ESP-IDF

Build something

- Open ESP-IDF command prompt
- Goto the folder with the code
- Type “idf.py set-target esp32” for esp32. Er zijn ook enkele andere versies ondersteund bij IDF 4.4.
- idf.py –list-targets (dus met twee maal - voor list)
- idf.py -p COM14 flash monitor to build and upload
  (dus niet de COM port vergeten, anders probeert hij het naar alle COM-poorten te uploaden).
- Belangrijk: Houd de boot-button van het esp board ingedrukt totdat hij probeert te uploaden.
- Met CTRL+]
- idf.py menuconfig
- This seems to fix the coredump (one of both in the file ”sdkconfig”):
  CONFIG_COMPILER_CXX_EXCEPTIONS=y
  CONFIG_COMPILER_CXX_EXCEPTIONS_EMG_POOL_SIZE=1024

FreeRTOS Timer queue length?

[esp-idf/Kconfig at master · espressif/esp-idf · GitHub]

```
    config
FREERTOS_TIMER_QUEUE_LENGTH
                                    int "FreeRTOS timer queue length"
                                    range 5 20
                                    default 10
                                    help
                                        FreeRTOS provides a set of timer related API
                                functions. Many of these functions use a standard
                                        FreeRTOS queue to send commands to the timer
                                service task. The queue used for this purpose is
                                        called the 'timer command queue'. The 'timer
                                command queue' is private to the FreeRTOS timer
                                        implementation, and cannot be accessed directly.

                                        For most uses the default value of 10 is OK.
```

Pins for I2c, PWM, motor PWM, SPI, UART, CAN, touch sensors

[ESP32 Pinout - How to use GPIO pins? Pin mapping of ESP32 (microcontrollerslab.com)]

Tick time of free rtos instellen

CONFIG_FREERTOS_HZ

Book title

Design like a Robot!

Easy Multitasking on ESP32 with CleanRTOS

Meer een C++ dingetje, denk ik:
Als je object-members niet in de initializer-list initialiseert, maar erbuiten, dan zijn ze nog ongeinitialiseerd binnen de initializer-list en kun je ze daar niet gebruiken.

Probleem: ESP IDF builds executeren veel trager op WROOM dan arduino IDE builds

Vermoedelijke oorzaak:

Na aanpassen van sdkconfig naar onderstaande was het opgelost:

```
# System
CONFIG_ESP_SYSTEM_EVENT_TASK_STACK_SIZE=4096
CONFIG_FREERTOS_UNICORE=n
CONFIG_FREERTOS_HZ=1000
CONFIG_ESP32_DEFAULT_CPU_FREQ_240=y
CONFIG_ESP32_DEFAULT_CPU_FREQ_MHZ=240
CONFIG_ESPTOOLPY_FLASHMODE_QIO=y
CONFIG_ESPTOOLPY_FLASHFREQ_80M=y
```

(en ook ergens een optie om geen verbose maar info te printen en om voor performance te compilen ipv voor debug. Dat geeft echter minder goede tracability van evt fouten, las ik ergens, en de performance gain was beperkt.)

Grootste boosdoener: de frequenties. Met name de FREERTOS_HZ.
Die stond standaard op 100. Gevolg: VTaskDelay(100) wacht niet 0.1s maar 1s. Daar gaat je button timing 😊.

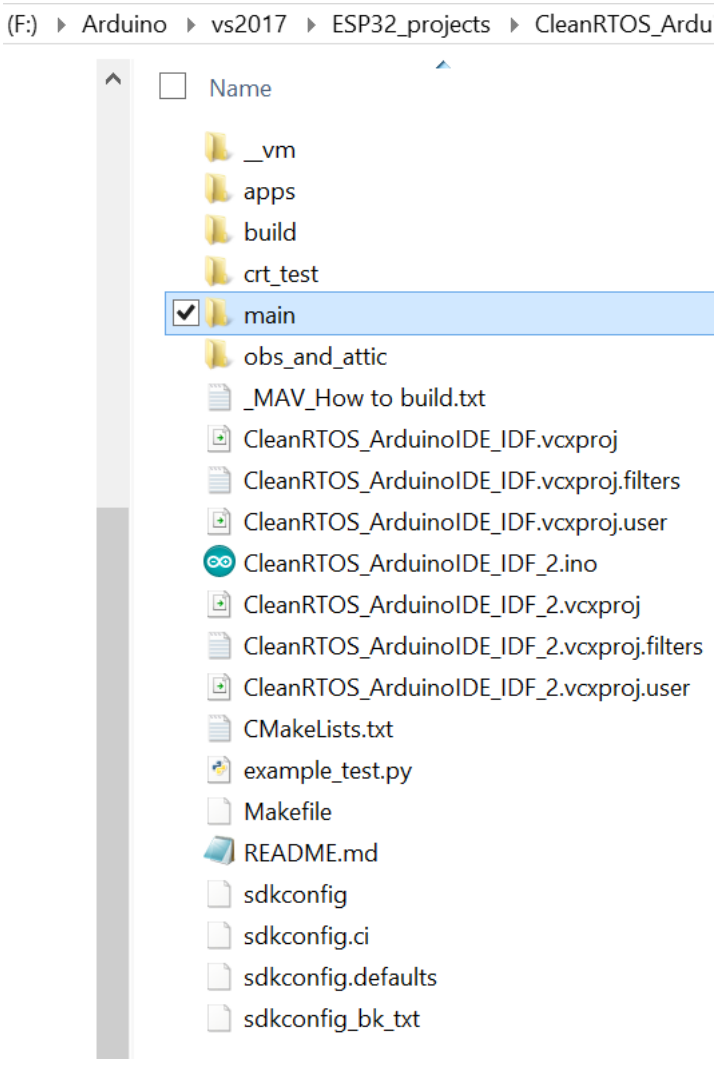
Probleem: ninja flash wrong boot mode detected

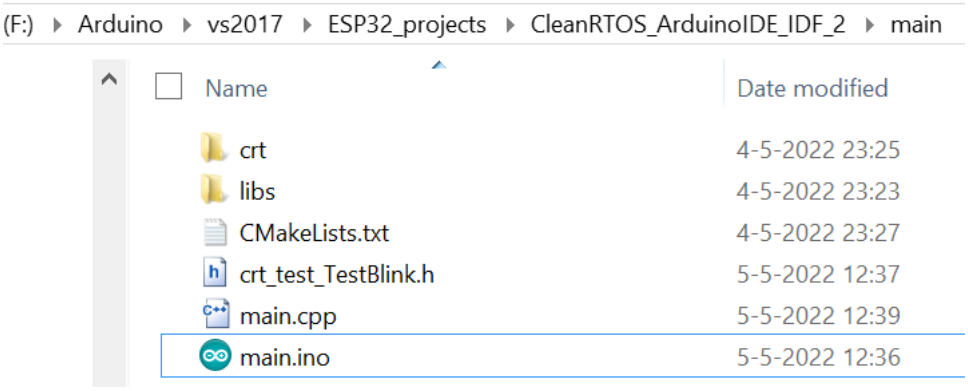
Oplossing: houd boot button ingedrukt tijdens/aan het eind van het linken. Laat hem pas los tijdens het “uploaden”.

Gekke crashes – stack overflow of canary error

Voorkom het dereferencen van null pointers. Alloceer voldoende grote stack sizes aan de tasks.

Laatste bouw-systeem:





Building ESP code

Note: the folder named "main" contains by default:
* a subfolder "libs" (Other libraries can be added here)
* a subfolder "crt" (The Clean RTOS lib)

* your application The "main" folder contains your application.
You can put your own application there

or copy the contents of an application folder within the
"apps" folder or the "crt_test" folder to the "main" folder)

* the application that you like to build
* the application must contain a file named main.cpp.

Code for ESP32 can be built in next 2 ways:

1. Using ESP-IDF 4.4. CMD prompt.
In that case, go to the directory that CONTAINS the FOLDER named "main".
(this parent directory, that is, not the folder callled "main")

type in the prompt:
idf.py -p COM4 flash monitor

(if the device is connected to another port, select that one.
you can find out about the port in either arduino IDE or in
windows device settings menu)

* a CMakeList.txt file that specifies which folders to include in the build
for that application.
* the file main.cpp must contain a function called "app_main"

* typically, you can rename a main.ino file to main.cpp and comment out
inside it: "#define CRT_ARDUINO_ID" (see crt_test as examples)

2. Using the Arduino IDE
* typically, you can rename a main.ino to main.cpp and uncomment
inside it: "#define CRT_ARDUINO_ID" (see crt_test as examples)
* it will then use setup() and loop() rather than app_main()
* In the Arduino IDE, select the main.ino (in the "main" folder)
and build it.

IDF monitor: how to pause and resume scrolling

Answer: Pause key for pausing, then space for resuming

ledcAttachPin

ledcWrite

Build probleem: na bouwen:

rst:0x10 (RTCWDT_RTC_RESET),boot:0x33 (SPI_FAST_FLASH_BOOT)
invalid header: 0xffffffff
invalid header: 0xffffffff
invalid header: 0xffffffff
invalid header: 0xffffffff

Geprobeerd (half afgaand op een forum):
in sdkconfig

~~Veranderd: CONFIG_BOOTLOADER_OFFSET_IN_FLASH=0x1000~~
~~naar: CONFIG_BOOTLOADER_OFFSET_IN_FLASH=0x0000~~

Nee, dat lijkt geen goed idee: dat zou de bootloader overschrijven.
Dat forum ging over het branden van de bootloader, niet van de code.

Uiteindelijk leek de oorzaak te maken te hebben met de (7v-) servo-voeding.
Als ik die tijdens het branden loskoppelde, ging het goed.

Bouw probleem: no cmake_minium_required command is present

```
F:\Arduino\vs2017\ESP32_projects\CleanRTOS_ArduinoIDE_IDF_4\main>idf.py -p COM4 flash monitor
Executing action: flash
Running cmake in directory f:\arduino\vs2017\esp32_projects\cleanrtos_arduinoide_idf_4\main\build
Executing "cmake -G Ninja -DPYTHON_DEPS_CHECKED=1 -DESP_PLATFORM=1 -DCCACHE_ENABLE=1 f:\arduino\vs2017\esp32_p
...
CMake Warning (dev) in CMakeLists.txt:
  No project() command is present.  The top-level CMakeLists.txt file must
  contain a literal, direct call to the project() command.  Add a line of
  code such as

    project(ProjectName)

  near the top of the file, but after cmake_minimum_required().

  CMake is pretending there is a "project(Project)" command on the first
  line.
This warning is for project developers.  Use -Wno-dev to suppress it.

-- The C compiler identification is unknown
-- The CXX compiler identification is unknown
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - failed
-- Check for working C compiler: C:/TDM-GCC-32/bin/gcc.exe
-- Check for working C compiler: C:/TDM-GCC-32/bin/gcc.exe - broken
CMake Error at H:/Espressif/tools/cmake/3.20.3/share/cmake-3.20/Modules/CMakeTestCCompiler.cmake:66 (message):
  The C compiler

    "C:/TDM-GCC-32/bin/gcc.exe"

  is not able to compile a simple test program.

  It fails with the following output:

    Change Dir: F:/Arduino/vs2017/ESP32_projects/CleanRTOS_ArduinoIDE_IDF_4/main/build/CMakeFiles/CMakeTmp

    Run Build Command(s):H:/Espressif/tools/ninja/1.10.2/ninja.exe cmTC_82bb1 && [1/2] Building C object CMake

    FAILED: CMakeFiles/cmTC_82bb1.dir/testCCompiler.c.obj
    C:\TDM-GCC-32\bin\gcc.exe    -o CMakeFiles\cmTC_82bb1.dir\testCCompiler.c.obj -c testCCompiler.c
    ninja: build stopped: subcommand failed.
```

Oorzaak: niet compileren vanuit de main dir, maar vanuit de directory erboven.

Arduino IDE examples gebruiken en bouwen met ESP IDF

Gebruik de tips uit Kolban (een paar git pulls, zodat de arduino libraries voor esp32 in een “component” komen). Vervolgens de main.c naar main.cpp renamen.

Serial.print etc gebruiken met ESP IDF

Dat kan, als je een baudrate van 115200 kiest. De ESP gebruikt die immers ook voor zijn “native” communicatie.

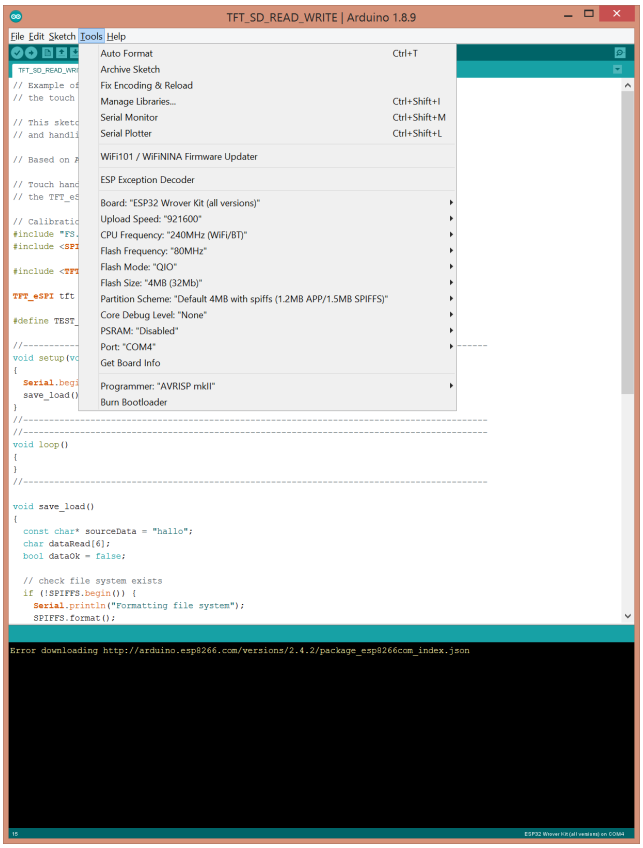
0x40080400: _init at ??:?

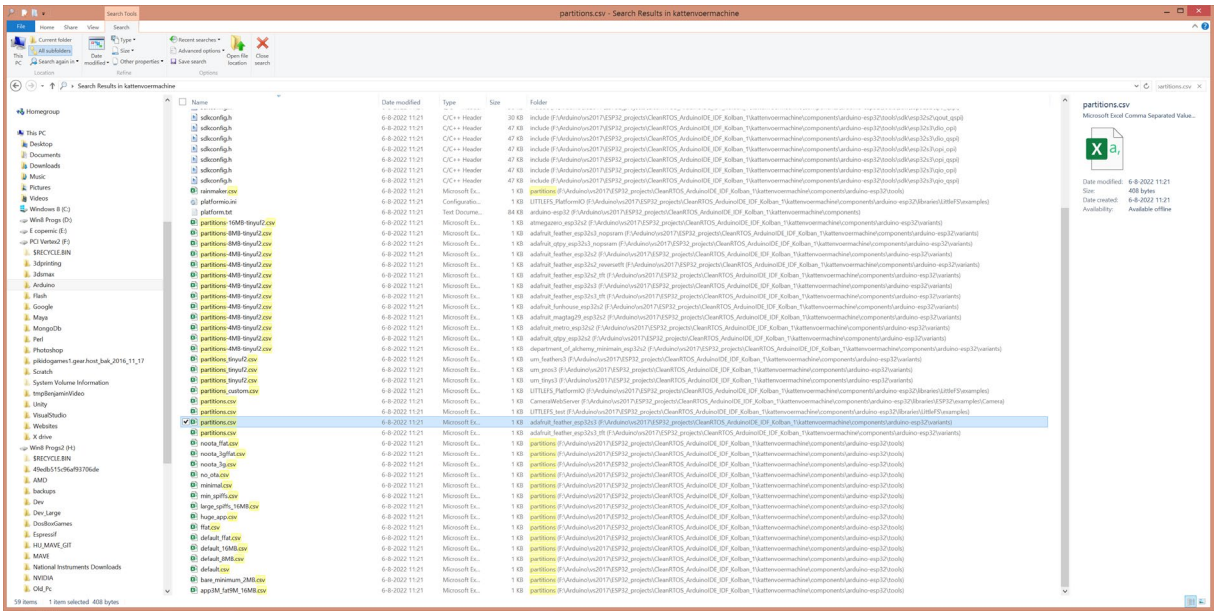
Dit wordt vermoedelijk veroorzaakt door de PSRAM read error als er geen PSRAM is, en je dat wel aangeeft in je config.

Het gebruik van een TFT-SD card (SPIFF) met IDF build werkt niet

Het geeft bij SPIFF.begin() een error. Bij bouwen met de arduino IDE werkt het wel.

Op internet grasduinend heeft dat vermoedelijk te maken met een andere “partitie configuratie die ingesteld staat bij de arduino IDE”:





Nog een paar keer proberen, zegt ie of sdkconfig (denk ik):
LINES value must be >=2 and <=100: got 101

Misschien heb ik zelf eens een lijn toegevoegd..

Even een lijn met commentaar verwijderen.

`CONFIG_ADC_DISABLE_DAC=y nu uitgecommentarieerd.`

```
Er blijkt nu ook een commando te zijn:
idf.py partition-table

Dat resulteert in:
Partition table binary generated. Contents:

*****************************************
# ESP-IDF Partition Table
# Name, Type, SubType, Offset, Size, Flags
nvs,data,nvs,0x9000,24K,
phy_init,data,phy,0xf000,4K,
factory,app,factory,0x10000,1M,
```

En verder een commando waarmee je dit naar de esp32 kunt flashen.
Die laat de huidige partition settings zien, vermoedelijk op basis van sdkconfig.

Yep, op basis van de geselecteerde csv in sdkconfig.
Zie:
[Partition Tables - ESP32 - — ESP-IDF Programming Guide latest documentation (espressif.com)]

Die van de arduinoIDE zijn kennelijk hier te vinden:
D:\Users\mave\AppData\Local\Arduino15\packages\esp32\hardware\esp32\2.0.2\tools\partitions

(zie [Partition Schemes in the Arduino IDE – Robot Zero One])

Uit die folder kopieer ik default.csv naar mijn kattevoermachinefolder, en rename het naar partitions.csv, want in sdkconfig staat:
`CONFIG_PARTITION_TABLE_CUSTOM_FILENAME="partitions.csv"`

Dat heeft helaas het probleem niet opgelost.
Dan: in de regel erboven zetten:
`CONFIG_PARTITION_TABLE_CUSTOM = y`

(was uitgecommentarieerd).
Dat hielp niet maar bovenstaande lijn werd weer automatisch uitgecommentarieerd.
Vermoedelijk moet er iets anders geset worden. Misschien erboven
CONFIG_PARTITION_TABLE_SINGLE_APP=y uitcommentarieren.

Toen kwam er een melding dat het niet in 2MB flash size past. Arduino IDE gaf aan dacht ik dat er 4MB beschikbaar is.
Dus verander ik dat ook:
CONFIG_ESPTOOLPY_FLASHSIZE_2MB=y uitcommentarieren en
CONFIG_ESPTOOLPY_FLASHSIZE_4MB=y aancommentarieren.

Nu zie je dat hij begint met bouwen met de nieuwe custom partition table.

Er staan allerlei partities in: nvs, ota, app0, app1 en spiff. Geen idee waar dat allemaal voor is.

Yep, that did the trick.

Ehm, OTA staat kennelijk voor “Over the air” updates, die software updates via wifi (ipv usb) mogelijk maakt.

Samengevat:

1. In Arduino IDE werkt iets: kijk welke partition scheme wordt gebruikt (in Tools menu).
2. Kopieer de betreffende csv uit:
   D:\Users\mave\AppData\Local\Arduino15\packages\esp32\hardware\esp32\2.0.2\tools\partitions
   naar de project folder voor je ESP32 project (waar je altijd idf.py -p COM4 flash monitor intypt), en verander de naam in partitions.csv
3. Verander in sdkconfig:
   Comment out `CONFIG_PARTITION_TABLE_SINGLE_APP`
   Comment in `CONFIG_PARTITION_TABLE_CUSTOM = y`
   comment out `CONFIG_PARTITION_TABLE_CUSTOM_FILENAME="partitions.csv"`
   comment out `CONFIG_ESPTOOLPY_FLASHSIZE_2MB=y`
   comment in `CONFIG_ESPTOOLPY_FLASHSIZE_4MB=y` (want dat had die default partition met SPIFF nodig – zo bleek uit een foutmelding zonder dat).
4. Als je nu bouwt, werkt SPIFF.begin() hopelijk.

NB:
SPIFFS is **a lightweight filesystem created for microcontrollers with a flash chip, which is connected by SPI bus, like the ESP32 flash memory**

Maar waar lijkt het nu op: Het werkt ook zonder de SD card.
Samengevat: mijn SPIFFS voorbeeld slaat het niet op de SD card op, maar in een deel van de ESP32 flash memory.

Yep, het is 1.5MB van het onboard flash memory, zie:
SPIFFS in ESP32 (tutorialspoint.com)

Gebruik van een SD card is eenvoudig. Sluit hem aan als in in dat ILI9341 voorbeeld met sdcard. Gebruik gewoon het standaard sd card uit de arduino IDE.

Json dan?

Great example en uitleg (including filtering streaming input van een website): ArduinoJson: Efficient JSON serialization for embedded C++

Het JsonConfigFile laat prachtig een combinatie zien van de ArduinoJson met de SD library.

Niet Clean maar FullClean

Gebruik niet
idf.py clean
maar
idf.py fullclean

(dat cleant meer)

Component maken

Je kunt ipv makelist bij main aanpassen voor includen van libs, de lib folder in de components folder zetten. Vervolgens (vrijwel) dezelfde CMakeLists file in de betreffende list file toevoegen, zoiets:
Edit following two lines to set component requirements (see docs)

```
set(COMPONENT_REQUIRES )
set(COMPONENT_PRIV_REQUIRES )

set(COMPONENT_SRCS)
set(COMPONENT_ADD_INCLUDEDIRS
"."
"src"
"src/internals"
"examples/AllWaitables"
"examples/Flag"
"examples/Queue"
"examples/Timer"
"examples/HelloWorld"
"examples/TwoTasks"
"examples/MutexSection"
"examples/Pool"
"examples/Handler"
"examples/Logger"
"examples/TenTasks"
)
```

register_component()

Dan wordt de component gepre-compileerd (minder vaak gebouwd) en hoef je die paden niet meer in de CmakeLists file op het main niveau te setten.

Verwerken van feedback van Stijn: maak static function deel van base class

Spinlock vs mutex

Spinlock is an aggressive mutex. In mutex, if you find that the resource is locked by someone else, you (the thread/process) switch the context and wait (*non-blocking*).

Whereas spinlocks *do not* switch context and keep spinning. As soon as the resource is free, they go and grab it. In this process of spinning, they consume many CPU cycles. Also, on a uni-processor machine, they are useless.

Caution – Spinlock causes the process to be uninterruptible. Thus Spinlock is feasible only for a short wait.

Spinlocks can be used with portENTER_CRITICAL/portEXIT_CRITICAL, dan wel taskENTER_CRITICAL / taskEXIT_CRITICAL
Beide functies zijn identiek en blokkeren de interrupts op BEIDE cores.

Op een andere plek:
ESP-IDF FreeRTOS SMP Changes - ESP32-C3 - — ESP-IDF Programming Guide v4.3 documentation (espressif.com)

Staat juist dat de andere core alleen blokkeert als die gelijktijdig op diezelfde spinlock wacht. (maar wel spinning – dus zonder zijn taak af te staan).

Conclusie: als je spinlocks gebruikt om op de user core aan resource sharing te doen, blokkeert de wifi-core niet.
... mmm... en wat als de wifi-core zijn periodieke task interrupt verstuurt? .. Dat zal wel niet gebeuren, omdat wel de interrupts op BEIDE cores geblokkeerd worden. Dus ook ISR van pinnen..

Atomic operations on ESP32

Volgens iemand zijn 32 bit reads en writes atomic, maar updaten niet.

Misschien dat daarmee een veilige circulaire buffer gemaakt kan worden zonder multiple-access risico's?

Vb:

```
Void addToBuffer(number:int)
{
        if(busy

}
```

Atomic compare and swap operations:
portATOMIC_COMPARE_EXCHANGE_8
portATOMIC_COMPARE_EXCHANGE_16
portATOMIC_COMPARE_EXCHANGE_32

Ik krijg LittleFS maar niet gecompileerd.

https://github.com/joltwallet/esp_littlefs

En nog eentje van lorol:

https://github.com/lorol/LITTLEFS/blob/master/src/esp_littlefs.c

Environment / configuratieconstanten

Een manier om die te zetten, lijkt:

Zet een sdkconfig.ci file in de directory van je CMakeLists file.
Daarin kun je dan regels zetten zoals:

CONFIG_EXAMPLE_FORMAT_IF_MOUNT_FAILED=y

FreeRTOS queue

NB: de freeRTOS queue (die de CleanRTOS queue ook wrapt), schrijft in zijn write functie kopieen van de aangeboden objecten middels memcopy. Daarbij worden dus niet de copy-constructors of operator= van het betreffende object gebruikt.

Het is daarom van belang dat het betreffende object eenvoudig kopieerbaar is, en niet bijvoorbeeld references bevat (naar variabelen uit een base-class).
Als je references met memcopy copieert, wijzen ze immers nog steeds naar het oorspronkelijke (mogelijk tijdelijke-) object.

ESP32-S3 Pinout

ESP32 S3 Pin Reference · bdring/FluidNC Wiki · GitHub

Windows11 detecteert de ESP32-S3 (WROOM-1 devkit) niet

Probeer driver van:

https://www.silabs.com/documents/public/software/CP210x_VCP_Windows.zip
-> nope, dat hielp niet.

Aan het project kattenvoermachine heb ik een dependency toegevoegd:
idf.py add-dependency esp_tinyusb~1.0.0

Dat heeft echter niet geleid tot vindbaarheid van de poorten van de ESP32-S3 devkit.

Wat bleek: de VR-bril-link-cable, welke ik gebruikte, is kennelijk **stuk of ongeschikt**.
Een andere USB-C kabel werkte wel.

Access voor nieuwe gebruikers

Nieuwe gebruikers maken contact met de esp32 als accesspoint (pw protected).
Ze kunnen dan de wifi-credentials invullen. Vervolgens kunnen ze het via local wifi gebruiken.

Iemand heeft daar al een library voor gemaakt (te installeren via de Arduino IDE):
https://github.com/tzapu/WiFiManager

ESP32-S3: Serial.print not working (in ArduinoIDE)

Oplossing: Zet de instelling “CDC USB on boot” op disabled.

All-in one voor ESP32-S3

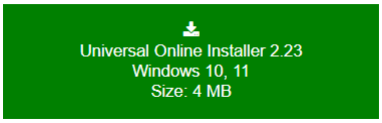
- Snor hem op in de netwerken, en connect.
- Stel je WIFI credentials in.
- Daarna kun je er een server op draaien.
- Die ook met andere servers contact kan maken.
- En via een locale DNS – naam makkelijk gevonden kan worden.

Het werkt zo awesome, dat ik de code hieronder even paste (met lettersize 1):

Installatie voor studenten:

2023_09_25

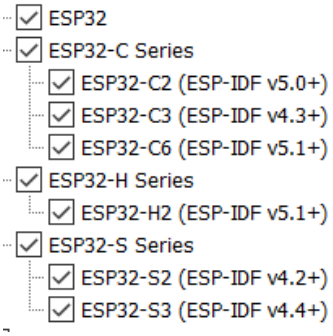
Ik kies nu voor:



Daarbinnen kies ik voor v5.1.1 (release version).
NB: voorheen gebruikte ik 4.4.

Ik verander de driveletter van het aanbevolen pad in D:
(het komt dan in: D:\Espressif\frameworks\esp-idf-v5.1.1)

Ik kies voor de default geselecteerde componenten + vinkt alle typen ESP32 chips aan:



Vervolgens wordt er 1,1 GB gedownload..

Installation has failed with exit code 128:
2023-09-25 13:07:47.246 Running command: D:\Espressif\tools\idf-git\2.39.2\cmd\git.exe -C D:\Espressif\frameworks\esp-idf-v5.1.1 config --local core.fileMode false

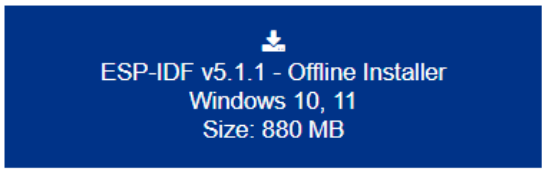
fatal: --local can only be used inside a git repositor

chat gpt tip gevolgd: ga naar de frameworks directory git bash, en type daar:
git clone -b v5.1.1 --recursive https://github.com/espressif/esp-idf.git D:\Espressif\frameworks\esp-idf-v5.1.1

Vervolgens laat ik git bash maar even open staan (hopelijk helpt het met rechten oid).
En start de install (die in downloads staat) opnieuw. Dit keer kies ik die directory:
D:\Espressif\frameworks\esp-idf-v5.1.1 als existing directory om in te installeren.
(De tools erna in D:\Espressif)

Yo, nu slaagt het wel. Maar nu opent hij in bovenstaande directory, maar heeft hij nog eens 20GB geinstalleerd in :
D:\Espressif\frameworks\Espressifframeworksesp-idf-v5.1.1
Dat is ook niet de bedoeling.
Aan de data kan ik zien dat op D:\Espressif\frameworks\esp-idf-v5.1.1
nog de git-clone versie staat.
Die verwijder ik.
De complete Espressif folder deleten..

Nu anders: ik download en installeer de laatste offline versie:



Weer alle ESP32 chips aangevinkt.

Weer D:\Espressif

Powershell opent na installatie in D:\Espressif\frameworks\esp-idf-v5.1.1
pinnen aan desktop.

Test:
esp32-s3 wordt niet herkend (met usb-c snoer van de externe disk).
USB Device Driver - ESP32-S3 - — ESP-IDF Programming Guide latest documentation (espressif.com)

Twee verschillende usb-c snoeren in de laptop geprobeerd: geen detectie. Ook niet met die van de externe disk.
Vervolgens een usb-a naar usb-c snoer geprobeerd. Toen werd de S3 wel gedetecteerd.

(zie: 456 Hassle-Free ESP32 USB (ESP32-C3, ESP32-S2, ESP32-S3) - YouTube
Je kunt dus uart(rtx,sdx) (Serial0) en CDC/JTag USB (Serial) parallel gebruiken.
(leuk van de S3: die heeft alles in een: bovenstaande + OTG USB (voor device emulatie).
door de ingebouwde serial to uart converter kun je met een losse module zonder veel extra componenten een eind komen.. en heeft veel extra bruikbare pins)

Check in device manger of met Arduino ide wat de com-port is.
Dan:
Ga naar de folder waar je code (en je main dir) in staat, dan:
cd examples
cd get-started
cd helloworld

Check comport in device manager of Arduino ide. In mijn geval COM4.
idf.py -p COM4 flash monitor

CTRL+] voor afbreken van monitor.

Git Bash

Maak folders:
C:\

ESP32

Dan: kies een van de teamleden uit die eigenaar wordt van de repo van de lasergame repo van het team.
Alleen die persoon, doet het volgende:
1. Maak op github een nieuwe, volledig lege private repo aan met de naam lasergame
   (dus zonder readme.md, .gitignore of whatever).

2. Ga in Git Bash naar een folder waarbinnen een folder wilt maken voor het lasergame project.
Bijvoorbeeld C:\ESP32 (of een andere lokale folder naar keuze).

3. git clone mavehu/lasergame-template lasergame

4. git remote remove origin
5. git remote add origin [URL van je nieuwe private repo]
6. git push -u origin master

~~type in die folder:~~

~~git clone https://github.com/espressif/esp-idf-template.git lasergame~~

~~Dat levert een basic hello world applicatie.~~

Mkdir components

Cd components

git clone https://github.com/espressif/arduino-esp32.git

(eventueel update submodules ?

git submodule update --init

)

Rename main.c naar main.cpp.
En gebruik die met setup en loop.

idf.py menuconfig

……..Okee, de lasergame template is nu denk ik klaar. Let's test:

Ik maak op TheMave een private test_lasergame repo

Ik haal die binnen op een folder naar keuze met:

3. git clone "https://github.com/mavehu/lasergame-template.git" lasergame
4. cd lasergame
5. git remote remove origin
6. git remote add origin [volledige URL van de nieuwe private team repo inclusief .git]
7. git push -u origin master

De andere teamleden clonen nu die team-repo ook, met:
git clone [URL van de private team repo] lasergame

Vervolgens doet elk teamlid voor zich:

Cd lasergame
cd components

git clone https://github.com/espressif/arduino-esp32.git

$ git clone https://github.com/espressif/arduino-esp32.git

Cloning into 'arduino-esp32'...
remote: Enumerating objects: 52533, done.
remote: Counting objects: 100% (1939/1939), done.
remote: Compressing objects: 100% (776/776), done.
remote: Total 52533 (delta 1015), reused 1687 (delta 903), pack-reused 50594
Receiving objects: 100% (52533/52533), 2.22 GiB | 19.01 MiB/s, done.
Resolving deltas: 100% (31692/31692), done.
error: unable to write file H:/My Drive/HuDev/ESP32/test_lasergame/components/arduino-esp32/.git/objects/pack/pack-24a8a9366b791ecd5a2b2cab183a79ac07911a9d.pack: No such file or directory
fatal: unable to rename temporary '*.pack' file to 'H:/My Drive/HuDev/ESP32/test_lasergame/components/arduino-esp32/.git/objects/pack/pack-24a8a9366b791ecd5a2b2cab183a79ac07911a9d.pack'
fatal: fetch-pack: invalid index-pack output

Conclusie: op Google drive kun je geen code zetten die met .pack files werkt.

idf.py fullclean

idf.py menuconfig

idf.py build

idf.py set-target esp32s2 # For ESP32-S2

idf.py set-target esp32s3 # For ESP32-S3

Dependencies lock doesn't exist, solving dependencies. CMake Error at D:/Espressif/frameworks/esp-idf-v5.1.1/tools/cmake/build.cmake:540 (message): HINT: Component "espressif/esp_tinyusb" has suitable versions for other targets: "esp32s2", "esp32s3". Is your current target "esp32" set correctly? ERROR: Solver failed processing dependency "espressif/esp_tinyusb" from the manifest file "C:/HuDev/test_lasergame/main\idf_component.yml". Cannot find versions of "espressif/esp_tinyusb" satisfying "~1.0.0" for the current target "esp32". Call Stack (most recent call first): D:/Espressif/frameworks/esp-idf-v5.1.1/tools/cmake/project.cmake:547 (idf_build_process) CMakeLists.txt:6 (project) -- Configuring incomplete, errors occurred! See also "C:/HuDev/test_lasergame/build/CMakeFiles/CMakeOutput.log". cmake failed with exit code 1, output of the command is in the C:\HuDev\test_lasergame\build\log\idf_py_stderr_output_22872 and C:\HuDev\test_lasergame\build\log\idf_py_stdout_output_22872

Bij S2 en S3 geen probleem, maar USB as device werkt niet voor normale ESP32.
Dus in dat geval:
ga naar main/idf-component.yml en commentarieer de volgende regel uit:
espressif/esp_tinyusb: “~1.0.0”

Wat blijkt: esp32 (zonder S2 en S3) kan alleen gebruikt worden met ESP-IDF versies tussen 4.4.0 en 4.4.99

(zucht!!)

Dus 5.1.1. deinstalleren en 4.4… installeren. Dit keer maar op de C-schijf (is hopelijk sneller).

TIP: Simuleer je chip online op Wokwi

.. dat kan veel dev tijd schelen.