



# Open Rapid Control Prototyping, Education and Design Tools

*Dion Beqiri*

Contact: [beqirdio@gmail.com](mailto:beqirdio@gmail.com)

Supervisor: *Ing. Pavel Píša, Ph.D.*

Department of Control Engineering

Bachelor's Thesis

June 2022



# Goals

- Understanding and extending the pysimCoder project
- Vector signals feature
- Adding support for GNU/Linux targets:
  - MZ\_APO Zynq based educational kit
  - RaspberryPi board (extension)
- Adding support for ESP32-C3 (RISC-V) board running NuttX RTOS
- Controlling some peripheral using combination of NuttX and pysimCoder

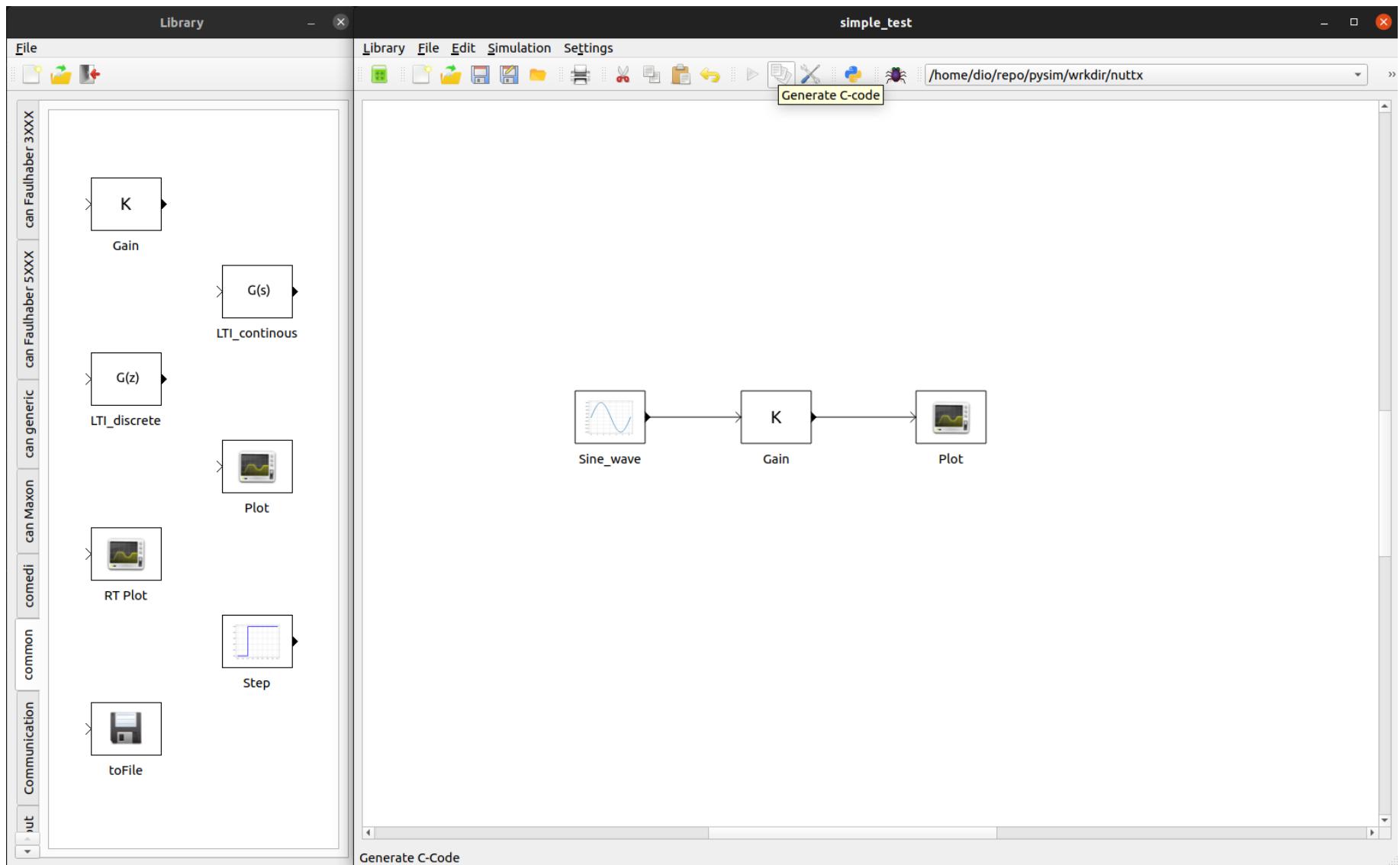


# PysimCoder

- Graphical editor for design of control systems
- Can generate real-time RPC code for various targets
- Similar in style to Simulink, Xcos, etc...
- Advantages:
  - Lighter, faster, compatible with any OS
  - Open source and freely available
  - Uses Python for most of its functions
- Disadvantages:
  - Less development, missing features, and poor (code) stability



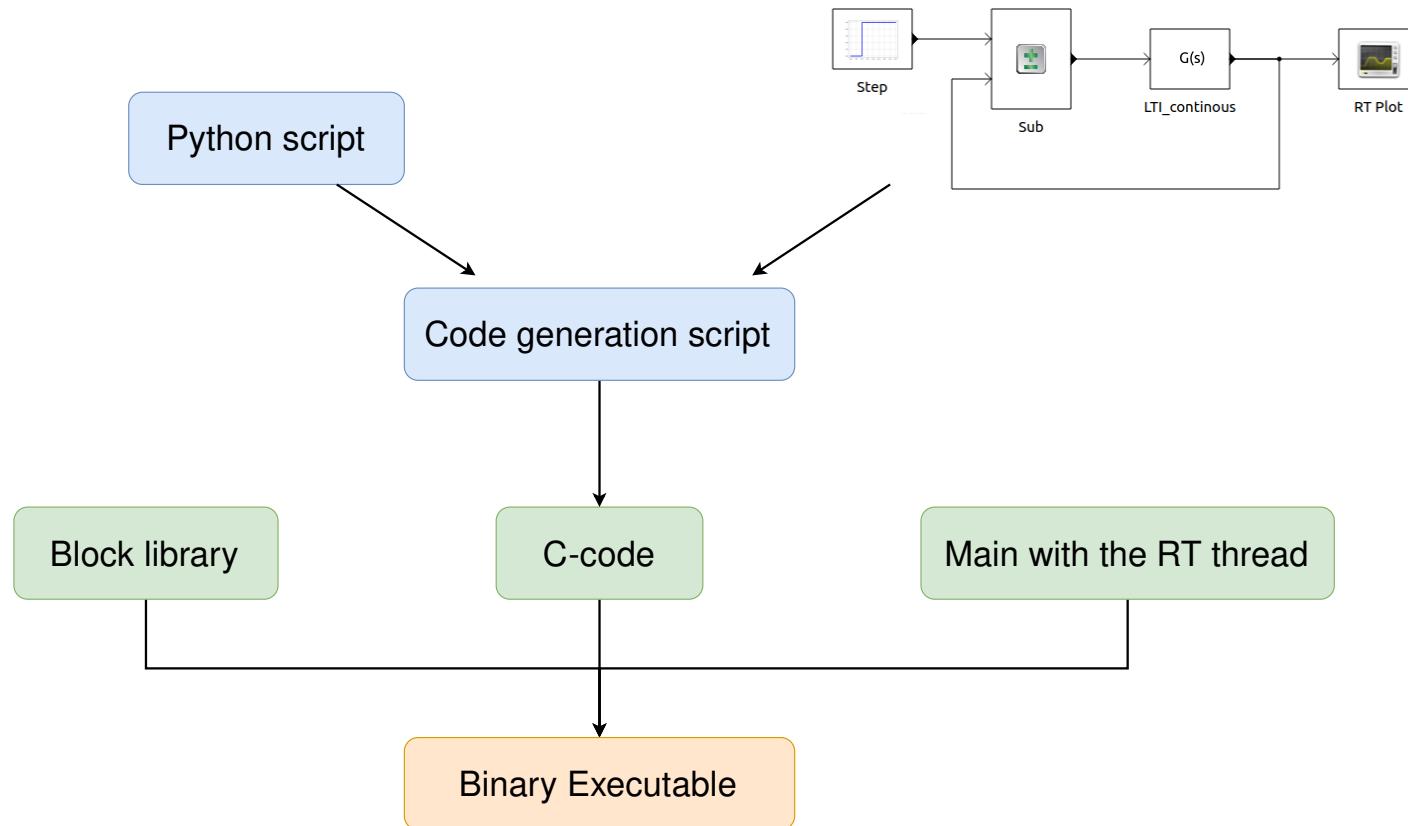
# PysimCoder





# PysimCoder

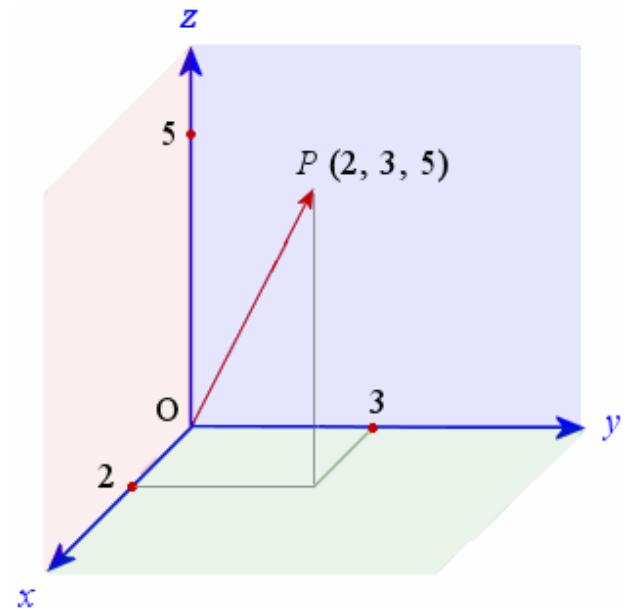
## Basics of Code Generation





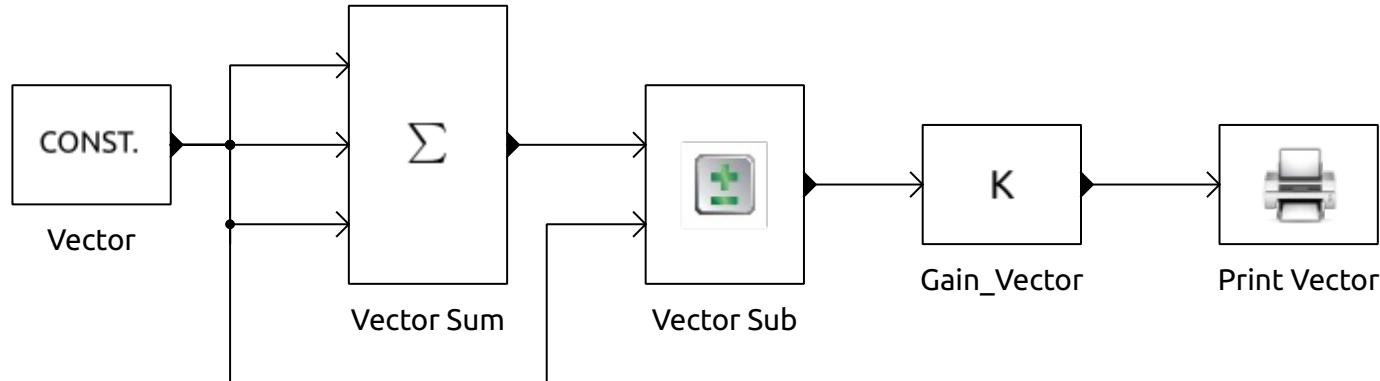
# Vectors

- Importance of vectors in mathematics and control systems engineering
- Existing support for vector dimensions
- Enabling vector signals in pysimCoder
- Static setting of dimensions
- Automatic setting of dimensions





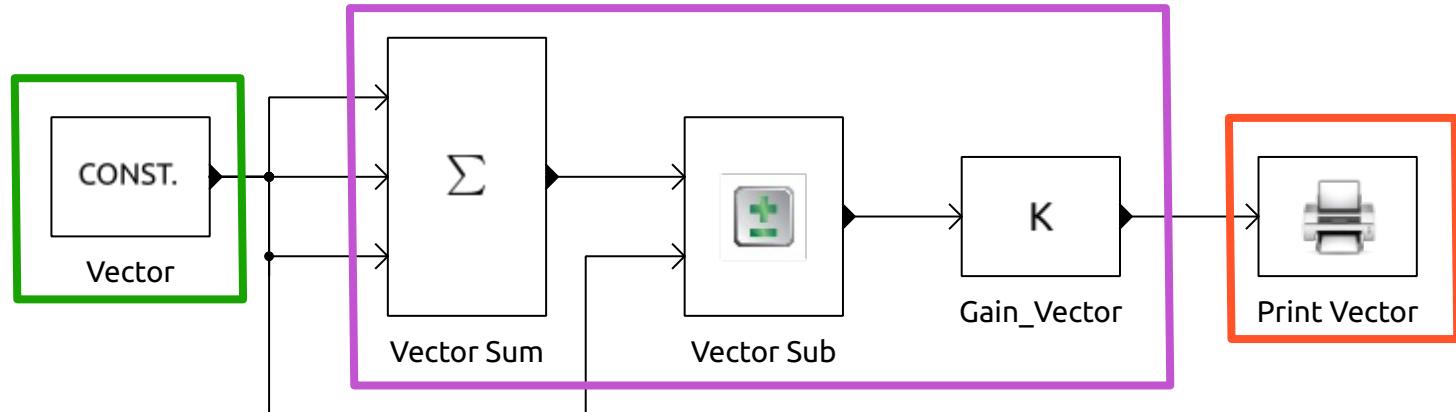
# Vectors



- New blocks for vector experimentation
- Mostly mathematical blocks
- Some blocks know their dimensions, others don't
- Static dimensions → only human error is possible



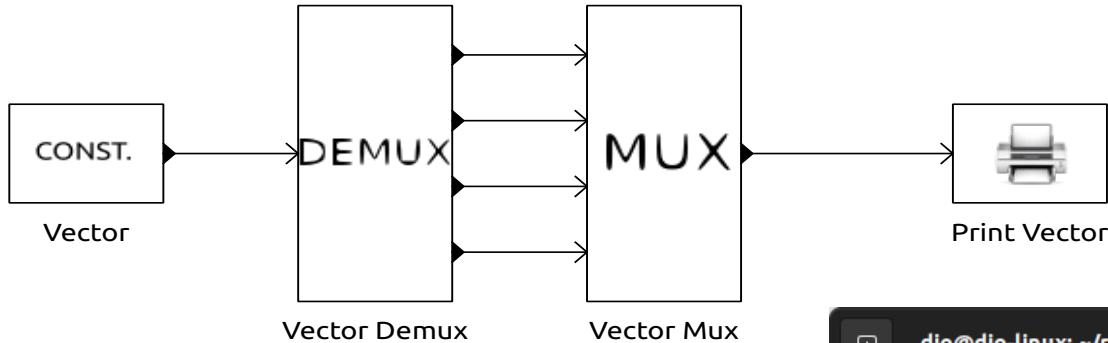
# Vectors



- Algorithm for setting dimensions without user input
- Assumption that blocks are ordered sequentially
- Propagation path of dimensions:
  - **Source blocks** (know dimensions internally)
  - **Middle blocks** (get dimensions externally)
  - **Destination blocks** (externally also)
- Middle blocks contain variable for I/O dimension ratio



# Vectors



- Multiplexer combines scalar values into a vector signal
- Demultiplexer is inverse
- Link between vector-able and normal blocks
- Testing on each other for verification

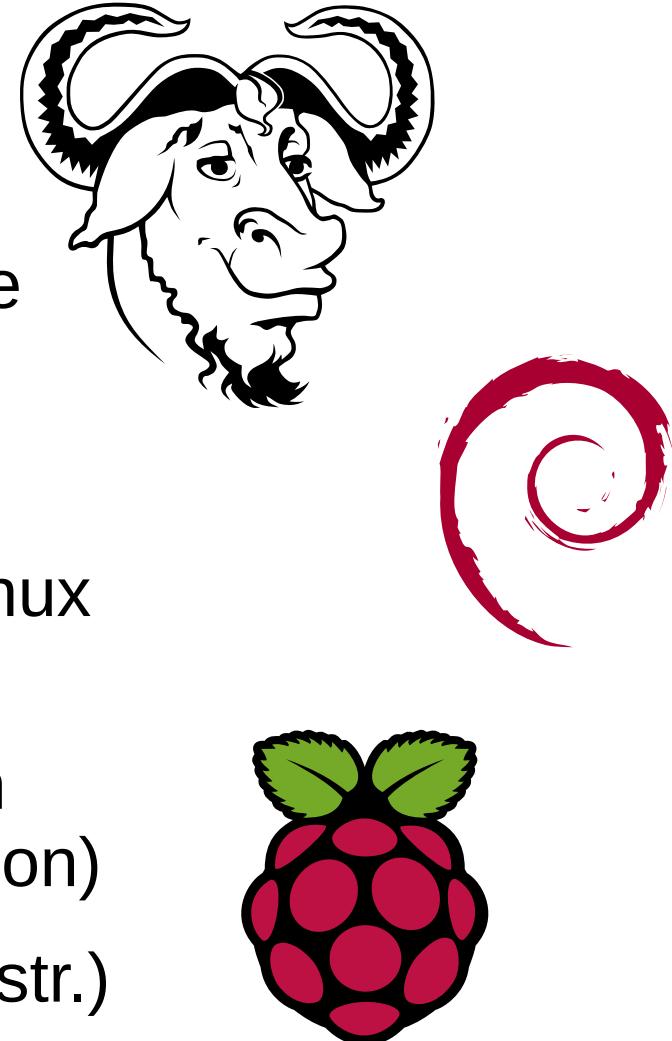
```

dio@dio-linux: ~/repo/pysim/wrk...
9.900000 [1.000000, 2.000000, 3.000000, 4.000000]
9.910000 [1.000000, 2.000000, 3.000000, 4.000000]
9.920000 [1.000000, 2.000000, 3.000000, 4.000000]
9.930000 [1.000000, 2.000000, 3.000000, 4.000000]
9.940000 [1.000000, 2.000000, 3.000000, 4.000000]
9.950000 [1.000000, 2.000000, 3.000000, 4.000000]
9.960000 [1.000000, 2.000000, 3.000000, 4.000000]
9.970000 [1.000000, 2.000000, 3.000000, 4.000000]
9.980000 [1.000000, 2.000000, 3.000000, 4.000000]
9.990000 [1.000000, 2.000000, 3.000000, 4.000000]
10.000000 [1.000000, 2.000000, 3.000000, 4.000000]
  
```



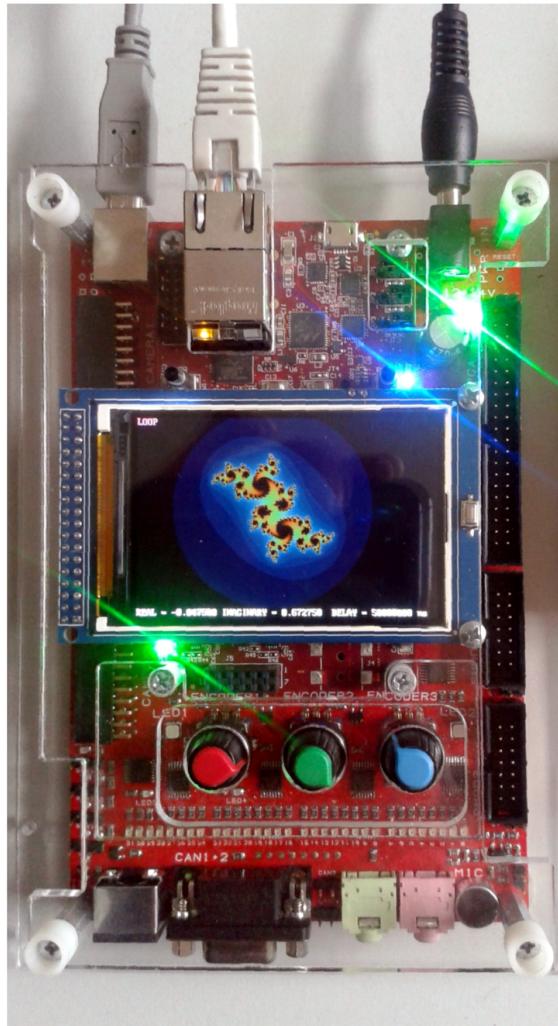
# GNU / Linux

- Popular operating system
- Supports real-time extensions
- Open source and freely available
- Increased usage in embedded systems
- pysimCoder supports various Linux targets
- MZ\_APO educational kit runs on real-time Linux (Debian distribution)
- RaspberryPi board (Raspbian distr.)





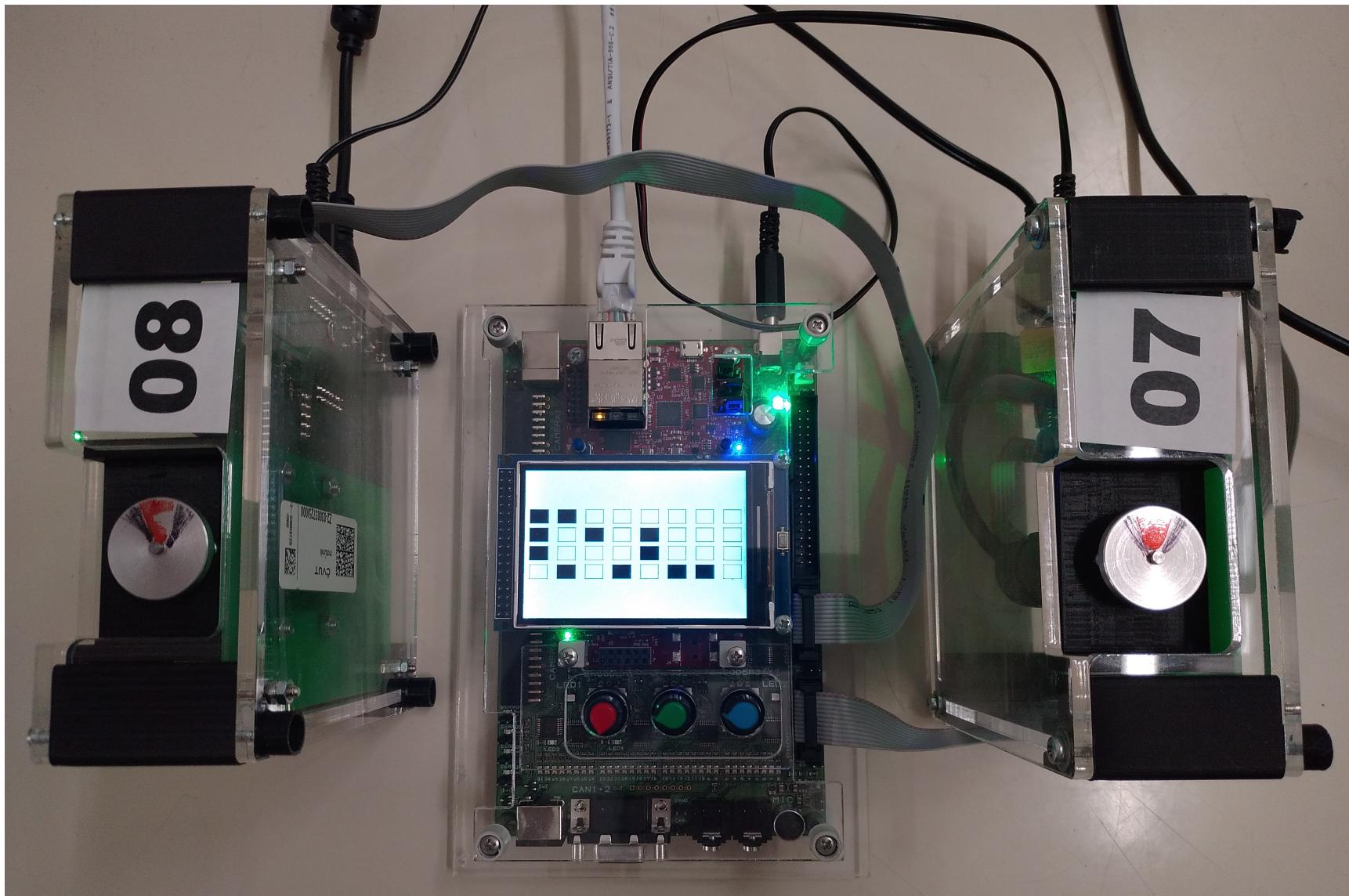
# MZ\_APO educational kit



- Existing implementation of MZ\_APO Simulink blocks:
  - Rotary knob incremental encoders
  - DC motor
- Adaptation into pysimCoder blocks
  - .xblk, .py, and .c files
- Construction of *Makefile* specific to Zynq based Linux targets
- DC motor follower with real-time PID tuning (using incremental encoders) demonstration



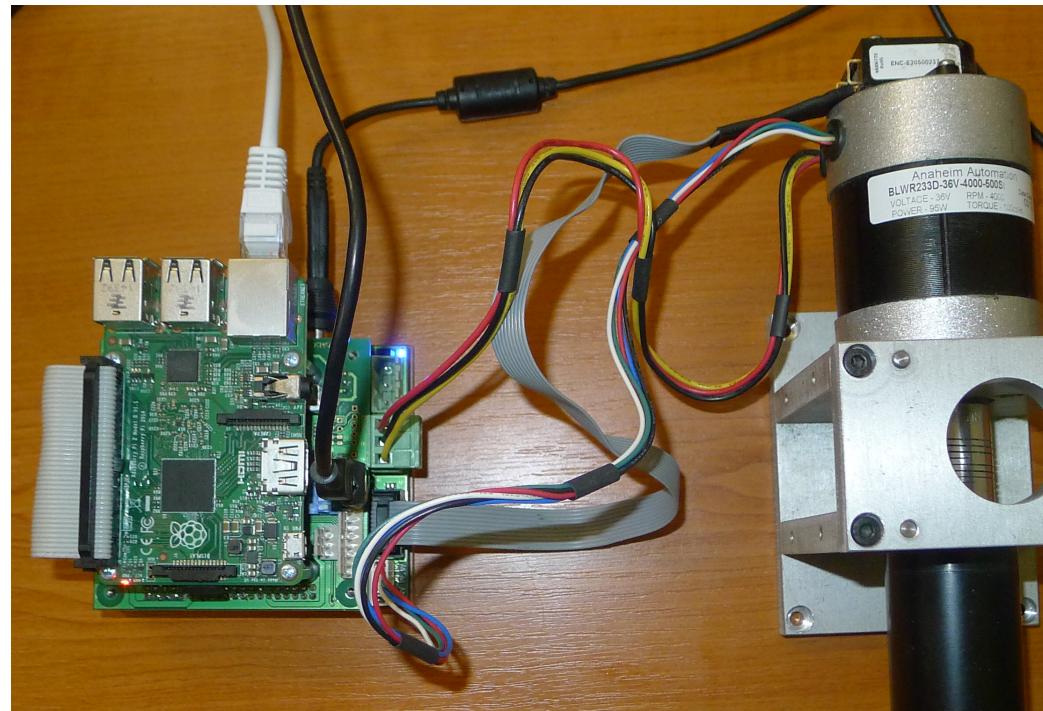
# MZ\_APO educational kit





# Raspberry Pi (PMSM control)

- RaspberryPi Model 2B (v1.1)
- Already supported in pysimCoder
- Existing Simulink project for controlling 3-Phase *Permanent Magnet Synchronous Motor*
- Possible adaptation to pysimCoder block and diagram



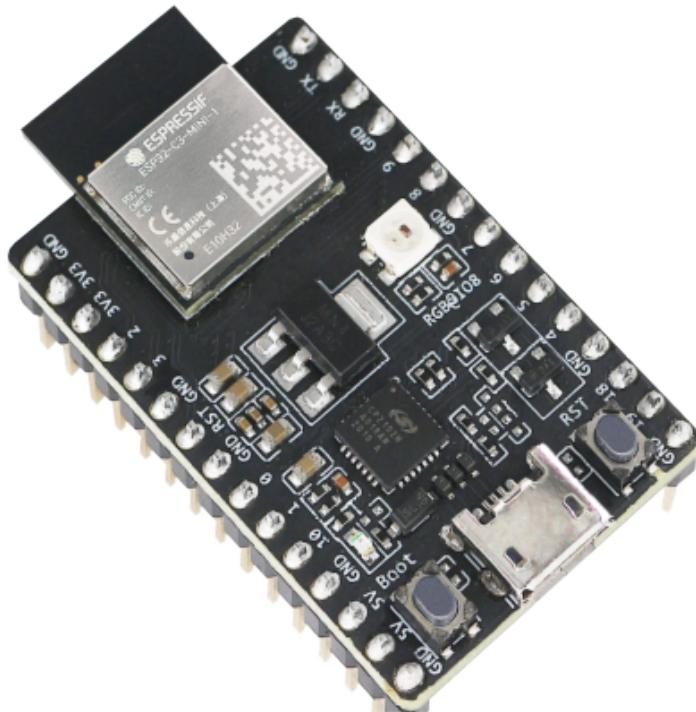


# Raspberry Pi (PMSM control)





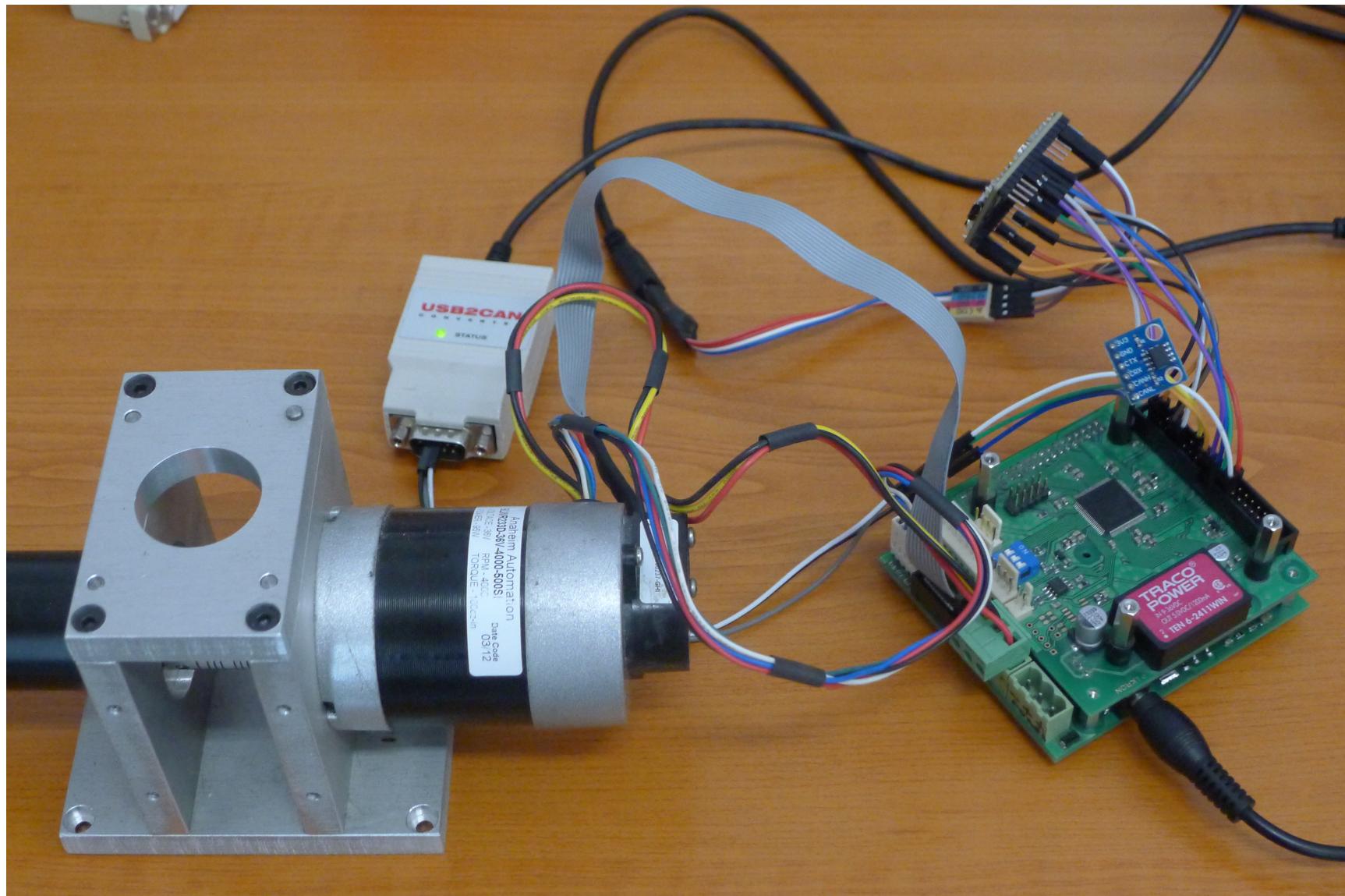
# NuttX and ESP32C3



- NuttX: Real Time Operating System
- ESP32C3-DevKitM-1
- Testing NuttX blocks on ESP32C3 board
- Replacing RaspberryPi with ESP32C3 for 3-Phase PMSM motion control
- Main changes for NuttX implementation



# NuttX and ESP32C3





# NuttX and ESP32C3

The screenshot illustrates the development environment for NuttX and ESP32C3, showing the following components:

- Library:** A block diagram library interface with tabs for input, linear, NuttX, and Raspberry.
- Diagram:** A detailed block diagram of a motor control system. It includes an ADC (A/D) block connected to a digital input (DigIn), a DAC (D/A) block connected to a digital output (DigOut), and a PulseGenerator block connected to an Integral block. The system also features an Inverse Park block, an Inverse Clarke block, a Phase3Motor block, a Sum block, a Gain block, and two CAN message blocks (canSDO\_SendMsg and canSDO\_SendMsg0). Various constant blocks (Const0, Const1, PWM\_EN, PWM\_EN0, PWM\_EN1, Const, Const2) provide parameters for the signal flow.
- pysimCoder.py:** A configuration dialog for generating code. It includes fields for Template Makefile (nuttx.tmf), Parameter script, Additional Objjs, Priority, Sampling Time (0.01), and Final Time (10).
- Terminal:** A terminal window showing the command-line build process for the ESP32C3. The command is:
 

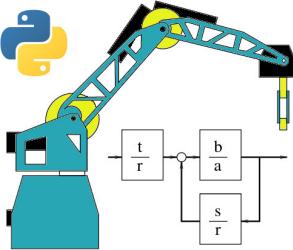
```
riscv64-unknown-elf-ld -melf32lriscv -L /home/dio/repo/pysim/wrkdir/nuttx/nuttx-export/libs -T /home/dio/repo/pysim/pysimCoder/CodeGen/nuttx/nuttx-export/scripts/esp32c3_out.ld -T /home/dio/repo/pysim/pysimCoder/CodeGen/nuttx/nuttx-export/scripts/esp32c3.ld -T /home/dio/repo/pysim/pysimCoder/CodeGen/nuttx/nuttx-export/scripts/esp32c3_rom.ld --entry=__start \
-o ../esp_nuttx_pm_hal /home/dio/repo/pysim/pysimCoder/CodeGen/nuttx/nuttx-export/startup/esp32c3_head.o \
    nuttx_main.o esp_nuttx_pm_hal.o nuttx_main-builtintab.o /home/dio/repo/pysim/pysimCoder/CodeGen/nuttx/lib/libpyblk.a --start-group -lsched -ldrivers -lboards -lc -lmm -larch -lapps -lbinfmt -lboard /usr/lib/gcc/riscv64-unknown-elf/10.2.0/rv32im/ilp32/libgcc.a --end-group
### Created executable: esp_nuttx_pm_hal
riscv64-unknown-elf-objcopy -O ihex ../esp_nuttx_pm_hal ..../esp_nuttx_pm_hex
```
- Log:** A terminal window showing the raw CAN bus data. The log shows multiple messages on the can0 interface, each consisting of a timestamp, source ID (601), and a sequence of bytes (e.g., [8] 22 31 20 00 01 00 00 00).



# Conclusions

- Vectors:
  - Testing on more complex diagrams
  - Optimization of dimension setting algorithm
  - Placement of algorithm on editor level
- GNU/Linux:
  - More MZ\_APO hardware blocks
  - New RPi block only for specific hardware
- NuttX + ESP32C3:
  - PMSM motion control without PID due to unattainable sampling frequencies

# Final Remarks



- Learned a lot both in hardware and software
- Got to use various types of controllers and peripherals
- Meaningful contributions to an open-source project
- Learned to appreciate Linux OS (for development and real-time purposes)

