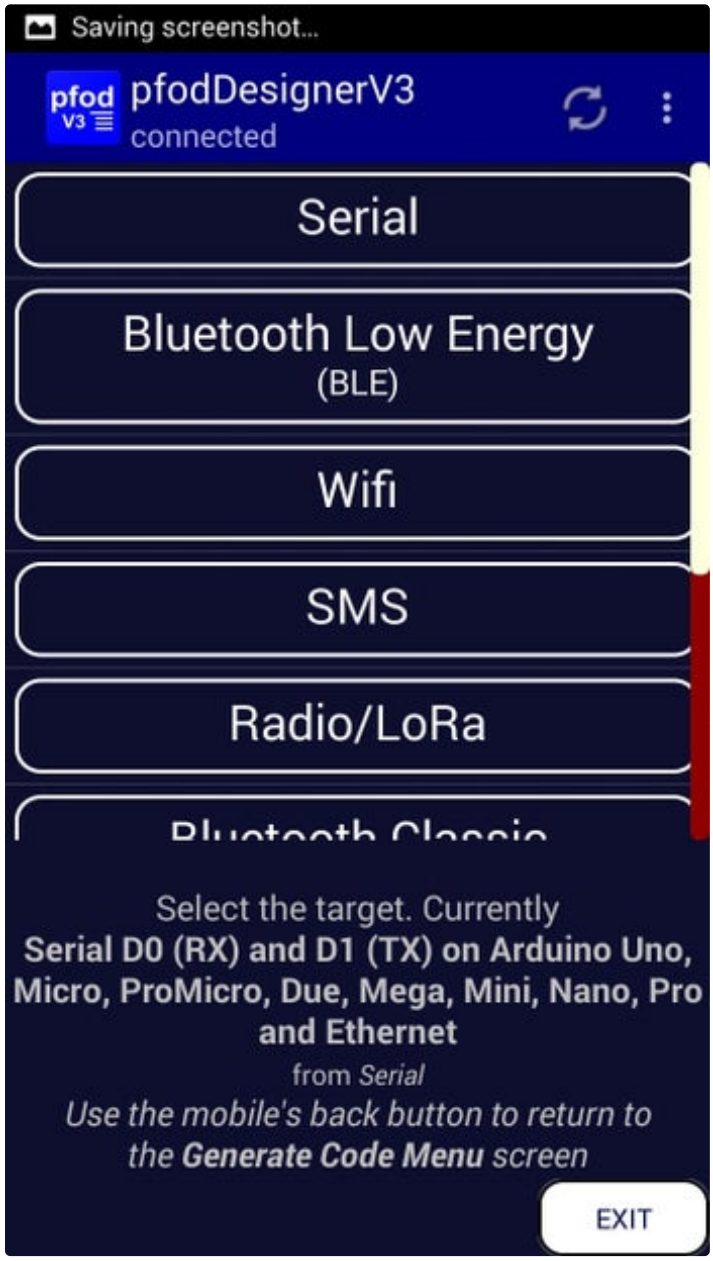
If you don't like the colours of font sizes or the text, you can easily edit them in pfodDesignerV3 to whatever you want and see a WYSIWYG (What You See Is What You Get) display of the designed menu.

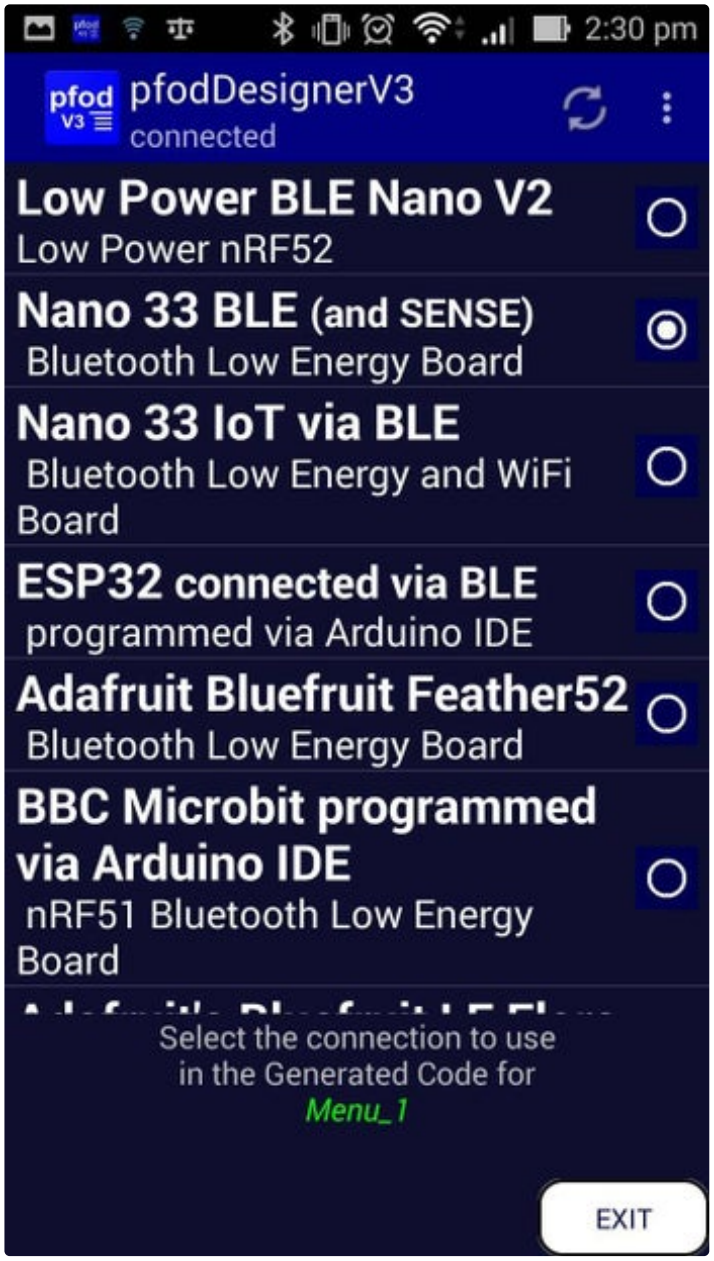
Now we will add a Chart button to display the A0 reading. The steps to does this in pfodDesignerV3 are shown in Adding a Chart and Logging Data The AtoD range is 0 to 1023 for 0 to 3.3V

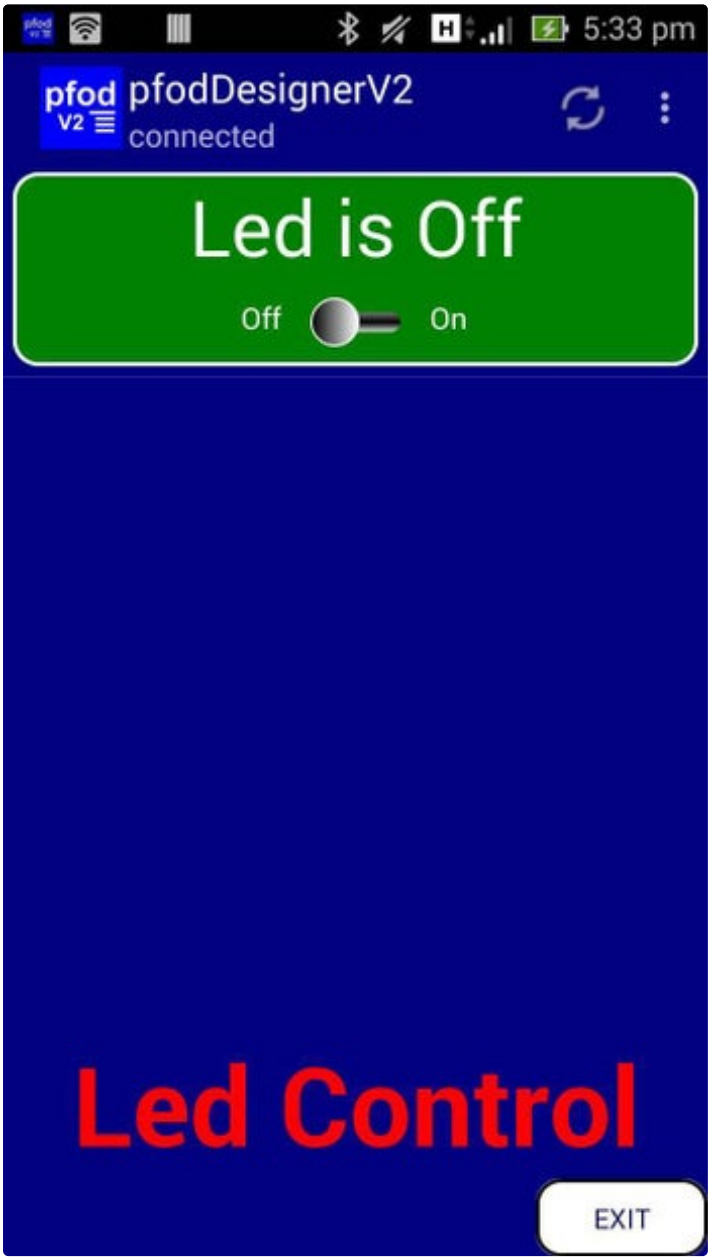
Generating the code from pfodDesignerV3 gives the this

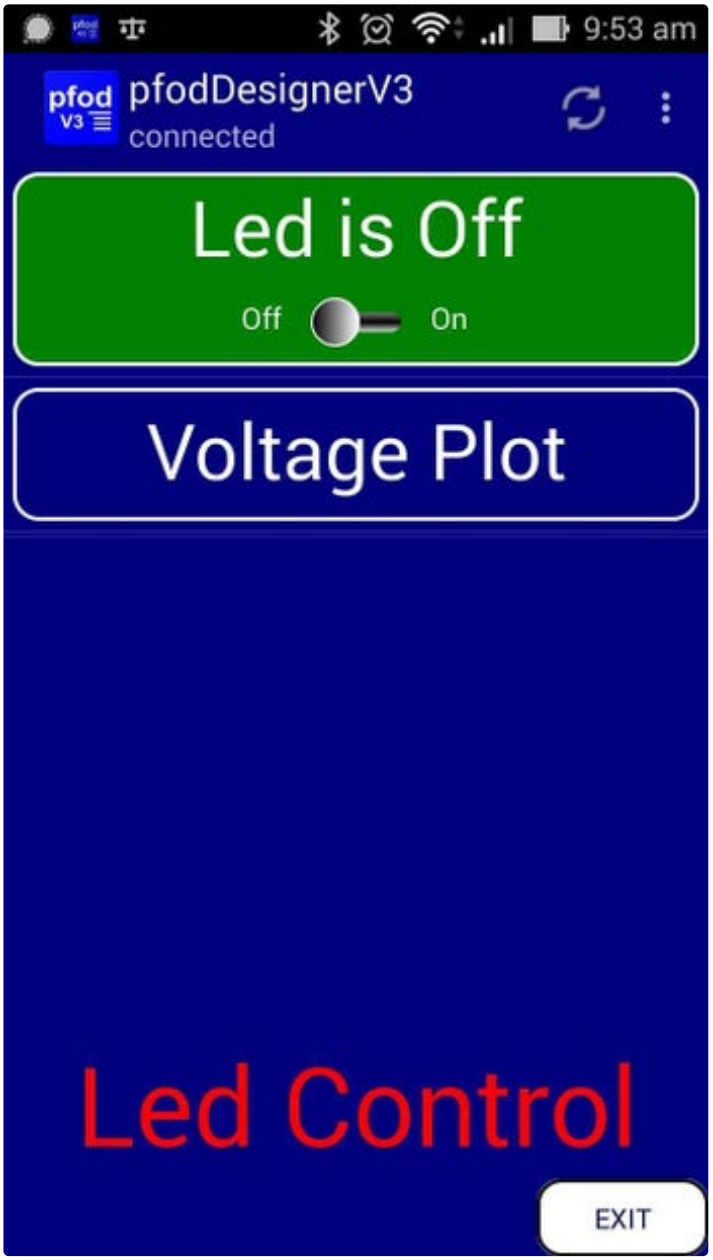

2:30 pm
pfodDesignerV3
connected
Edit existing Menu
Start new Menu
Delete empty Menu
Help
pfod Designer V3
This app lets you interactively design and view pfodApp menus. The Generate Code button then generates the required Arduino code to display these menus on your mobile, via pfodApp.
No Android or Arduino coding required.
Use the mobile's back button to navigate back through the screens.
EXIT


2:30 pm
pfodDesignerV3
connected
Target Serial
Click here to Set Target first
Preview Menu
Use the mobile's back button to return.
Edit Menu
Edit prompt
Add Menu Item
Refresh Interval None
Editing
Menu_1
Preview Menu previews the menu.
Edit Menu edits the menu items.
Edit prompt edits the menu's prompt.
Add menu item adds another menu item.
EXIT


Saving screenshot...
pfodDesignerV3
connected
Serial
Bluetooth Low Energy
(BLE)
Wifi
SMS
Radio/LoRa
Select the target. Currently
Serial D0 (RX) and D1 (TX) on Arduino Uno, Micro, ProMicro, Due, Mega, Mini, Nano, Pro and Ethernet
from Serial
Use the mobile's back button to return to the Generate Code Menu screen
EXIT


2:30 pm
pfodDesignerV3
connected
Low Power BLE Nano V2
Low Power nRF52
Nano 33 BLE (and SENSE)
Bluetooth Low Energy Board
Nano 33 IoT via BLE
Bluetooth Low Energy and WiFi Board
ESP32 connected via BLE
programmed via Arduino IDE
Adafruit Bluefruit Feather52
Bluetooth Low Energy Board
BBC Microbit programmed via Arduino IDE
nRF51 Bluetooth Low Energy Board
Select the connection to use in the Generated Code for
Menu_1
EXIT



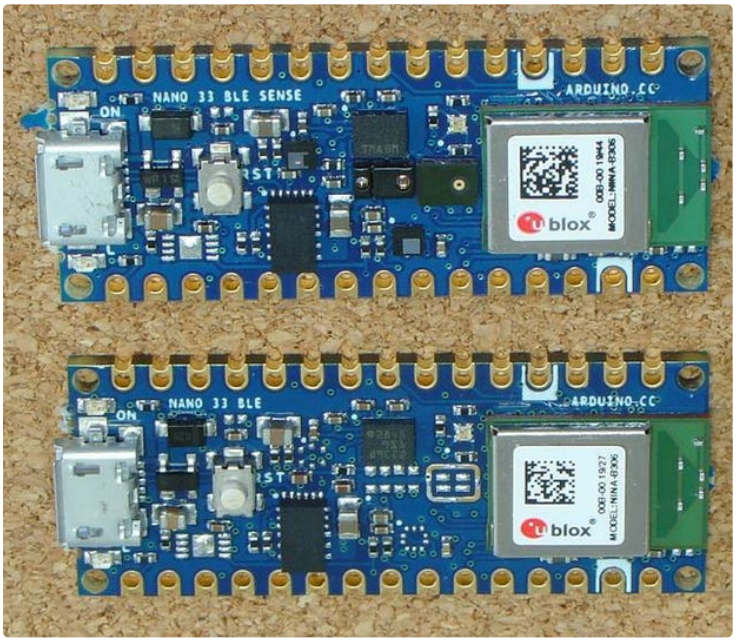


Step 3: NANO 33 BLE / Sense

Both the NANO 33 BLE and the NANO 33 BLE Sense connect via BLE. The NANO 33 BLE as an IMU (Inertial Measurement Unit). The NANO 33 BLE Sense has an IMU and a number of other sensor ICs. Each sensor requires it appropriate library to be installed. Here just the basic Digital and Analog I/O will be used, but you can readily replace the Analog plot data with a sensor's data.

Follow the Getting started with the Arduino NANO 33 BLE instructions to install the NANO 33 board support in the Arduino IDE and load and test with the Blink example. NOTE: If the board COM port is not found for programming you can double click the reset push-button to put the board into program mode until it is reset or power cycled. Also install the ArduinoBLE library from the Arduino IDE library manager.

Installing support for NANO 33 BLE / Sense

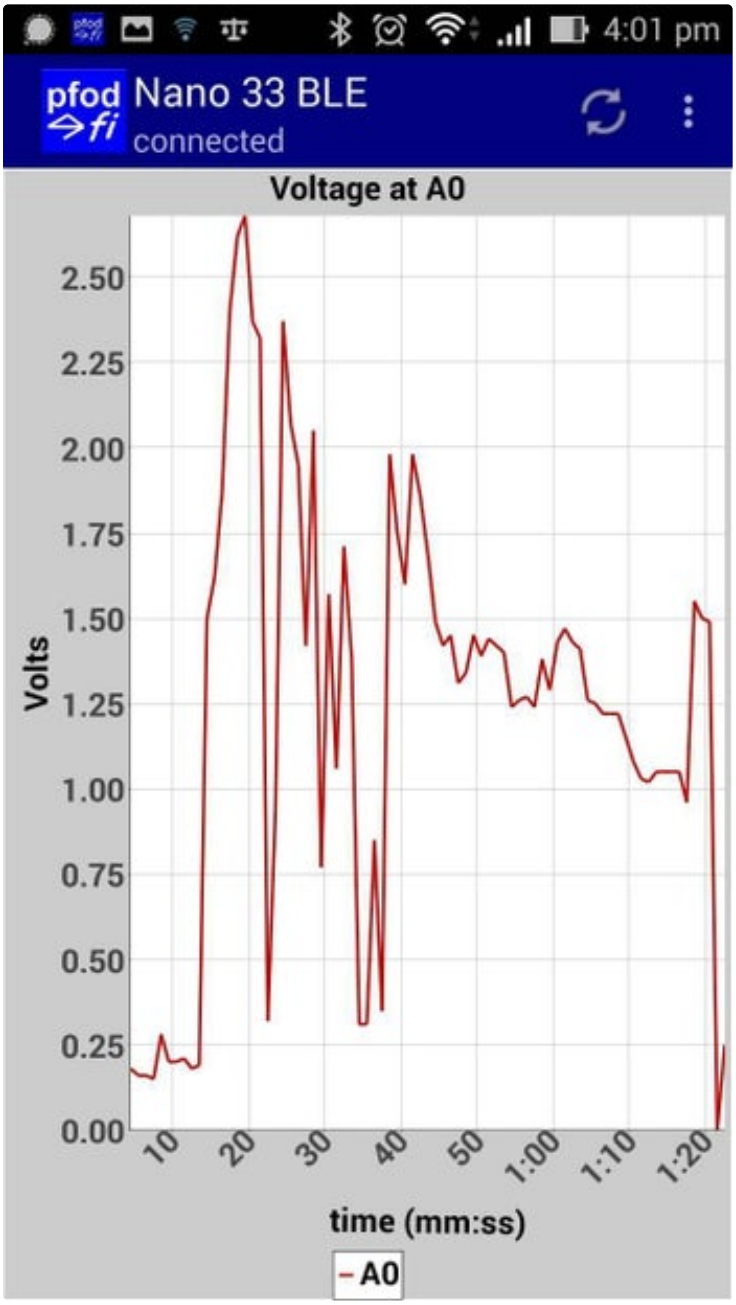


Step 4: Using PfodApp to Connect to the NANO 33 BLE

To connect using the pfodApp, load the Nano33BLE_Led_A0.ino sketch and then set up a BLE connection to the NANO 33 BLE (see the pfodAppForAndroidGettingStarted.pdf) then when you connect you will see the menu you designed above. Clicking in the Led button will toggle the led on/off. Clicking on the Voltage Plot button will show the plot of the voltage on A0. By default the plot data is also logged on your mobile for later use.

Sending BLE messages longer then 20 bytes

The BLE message size is limited to 20 bytes, pfodParser V3.32+ includes a pfodBLEBufferedSerial class which buffers the parser.print() and releases them 20 bytes at a time with a delay between each block to allow them to be sent via BLE. The code generated for the NANO 33 BLE connections includes this BLE buffer (defaults to 1024 byte buffer). The delay between blocks can adjusted by calling pfodBLEBufferedSerial's **setBLEBlockSendDelay(max)** which should match the pfodBLESerial's **setConnectionInterval(min,max)** max value. The delay is set by the maxConnectionCount (default 160 => 200mS)



Step 5: Using the Nordic NRF UART 2.0 App to Connect to NANO 33 BLE

You can also connect to the NANO 33 BLE using the free Nordic nRF UART 2.0 Open the app and click Connect and choose the NANO 33 BLE connection. You will see the CSV data for the A0 reading being sent every second.

To control your NANO 33 BLE/Sense from the nRF UART 2.0 you can modify the generated code to remove the pfodParser and add text commands. The modified sketch is Nano33BLE_UART_LED.ino

Nano33BLE_UART_LED.ino adds the SafeString library via the Library Manager and needs the millisDelay library installed from How to code Timers and Delays in Arduino

```
#include "ArduinoBLE.h"
 #include SafeString.h"
// install SafeString library V1.0.3+ via Arduino Library Manager
#include "millisDelay.h"
// download the millisDelay library from https://www.forward.com.au/pfod/ArduinoProgrammin...
// download the pfodParser library from http://www.forward.com.au/pfod/pfodParserLibrarie...
// pfodParser.zip V3.48+ contains pfodBLEBufferedSerial, pfodParser, pfodSecurity, pfodDelay, pfodSMS and pfodRadio
#include "pfodBLEBufferedSerial.h"
```

The **setup()** sets the Led output and connects the bleBufferedSerial and starts the plot timer.

```
void setup() {
   pinMode(Led_pin, OUTPUT); // output for 'Led' is initially LOW,
  digitalWrite(Led_pin, 0); // set output
  if (!bleSerial.begin()) {
    //  Serial.println("starting ble failed!");
    while (1);
  }
  bleBufferedSerial.connect(&bleSerial);
  plotDataTimer.start(PLOT_DATA_INTERVAL); // start plot timer
}
```

The next section defines the commands that will be recognized and the delimiters and the command timeout. The timeout will execute the last command if nothing else is received with in 0.3sec.

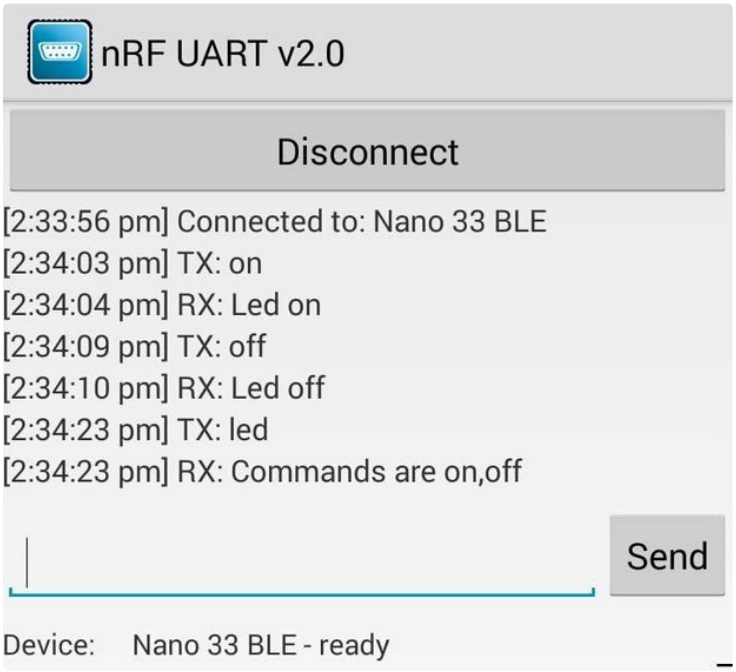
```
const size_t maxCmdLength = 5; // make SafeStrings at least large enough to hold longest cmd
 createSafeString(onCmdStr, maxCmdLength, "on");
createSafeString(offCmdStr, maxCmdLength, "off");

// input must be large enough to hold longest cmd + 1 delimiter
createSafeString(input, maxCmdLength + 1); //  to read input cmd + 1 delimiter
createSafeString(token, maxCmdLength + 1); // for parsing, capacity >= input.capacity()

char delimiters[] = " .,\r\n"; // space dot comma CR NL are cmd delimiters

millisDelay timeout;
unsigned long TIMEOUT_MS = 300; // 0.3sec
```

The **loop()** code is very simple. It collects chars from the BLE UART connection and then breaks it to tokens and executes any recognized command. If the command is invalid the message "Commands are on,off" is returned.

```
void loop() {
   if (input.read(bleBufferedSerial)) {  // read from Serial, returns true if at least one character was added to SafeString input
    timeout.start(TIMEOUT_MS); // restart a 0.3sec timer every time something is read
  }
  if (input.nextToken(token, delimiters)) { // process at most one token per loop does not return tokens longer than input.capacity()
    if (token == onCmdStr) {
      digitalWrite(Led_pin, 1); // set output
      bleBufferedSerial.print("Led on");
    } else if (token == offCmdStr) {
      digitalWrite(Led_pin, 0); // set output
      bleBufferedSerial.print("Led off");
    } else { // not a valid cmd ignore
      bleBufferedSerial.print("Commands are on,off");
    }
  }
  if (timeout.justFinished()) { // nothing received for 0.3secs, terminated last chars so token will be processed.
    input += delimiters[0]; // any delimiter will do
  }
//  sendData(); // uncomment this to send data every 1sec
}
```

Above is a sample screen from Nordic nRF UART 2.0

The 'led' command was not recognised so the message “Commands are on,off” was returned. The sketch can be easily extended to add your own command words. Multiple commands can be sent on the same line separated by space, dot or comma.



Step 6: NANO 33 IoT

The NANO 33 IoT can be connect to via either BLE or WiFi. The NANO 33 IoT has an IMU and a CryptoAuthentication IC, but no EEPROM. Here we will setting up a Serial BLE point to point connection and a Serial WiFi point to point connection.

Installing support for NANO 33 IoT and Upgrading the Firmware

First follow the Getting started with the Arduino NANO 33 IoT guide to install the board support for the NANO 33 IoT and run the Blink example to test the install. Next install the WiFiNINA_Generic library and the ArduinoBLE library via the Arduino IDE library manager.

Then load and run the CheckFirmwareVersion sketch from **File → Examples → WifiNINA_Generic → Tools → CheckFirmwareVersio**n The WifiNINA_Generic is at the bottom of the Examples list. Load and run this sketch and check the Monitor output (using 115200 baud setting)

If you get a message like the one below you will need to update the firmware

```
WiFiNINA firmware check.

Firmware version installed: 1.2.3
Latest firmware version available : 1.3.0

Check result: NOT PASSED
 - The firmware version on the module do not match the
   version required by the library, you may experience
   issues or failures.
```

To update the firmware, first load the **File → Examples → WifiNINA_Generic → Tools → FirmwareUpdater** sketch.

This sketch does not do the updating. It just communicates with the updating tool.

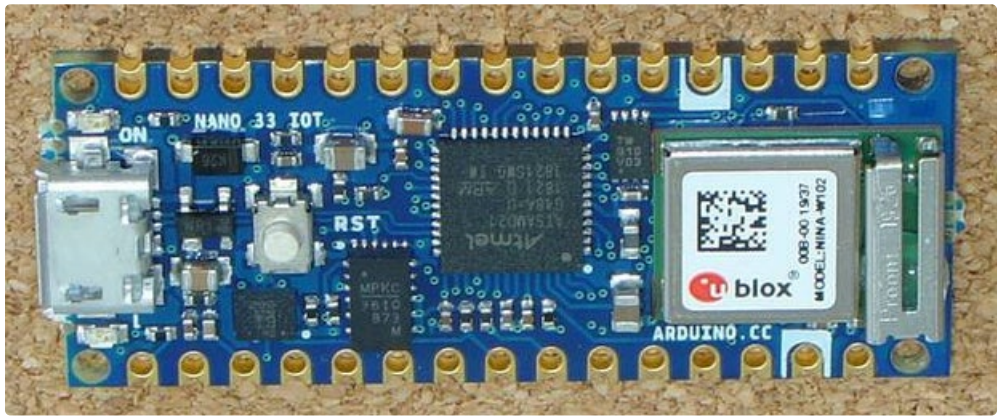
The Firmware and certificates Updater tutorial covers running the firmware updater tool ***BUT you must first install the latest version of the tool***. Download the latest firmware updater tool from https://github.com/arduino-libraries/WiFi101-FirmwareUpdater-Plugin/releases/latest The version used here was V0.10.10

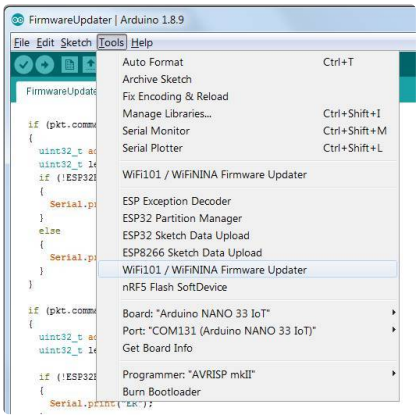
Create
a tools folder in your sketchbook, if it doesn't exist yet. It should already exist with the old version of the tool. Delete the old version i.e. delete the directory **../Arduino/tools/WiFi101** Then unzip the latest WiFi101-Updater-ArduinoIDE-Plugin-0.10.10.zip to the **tools** directory to get the WiFi101.jar in a directory like **.../Arduino/tools/WiFi101/tool/WiFi101.jar** Then stop and restart the ArduinoIDE.

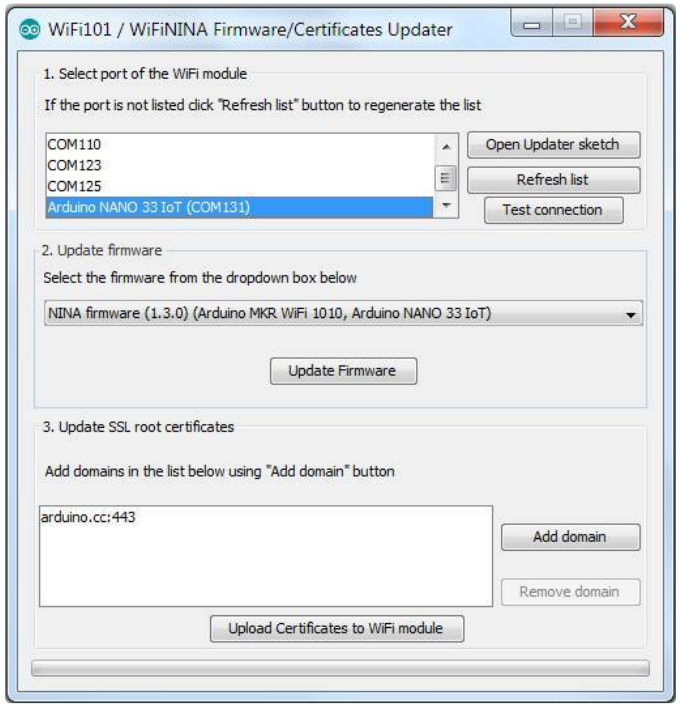
Open menu **Tools → WiFi101 / NINA Firmware Updater** If there are two listings choose the second one.

Scroll down the ports and select the Arduino NANO 33 IoT port and select the required firmware from the drop down. I.e. NINA 1.3.0 as noted in the CheckFirmwareVersion output above. If NINA 1.3.0 is not listed try the 'other' Firmware Updater.

Click Update Firmware button and the progress bar will indicate the progress. When it finishes there will be a Success dialog box.







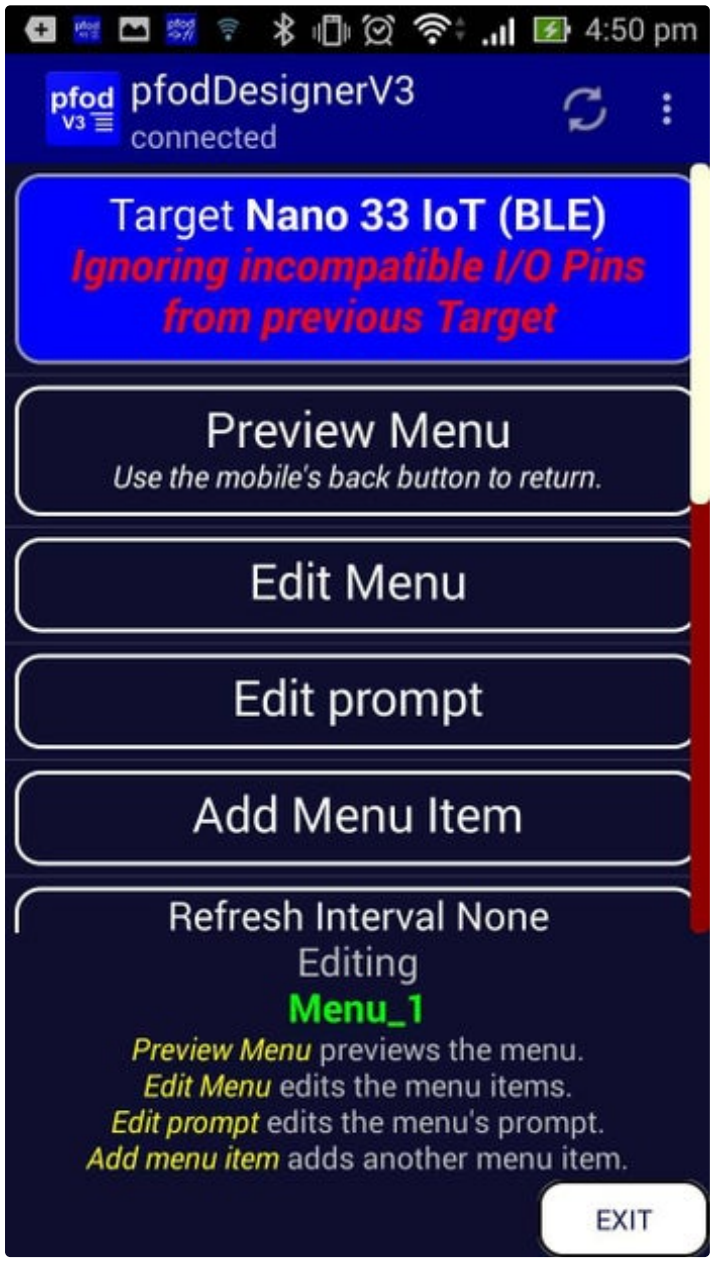
Step 7: Using PfodApp to Connect to the NANO 33 IoT Via BLE

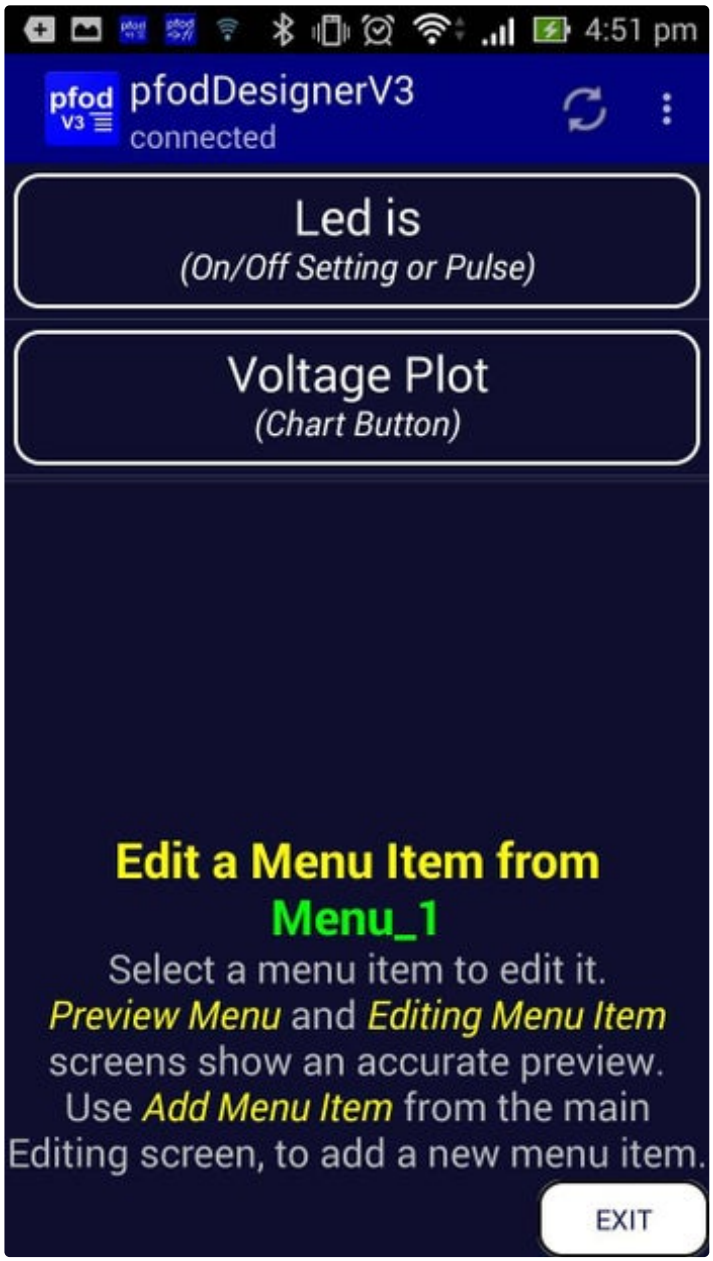
The NANO 33 IoT uses a different chip to the NANO 33 BLE and so the pin capabilities differ. Open the pfodDesignerV3 again. Choose **Edit Existing Menu** and the **Menu 1** and then change the target to **Bluetooth Low Energy → NANO 33 IoT via BLE**

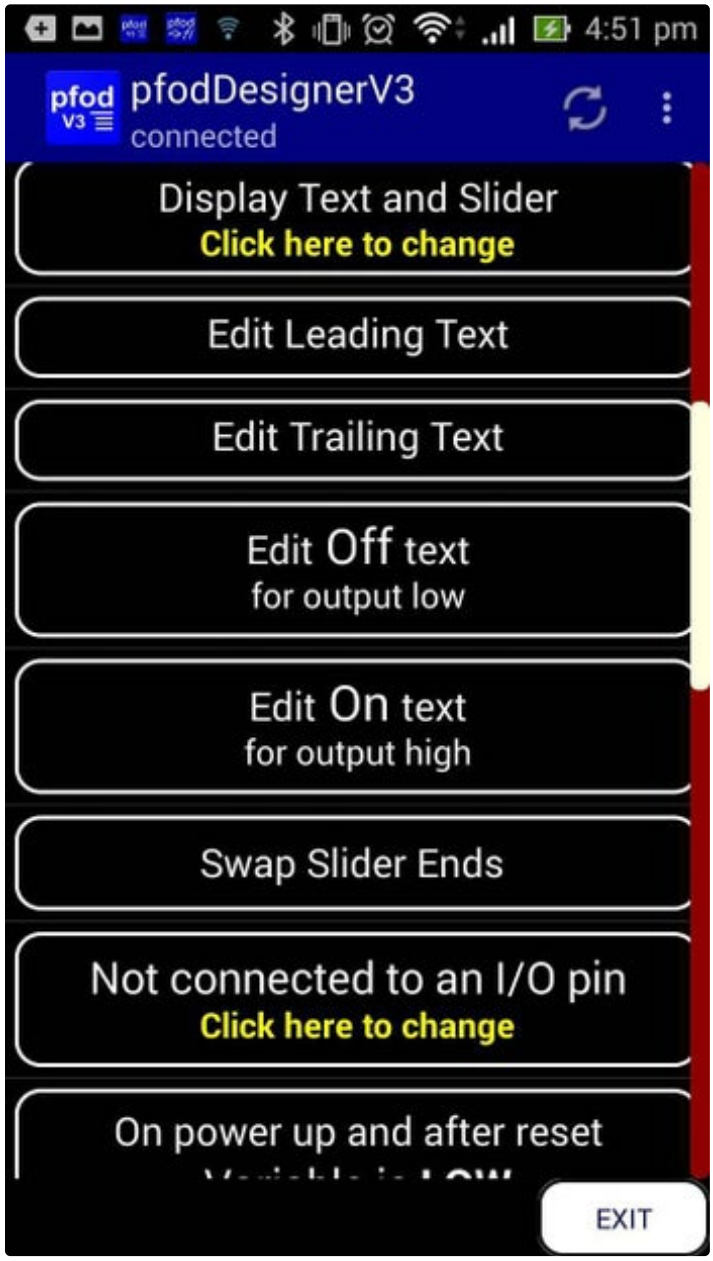
There will be a message *Ignoring incompatible I/O Pins from the previous Target*. The D13 pin on the IoT is different. It does not support PWM.

Click on **Edit Menu** and then on the **Led is** button and then scroll down to I/O pin to choose the D13 pin.

You can then go back and generate the code again. The sketch is Nano33IoTBLE_Led_A0.ino This sketch is the same as Nano33BLE_Led_A0.ino except for comments. The NANO 33 IoT via BLE connection uses the same libraries as the NANO 33 BLE and the pfodApp connection is similar (but with a different BLE address)

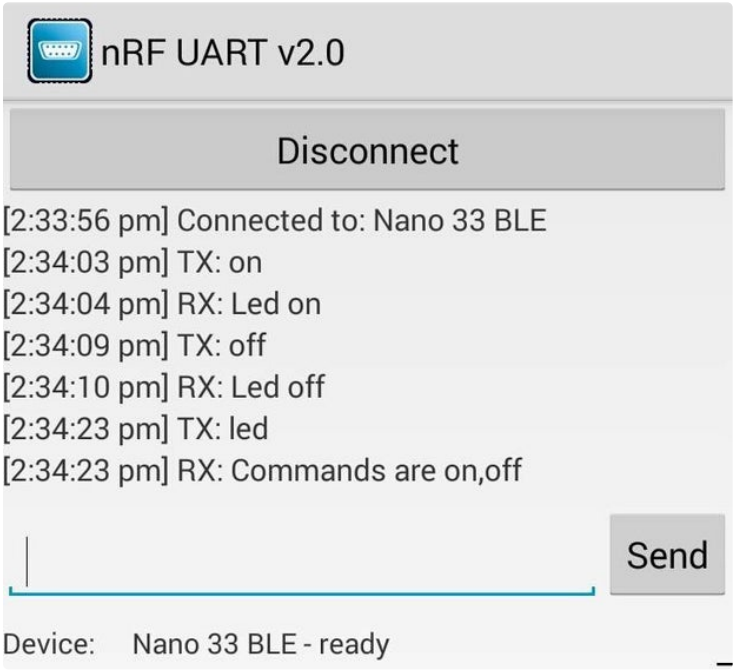






Step 8: Using the Nordic NRF UART 2.0 App to Connect to the NANO 33 IoT Via BLE

Because the NANO 33 IoT via BLE connection uses the same libraries as the NANO 33 BLE, The sketch to connect using the Nordic nRF UART 2.0 app is the same, i.e. Nano33BLE_UART_LED.ino as above in Step 5



Step 9: Using PfodApp to Connect to the NANO 33 IoT Via WiFi

Open the Existing Menu, previously designed in pfodDesignerV3 (above Create the Custom menu to turn the Arduino LED on and off and Plot A0) and click in **Target** and then **WiFi** and then **NANO 33 IoT connected via WiFi**. Go back and **Generate Code**

The resulting sketch is Nano33IoTWiFi_Led_A0.ino

```
Edit the top of the sketch to match your WiFi network and choose a
staticIP so that it is easy to connect to, typically 10.1.1.200 to
10.1.1.254 so as not to interfere with any other existing devices
```

```
#define WLAN_SSID       "myNetwork"        // cannot be longer than 32 characters!
#define WLAN_PASS       "myPassword"
const int portNo = 4989; // What TCP port to listen on for connections.
const char staticIP[] = "";  // set this the static IP you want, e.g. "10.1.1.200" or leave it as "" for DHCP. DHCP is not recommended.
```

Setting up a WiFi connection on pfodApp (see the pfodAppForAndroidGettingStarted.pdf) and connecting with pfodApp will show the same menu as above when connecting via NANO 33 BLE above. You can also add a 128bit shared private key to secure the connection. See A Simple WiFi/Arduino pfodDevice with 128 bit security and SipHash Secure Challenge and Response for micro-devices and Secret Key Generator for Secure Challenge and Response

Step 10: Using a Telnet Terminal Program to Connect to the NANO 33 IoT Via WiFi

To connect via a telnet terminal, the changes made are similar to those made for **Using the Nordic nRF UART 2.0 app to connect to NANO 33 BLE** above. The port number is changed to 23, the well know Telnet port number. The resulting sketch is Nano33IoTWiFi_UART_Led.ino

On PC's you can use TeraTerm to connect. Connecting using Telnet on Mac is bit more involved but still doable. On your Android mobile there are a number of terminal apps such as TCP Telnet Terminal Pro

Connecting using TeraTerm

Start TeraTerm. Use the menu option **Setup → Terminal** to tick **Local Echo**

Then use **File → New Connection** to open an Telnet connect to the IP you have set. Here I have used 10.1.1.211

TeraTerm sends some initial negotiation bytes which the Nano33IoTWiFi_UART_Led.ino sketch does not recognised as a command so it responds with the "Commands are on,off" message. You can then send the **on** and **off** commands.

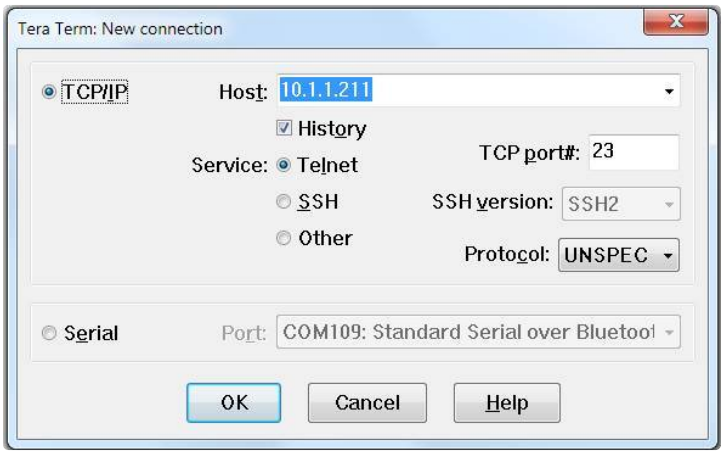
Again the sketch can be easily extended to handle other command words. Multiple commands can be sent on the same line separated by space, dot or comma.

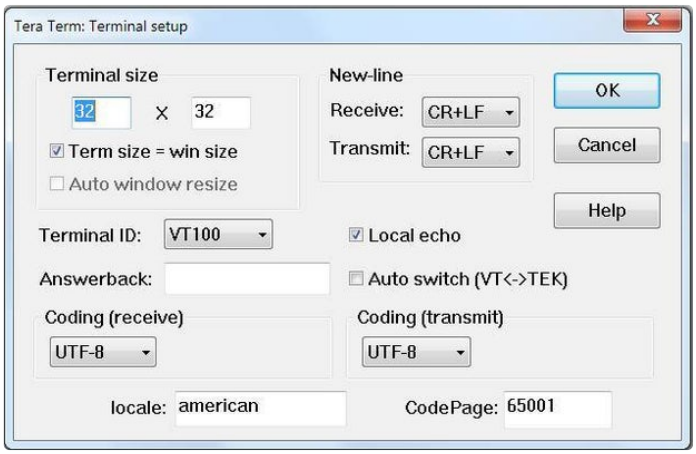
TeraTerm buffers you typing until you press Enter. Other Telnet programs may send each character as you type it. In that case if you don't type fast enough the timeout timer will finish and add a delimiter forcing a token to be returned. You can fix this by increasing the timeout setting to say 1sec

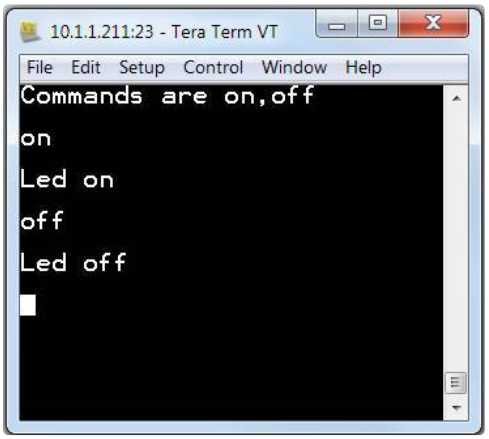
```
unsigned long TIMEOUT_MS = 1000; // 1.0sec
```

OR by commenting out the timer altogether.

```
if (input.read(bleBufferedSerial)) {  // read from Serial, returns true if at least one character was added to SafeString input
//  timeout.start(TIMEOUT_MS); // remove timer start so need to send delimiter like space or newline
}
```







Step 11: Conclusion

This tutorial has shown how you can easily set-up Arduino NANO 33 BLE, Sense and IoT to connect via Bluetooth Low Energy (and WiFi for the IoT).
No Android programming is required. pfodApp handles all of that.
No Arduino coding is required. The (free) pfodDesignerV2 generates complete sketches for each of these modules.

While the sketches generated by pfodDesigner are for use with pfodApp, as shown above you can readily modify them to use a Nordic BLE UART app or a telnet program (for WiFi). In the modified sketches two simple commands, on and off, where set-up but the sketches can be easily extended to handle other command words. Multiple commands can be sent on the same line.