**Hardware design**

* RMII[[1]](#footnote-1)
  + length matched (less than 3mm)
  + angle max 45deg
  + as short as posible
  + ground polygon around traces, many equaly placed vias
  + space between signals at lest of the witdth of the signal (idealy twice the height of the substrate) (3-W rule)
  + TX and RX separate
  + 50ohm (60ohm) impedance-controlled single-end traces[[2]](#footnote-2)
  + JLCPCB Substrate
    - Choosing
    - comparison
    - calculator
    - resulted width
    - <https://jlcpcb.com/impedance>
    - <https://jlcpcb.com/pcb-impedance-calculator>
  + ground reference – important (**find source**)
  + Delay matching – say why is not a concern in this case
  + <https://www.youtube.com/watch?v=xdUR3NzXUkc&t=876s>
  + Aktuálny delay na RMII cestách - 5.7ps/mm
* MDI
  + Simply describe differential and common impedance
  + <https://www.digikey.com/en/maker/projects/how-to-route-differential-pairs-in-kicad-for-usb/45b99011f5d34879ae1831dce1f13e93>
  + <https://resources.altium.com/p/how-design-differential-impedance-specification>
  + Traces between PHY device, transformer and RJ45 connector should be
  + designed with a differential impedance of 100Ω±10% (Dodd=50ohm).
  + KiCad calculator
  + Calculated - W:0.155 S:0.15 - Zdiff:100ohm
* ETH conn RJ45
  + chassis Ground plane around RJ452
* Power
  + Power plochy 3.3 hlavne (**find source**)
  + fencing vias - via okolo plošáku (nebudem používať, lebo nemám coplanar traces, nevylievam zem na TOP layer)
  + traces width – according to .txt
  + calculator – KiCad
* USB 2.0 USB-C
  + Zdiff = 90ohm Zcomm = 25ohm (W=0.19, S=0.15)
  + 3W
  + Ferrite bead
  + <https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/171/USB-2.0-Board-Design-and-Layout-Guidelines.pdf>

**ROS Software**

- installing WSL Ubuntu 22.04

- install ROS2 Humble

https://docs.ros.org/en/humble/Installation/Alternatives/Ubuntu-Development-Setup.html

- install and configure VcXsrv (XLaunch app) for runnig GUI apps

<https://turlucode.com/running-ros-inside-windows-10-with-gui-support-wsl/>

> install usbipd

<https://learn.microsoft.com/en-us/windows/wsl/connect-usb>

> usbipd attach --wsl --busid 1-2

> usbipd detach --wsl --busid 1-2

- install colcon

- create ws directory

- source this custom ws dir

- create custom ros2 package (ros2 pkg create RSS\_test\_pkg --build-type ament\_python --dependencies rclpy)

- downgrade setuptools to version 58.2.0

- build custom ros2 package (colcon build)

- create .py file in pkg

- make it exacutable

- run the script to test it

- install the node in setup.py

- :~/ros2\_ws\$ colcon build -> it will build the package in the directory

parameter --symlink-install will ensure that in every change of script, the build is not needed

- ros2 run <package\_name> <executable\_name> -> executable name is set in setup.py in package

(colcon build --symlink-install is needed only after creating new python script file (executable))

**DICTIONARY:**

ros2 package (name)

executable (name) (setup.py)

python script file

main function

ros2 nodes

colcon is the build tool

cmake is the build system

#!/usr/bin/env python3 - interpreter line, to tell interpreter we are going to use python3

-WSL access from LAN

<https://learn.microsoft.com/en-us/windows/wsl/networking>

-microRos platformIO

<https://www.youtube.com/watch?v=Nf7HP9y6Ovo&t=614s>

following this tutorial - Create a ROS2 Python Package - ROS2 Tutorial 1-11

<https://www.youtube.com/watch?v=c5DRTN2b2kY&list=PLLSegLrePWgJudpPUof4-nVFHGkB62Izy&index=2>

Cesty na WSL Ubutnu\_22.04\_new

* ROS2 /opt/ros/humble/
* uROS (it’s a package in ROS2) ~/microros\_ws
* ESP-IDF ~/esp/v5.0.5/esp-idf/export.sh

WIFI

* SSID: BO\_0M\_GW
* Pass: 3#IZ60P0n4UV

# Progress

* ~/uros\_test\_2 – funkčný projekt (esp-idf, uros component, nefunkčný ethernet – origo repo)
* Agenta používame na virtuálke – DHCP 192.168.1.101:8888
* WSL nemá prístup do ROS2 domény – treba vyriešiť
  + Vyriešime WSL agentom na win Pro novom ntb
* Vytvorené /home/martin/esp/projects/rss\_module
  + Submodule from <https://github.com/TaVodic/uros_mirror.git>
  + Main.c
    - publisher of angles (geometry msgs)
    - subscriber of threat level (int16)
* ROS2 packages používame na virtuálke – rss\_test\_pkg

## FreeRTOS

**scheduler** = part of the kernel responsible for deciding which task should be executing at any particular time

**context** = resources (the processor registers, stack, etc.)

**context switching** = saving the context and restoring the context

**idle task** = task running when no other tasks are able to run

**sleep =** spí, po čase sa chce vykonať

**block =** čaká na uvoňenie resources

**preemptive context switching =** forclly interrupted task

**„preempted“** = prerušený vyššou prioritou

**SMP symmetric multicore** = one instance of FreeRTOS that schedules RTOS tasks across multiple cores

**Time slice** = time between tick interrupts (periodic interrupt used by the RTOS to measure time)

comprise obsahovať

prior to its execution pred jeho vykonaním

Upon resumption  pri obnovení

Depicted zobrazené

## ESP-IDF Ubutnu22.04\_new (v5.0.5-dirty)

<https://docs.espressif.com/projects/esp-idf/en/release-v5.0/esp32/api-reference/peripherals/i2c.html>

1. http://ebook.pldworld.com/\_eBook/-Telecommunications,Networks-/TCPIP/RMII/rmii\_rev12.pdf [↑](#footnote-ref-1)
2. https://www.nxp.com/docs/en/application-note/AN13335.pdf [↑](#footnote-ref-2)