

WSIZ, 2023  
Computer systems architecture

# Lecture 15. Computer architecture for embedded applications

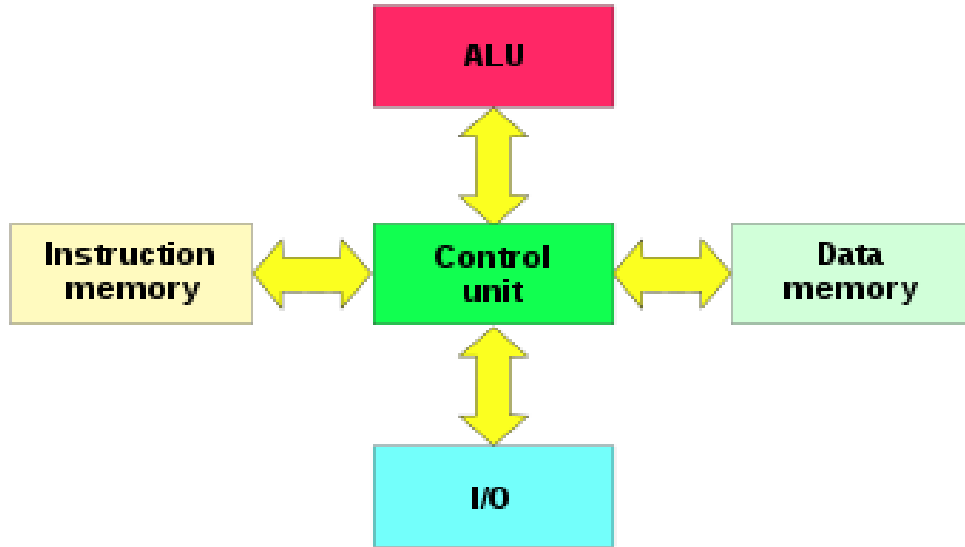
Dmitry Zaitsev

<http://daze.ho.ua>

# Basic directions

- Conventional micro-processor based control systems
- Field Programmable Gate Arrays (FPGAs) based control systems
- Intellectual embedded control architecture and technology of application design

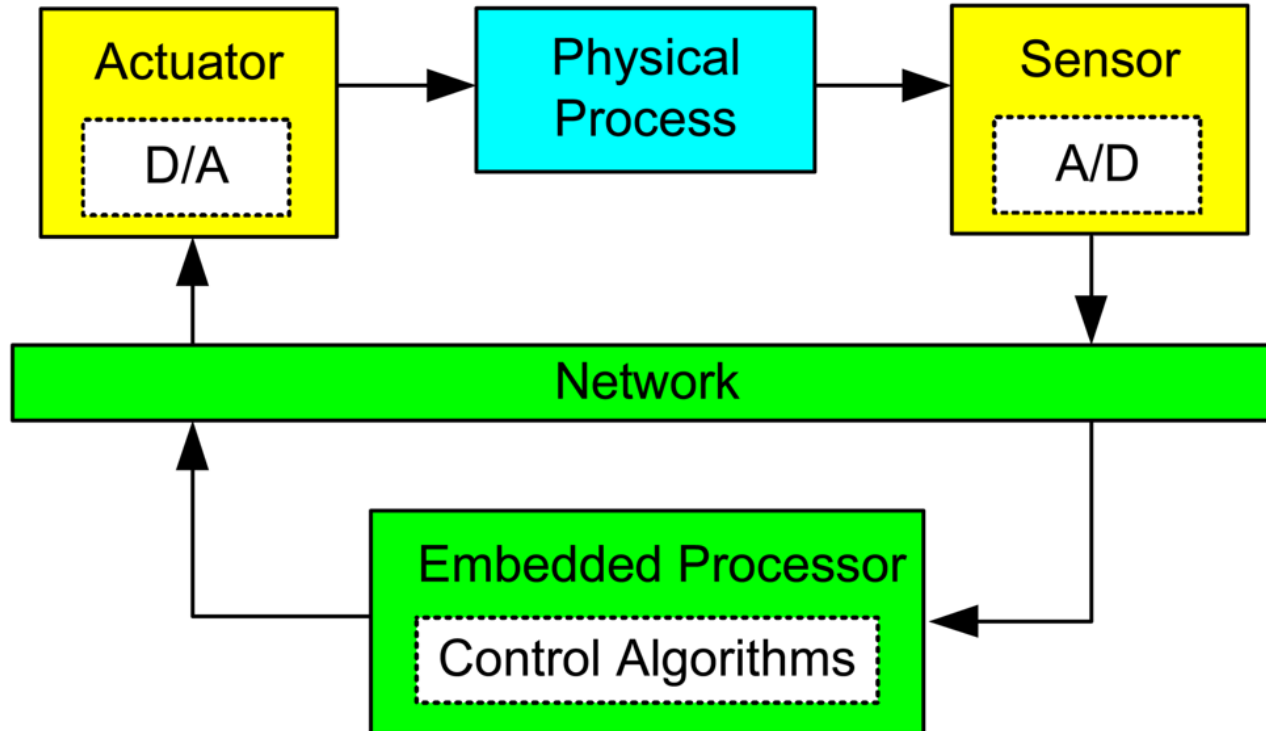
# Micro-processor based control



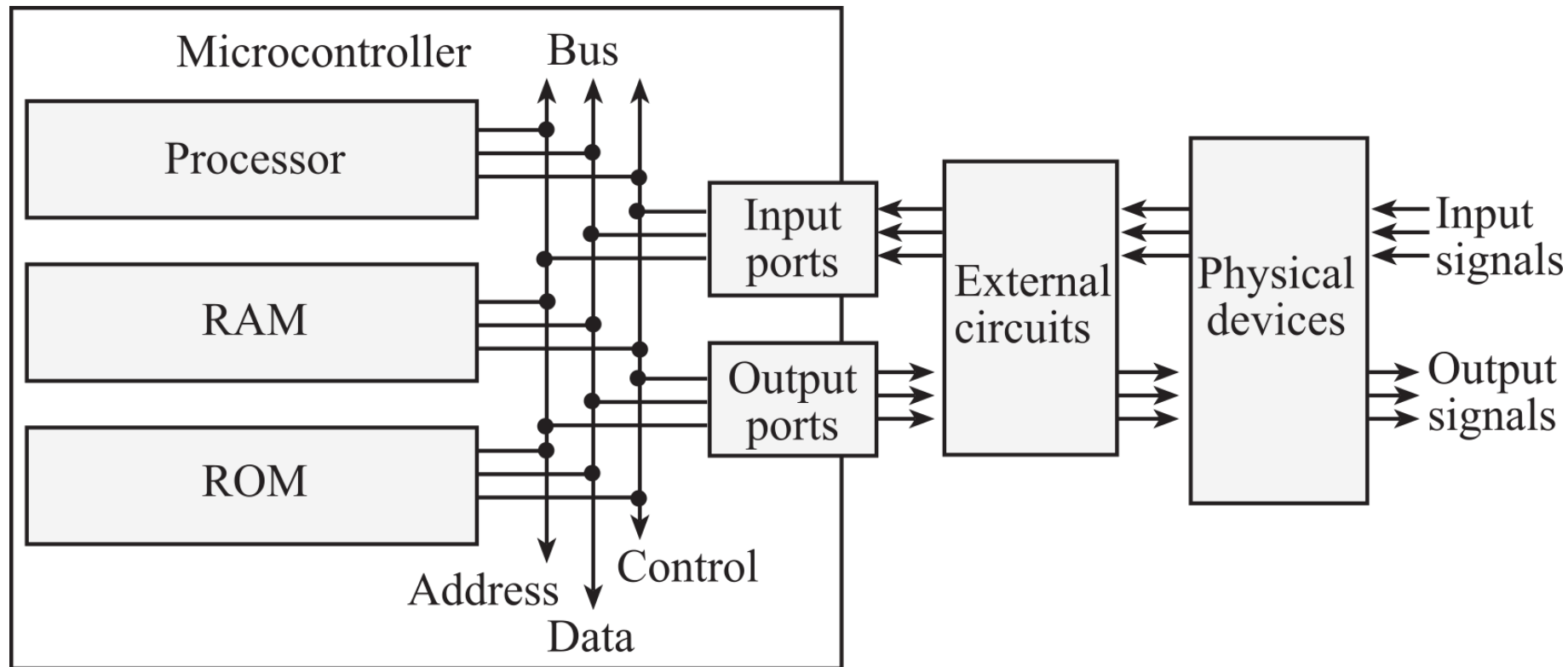
The Harvard architecture is a computer architecture with separate storage and signal pathways for instructions and data.

ARM is a family of reduced instruction set computer (RISC) instruction set architectures for computer processors, configured for various environments.

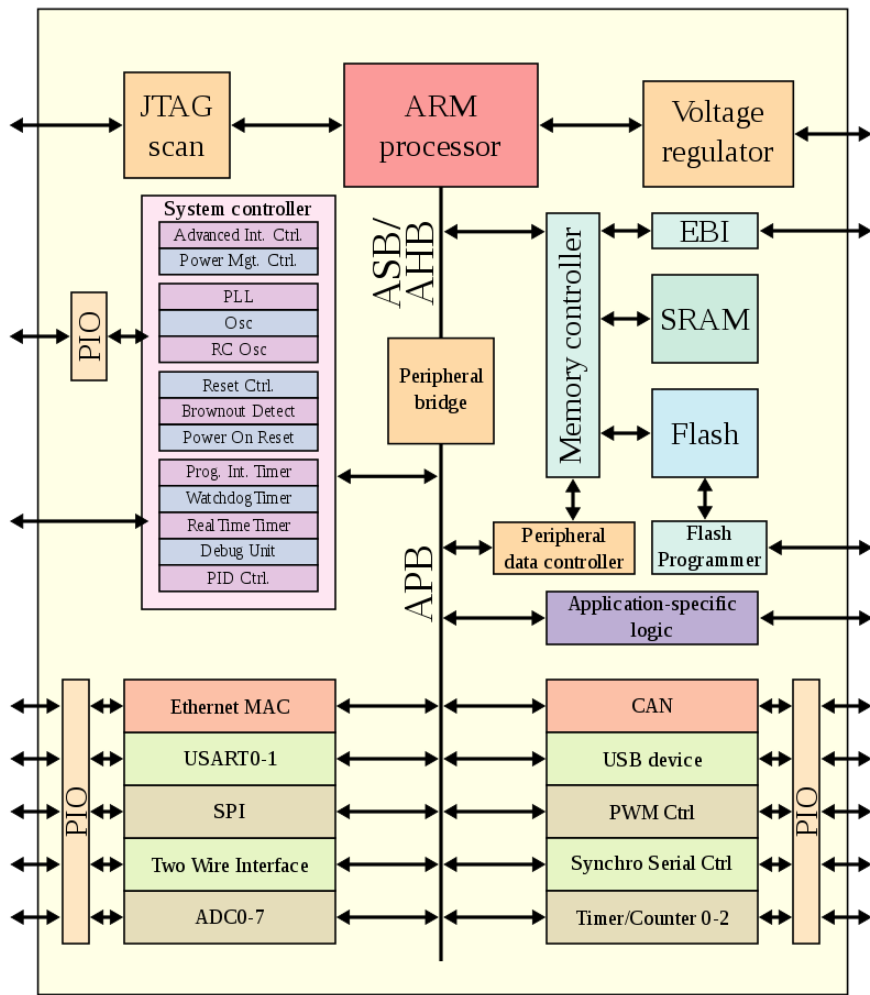
# Generalized embedded control architecture



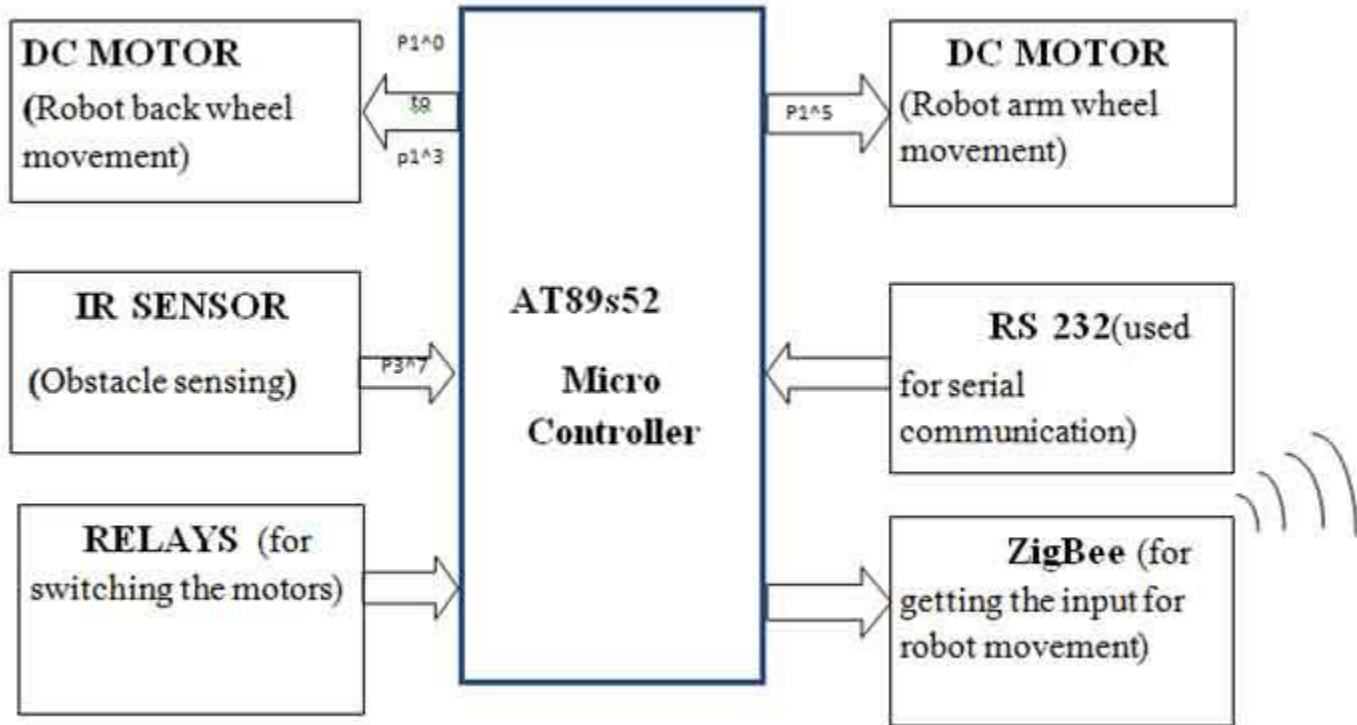
# Detailed embedded architecture



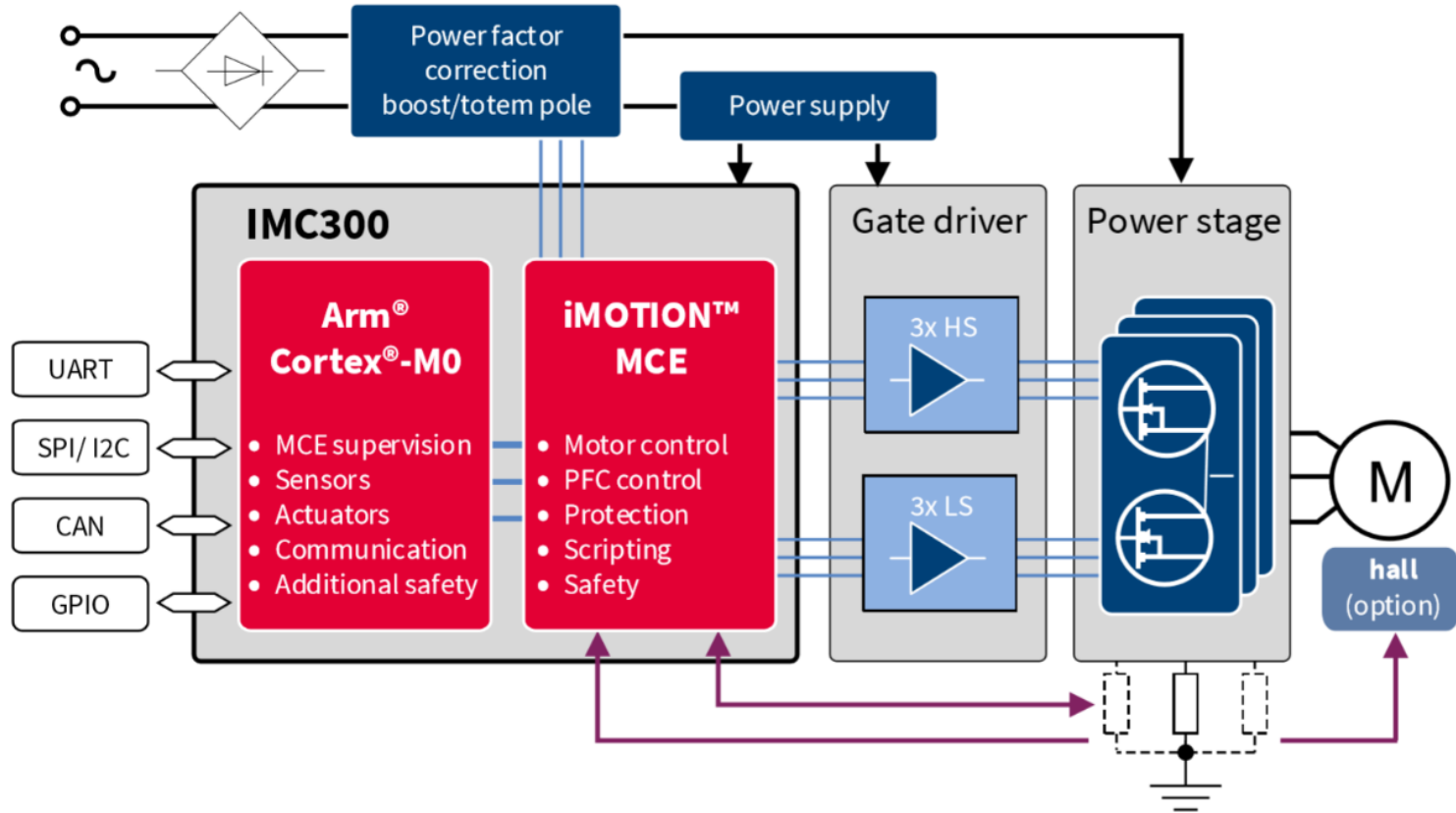
# Microprocessor-based system on a chip (arm.com)



# Embedded System Design for Robotic Arm Control

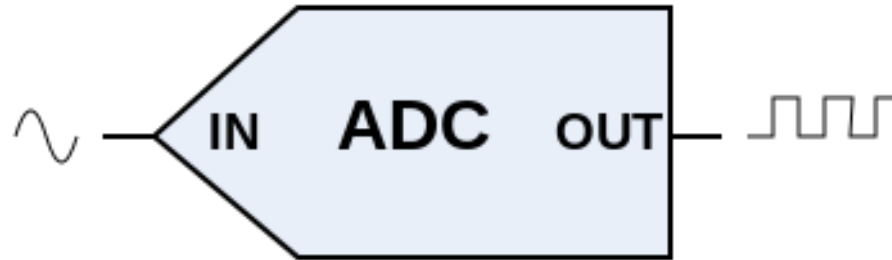
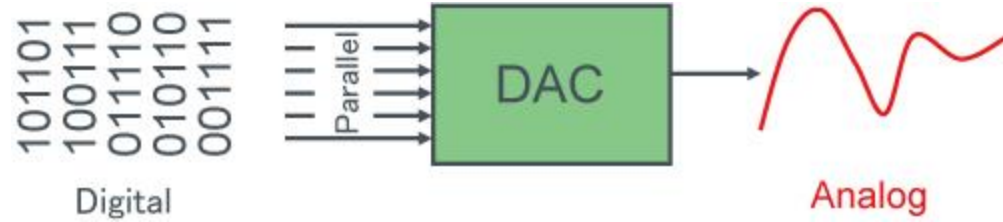


# Motor controller integrates Arm Cortex-M0 core



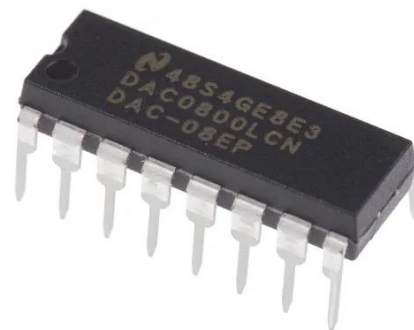
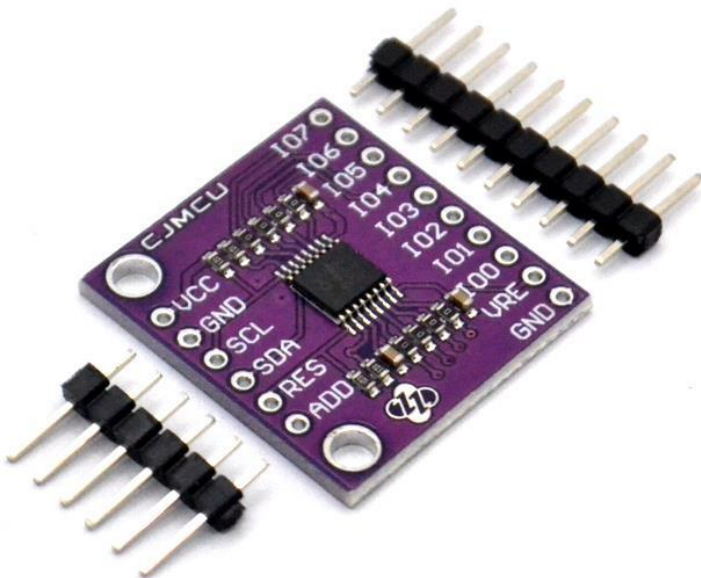


# Digital-analog and analog-digital converters

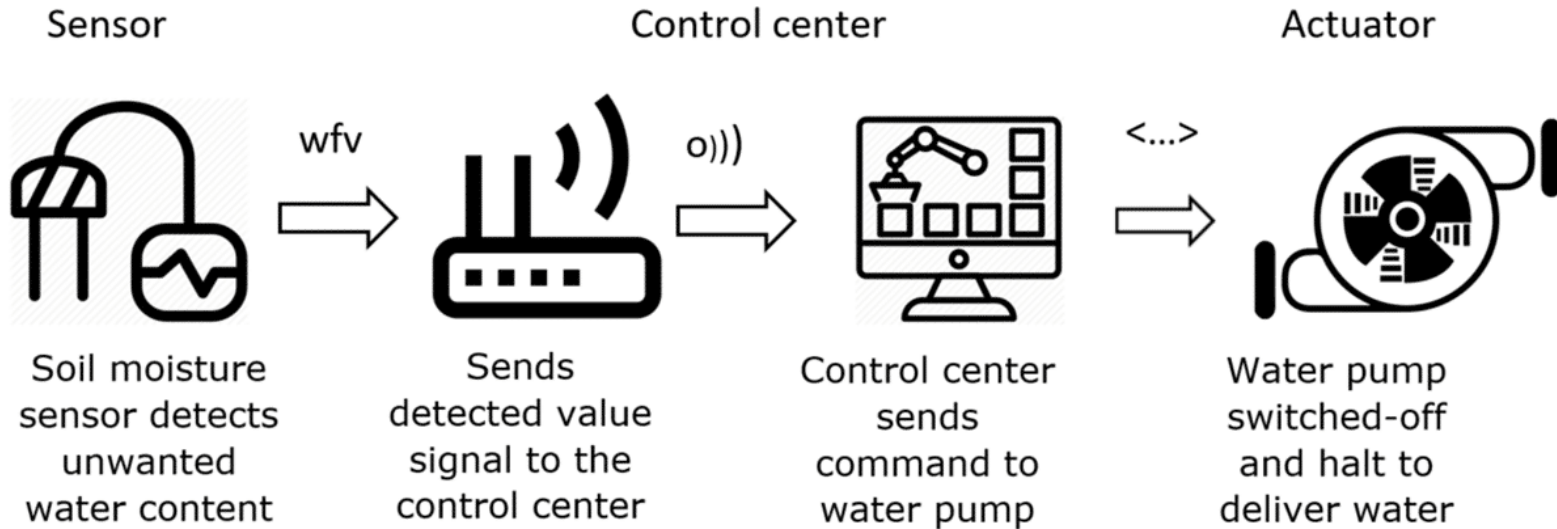


# DAC and ADC view

SZYTF



# Sensors and actuators



# Sensors and actuators view

## Difference Between Sensor & Actuator

**Actuator**



**Sensor**



AV104 Vibration sensor



VEVOR 400mm Dc 12V Linear Actuators

# Microchip studio

The screenshot displays the Microchip Studio interface with the ATtiny417 datasheet open. The left sidebar shows the 'PORT - I/O Pin Controller' section, with 'DIR' selected. The main window shows the 'Data Direction' register (DIR) details, including its name, offset, reset, and access. A table shows the bit fields for DIR[7:0], with bits 7-0 all being R/W. Below this, a text box explains that writing a '1' to PORT.DIR[n] configures and enables pin n as output pin. The right sidebar shows the 'I/O' register list, with 'DIR (PORTB)' selected. A red arrow points to the 'DIR (PORTB)' entry in the list, and a callout box indicates that clicking the register and pressing F1 opens the datasheet register description.

**ATtiny417 / ATtiny814 / ATtiny816 / ATtiny817**

**PORT - I/O Pin Controller**

- Features
- Overview
- Functional Description
  - Register Summary - PORT
  - Register Description - Ports
    - DIR**
      - DIRSET
      - DIRCLR
      - DIRTGL
      - OUT
      - OUTSET
      - OUTCLR
      - OUTTGL
      - IN
      - INTFLAGS
      - PINCTRL0, PINCTRL1, PINCTRL2, PIN
    - Register Summary - VPORT
    - Register Description - Virtual Ports
- BOD - Brownout Detector
- VREF - Voltage Reference
- WDT - Watchdog Timer
- TCA - 16-bit Timer/Counter Type A
- TCB - 16-bit Timer/Counter Type B
- TCD - 12-bit Timer/Counter Type D
- RTC - Real Time Counter

**Data Direction**

**Name:** DIR  
**Offset:** 0x00  
**Reset:** 0x00  
**Access:** -

Bit	7	6	5	4	3	2	1	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

**Bits 7:0 – DIR[7:0]: Data Direction**

This bit field selects the data direction for the individual pins n of the Port.

Writing a '1' to PORT.DIR[n] configures and enables pin n as output pin.

Writing a '0' to PORT.DIR[n] configures pin n as input pin. It can be configured by writing to the ISC bit in PORT\_PINnCTRL.

**I/O**

Name	Address	Value
DIR	0x420	0x00
DIRSET	0x421	0x00
DIRCLR	0x422	0x00
DIRTGL	0x423	0x00
OUT	0x424	0x00
OUTSET	0x425	0x00
OUTCLR	0x426	0x00
OUTTGL	0x427	0x00
IN	0x428	0xFC
INTFLAGS	0x429	0x00
PIN0CTRL	0x430	0x00
PIN1CTRL	0x431	0x00
PIN2CTRL	0x432	0x00
PIN3CTRL	0x433	0x00
PIN4CTRL	0x434	0x00
PIN5CTRL	0x435	0x00
PIN6CTRL	0x436	0x00
PIN7CTRL	0x437	0x00

**Click register, F1 opens datasheet register description**

# How FPGA looks like



# What is FPGA

- FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects allowing blocks to be wired together.
- Logic blocks can be configured to perform complex combinational functions, or act as simple logic gates like AND and XOR.
- In most FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.
- Many FPGAs can be reprogrammed to implement different logic functions, allowing flexible reconfigurable computing as performed in computer software.

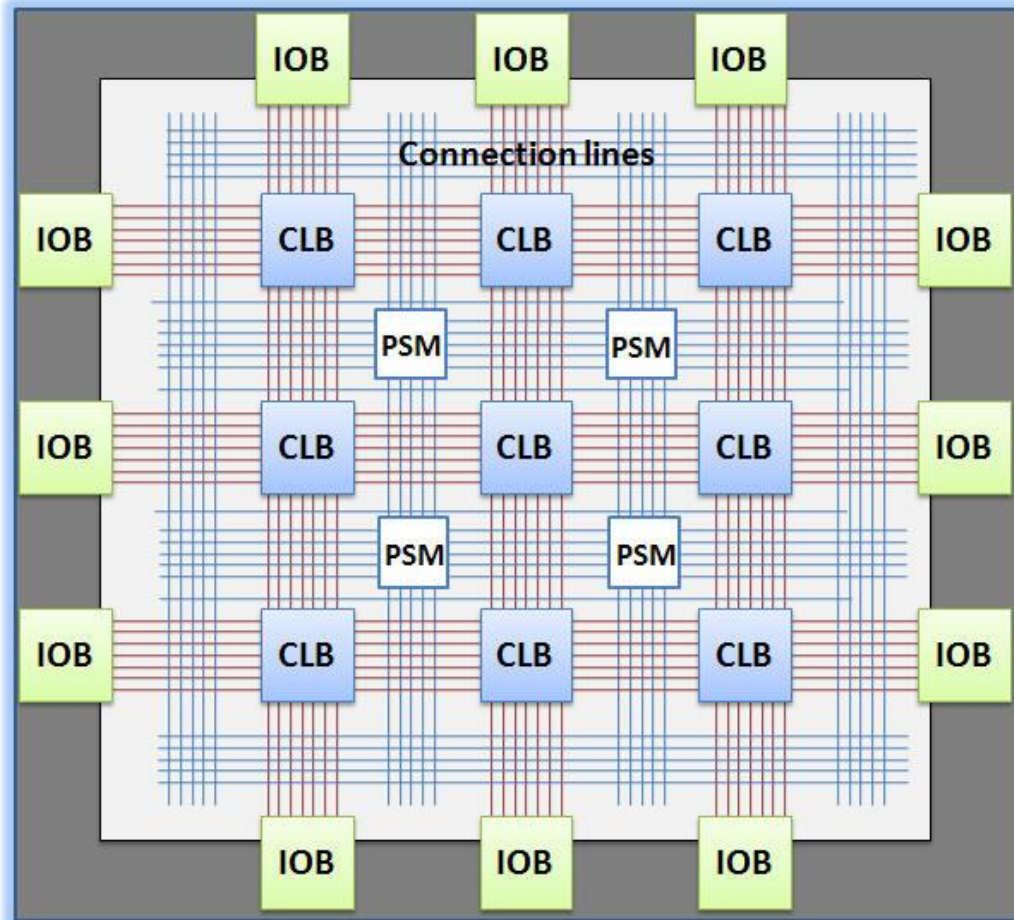


# Motivation and basic directions of FPGA application

- need for parallel computing
- dynamic requirements
- low price
- hardware prototyping
- hardware acceleration
- neural networks



# FPGA architecture



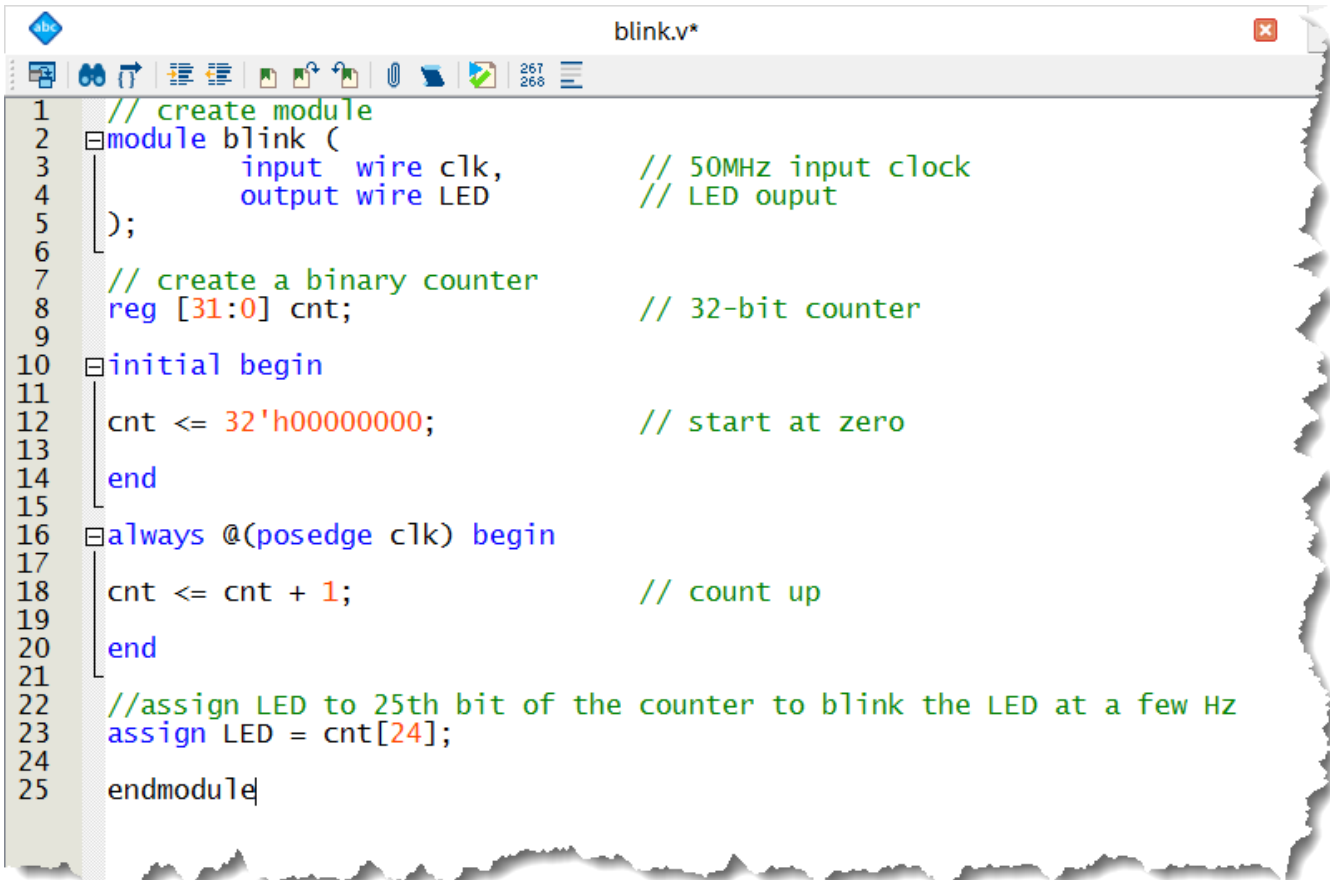
**IOB**  
Input Output Block

**CLB**  
Configurable  
Logic Block

**PSM**  
Programable  
Switch Matrix

**Connection lines**  
Single, Long  
Double, Direct

# FPGA designs with Verilog



```
1 // create module
2 module blink (
3     input wire clk,           // 50MHz input clock
4     output wire LED           // LED output
5 );
6
7 // create a binary counter
8 reg [31:0] cnt;               // 32-bit counter
9
10 initial begin
11     cnt <= 32'h00000000;       // start at zero
12
13 end
14
15
16 always @(posedge clk) begin
17     cnt <= cnt + 1;            // count up
18
19 end
20
21 //assign LED to 25th bit of the counter to blink the LED at a few Hz
22 assign LED = cnt[24];
23
24
25 endmodule
```

# AI powered embedded control

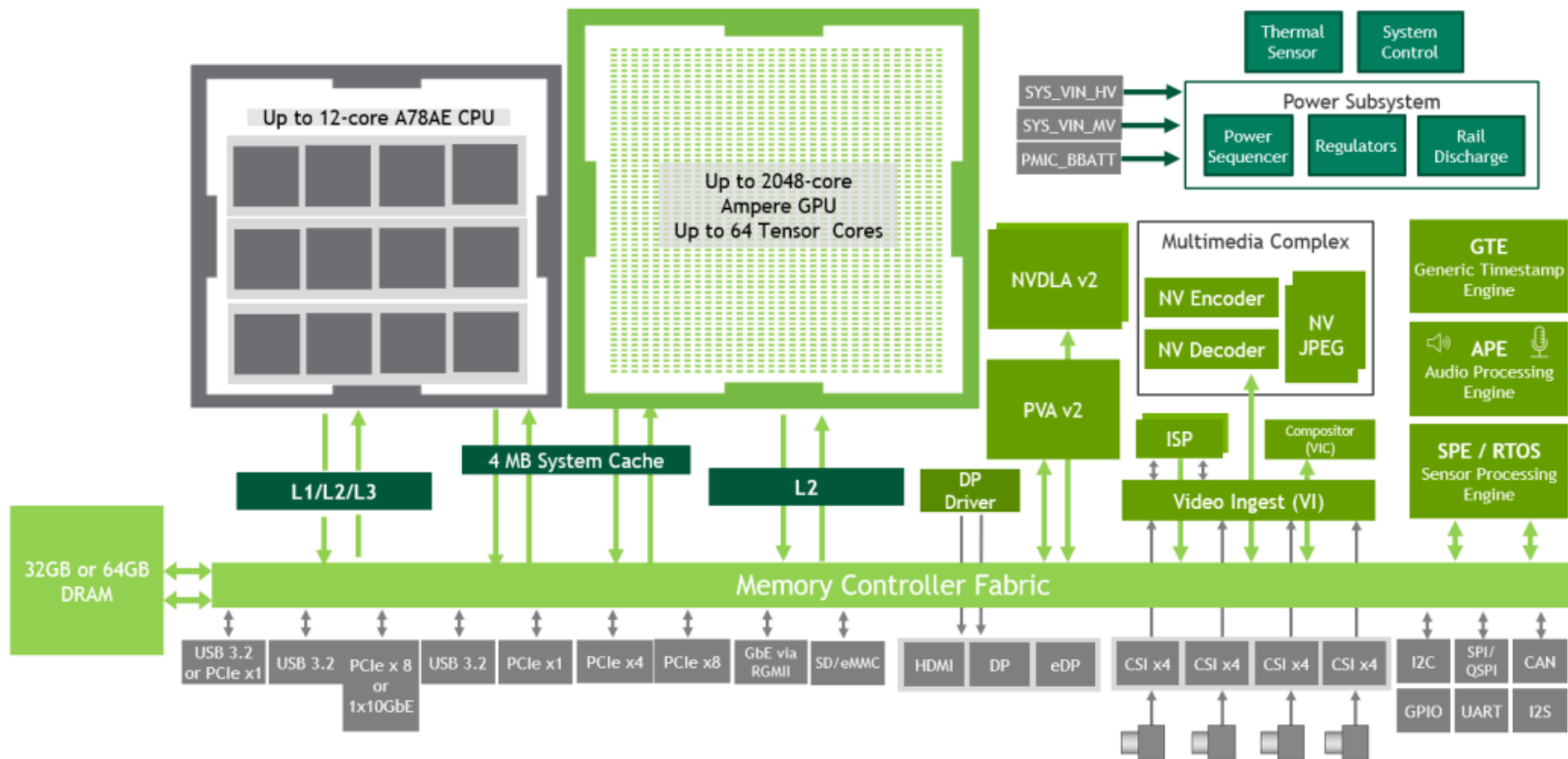
- Nvidia Jetson is a series of embedded computing boards from Nvidia. The Jetson TK1, TX1 and TX2 models all carry a Tegra processor (or SoC) from Nvidia that integrates an ARM architecture central processing unit (CPU). Jetson is a low-power system and is designed for accelerating machine learning applications.

# Seeed Studio Developer Kit 32 GB Jetson

## AGX Orin™ NVIDIA®

- GPU: NVIDIA Ampere architecture with 2,048 NVIDIA CUDA® cores and 64 Tensor cores
- CPU/CPU: 64-bit Arm® Cortex®-A78AE v8.2 12-core CPU/CPU
- 3MB L2 + 6MB L3
- DL Accelerator: 2 NVDLA v2.0
- Vision Accelerator: PVA v2.0
- Memory: 32GB 256-bit LPDDR5, 204.8GB/s
- Storage: 64 GB eMMC 5.1
- Video encoding: 2x 4K60, 4x 4K30, 8x 1080p60, 16x 1080p30 (H.265)
- Video decoding: 1x 8K30, 3x 4K60, 6x 4K30, 12x 1080p60, 24x 1080p30 (H.265)
- Power: 15W to 60W

# Jetson AGX Orin Series Functional Block Diagram

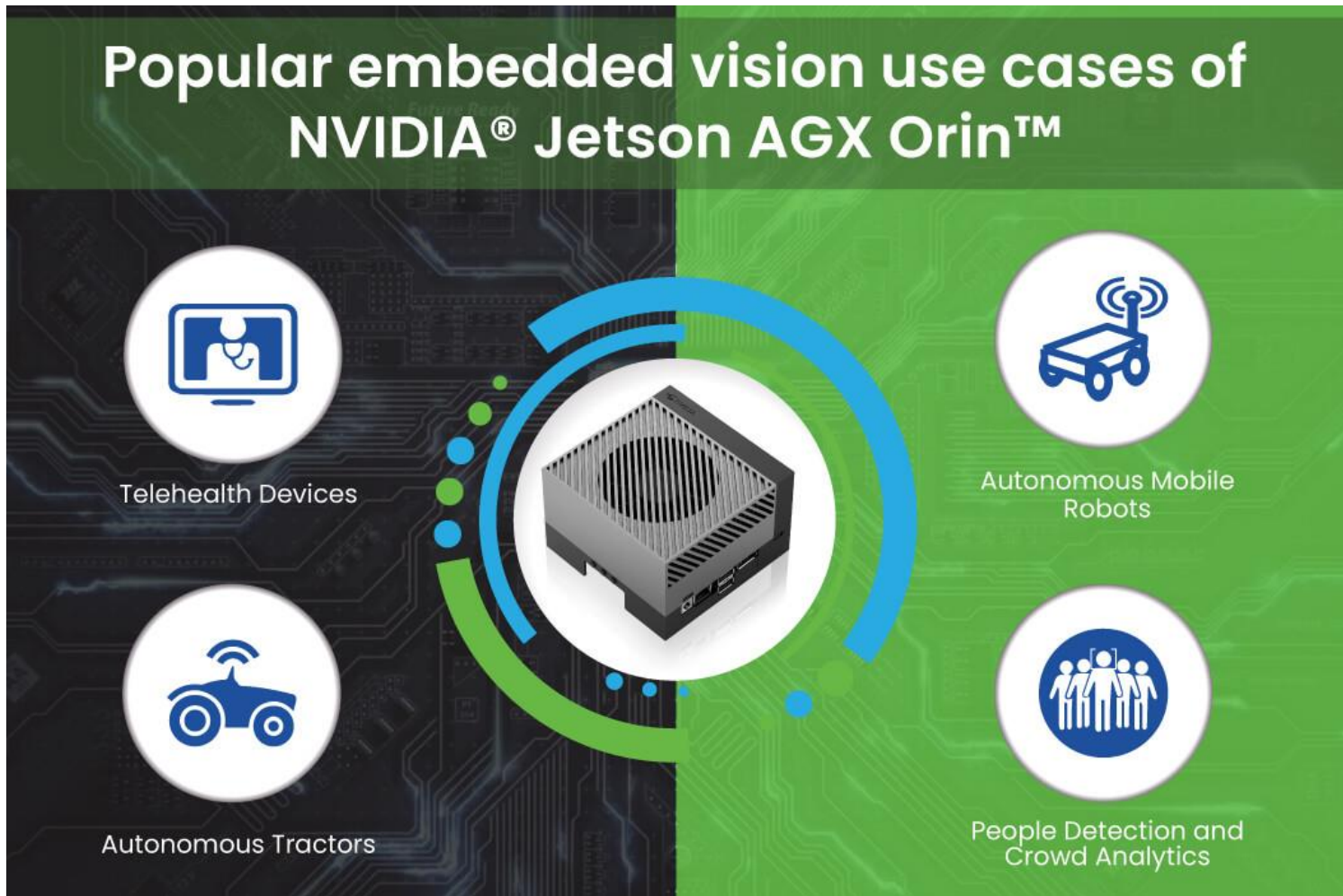


# How it looks



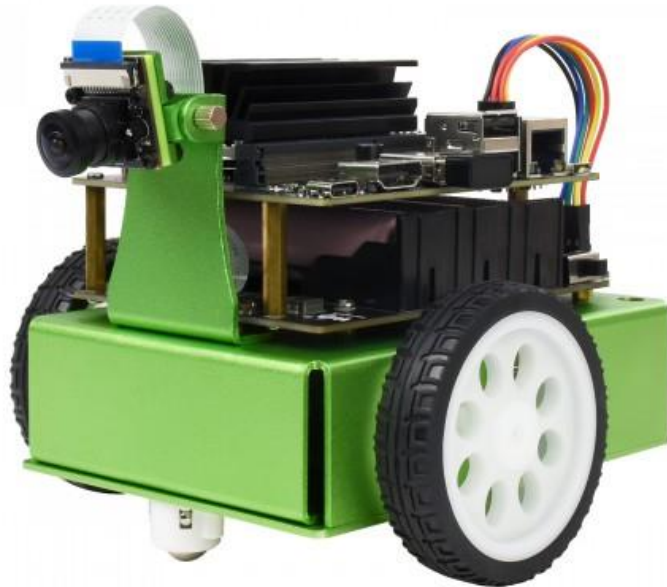
[Jetson AGX Orin 64GB Developer Kit](#)

# Popular embedded vision use cases of NVIDIA® Jetson AGX Orin





# JetBot – robot controlled by trained neural network in Python





# Parts to assemble



# NVIDIA Jetson Nano – computer for embedded AI applications

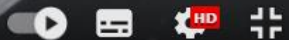


Jetbot Road Following and Collision Avoidance



0:22 / 0:57

Прокрутите экран вниз, чтобы посмотреть подробную информацию

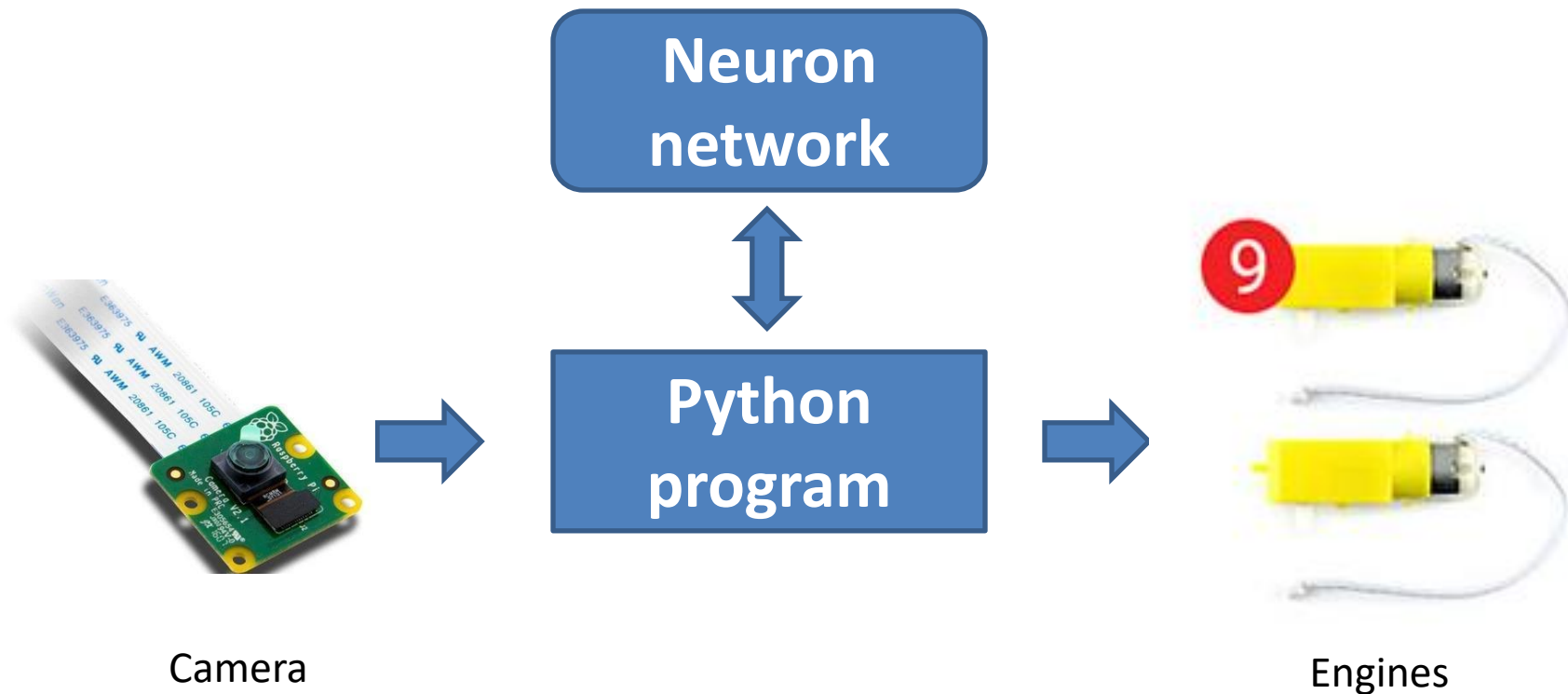


# How it works

- Autonomous vehicle follows road and avoids collisions
- Python program runs neuron network
- Python program provides connection of neuron network to camera and engines
- <https://youtu.be/8Hz2G2SK3KI>



# Python and AI based control



# How to do

- Test NANO Jetson Toolkit
- Assemble JetBot
- Collect images/video for collision avoidance or/and road following
- Train neuron network
- Run neuron network to drive JetBot

Python

# Jupyter notebooks for JetBot

tesla/stock\_predictions | Kaggle | Tesla, Inc. (TSLA) Stock Historical | jetbot/notebooks/collision\_avoidance

github.com/NVIDIA-AI-IOT/jetbot/tree/master/notebooks/collision\_avoidance

Product Solutions Open Source Pricing Search Sign in Sign up

NVIDIA-AI-IOT / jetbot Public Notifications Fork 933 Star 2.6k

<> Code Issues 13 Pull requests 6 Discussions Actions Projects Wiki Security Insights

master jetbot / notebooks / collision\_avoidance / Go to file

tokk-nv Add missing descriptions be2ca63 on Jan 9, 2021 History

..		
data_collection.ipynb	Merge branch 'master' into fix-typos	2 years ago
live_demo.ipynb	Merge branch 'master' into fix-typos	2 years ago
live_demo_resnet18.ipynb	Add missing descriptions	2 years ago
live_demo_resnet18_build_trt.ipynb	Split Collision Avoidance TRT inference notebook	2 years ago
live_demo_resnet18_trt.ipynb	Add missing descriptions	2 years ago
train_model.ipynb	Merge branch 'dev-headroom' of https://github.com/tokk-nv/jetbot into...	2 years ago
train_model_plot.ipynb	Fix typos in notebooks/collision_avoidance/	2 years ago
train_model_resnet18.ipynb	Change parameters to stabilize training across platforms	2 years ago

Activate Windows. Go to Settings to activate Windows.

Type here to search 55°F Cloudy 6:46 PM 1/7/2023

# Collision Avoidance - Train Model

Welcome to this host side Jupyter Notebook! This should look familiar if you ran through the notebooks that run on the robot. In this notebook we'll train our image classifier to detect two classes `free` and `blocked`, which we'll use for avoiding collisions. For this, we'll use a popular deep learning library *PyTorch*

```
In [ ]: import torch
import torch.optim as optim
import torch.nn.functional as F
import torchvision
import torchvision.datasets as datasets
import torchvision.models as models
import torchvision.transforms as transforms
```

## Upload and extract dataset

Before you start, you should upload the `dataset.zip` file that you created in the `data_collection.ipynb` notebook on the robot.

You should then extract this dataset by calling the command below

```
In [ ]: !unzip -q dataset.zip
```

You should see a folder named `dataset` appear in the file browser.

## Create dataset instance

Now we use the `ImageFolder` dataset class available with the `torchvision.datasets` package. We attach transforms from the `torchvision.transforms` package to prepare the data for training.

```
In [ ]: dataset = datasets.ImageFolder(
```

Activate Windows  
Go to Settings to activate Windows.