# Handover document (software)

This section aims to get the future team to quickly up to know how the actual device and sub-communication systems work from the software perspective. A thorough reading of this section is able to:

1. Be familiar with the selected Arduino board

2. Understand the communications chipset on Arduino (Bluetooth module)

3. Short summary of each sub-module that already compiled in the code. (Buzzer, Screen, Button, LED , CO2 sensor , Temp sensor, Acetone sensor)

4. Suggestions about the future works from software side.

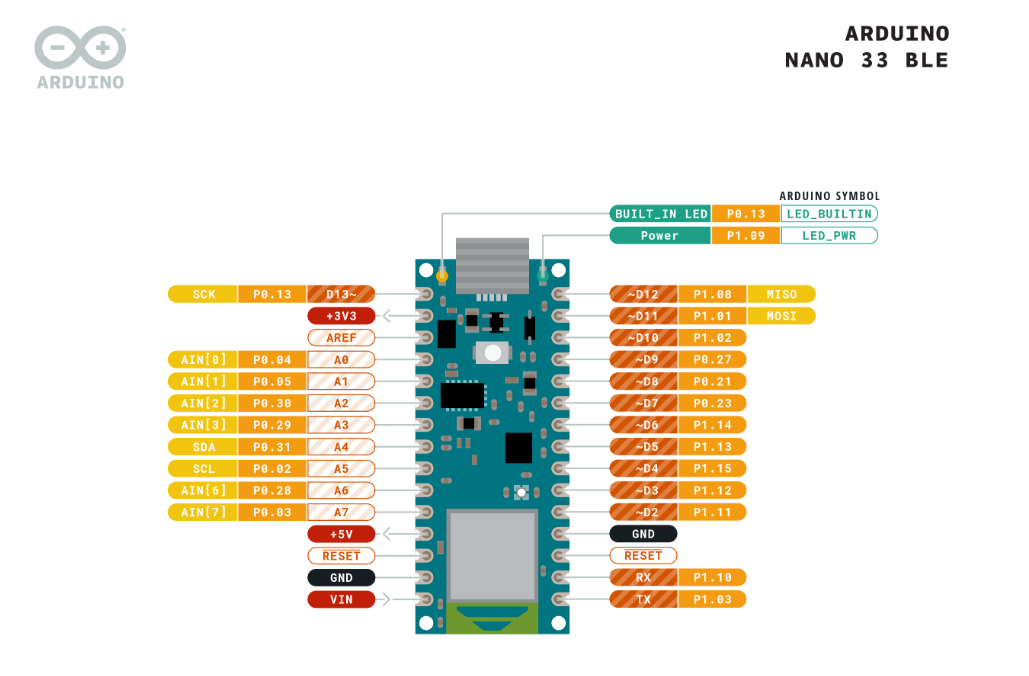
### Arduino Nano 33 BLE

The Arduino Nano 33 BLE is well-known as an evolution of the traditional Arduino Nano but featuring a lot more powerful processor (nRF52840 from Nordic Semiconductors <https://content.arduino.cc/assets/Nano_BLE_MCU-nRF52840_PS_v1.1.pdf>) This allows developer to write larger programs with a lot more variables than traditional Arduino Uno ( New Nano typically has default 1MB of program memory <https://store.arduino.cc/usa/nano-33-ble>).

Arduino Nano 33 BLE has two pairs of serial ports. One is the USB connector of the board which directly connected to the native USB of the NINA B306 module, it enables you can use the Nano 33 BLE as a client USB peripheral. This port can also be used as a virtual serial port using the Serial object in the Arduino programming language. Alternatively, the RX0 and TX1 pins are a second serial port available as Serial1.

Additionally, the Arduino Nano 33 BLE has the ability to change its analog read resolution (default is 10-bits and can support up to 12-bit ADC). <https://www.arduino.cc/reference/en/language/functions/zero-due-mkr-family/analogreadresolution/>. Above link is showing to how to change the ADC resolution on Nano.

For people who used Arduino for their projects before, you should be able to directly start the coding stuff since the coding syntax is quite similar. Besides, keep in mind that updates your libraries to Nano version, and carefully read through Pin Diagram and adjust the changes correspondingly.

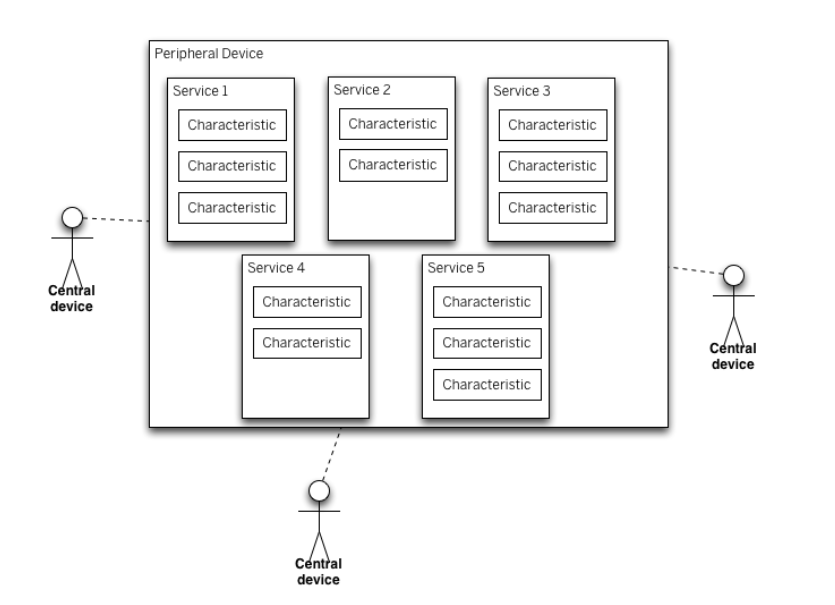


Otherwise, if you have no background of C/C++/Arduino coding at all, strongly recommend reading the official Arduino 33 BLE getting started page <https://www.arduino.cc/en/Guide/NANO33BLE#toc8>. Follow up each step and understand the inbuild examples before further adjust/rewrite ketone project codes.

### Arduino Nano 33 BLE Bluetooth 4.0 module

‘BLE’ stands for Bluetooth Low Energy. It is optimised for low power use at low data rates and was originally designed to operate from lithium coin cell batteries. Ideally, it is the fittest Bluetooth choice for our breath analyser.

Unlike standard Bluetooth communication basically based on an asynchronous serial connection (UART). Bluetooth LE acts as a host device/advertising screen (called a peripheral device in Bluetooth LE) to keep publishing/advertising information while the module is enabled. And phone/app (called a central device in Bluetooth LE terms) is working like audience, it can receive/read any of information which it cares.



Central devices are clients. They read data from peripheral devices. Peripheral devices are servers. They provide data from sensors as readable characteristics. A BLE peripheral will provide services, which in turn provide characteristics. Services are identified by unique numbers known as UUIDs. You know about UUIDs from other sources (check its definition) and custom services have a 128-bit UUID. You can customise your own services that contain Acetone reading characteristic which was already successfully implemented in current design (link to the short video).

**Compiled components/modules**

All the source codes can be found on the repository separately. (what materials supposed to have currently in my mind are 1. One complete code for design 10, break the complete code to induvial modules based on different components). Rather than covering code at all, this section will briefly introduce the components we have built in our design.

1. **Wake-up button**

It is used for waking up the device for each time usage, the heating element will automatically start.

1. **Buzzer**

Use ‘Piezo buzzer’ as the additional signalling to indicate a change of device status (eg. to signal device is ready for a breath, or to signal that a result has been determined, 'Beep' sound)

1. **Contacted Temperature sensor**

This temperature sensor is fixed on the Acetone sensor to monitor its temperature in real-time.  the Arduino Nano might guarantee the Acetone sensor is working in a stable temperature range (345-355 degree) based on monitored readings when user taking a breath test.

1. **LED**

‘**Blink**’: Warning user the device is heating up; ‘**ON**’: the result of the breath test is invalid.

1. **OLED screen**

The Screen is the core display platform when users do the breath test without a smartphone. It not only can display the changes of device status (switch between animated icons and text), but also allow user directly monitor the Acetone temperature during heating or have numbers showing the Acetone reading once finish the test.

1. **CO2 sensor**

The readings from the CO2 sensor are used for further determining whether user has taken a deep breath or not.

1. **Acetone sensor**

The Arduino Nano will receive the measurement of changes in current that sense by Acetone sensor, then do the mathematical conversion between the raw value and the concentration of Acetone in ppm(unit)