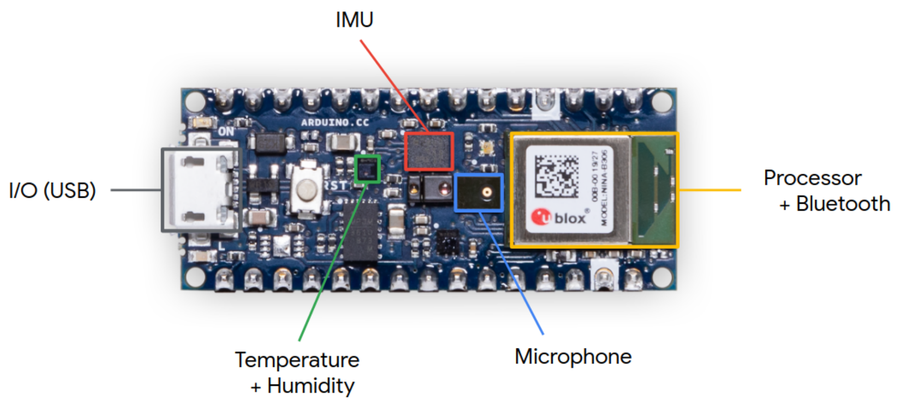
Embedded Systems

In this reading we will explore the diversity of embedded systems.

If you take a few minutes to look around the room you're in, you'll no doubt recognize the proliferation of embedded hardware within our world. Embedded systems are increasingly ubiquitous because they serve to complete specific computational tasks within their environment, allowing us to not only collect all sorts of data from distributed sensors but also to process the resulting information locally and act accordingly. That we can accomplish this *in situ*, or within the environment that this hardware is embedded within, at a fraction of the cost and physical scale of most general purpose computing hardware, opens the door to many advancements in automation and generally to the creation of smart, connected things.

While the specifications and capabilities of a given embedded system ought to be tailored for its application, in general we can look to the common construct for an embedded system that links sensing to processing and later actuation through the process of *transduction*, the conversion of one form of energy into another, for throughlines. In sensing, we convert energy from a physical phenomenon into an electrical signal we can go on to digitize and compute with. In actuation, the converse. In this course, we'll focus predominantly on the central element of said construct: the computing hardware, and review specifications and technical considerations that vary between implementations of embedded systems.



Development Boards

In the context of this course, we have talked about and deployed a microcontroller unit (or MCU) development board from Arduino, the Nano 33 BLE Sense. Here, we distinguish the nRF52840 MCU from the development board it is soldered to. What's interesting about this board is that despite its relatively small size, it is jam-packed with many of the representative elements of an embedded system, all on one printed circuit board (PCB): a collection of surface mount sensors, a form of actuator in the on-board programmable RGB LED, and an integrated MCU-BLE module, where BLE is an acronym for a branch of the Bluetooth standard called Bluetooth Low Energy. Further still, the PCB that the MCU is soldered to serves to 'break out' additional IO from the controller to provide interfaces for external, off-board modules and connections to daughter boards. So, while in many ways the Nano 33 BLE Sense is representative of an embedded system, it is separated from commercial implementations in that it serves no particular purpose, other than the open-ended development of an application that is. Put another way, a development board has the advantage of enabling many potential use cases. Further, you don't have to go through the process of designing circuitry, capturing schematics, or physical layout for such a board to get started prototyping a concept that could go on to be manufactured with specific intent later. The tradeoffs you make in employing a development board often involve board size and cost.

Board Size

Board size, and shape, have fairly meaningful implications for use in the field, given these simple parameters constrain where and how a system can be deployed. Some environments or contexts within which we'd like to embedded a system are forgiving (like the Google Home, say), but other manifestations (smart glasses, for example) depend wholly on the requisite hardware being quite tiny and perhaps of unique form. Somewhat obviously, board size is determined by the number and physical size, or package, of the components that live upon it. At a minimum, an embedded system must include a MCU chip alongside a power source, often a lithium polymer battery, and power circuitry. The nRF52840 MCU on the Nano 33 BLE Sense lives within a MCU-BLE module, the U-Blox NINA-B306, which spans 10 by 15 mm, dimensions that include a trace antenna. The MPM3610 step-down converter used to down regulate the 5V delivered to the board over USB occupies 3 by 5 mm, alongside small SMD passives. The entire board, meanwhile, spans 18 by 45 mm. While impressive, clearly a purpose-designed PCB could remove some of the sensors and IO breakout unnecessary for a specific application to reduce the overall scale even further, perhaps by about 60 to 70%.

Cost

Unsurprisingly, the cost of a development board scales with its MCU's compute capability and general feature set. In selecting the appropriate MCU for an application, it is important to remember that MCUs are often deployed to complete specific tasks, where the complexity it will face is predetermined or constrained. For very simple tasks, we highlight the ATTINY85 (an 8-bit processor with 8 kB of flash) that can, depending on the package selected, cost well less than $1 EA and even less at scale, perfect for simple computing requirements. In the context of TinyML, the AI-capable NINA-B306 module featuring the Nordic Semiconductor nRF52840 chip (an ARM Cortex M4) and all-in-one BLE hardware (including antenna) costs about $10 EA, and is more generally representative. In view of the on-board sensors and design costs, the Arduino Nano 33 BLE Sense costs just over $30, not even an hour's time of an electrical engineer who might design such a board — a tremendous value. In general, the boards we originally considered for this class span from $2 (BluePill) to $54 (Disco-F746NG).

You can find a list of the boards we considered in the table below. Ultimately, our staff selected the Arduino Nano 33 BLE Sense for its versatility, in providing a large selection of on-board sensors, accessible IO via breakout pins, and a BLE module for projects that involve wireless communication, with reasonably representative compute specifications for resource constrained hardware. We'll explore the compute specifications in more detail in the next video and reading.

TinyML Development Board Comparison

▨ officially TFLM supported ▢ unofficially compatible boards

Board	MCU	CPU	Clock	Memory	IO	Sensor(s)	Radio
Arduino Nano 33 BLE Sense	Nordic nRF52840	32-bit ARM Cortex-M4F	64 MHz	1 MB flash 256 kB RAM	x8 12-bit ADCs x14 DIO UART, I2C, SPI	Mic, IMU, temp, humidity, gesture, pressure, proximity, brightness, color	BLE
Espressif ESP32-DevKitC	ESP32 D0WDQ6	32-bit, 2-core Xtensa LX6	240 MHz	4 MB flash 520 kB RAM	x18 12-bit ADCs x34 DIO** UART, I2C, SPI	Hall effect, capacitive touch***	WiFi, BLE
Espressif EYE	ESP32 D0WD	32-bit, 2-core Xtensa LX6	240 MHz	4 MB flash* 520 kB RAM	SPI via surface pads	Mic, camera	WiFi, BLE
Teensy 4.0	NXP iMXRT1062	32-bit ARM Cortex-M7	600 MHz	2 MB flash 1 MB RAM	x14 10-bit ADCs x40 DIO** UART, I2C, SPI	Internal temperature, capacitive touch	None
MAX32630FTHR	Maxim MAX32620	32-bit ARM Cortex-M4F	96 MHz	2 MB flash 512 kB RAM	x4 10-bit ADCs x16 DIO UART, I2C, SPI	Accelerometer, gyroscope	BLE

*this board also features 4 MB flash and 8 MB of PSRAM external to the MCU, **shared programmable functions, ***with external touch pads

Board	ASIC	DSP	Clock	Memory	IO	Sensor(s)	Radio
Himax WiseEye WE-I Plus EVB	HX6537-A	32-bit ARC EM9D DSP	400 MHz	2 MB flash 2 MB RAM	x3 DIO I2C	Mic, accelerometer, camera	None

We include the Himax WiseEye as an officially supported example of hardware optimization for TensorFlow Lite that calls on an application specific integrated circuit (ASIC).