

uTensor

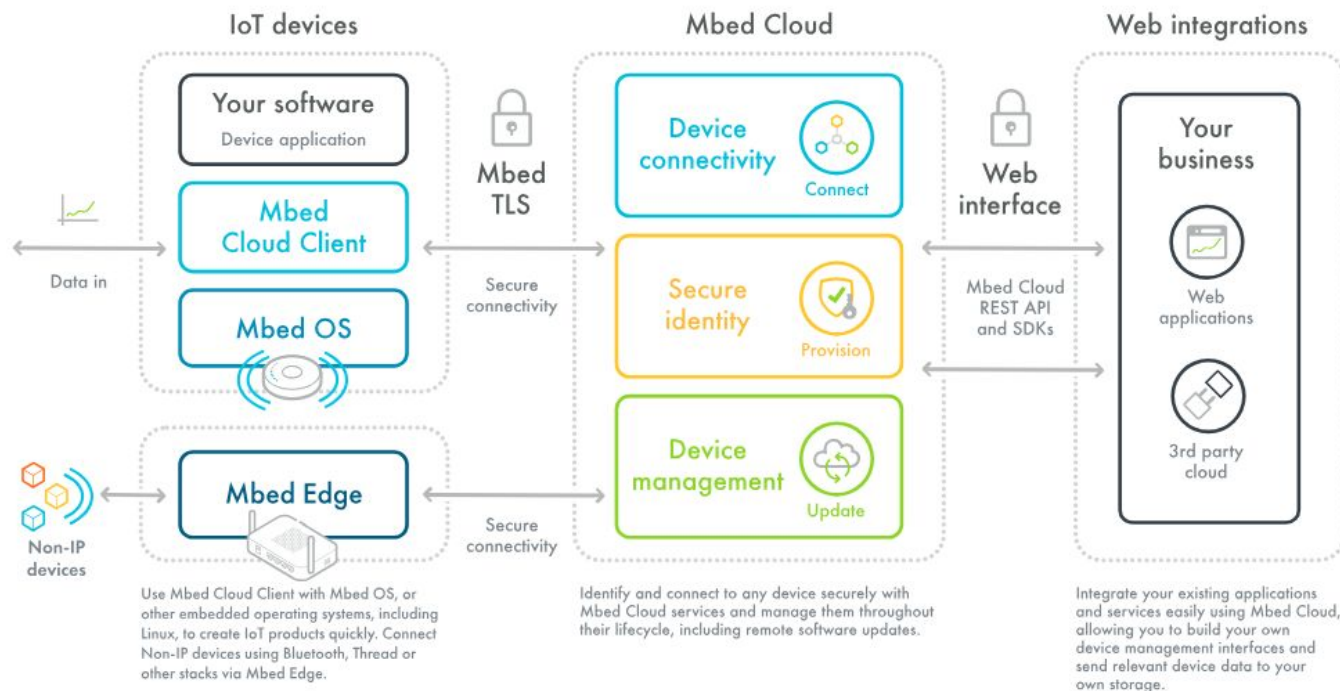
Tensorflow for microprocessors

Emilio Jose Coronado Lopez
#SeoulAI

arm MBED

<https://www.mbed.com/en/>

How Mbed Works



<https://os.mbed.com/platforms/>

OS Home

Hardware ▾

Code

Documentation ▾

Questions

Forum

Compiler

Boards

Filter

Mbed Enabled

☐ Mbed Enabled

Mbed OS support

☐ Mbed OS 2

☐ Mbed OS 5.4

☐ Mbed OS 5.5

☐ Mbed OS 5.6

☐ Mbed OS 5.7

☐ Mbed OS 5.8

Target vendor

☐ AnalogDevices

☐ ARM

☐ Maxim Integrated

☐ Nordic Semiconductor ASA

☐ Nuvoton

☐ NXP Semiconductors

☐ Realtek

☐ Renesas

☐ Silicon Labs

☐ STMicroelectronics

☐ Toshiba

☐ WIZnet

Platform vendor


☐ AnalogDevices

☐ ARM

☐ Avnet Silica

☐ BRC Make it Digital Campaign

Boards




arm

Mbed

Enabled

mbed LPC1768

- Cortex-M3, 96MHz
- 512KB Flash, 32KB RAM




arm

Mbed

Enabled

mbed LPC1114U24

- Cortex-M0, 48MHz
- 32KB Flash, 8KB RAM




arm

Mbed

Enabled

FRDM-KL25Z

- Cortex-M0+
- 128KB Flash, 16KB RAM
- USB OTG



arm

Mbed

Enabled

NXP LPC800-MAX

- Cortex-M0+
- 16KB Flash, 4KB RAM

<https://os.mbed.com/cookbook/Homepage>

[OS Home](#)

[Hardware ▾](#)

[Code](#)

[Documentation ▾](#)

[Questions](#)

[Forum](#)

[Compiler](#)

[Cookbook](#) » [Homepage](#)

Homepage

Introduction and Help

- [About the Cookbook](#) - What it is for, how to use it, and how you can contribute
- [deadmbed](#) - Having trouble with your mbed working?
- [Course Notes](#) - Course notes being developed to support workshops, lectures and self learning
- [Short Course Notes](#) - Course notes used in [Ngee Ann Polytechnic](#).
- [IoT DeepDive Workshop Meetup](#) - slides and written documentation for an mbed centric meetup series covering various IoT Technologies.
- [WikiSyntax](#) - for [developer.mbed.org](#) (how to add links to posts, inline code in posts, add videos, general markup) very useful for creating good forum posts and code documentation on notebook pages

Components and Libraries

This section is for information about different reusable building blocks; primarily components and the libraries, code and information to make use of them. For more about Libraries, see [Working with Libraries](#).

See also the main [components database](#).

TCP/IP Networking

Table of Contents

1. [Introduction and Help](#)
2. [Components and Libraries](#)
3. [TCP/IP Networking](#)
4. [USB](#)
5. [Custom Peripheral Drivers](#)
6. [LCDs and Displays](#)
7. [Audio](#)
8. [Wireless](#)
9. [Motors and Actuators](#)
10. [Sensors](#)
11. [Compass](#)
12. [NFC/RFID](#)
13. [Barcode](#)
14. [Temperature](#)
15. [Clocks and Oscillators](#)
16. [External ADC/DAC](#)
17. [Interfaces and Drivers](#)
18. [Storage, Smart Cards](#)
19. [Magnetic, Proximity Card Readers](#)
20. [Digital Signal Processing](#)

<https://github.com/janjongboom/mbed-simulator>

Arm Mbed OS simulator

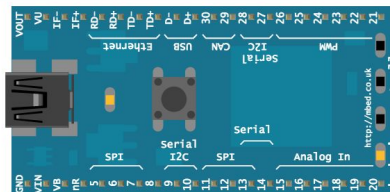
GitHub project

Blinky

[Load demo](#)

Run

```
1 #include "mbed.h"
2
3 DigitalOut led(LED1);
4
5 int main() {
6     while (1) {
7         led = !led;
8         printf("Blink! LED is now %d\n", led.read());
9
10        wait_ms(500);
11    }
12 }
13
```



Serial output

[illegible]

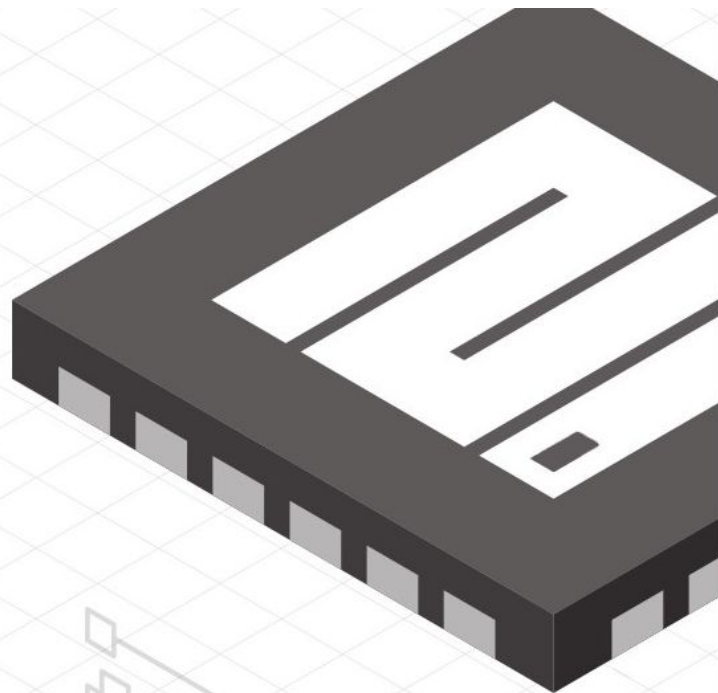
MicroPython

MicroPython is a lean and efficient implementation of the **Python 3** programming language that includes a small subset of the Python standard library and is optimised to run on microcontrollers and in constrained environments.

The MicroPython **pyboard** is a compact electronic circuit board that runs MicroPython on the bare metal, giving you a low-level Python operating system that can be used to control all kinds of electronic projects.

MicroPython is packed full of advanced features such as an interactive prompt, arbitrary precision integers, closures, list comprehension, generators, exception handling and more. Yet it is compact enough to fit and run within just 256k of code space and 16k of RAM.

MicroPython aims to be as compatible with normal Python as possible to allow you to transfer code with ease from the desktop to a microcontroller or embedded system.



TEST DRIVE A PYBOARD

BUY A PYBOARD

USE MICROPYTHON ONLINE

JerryScript

A JavaScript engine for **Internet of Things**

JerryScript is the lightweight JavaScript engine intended to run on a very constrained devices such as microcontrollers:

- Only few kilobytes of RAM available to the engine (<64 KB RAM)
- Constrained ROM space for the code of the engine (<200 KB ROM)

The engine supports on-device compilation, execution and provides access to peripherals from JavaScript.

To learn more, please visit the [project on GitHub](#).

If you want to try JerryScript take a look at our documentation. It includes [Getting Started](#) guides and [JerryScript APIs reference](#).

Please report all bugs and feature requests on JerryScript [issue tracker](#). Feel free to ask questions on [issue tracker](#)

<http://jerryscript.net/>

<https://github.com/jerryscript-project/jerryscript>

Johnny-Five

The JavaScript
Raspberry Pi
Robotics & IoT Platform





Johnny-Five is the [JavaScript Robotics & IoT Platform](#). Released by [Bocoup](#) in 2012, Johnny-Five is maintained by a

<http://johnny-five.io/>

uTensor

Be the first to clip this slide



Machine learning
on microcontrollers


Jan Jongboom
Tech Power Summit
14 April 2018

1 of 34

Machine learning on microcontrollers - Tech Power Summit 2018

384 views

Share Like Download ...

 Jan Jongboom, Developer Evangelist IoT at ARM
+ Follow

in f t

<https://www.slideshare.net/janjongboom/machine-learning-on-microcontrollers-tech-power-summit-2018>

<https://github.com/uTensor/uTensor>

Extremely light-weight Deep-Learning Inference framework

Mbed importable runtime library.

[Utensor-cli](#): Offline-tool which generates embedded C++ code base on supplied, quantized-inference-graph.

<https://github.com/uTensor/uTensor>

Mbed Runtime Library

- Tensor Classes
 - Data Holder
 - Virtual Memory
- Operators Classes
 - C reference implementation
 - Basic operators: MatMul, Add, ReLu, Reshape, Max, Min, ArgMax, Quantization Ops, etc
- Context Class
 - A resource management class
 - An interface to utensor-cli's code generation
 - Describes a graph

Quantize

<https://petewarden.com/2016/05/03/how-to-quantize-neural-networks-with-tensorflow/>



How to Quantize Neural Networks with TensorFlow

MAY 3, 2016
By Pete Warden
in **UNCATEGORIZED**
52 COMMENTS



Picture by Jaebum Jo

I'm pleased to say that we've been able to release a first version of [TensorFlow's](#) quantized eight bit support. I was pushing hard to get it in before the [Embedded Vision Summit](#), because it's

FOLLOW @PETEWARDEN ON TWITTER

Tweets by @petewarden

Pete Warden Retweeted

[hardmaru](#)
@hardmaru

Video Compression through Image Interpolation, Wu et al. "Our deep video codec outperforms today's prevailing codecs, such as H.261, MPEG-4 Part 2, and performs on par with H.264."
arxiv.org/abs/1804.06919



[Embed](#)

[View on Twitter](#)

RSS - Posts

RECENT POSTS

[Enter the OVIC low-power challenge!](#)

[Speech Commands is now larger and cleaner!](#)

[The Machine Learning Reproducibility Crisis](#)

[Why Low-Power NN Accelerators Matter](#)

[Blue Pill: A 72MHz 32-Bit Computer for \\$2!](#)

RECENT COMMENTS

MNIST Demo

<https://github.com/uTensor/utensor-mnist-demo>

Basic MNIST handwriting recognition with uTensor

uTensor reduces AI-inference cost significantly, bringing AI to Cortex M devices.

This tutorial discusses at a high level what TensorFlow graphs are and how to start using uTensor to build a handwriting recognition application. This involves training a fully-connected neural network on the MNIST dataset on a host machine, generating embedded code, and building an mbed application that classifies handwritten digits based on user input.

Introduction

TensorFlow is an open-source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, and the graph edges represent tensors, a type of multidimensional data array.

uTensor translates these TensorFlow graphs into C++ code, which you can run on embedded devices.

Running artificial intelligence on embedded systems involves 3 main steps

<https://github.com/uTensor/uTensor>

Build a model on Tensorflow, and train it on a PC as usual.

Setting up a Mbed project with uTensor.

Convert the model to a quantized graph to C++ code with the uCli tool.

Import the graph into the main.cpp file to perform inference.

Compile and Copy the binary into the device.

Have Fun !!

Live Demo

Key Features

- STM32F413ZHT6 microcontroller featuring 1.5 Mbytes of Flash memory and 320 Kbytes of SRAM, in LQFP144 package
- On-board ST-LINK/V2-1 supporting USB re-enumeration capability
- USB ST-LINK functions:
 - Virtual COM port
 - Mass storage
 - Debug port
- 240x240-pixel LCD with parallel interface and touch-panel connector
- 8-Mbit PSRAM; 512K word x 16bits
- 128-Mbit Quad-SPI Flash memory
- I²S audio codec
- Jack connector for Audio line with microphone input and stereo output
- Two on-board ST-MEMS microphones
- Connector extension for up to 5-MEMS microphones.
- USB OTG FS with Micro-AB connector
- Connector for microSD™ card
- Integrated Wi-Fi® module 802.11 b/g/n



Thank you