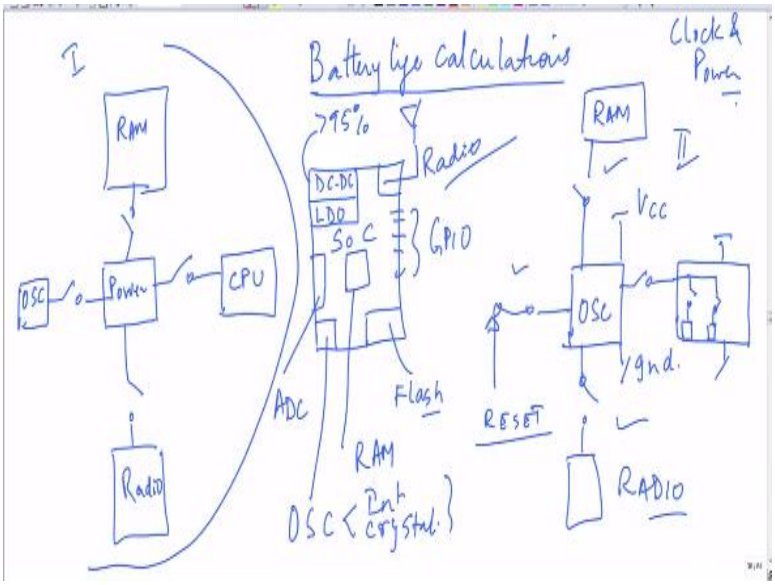
Design for Internet of Things
Prof. T V Prabhakar
Department of Electronic Systems Engineering
Indian Institute of Science-Bengaluru

Lecture - 30
Introduction to Low Power Software

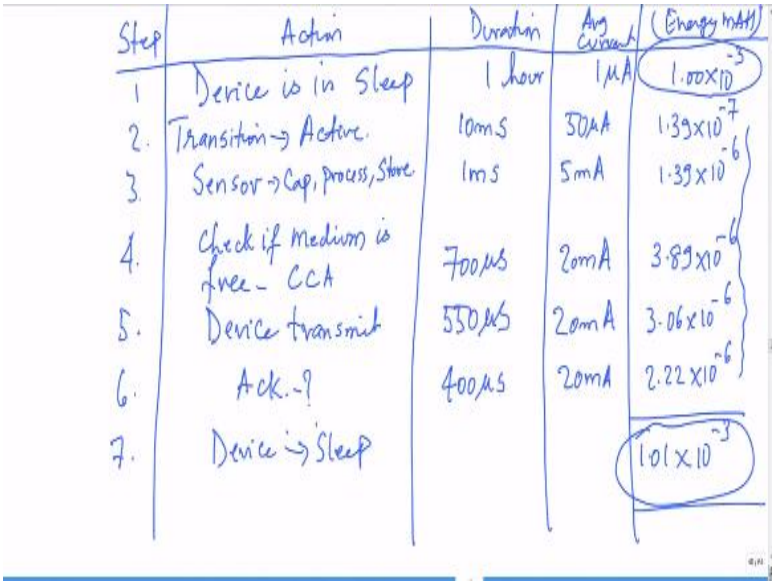
I want to introduce you to this topic of power management algorithms. Remember one familiar picture that I have been drawing and I want to draw your attention to this one.

(Refer Slide Time: 00:39)



What does this say? There is an SOC here. There is an ADC where you will be connecting all your sensors and you would essentially try to do a battery life calculation by doing the following, right?

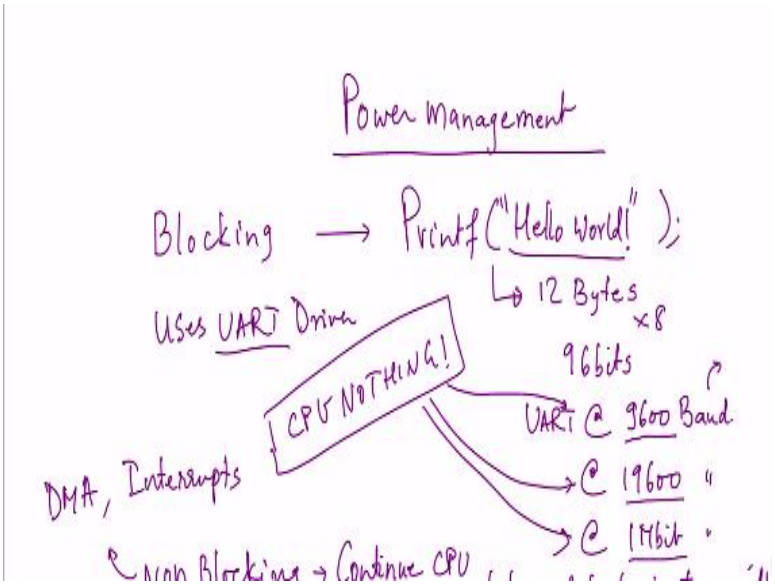
(Refer Slide Time: 00:54)



Devices in sleep. One hour, one average current is one microampere. Energy milliampere is given in this, milliampere hour, the unit for energy is this. Essentially it is (amp x time). If you multiply this into the voltage you will actually get the energy. So because it is (voltage x current x time). But usually energy milliampere hour is mentioned because V is assumed to be, voltage is assumed to be constant.

Idea is that the currents that you are measuring are because of the fact that you wrote good software, good power management software, okay? Take this case ADC. Question is what was the software that you wrote to acquire data? Nobody knows this, right? Unless you design your device driver properly, unless you make your low level driver properly, you are not going to achieve the kind of low power that is required for your IoT node. What does it mean is the question.

(Refer Slide Time: 02:04)



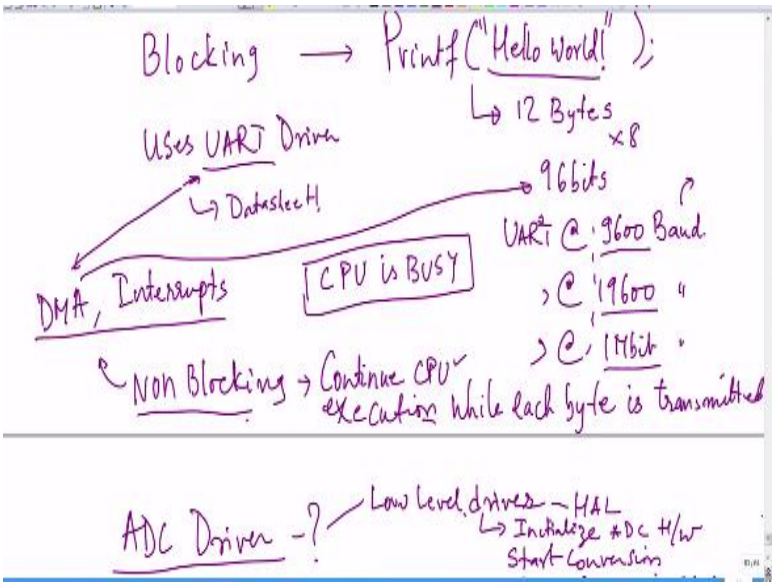
Let me draw your attention to this picture, okay? Take the very simplest possible example. You have let us say a printf statement which says “hello world” right? This hello world is essentially 12 bytes, which means it is 96 bits. And this is an output statement which means you are expecting it to print it to let us say a screen or on some output device and you would typically use a UART driver.

Let us say your UART hardware can only support 9600 baud, okay. Similarly 19,600. Similarly, one megabit per second, okay. What is essentially saying is this is an old UART, it supports very slow rate. And as you move on, it is increasing its baud rate 19,600 and 1 megabit. What would you do is the question. Now, I will give you an exercise.

If your UART hardware supports only 9600 baud, what is that time associated for transfer of these many bytes is the question. Now how much will it be if it is 19,600? What will the time be if it is one megabit baud right, is the question. Obviously, I would not do the calculation, but I am sure you will appreciate that a slow UART driver would take away quite a bit of time for it to transfer this 12 bytes.

And what is the CPU doing during that time? CPU is doing nothing. That is the problem, okay. But amount of time CPU is doing nothing is very small if it is here, for sure. But here is okay. But here is significant, is it not? That is what we are trying to say. Therefore, how should you write the driver? This is a very simple example. If it is ADC is even more complicated. I will tell you why.

(Refer Slide Time: 04:22)



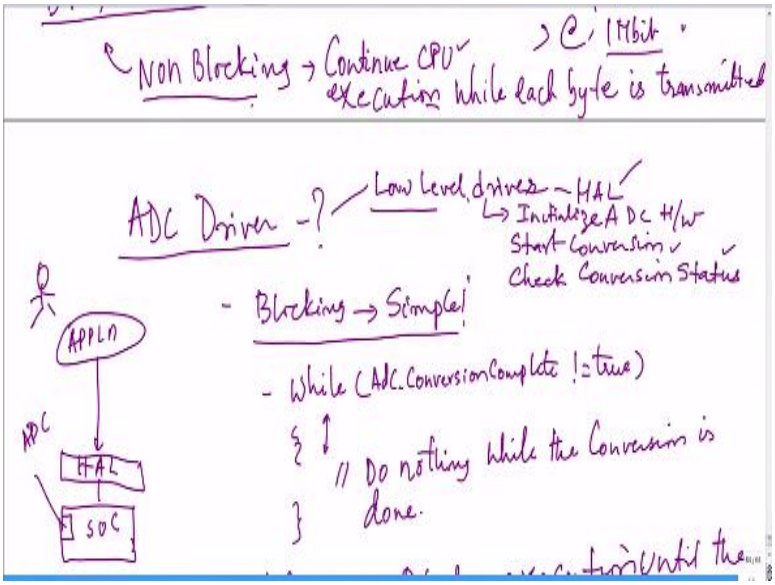
As we go along, you will realize that you should never write a driver which is looking like this blocking kind of driver. This is actually a blocking driver because CPU is doing nothing during that time. Therefore, the best way would be to write non-blocking. And how do you do it with non-blocking? You would either use interrupts or you would use DMA.

Now DMA with UART will be a good combination. But you have to check your controller whether DMA and UART can be used together. In other words, whether DMA can be configured for any UART operation you have to find out from your controller and that is where the data sheet will come into picture in a big way.

So please look up the data sheet and its capabilities to know whether UART can actually get configured via the DMA so that DMA can take care of transferring these 96 bits, whatever be the baud rate, so that CPU can continue to do its execution. CPU can, CPU is busy. Earlier I drew CPU is doing nothing. Now I am saying CPU can continue to be busy doing whatever it has to, okay, if you configure for DMA or you can also use interrupts.

Interrupt is another way. This way, essentially you are trying to do non-blocking where the CPU will continue execution while each byte is being transmitted via the UART port to the output, okay?

(Refer Slide Time: 06:27)



So this can be an issue and while this printf was a very simple statement for you ADC can start getting very complicated, okay? It is possible that the ADC sensor is not free, okay? Sensor is not free, you have written a blocking ADC driver, the device driver is blocking. So CPU will never come out, okay.

Or maybe at that instant when ADC was doing a conversion, it got stuck for some reason and the conversion never took place. So now you are waiting for ADC sensor input to be converted to some code word eternally, right? And what will happen when you switch it on eternally? Battery is getting consumed continuously. And that will have an impact on the lifetime of your system.

So therefore, think about everything when you talk about power management, think about doctors, think about stethoscope, right? You need a steth equivalent, which you put on the patient to find out how the patient is performing, right? So you need tools like that, which will tell you something about the way you wrote your software. Where is it getting stuck, what is happening?

This information, this clarity should come in everything that you do. But before we go and see anything, let us continue the ADC driver, device driver. You can basically divide the ADC driver into a low level driver, which essentially can take care of some form of hardware abstraction, right? It can do hardware extraction. It provides a hardware extraction to the application.

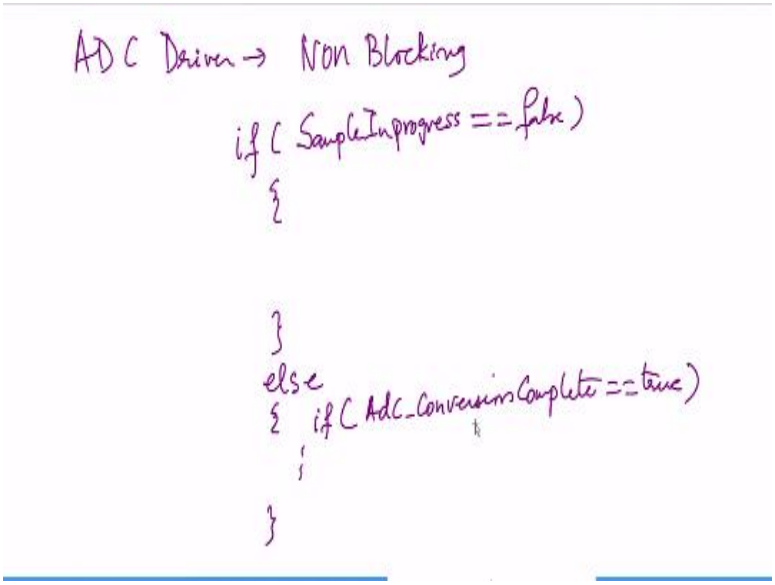
So the application, you write your application as though you are talking in a generic way to the, this is your HAL and this is your SOC and in this SOC this is your ADC, okay? So here you are hiding everything as far as this application is concerned. Human is sitting here, okay? You are only writing an application to the, a layer a software layer called the hardware abstraction layer.

What does it do? It initializes the ADC hardware. It does initialization of ADC hardware. It can do start conversion. It can check the conversion status. All of this is possible. But now you have to see how you are writing that your low level driver, if you want to use this HAL in an effective way. Let me give you an example of what ADC blocking can be. Blocking codes are damn simple, very simple.

All you need is a while loop, okay? What are you doing? While ADC conversion is not completed, do nothing while the conversion is done. You are in an eternal loop. So simple. You continue to wait until the ADC conversion is complete. And once it is done you come out. Well that is not obvious from anything here. It is just entering the while loop and waiting there eternally till the conversion is completed.

This is blocking. Something happens to the sensor, something happens to the ADC, this while loop will continue to get executed and it will never come out. So it is blocking but while it is blocking it is also quite simple in a way.

(Refer Slide Time: 09:57)

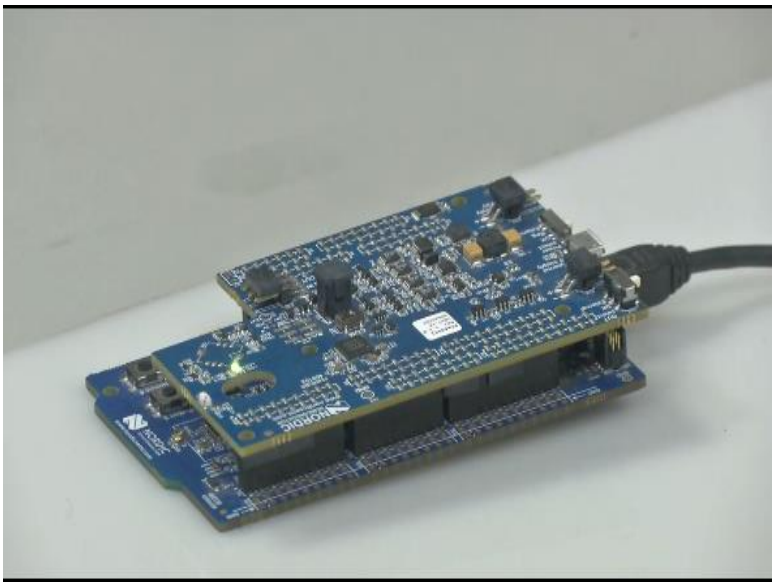


This is a non-blocking driver. What have you done? You got rid of the while loop and you replaced it with a if loop okay, you have replaced it with if. So it is completely non-blocking. Sample in progress is false, you enter the loop else. If there is an if else loop, then again you check if ADC conversion complete is not true or is true, you do something and you come out.

Essentially, you are replacing the while loop with a if else loop. And that will take care of broadly trying to be non-blocking by nature, okay. So that is what I wanted to say. Now coming to the steth part of it, you need a steth equivalent. You need to check right? How do you know whether your code is performing well or not? For that, you definitely need a development board.

And you develop, you definitely need some amount of hardware tools, which will allow you to debug, check your software, see if your systems are working well. That you must be, you must basically evaluate it.

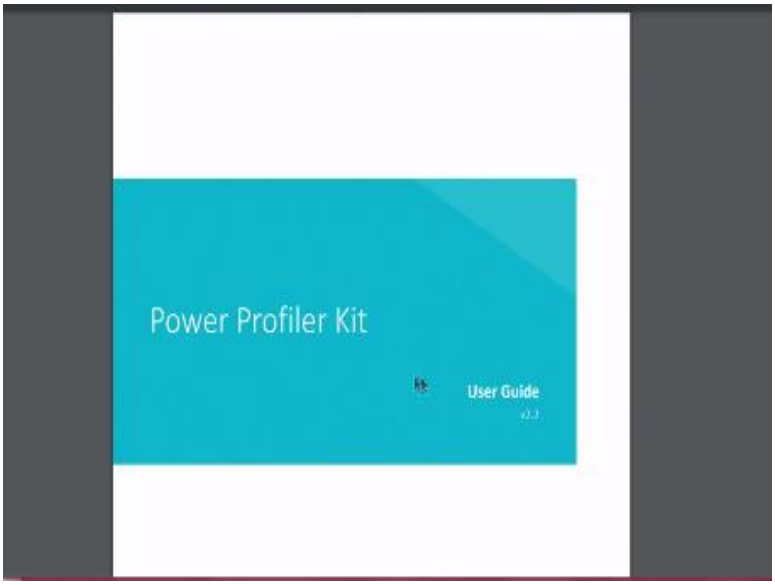
(Refer Slide Time: 11:14)



For that what I want to show you, point you to this piece of hardware has two boards, okay. Down is a development board. This is a normal nordic development board. You can either use 52832, or you can use 52840. Any one of the two boards you can use. This is a development board. Top of this is what is known as the power profiler kit, it is called PPK. And this is version 1.

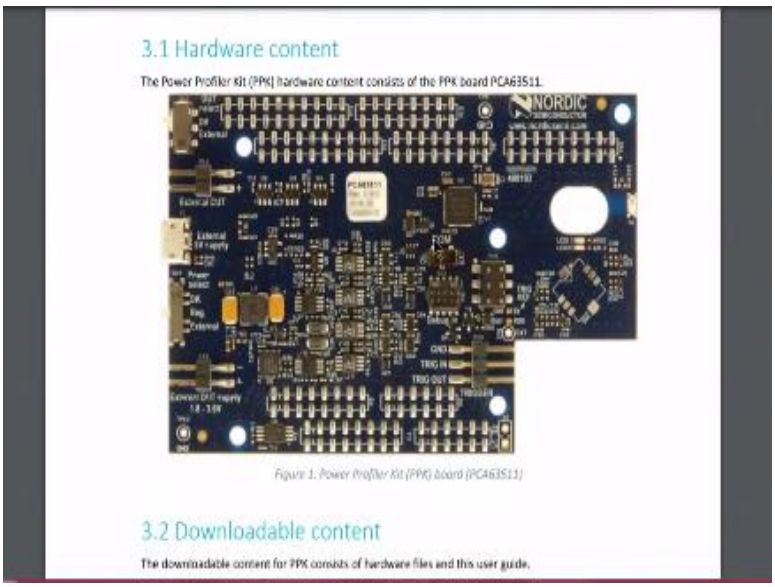
You see there are some headers on this. And there are mating headers on the top. Dump your development code here and check if your development code is functioning as per the energy requirements by using this piece of hardware, which is like I mentioned, like a stethoscope. And this PPK essentially will give you a current profile. It will tell you if there are any issues, okay. So let us see, what is this power profiler kit?

(Refer Slide Time: 12:15)



For that I point you to this screen here, okay. This is a user guide. This is from Nordic. I chose Nordic because we use Nordic quite comfortably in the lab. It tells you a lot of details about this kit. I do not think we have time to cover every aspect of this.

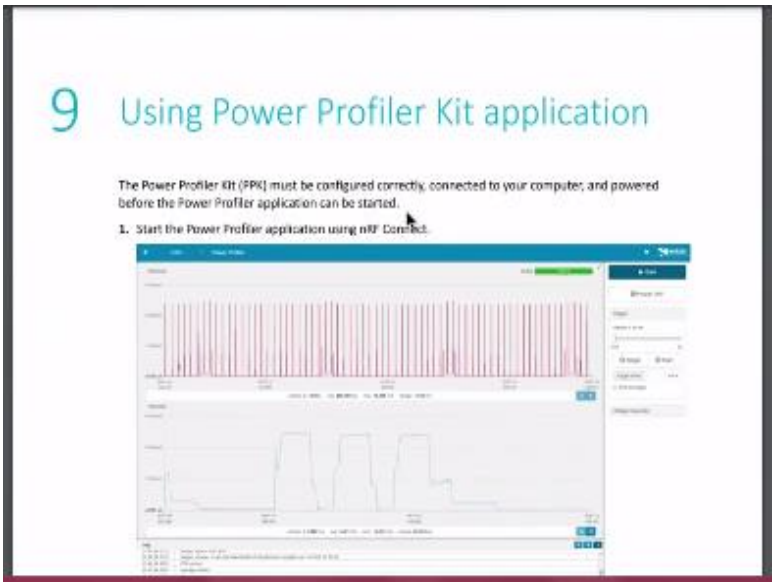
(Refer Slide Time: 12:32)



But let me point you to the bare minimum things that you need to know. But I would strongly encourage you to study this manual yourself to understand how actually the power profiler kit works along with the development board. There is lot of software, there is a quick start guide and it gives you an overview of what are the measurements it does, okay? What are the pin outs?

How do you connect it? How do you do power selection? So many related configurations are out there. You have to read the manual carefully, and then dump your actual testing software to see how good is your software in terms of its power consumption, right? So that is what we have to do.

(Refer Slide Time: 13:22)



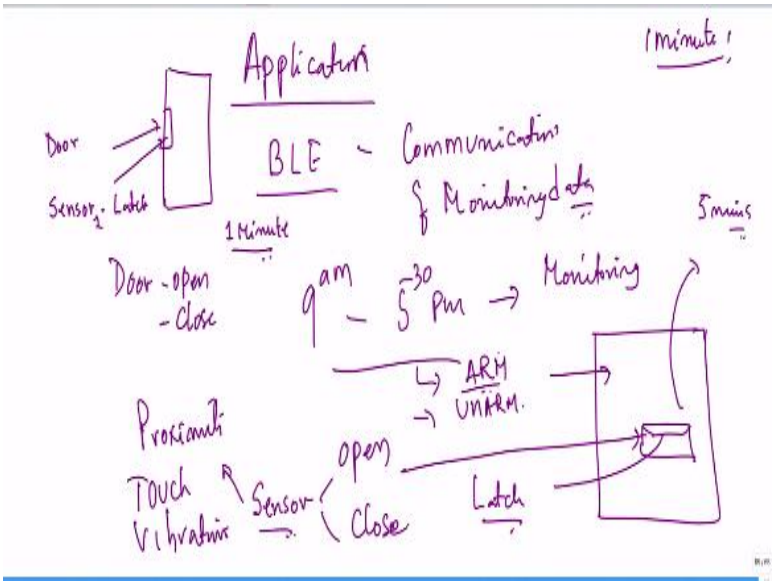
So let me point you to this particular screen here. See this screen is very useful, folks. Because it will tell you all that you did with with respect to your software. How good is this? How good did you write it? Top one tells you something, bottom one tells you something else, okay? But very importantly, read the fine print here. You cannot perhaps see it so well on the screen.

But I can tell you it is realing out numbers. Average current it is saying is 250 microampere, it is right here, okay? It tells you the maximum current which is 12.4 milliamperes, okay. It is also telling you in terms of charge, which is in microcoulombs 1.753 microcoulombs. It is telling you the measurement window over which it did, okay. And similarly, it is also telling you with respect to time on the x axis and current on the y axis.

It is telling you what is the average current. It is telling you the maximum current. It is telling you the microcoulombs over that window of whatever 5.850 milliseconds. Basically it is expanding this screen here and showing you in great detail the current consumption, okay. Very useful, right? Otherwise, how will you know what exactly is happening with your embedded software, which you developed.

How good is your driver, right? So these issues will have to be measurable and you should be able to quantify them. This is the key takeaway from this particular aspect, okay? So now before we go and see a demonstration of the actual system, let me show you what exactly the application that we have in mind.

(Refer Slide Time: 15:18)



The application is very simple, okay? The application is, see folks, you can have many cases where you are using Bluetooth Low Energy from communication of monitoring data. Think about the following folks. You get up in the morning, you come to office, you are working between 9am and you are working till let us say 5:30pm. During this period, you want to do some monitoring, okay.

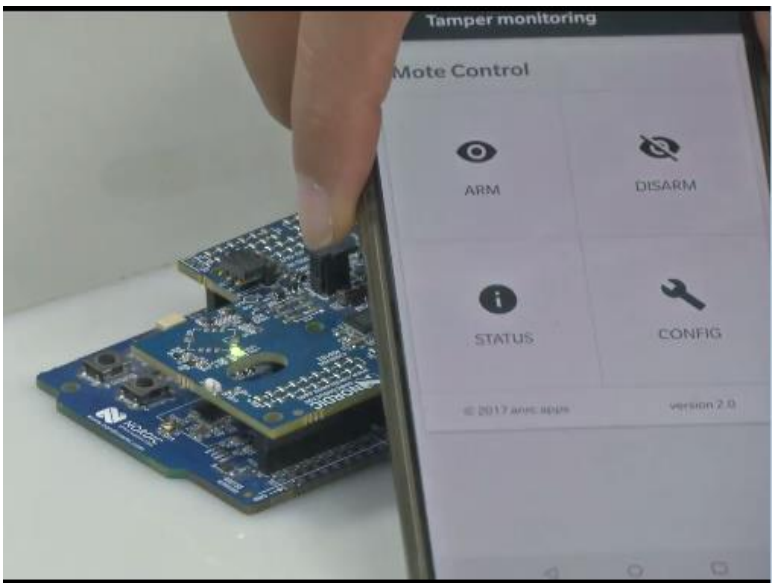
I will give you an example. You want to let us say, monitor a latch of your door. To open the door, there is access to some, let us say radioactive material. So you just basically want to monitor a closed area and you want to monitor if someone pressed, opened a latch, okay. Now let us say, I put a sensor for detecting a latch open or close. Well, I can put any type of sensor.

I can put proximity, okay. I can put that sensor. I can put vibration sensor. I can do anything, right? And I will want to, you know I want to monitor this continuously. If someone opens it, I should know it in, let us say in one minute I want to know. No one should enter this place. And so you should not be here for more than, let us say 5 minutes, okay?

But you need an alert in one minute that someone entered, someone touched the latch. Alright. But touching the latch need not mean that the door has opened, right? People can play mischief. So that is up to you, how you detect, not only the latch was touched, the door was also opened. So you need not only latch but also door open close. So I will say door open close. Okay, very simple.

You now go back and abstract it. You have an ADC. This ADC has two ports, one sensor for latch, another sensor for door open close, right? And you need this information in exactly one minute. Now what should it do? Beyond 5:30 is the question. Therefore, your application starts by arming it between 9 and 5:30 and unarming it after 5:30pm. So there is no need to arm any latch sensor, okay. So that is what you want to do.

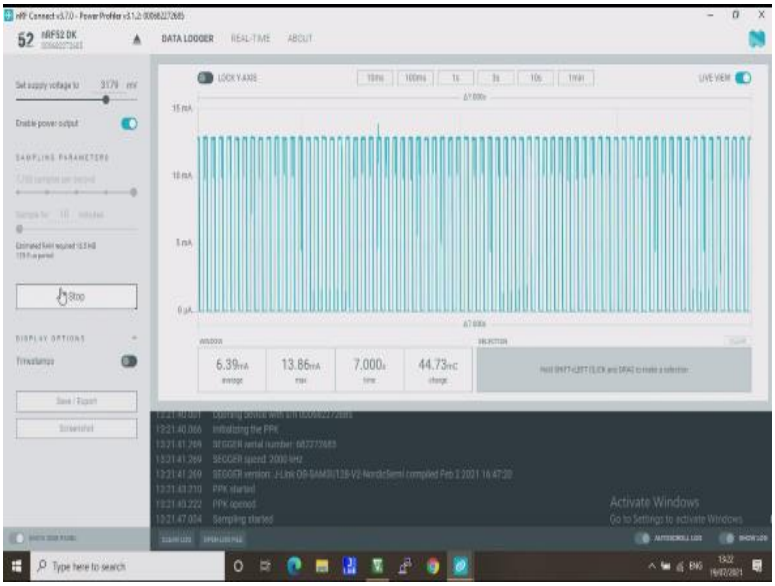
(Refer Slide Time: 18:17)



Now this is the screen of interest. You see, there are four buttons on the app. Mode control, ARM, DISARM, STATUS, and CONFIG. These are the four buttons. Remember folks, we wrote a full Android app to do these operations and we will be

controlling a development board and on top of the development board is this power profiler kit, okay. So now your hardware for your demonstration is these three. How does the screen look? And how do we start everything?

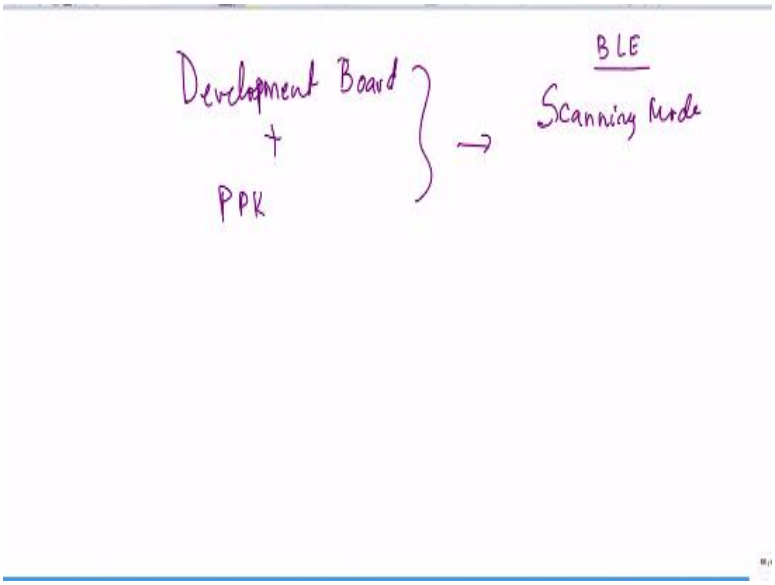
(Refer Slide Time: 18:57)



For that let us go to this window which is another laptop and let us start the demonstration from the beginning. Let us go slowly and let us start by section by section, okay. Now you see we are loading the system and we will now start demonstration.

You see now what is actually happening is the sensor node development kit along with the power profiler kit is essentially showing you some amount of data. It is showing you along the x axis is time. Along the y axis is current.

(Refer Slide Time: 19:48)

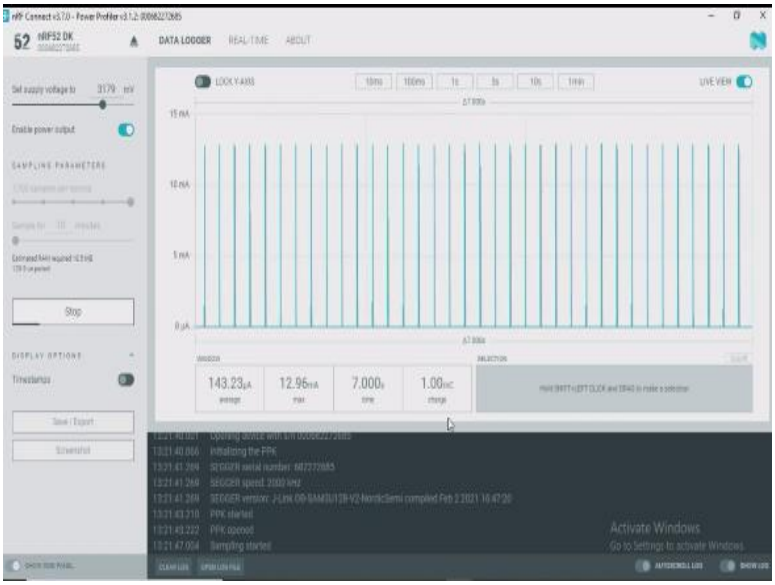


It is telling you the fact that the node is in what is known as, let me write it here the receiver node, the mode of interest, the development board, along with PPK is demonstrating that it is in scanning mode, okay. Do not worry so much about it right now because this is something that when we study Bluetooth Low Energy, we will be studying about it.

But right now what it is showing you is the scanning mode. In the scanning mode, what is happening? Now let us look at this screen here. If you see carefully on the screen, in the second button you see that it is taking a maximum of 13.03 milliamperes and average current is 6.4 milliamperes, time is 7 seconds, charge in microcoulombs is 44 point some seven six or seven five microcoulombs.

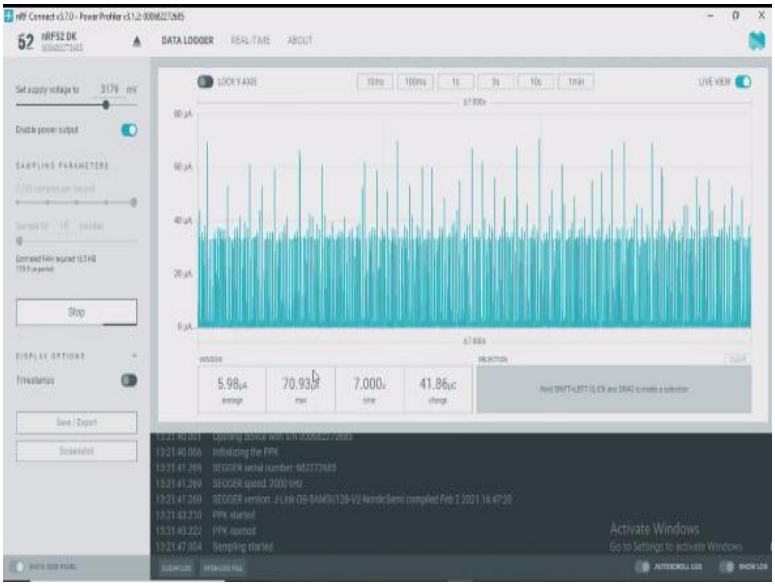
Now what I will do, I will take this phone and I will press a button on this phone, which will arm the system. Great, there you are.

(Refer Slide Time: 20:59)



Now you see moment I gave arm command it went into a peripheral mode and it started communicating back to my mobile phone and you will see that it is in active mode for 30 seconds and it is in sleep mode for 30 seconds and the current consumption is maximum of 12.94. And the system is now going into something dramatic.

(Refer Slide Time: 21:30)



Now you see, you will be able to see that this is sleep mode. In the sleep mode for 30 seconds, let us see this should go also to 30 seconds. It is consuming 70.81 microamperes. Average is just 5.9, about 6 microamperes, right? For 30 seconds, it will last that long. And then you are back again folks. It is back to 13.23 maximum and 6.38 minimum, right. So it has gone back to what is known as the scanning mode. So that is the important part.

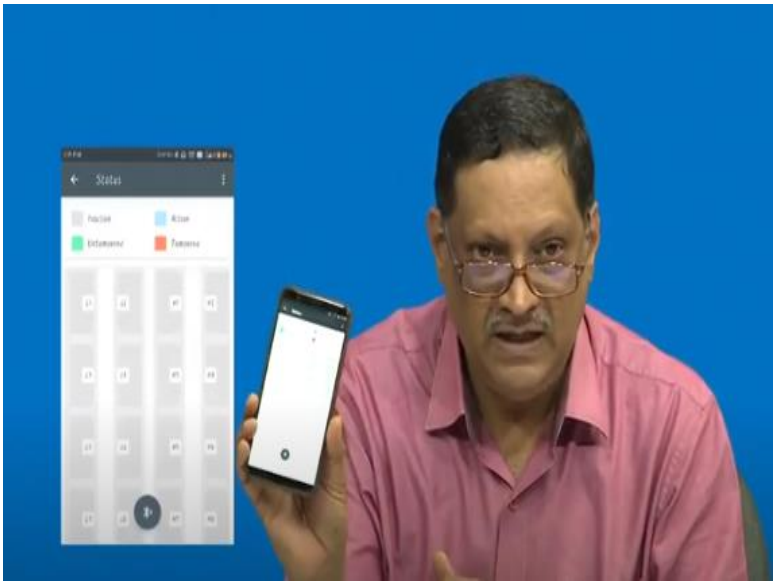
So now what is actually happening is during the period, when you go back to arming mode, when you it goes back to peripheral mode, it is as I mentioned 30 seconds in active and sleep. Any action on the latch that happens between either the sleep or the active time is immediately indicated to the mobile phone. Well, if it is in active mode, there is no requirement to do anything special.

But if the node is in sleep mode, it has just went to sleep when there was an activity on the latch, it has to wait for the 30 seconds to elapse, then it will come back to active mode and send out the fact that the latch was handled or whether the latch was operated upon and so on. But if it is in active mode, there is no such requirement, it will immediately send it to the phone, my phone saying that a latch was opened.

So you see folks current consumption, application in mind, sensing, all of this is now bundled together. With these numbers that you see, you should be able to estimate how long does the battery last for this application, right? So that is the story behind this demonstration that we put together. So the other button that you see here is status. If you press the status button, see the application is designed in a very clever way.

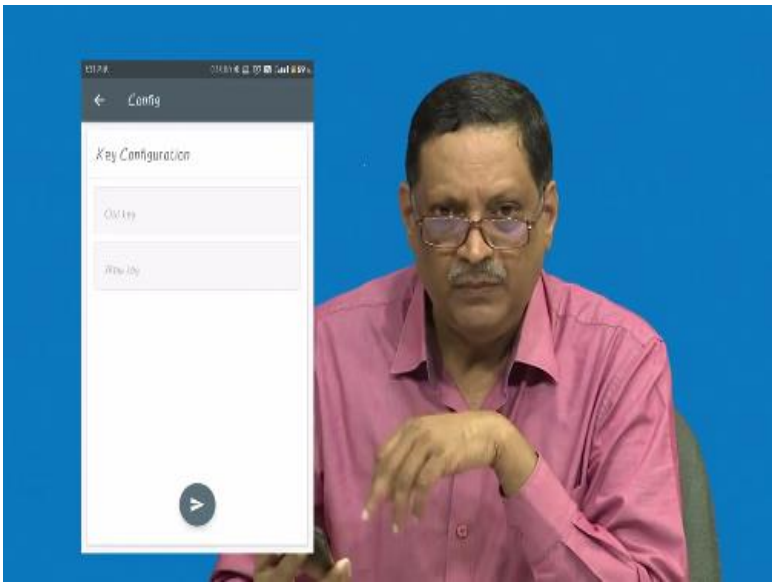
It will never do a transmission of the tamper status unless you press a button here, okay, that is the idea. Moment I press a button, it will say at what time the latch was opened and whether it was tampered or not. That information I get once I press this button.

(Refer Slide Time: 24:11)



You see now the screen changed and it will tell me what is the status, whether it was tampered or untampered, was it active, inactive and so on.

(Refer Slide Time: 24:23)



There is also this other button related to config. Config is nothing but about the keys, encryption keys, right? You can configure the encryption keys so that you have secure communication between the embedded device and the mobile phone.

So this demonstration, just putting together the sensors, the application of interest, which was to be tested for its current consumption on the development board using the power profiler kit, and checking whether you have coded your drivers, your application correctly, so that current consumptions can be monitored, used for estimation of lifetime of this application. Thank you very much.