CEC Continuing Education Center

DesignNews

Machine Learning in Microcontrollers

DAY 3 : Tools for Machine Learning in Microcontrollers

Sponsored by

Digi-Key ELECTRONICS

BENINGO
EMBEDDED GROUP

informa markets

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

THE SPEAKER

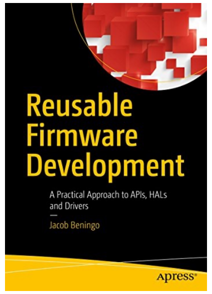
Jacob Beningo

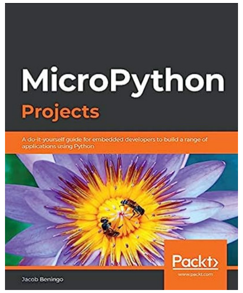
Visit 'Lecturer Profile'

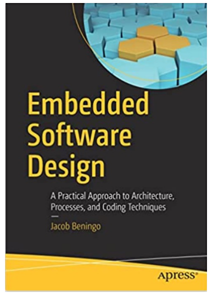
Beningo Embedded Group - President

Focus: Embedded Software Consulting and Training

Specializes in creating and promoting embedded software **excellence** in businesses around the world.


Reusable Firmware Development
A Practical Approach to APIs, HALs and Drivers
Jacob Beningo
Apress


MicroPython Projects
A do-it-yourself guide for embedded developers to build a range of applications using Python
Jacob Beningo
Packt
www.packt.com


Embedded Software Design
A Practical Approach to Architecture, Processes, and Coding Techniques
Jacob Beningo
Apress

Blogs for:
- DesignNews.com
- Embedded.com
- EmbeddedRelated.com
- MLRelated.com

Visit www.beningo.com to learn more …

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

Course Sessions

- AI and ML for Microcontrollers
- Writing Embedded Software with ChatGPT and Open.AI
- **Tools for Machine Learning in Microcontrollers**
- Training a Model for the STM32
- Deploying Machine Learning Models

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

1

ChatGPT to Debug Code

ChatGPT to Debug Code

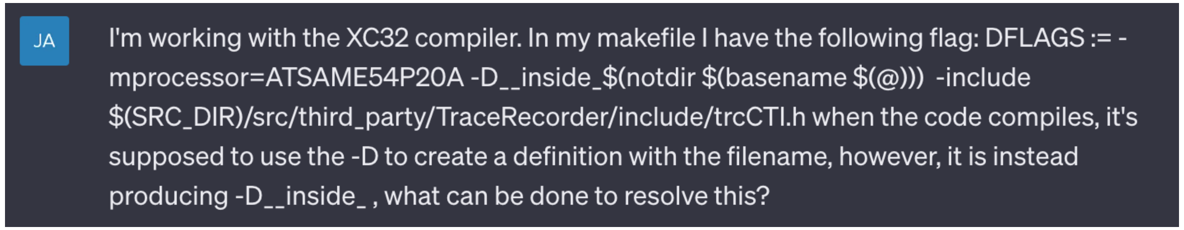
Example Debug Problem:

Integrating Percepio Tracealyzer into ThreadX, an RTOS, using a custom command line driven build system (designed for use with DevOps). The TraceRecorder requires a macro expansion to perform linker voodoo for function replacement. `D__inside_$(notdir $(basename $(@)))`

The macro expansion during compilation produces `__inside_`

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

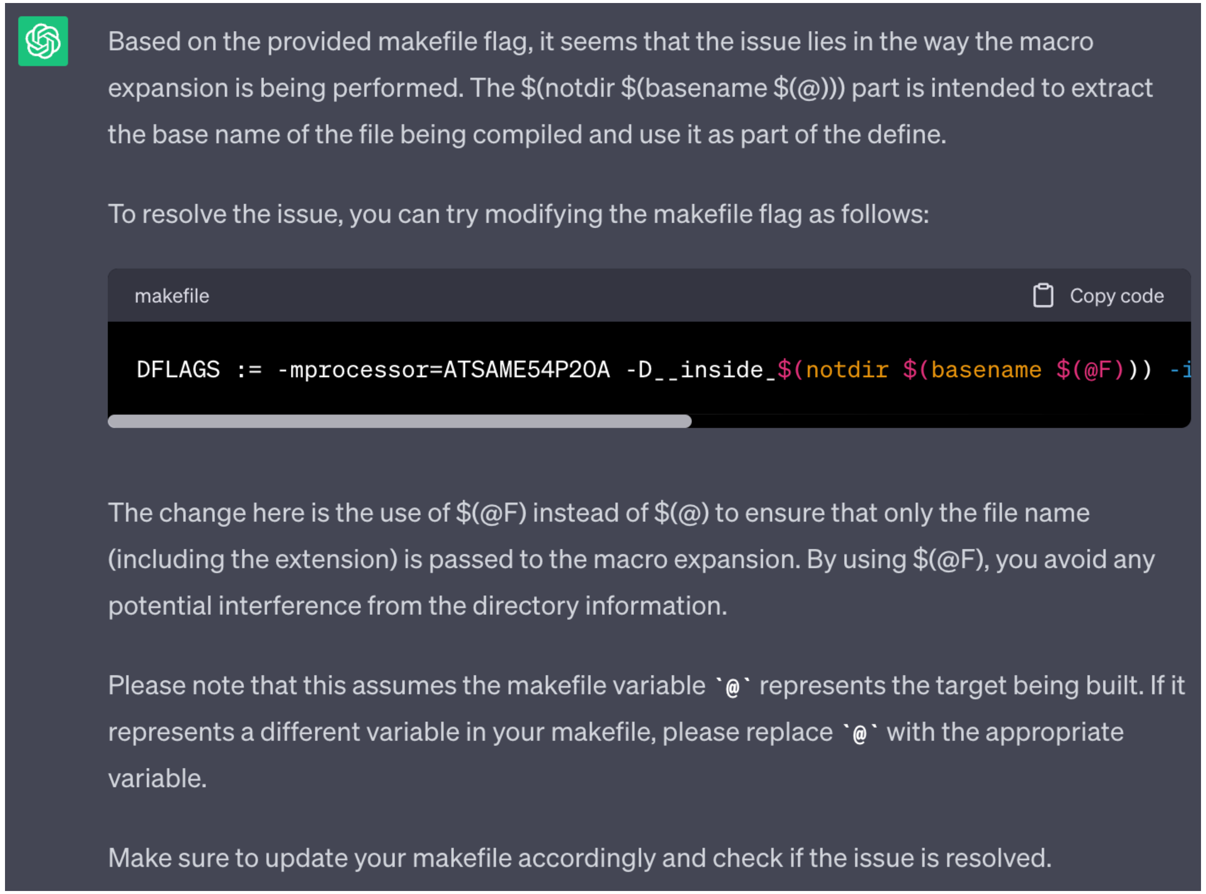
ChatGPT to Debug Code



JA

I'm working with the XC32 compiler. In my makefile I have the following flag: DFLAGS := -mprocessor=ATSAME54P20A -D__inside_$(notdir $(basename $(@))) -include $(SRC_DIR)/src/third_party/TraceRecorder/include/trcCTI.h when the code compiles, it's supposed to use the -D to create a definition with the filename, however, it is instead producing -D__inside_ , what can be done to resolve this?

- Initial recommendation didn't work
- The recommendation made me question the order of definitions, includes, and flags in the build
- AI was wrong, but generated ideas and paths to explore!
- Next step: Submit the linker file and see what it suggestions



Based on the provided makefile flag, it seems that the issue lies in the way the macro expansion is being performed. The $(notdir $(basename $(@))) part is intended to extract the base name of the file being compiled and use it as part of the define.

To resolve the issue, you can try modifying the makefile flag as follows:

```makefile
DFLAGS := -mprocessor=ATSAME54P20A -D__inside_$(notdir $(basename $(@F))) -i
```

The change here is the use of $(@F) instead of $(@) to ensure that only the file name (including the extension) is passed to the macro expansion. By using $(@F), you avoid any potential interference from the directory information.

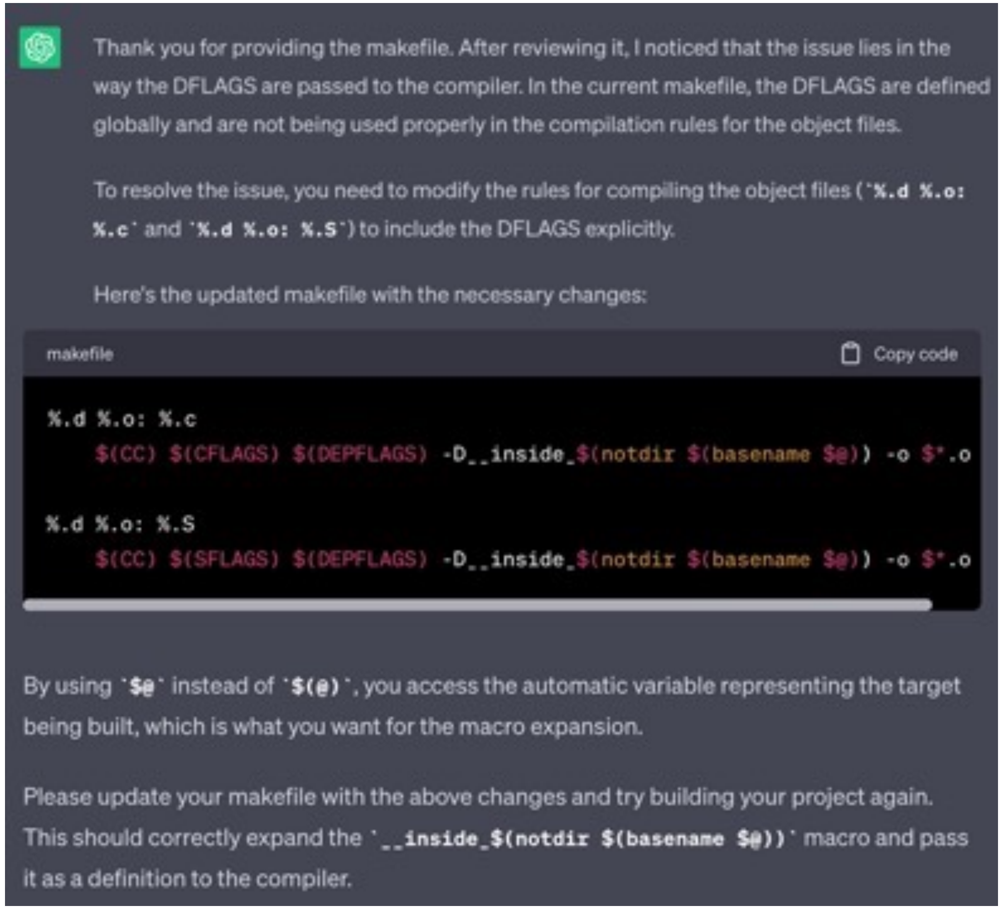
Please note that this assumes the makefile variable `@` represents the target being built. If it represents a different variable in your makefile, please replace `@` with the appropriate variable.

Make sure to update your makefile accordingly and check if the issue is resolved.

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

ChatGPT to Debug Code

- Based on how the custom makefile was configured, the vendor suggested macro expansion was not appropriate!
- Adjustments to the macro expansion were necessary.
- Adjustments to the build recipes were also necessary.
- The changes were not obvious and may have taken a developer quite some time to find! The entire debug session lasted 10 minutes.



Thank you for providing the makefile. After reviewing it, I noticed that the issue lies in the way the DFLAGS are passed to the compiler. In the current makefile, the DFLAGS are defined globally and are not being used properly in the compilation rules for the object files.

To resolve the issue, you need to modify the rules for compiling the object files (`%.d %.o: %.c` and `%.d %.o: %.S`) to include the DFLAGS explicitly.

Here's the updated makefile with the necessary changes:

```makefile
%.d %.o: %.c
	$(CC) $(CFLAGS) $(DEPFLAGS) -D__inside_$(notdir $(basename $@)) -o $*.o

%.d %.o: %.S
	$(CC) $(SFLAGS) $(DEPFLAGS) -D__inside_$(notdir $(basename $@)) -o $*.o
```

By using `$@` instead of `$(@)`, you access the automatic variable representing the target being built, which is what you want for the macro expansion.

Please update your makefile with the above changes and try building your project again. This should correctly expand the `__inside_$(notdir $(basename $@))` macro and pass it as a definition to the compiler.

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

Have you tried to use AI to debug a makefile or code?

- Yes
- No
- No but will try it now!

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

2

Machine Learning on MCUs

Machine Learning on MCUs

NPU’s

GPU’s

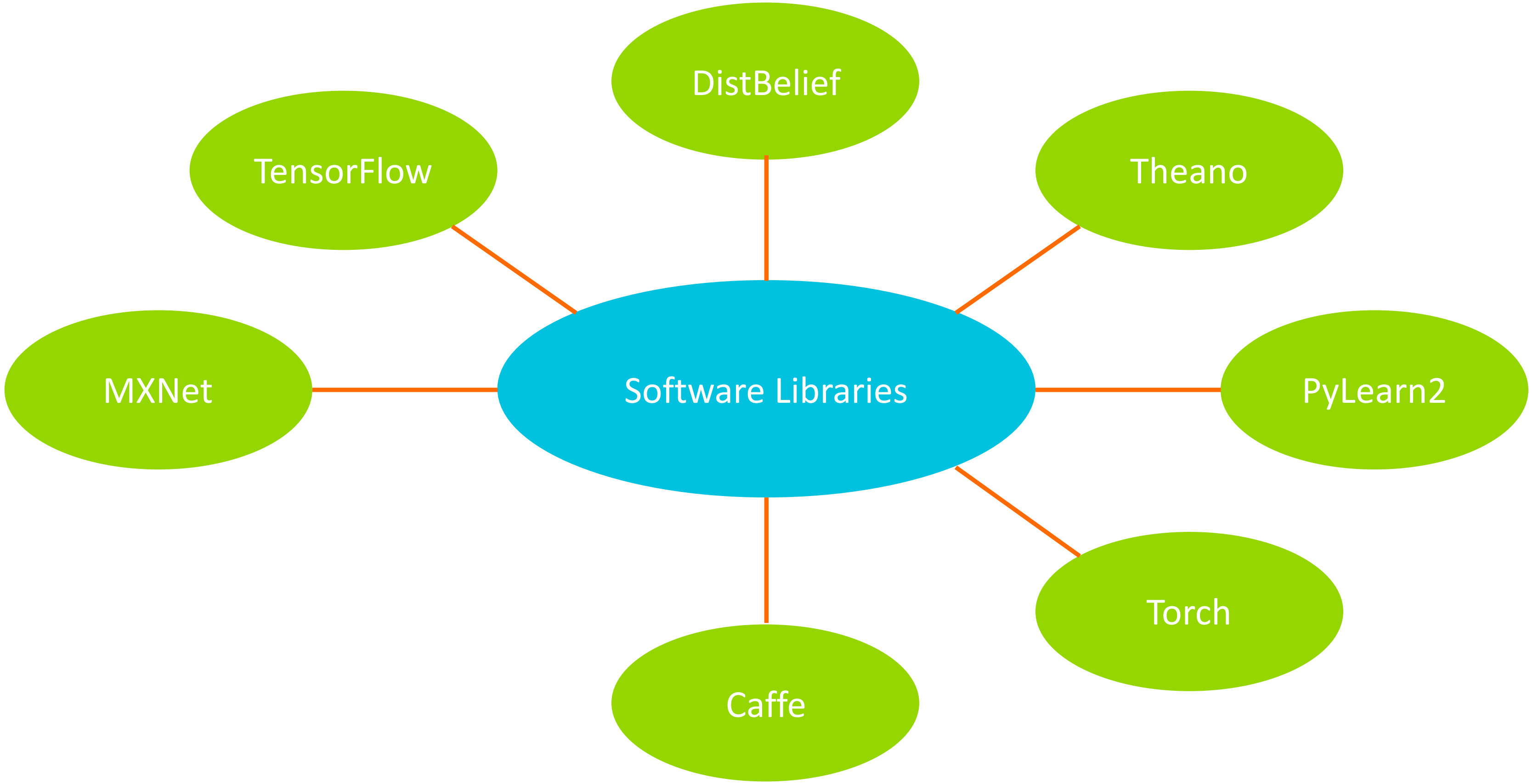
CPU’s

uNPU’s

MCU’s

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

Machine Learning on MCUs



Software Libraries

DistBelief

TensorFlow

Theano

MXNet

PyLearn2

Torch

Caffe

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

Machine Learning on MCUs – AI Frameworks for MCUs

- Tensorflow Light for Microcontrollers
- PyTorch Mobile
- Edge Impulse
- Deeplite
- Qeexo
- SensiML

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

Which of these tools have you used?

Tensorflow Light for Microcontrollers
PyTorch Mobile
Edge Impulse
Deeplite
Qeexo
SensiML
None

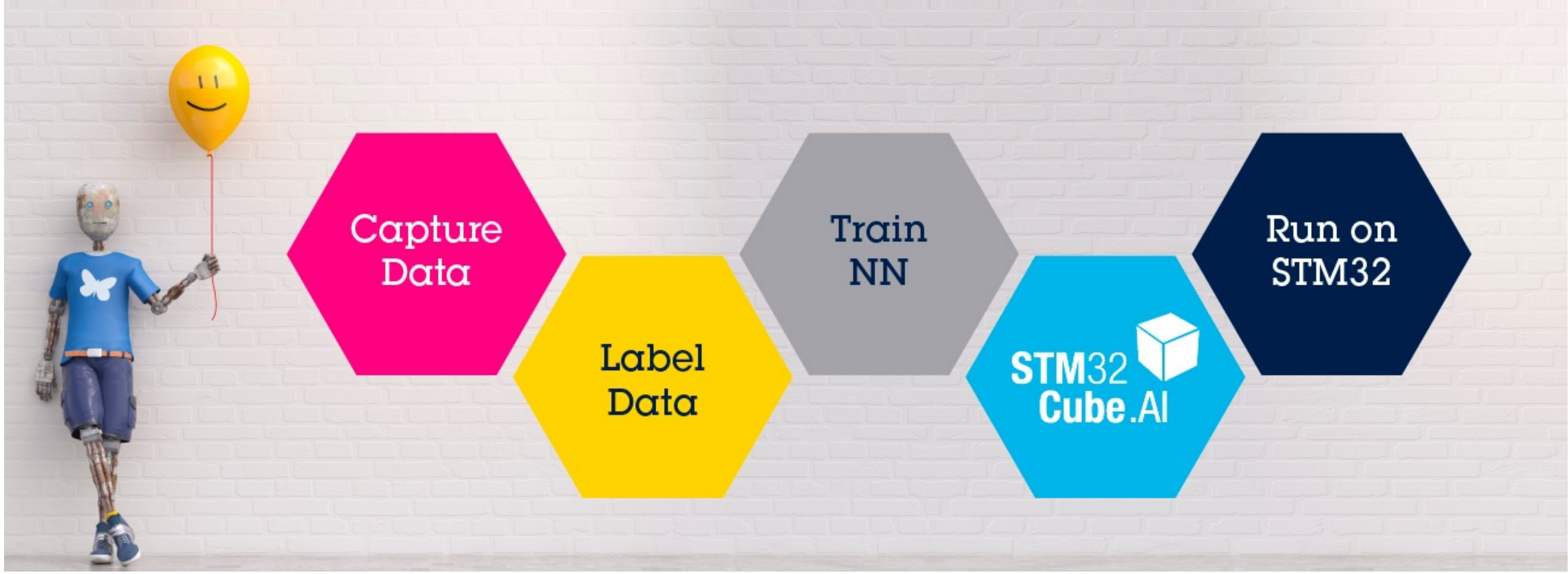
© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

3

Machine Learning with STM32

Machine Learning with STM32

5 Steps to Deploy a Neural Network using STM32Cube.AI



© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

Machine Learning with STM32

Train Neural Network using any major AI frameworks

K
TensorFlow Lite
ONNX
MATLAB
PyTorch

And more

Convert NN into optimized code

STM32 Cube.AI

Select most appropriate MCU
Review computation and memory consumption per layer

Run on optimized runtime

STM32 run-time

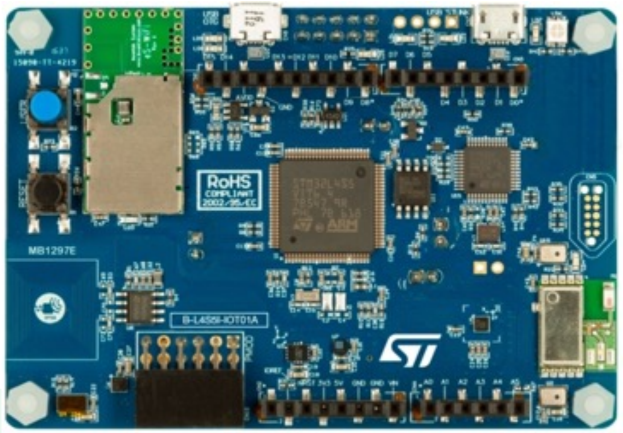
Validate code directly on target
Get accuracy and inference time
Optimize memory usage

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

Machine Learning with STM32

STM32L4S5-IOT01A

Capture Data



Inference on STM32L4S5

IoT Device Connectivity

- STM32L4+ ultra low-power processor, Arm Cortex-M4 at 120 with 2MB internal Flash, 640kB internal RAM
- 64-Mbit Quad-SPI Flash memory
- 802.11 b/g/n compliant Wi-Fi® module (ISM43362-M3G-L44)
- 2 digital omnidirectional microphones (MP34DT01)
- High-performance 3-axis magnetometer (LIS3MDL)
- 3D accelerometer and 3D gyroscope (LSM6DSL)
- Time-of-flight and gesture-detection sensor (VL53L0X)

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

What is your experience with ML on STM32 MCUs?
-Beginner
- Intermediate
- Advanced
- Other

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

4

Going Further

AI and ML Resources

- Jacob’s AI Blogs
- Jacob’s CEC courses
- Jacob’s ML Blogs

- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

www.beningo.com

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

Digi-Key Electronics

BENINGO
EMBEDDED GROUP

informa markets

© 2023 Beningo Embedded Group, LLC. All Rights Reserved.