Lazarus für ESP8266 mit FpcUpDeluxe einrichten

Meine Quellen:

https://wiki.lazarus.freepascal.org/Xtensa
https://forum.lazarus.freepascal.org/index.php?topic=57725.msg430121#msg430121
https://de.ubunlog.com/pyenv-installiert-mehrere-Python-Versionen-auf-Ihrem-System/
https://github.com/espressif/ESP8266_RTOS_SDK/tree/d412ac601befc4dd024d2d2adcfaa319c7463e36/components/esp8266/include/driver
https://github.com/ccrause/fpc-esp-freertos

Alle orangenen Stellen müssen angepasst werden! Und keine Leerzeichen in den Pfaden verwenden!

Mein System:

Linux Mint 22.1 Cinnamon
FPC 3.3.1 x86_64-linux-gtk2
Lazarus 4.99 (rev main_4_99-2003-g2448ebf956)

FpcUpDeluxe:

FPCUPdeluxe V2.4.0f for x86_64-linux-gtk2
von hier:
https://github.com/newpascal/fpcupdeluxe/releases/latest

Die Version: fpcupdeluxe-x86_64-linux

Installation FpcUpDeluxe:
Zuerst folgende Dateien im Terminal installieren:

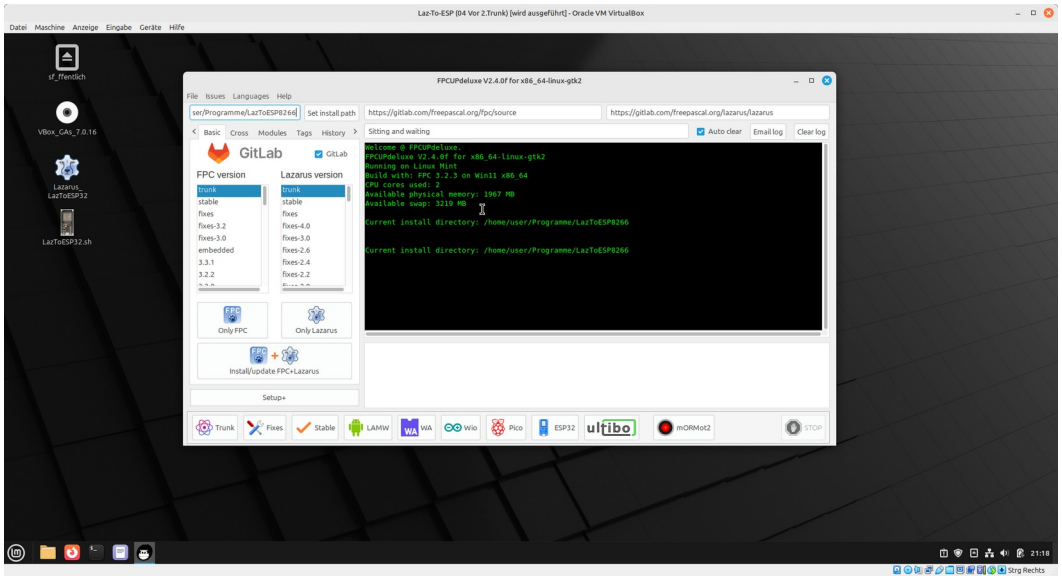
```
sudo apt-get install libx11-dev
sudo apt-get install libgdk-pixbuf2.0-dev
sudo apt-get install libpango1.0-dev
sudo apt-get install libgtk2.0-dev
sudo apt-get install freeglut3-dev
sudo apt-get install git
```

Dann kopiert man sich die Installationsdatei am besten in einen neuen Ordner und klickt diese dort mit der rechten Maustaste an. Dann Eigenschaften und unter Zugriffsrechte bei „Der Datei erlauben sie als Programm auszuführen“ einen Haken machen und schließen.

Lazarus Trunk (jetzt Main) installieren:

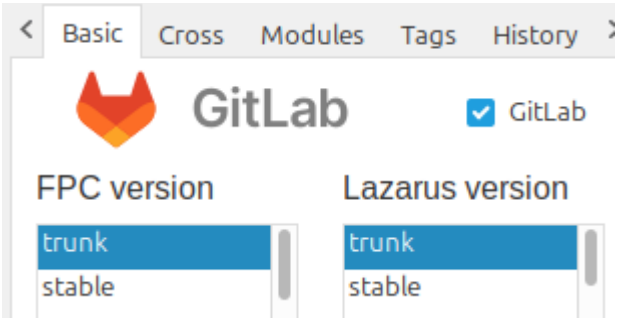
Zuerst legt man einen neuen leeren Ordner an in dem die Installation erfolgen soll. Bei mir /home/user/Programme/LazToESP8266.

Nun FpcUpDeluxe starten.



Jetzt auf „Set Install Path“ klicken und zu dem vorher neu angelegten Ordner navigieren.

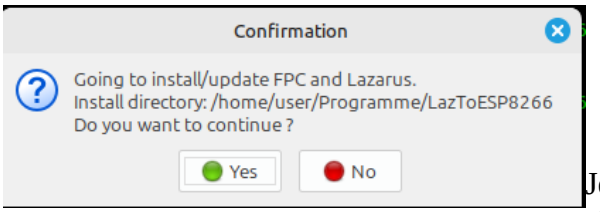
Nun Trunk und Trunk anwählen:



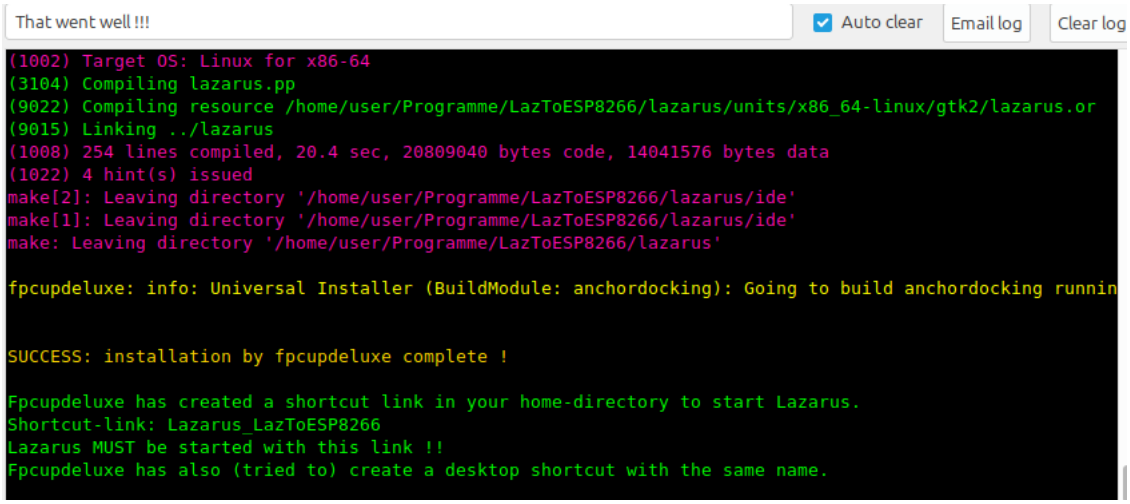
Optional: Auf „Setup+“ klicken. Die Option Be extra verbose liefert mehr Ausgaben im Logfenster und wenn man möchte kann man gleich die Docked Lazarus IDE mit installieren.

Jetzt „install/update the selected version of both FPC and Lazarus“ klicken:

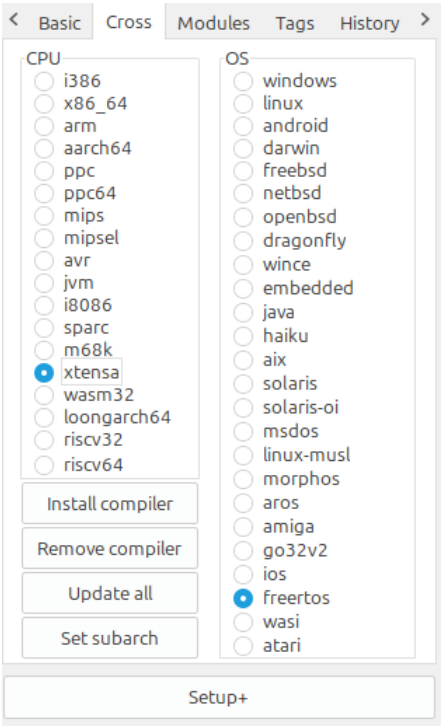
Mit Yes bestätigen:



Jetzt beginnt die Installation und mit etwas Glück läuft sie ohne Probleme durch. Da die Trunk (Main) Versionen nicht stabil sind kann es vorkommen das die Installation mal einige Tage nicht funktioniert. Dann einfach öfter mal probieren.

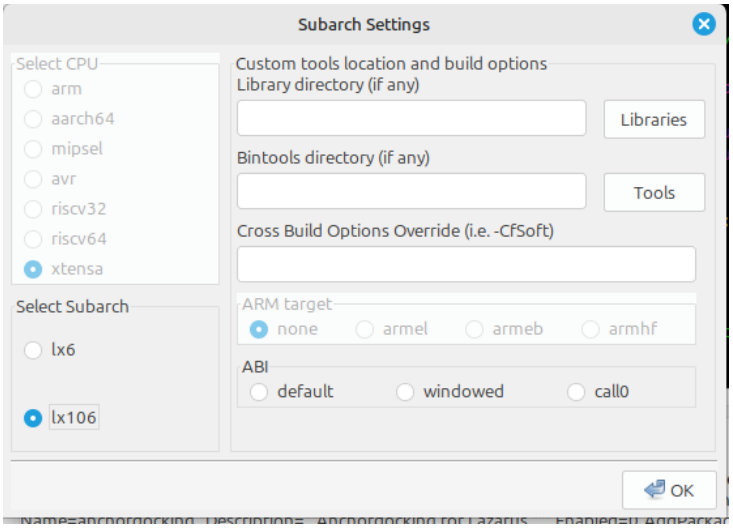


Wenn man möchte kann man Lazarus nun starten. Hat man die Option Dockedform gewählt sollte man nun bei Modern IDE und Modern Form Editor den Haken setzen. Dann „Start IDE“. Zum Umstellen auf Deutsch „Tools, Options, General, German, okay“. IDE schließen und wieder starten. Nun ist alles auf Deutsch eingestellt. Lazarus schließen.

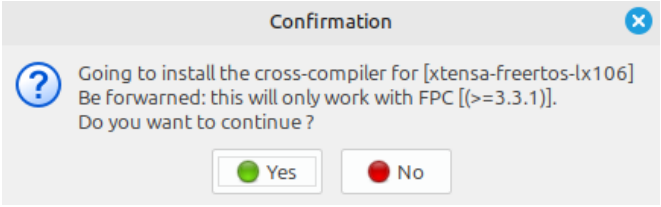


Weiter gehts in FpcUpDeluxe. Hier den Reiter Cross aktivieren. Dort bei CPU extensa und bei OS freertos wählen.

Nun „Set subarch“ klicken. Dort einen Haken bei lx106 machen und Ok.

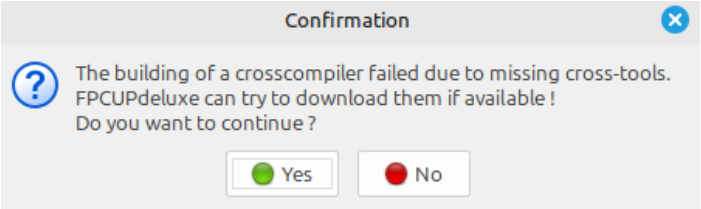


Nun „Install Compiler“ klicken. Es kommt folgendes Fenster:

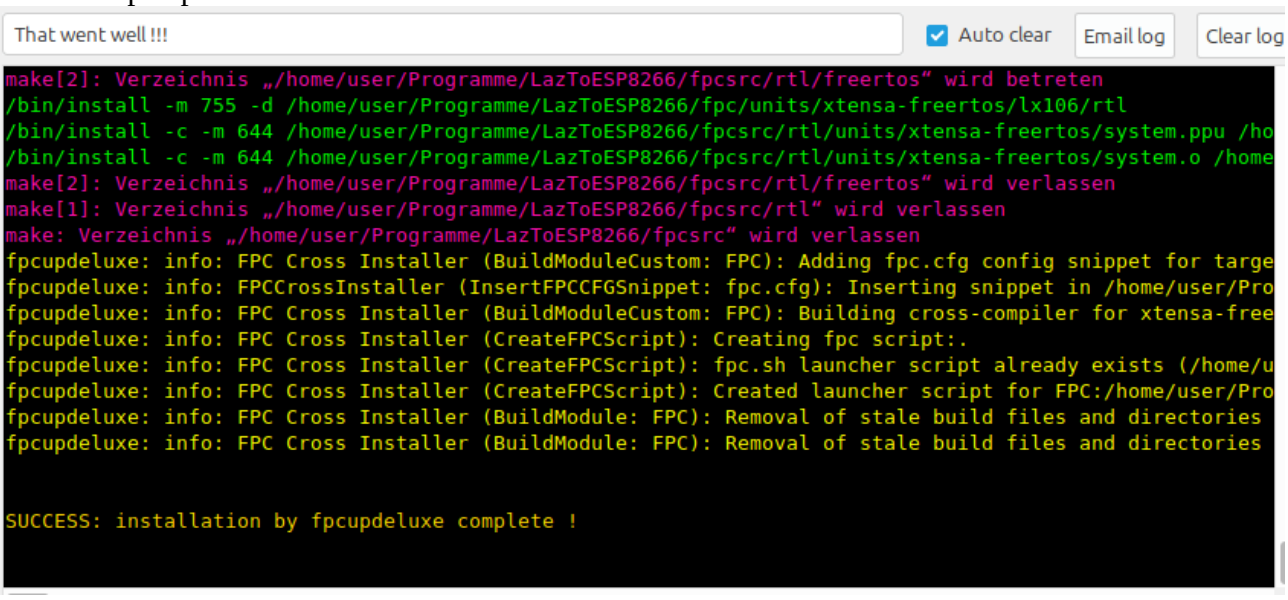


Dies mit Yes bestätigen.

Nun das downloaden der benötigten Dateien bestätigen:



Nun hat FpcUpDeluxe alles installiert:



Anpassen der Python-Version:

Unter /home/user/Programme/LazToESP8266/cross/bin/xtensa-freertos/esp-rtos-3.4 findet man eine Datei requirements.txt. Darin stehen die benötigten Phytonpakete bzw. installiert diese Datei die benötigten Packete.
Damit Lazarus mit den benötigten Paketen laufen kann muss man es mit einer passenden Python-Version starten. Dafür muss zunächst eine Virtuelle Umgebung geschaffen werden. Dazu benötigt man das Programm pyenv.

Führen Sie folgende Befehle in Ihrem Terminal aus, um pyenv zu installieren:

```
sudo apt-get install -y make build-essential git libssl-dev zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev libncursesw5-dev xz-utils tk-dev

curl -L https://raw.githubusercontent.com/pyenv/pyenv-installer/master/bin/pyenv-installer | bash

nano ~/.bash_profile
```

Nun die folgenden Zeilen am Ende der Datei hinzu fügen, hier müssen wir "**USER**" durch Ihren Systembenutzernamen **ersetzen**.

```
export PATH="/home/USER/.pyenv/bin:$PATH"
eval "$(pyenv init -)"
eval "$(pyenv virtualenv-init -)"
```

Die Änderungen mit Strg + O, Enter speichern und mit Strg + X nano beenden.
Um pyenv dem System bekannt zu machen muss (immer beim neu Öffnen des Terminals) folgender Befehl eingegeben werden:

```
source ~/.bash_profile
```

Möchte man wissen welche Python-Versionen in unserem System verfügbar sind folgenden Befehl eingeben:

```
pyenv install --list
```

Meine Arduino IDE verwendet 3.7.2. Leider lies sich diese Version bei mir nicht installieren. Ich habe mich deshalb für 3.7.17 entschieden.

Jetzt die Version 3.7.17 installieren:

```
pyenv install   3.7.17
```

Um die Version 3.7.17 von Python zu aktivieren:

```
pyenv global  3.7.17
```

Terminal nicht schließen. Lazarus muss in diesem Zustand aus dem Terminal heraus gestartet werden damit es unter pyenv läuft!!

Hat man Lazarus aus versehen geschlossen bzw. möchte man aus irgend einem Grund Lazarus unter pyenv starten kann man das mit diesen Befehlen tun oder sich gleich eine kleine Start-Bash erstellen:
Einen Texteditor öffnen und folgendes eingeben:

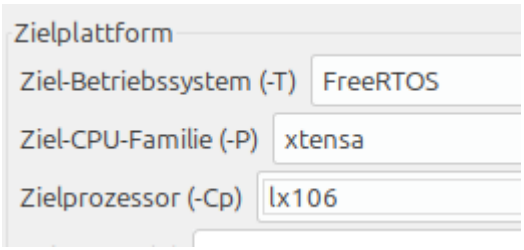
```
#!/bin/bash
cd
source ~/.bash_profile
pyenv global 3.7.17
echo Es ist folgende Python_Version aktiv:
python3 --version
cd /home/user/Programme/LazToESP8266/lazarus
./lazarus
```

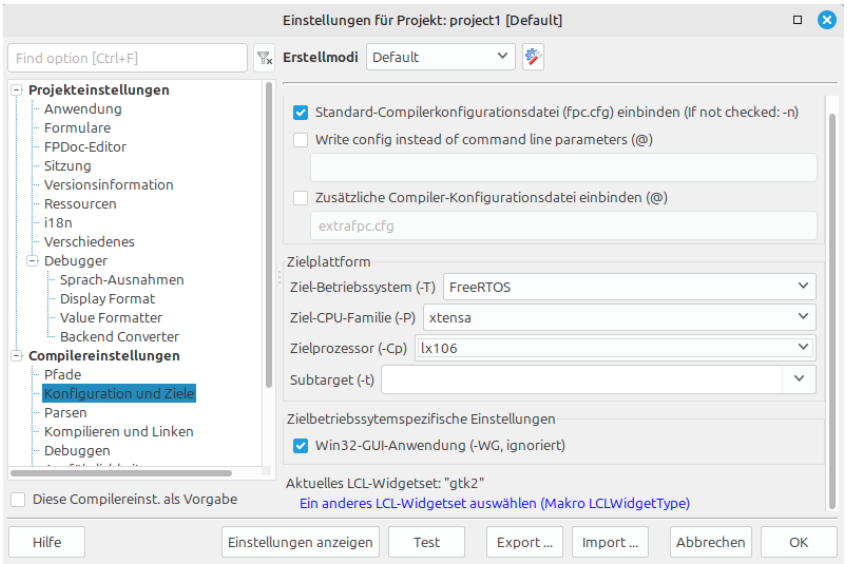
Ins Lazarusverzeichnis wechseln und starten:

```
cd /home/user/Programme/LazToESP8266/lazarus
./lazarus
```

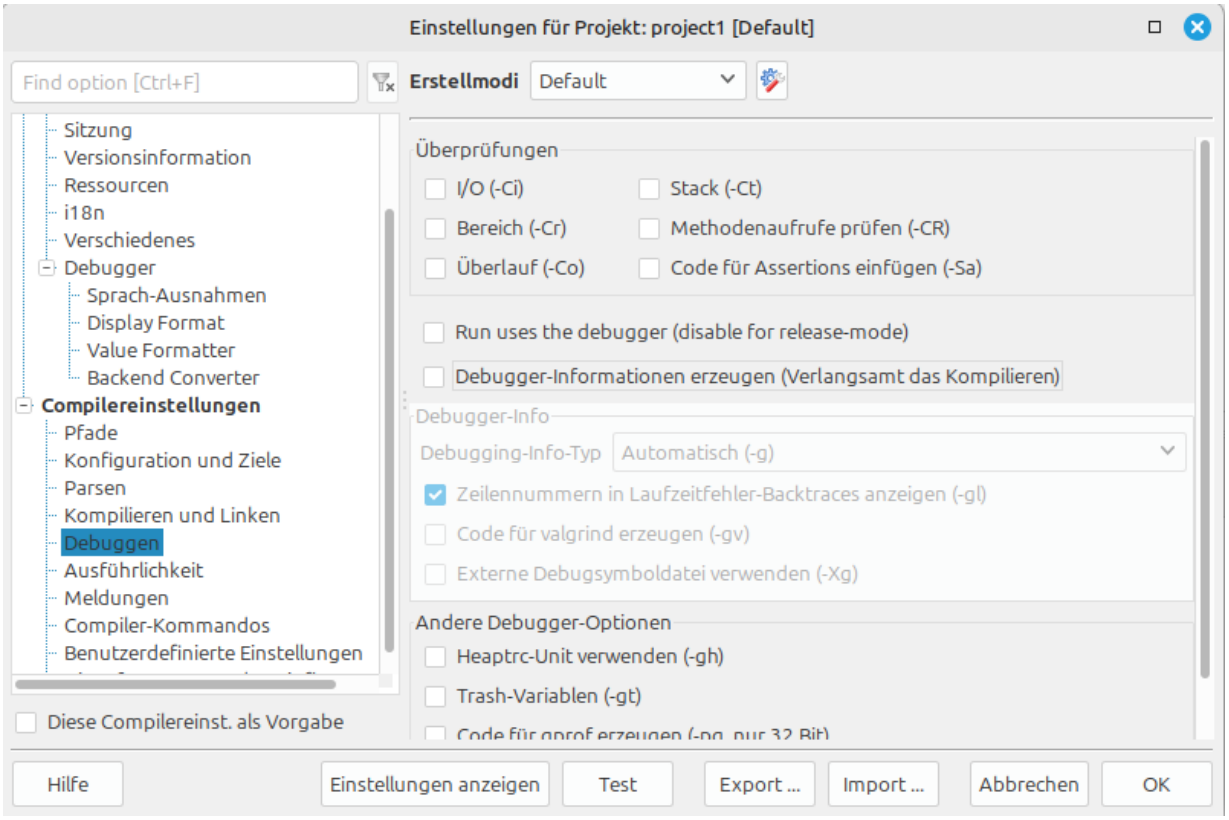
Nachdem Lazarus gestartet ist muss noch der passende Erstellmodi eingestellt werden:
Projekt, Projekteinstellungen, Konfiguration und Ziele

Das hier Einstellen:





Dann in dem Reiter „Debuggen“ und dort alles deaktivieren:



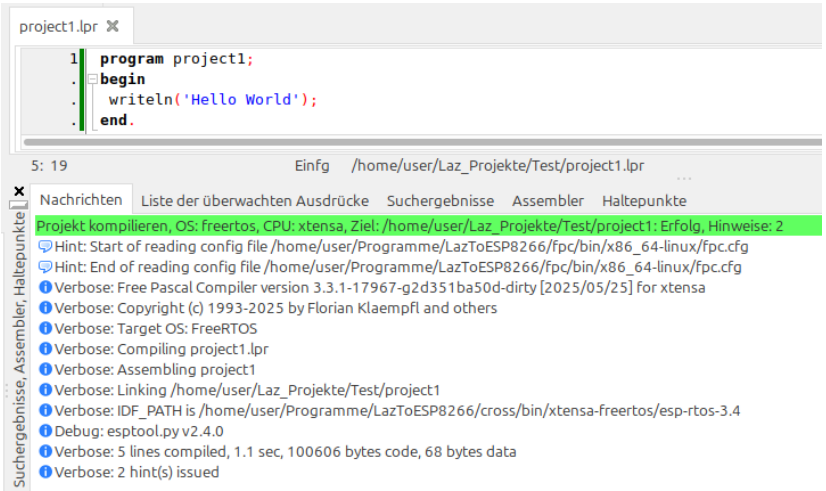
Wenn man möchte das diese Einstellung bei einem neuen Projekt gleich drin steht muss man einen Haken bei „Diese Compilereinst. als Vorgabe“ machen. Mit Ok das Fenster schließen.

Jetzt auf Datei, Neu, Einfaches Programm und folgendes Minimal Projekt eingeben:

```
program project1;
begin
 writeln('Hello World');
end.
```

Das Mini Projekt mit Speichern Unter in ein eigenes Verzeichnis abspeichern. Dann auf Start und Kompillieren (oder Strg+F9).

Ergebnis:



Nach dem ersten Mal kompillieren (**mit pyenv**) befindet sich unter /home/user/Programme/LazToESP8266/cross/bin/xtensa-freertos/esp-rtos-3.4/ der Ordner venv. Darin findet man nun u. a. die benötigten Packages.
Ab diesem Zeitpunkt kann Lazarus ganz normal über das von FpcUpDeluxe angelegte Starter Icon geöffnet werden.

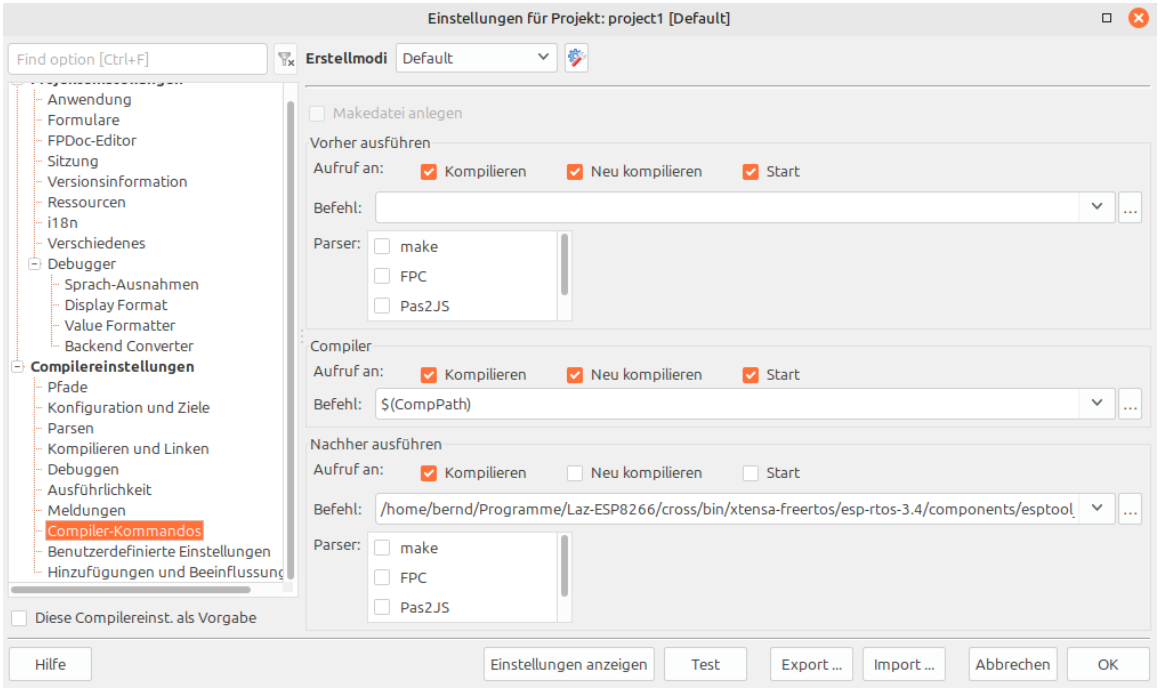
Damit das Kompillierte Programm auf den ESP8266 geflasht wird muss noch folgendes getan werden:

Der Befehl zum Flashen schaut in etwa so aus. Es müssen die Pfade angepasst werde!

/home/user/Programme/LazToESP8266/cross/bin/xtensa-freertos/esp-rtos-3.4/components/esptool_py/esptool/esptool.py --chip esp8266 --port "/dev/ttyUSB0" --baud 115200 --before "default_reset" --after "hard_reset" write_flash -z --flash_mode "dio" --flash_freq "40m" --flash_size "4MB" 0x0 /home/user/Programme/LazToESP8266/cross/lib/xtensa-freertos/lx106/bootloader.bin 0x10000 project1.bin 0x8000 /home/user/Programme/LazToESP8266/cross/lib/xtensa-freertos/lx106/partitions_singleapp.bin

Unter VirtualBox:
PlatformIO [Errno 13] Permission denied: '/dev/ttyUSB0'
ESP8266 anstecken.
sudo chmod a+rw /dev/ttyUSB0

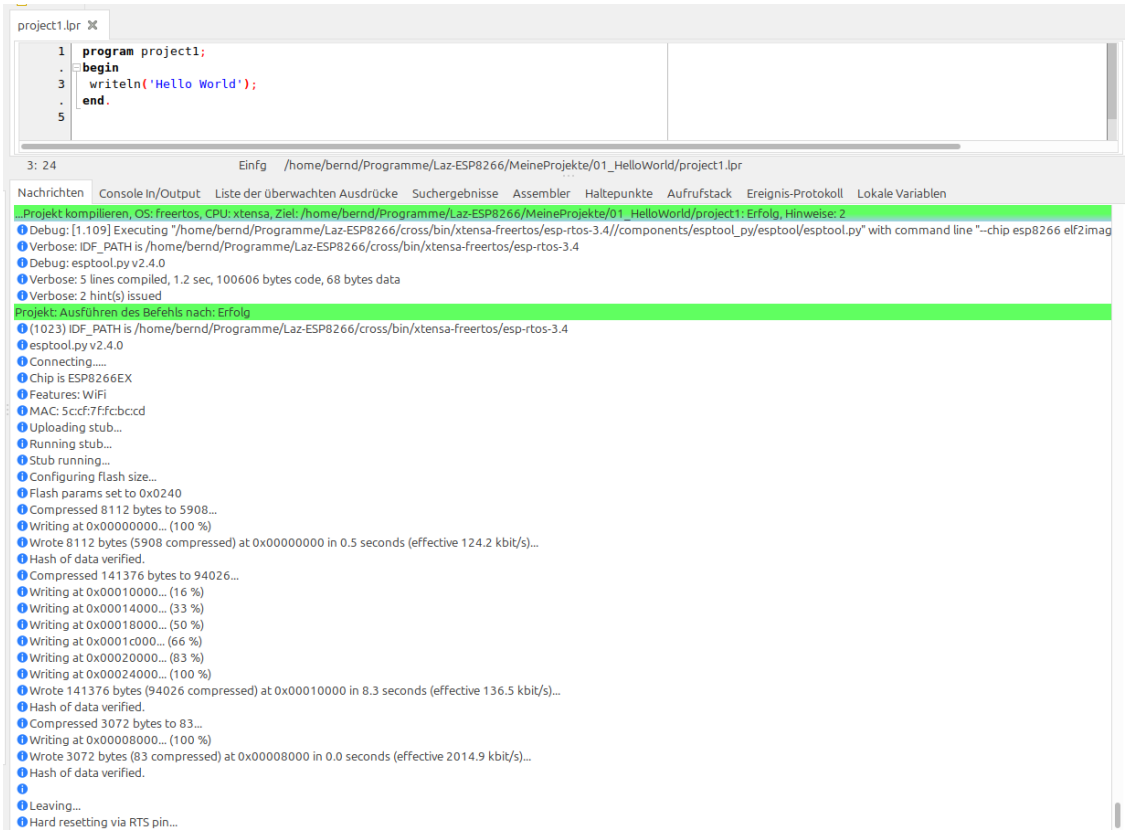
Dieser Befehl muss nun bei Projekt, Projekteinstellungen, Compiler-Kommandos in die Zeile Befehl bei Nachher ausführen kopiert werden. Die Haken bei Neu kompilieren und Start entfernen! Achtung Vorne und Hinten keine Leerzeichen!



Wenn man möchte Haken bei Diese Compilereinst. als Vorgabe. OK.

ESP8266 anstecken und dann kompillieren [Strg+F9]

So sollte es dann in Lazarus aussehen:



Das erste Blinker Programm.
Das Programm hab ich hier gefunden:
https://forum.lazarus.freepascal.org/index.php?topic=57725.msg430121#msg430121

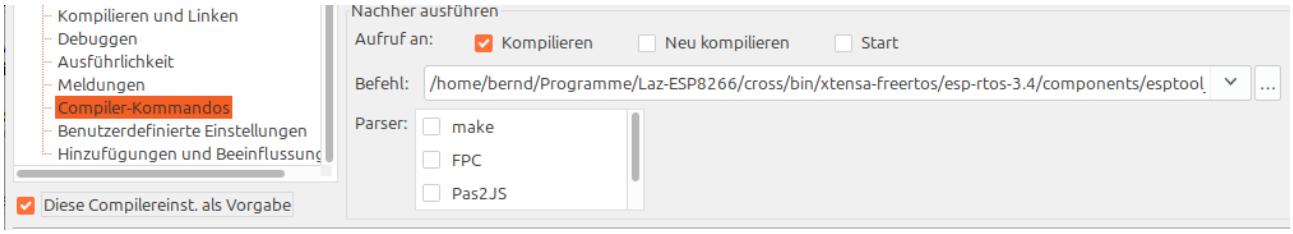
Lazarus ganz normal öffnen. Datei, Neu, Einfaches Programm, Ok.
Diesen Quelltext eingeben:

```
program project1;

type
  Tgpio_config = record
    pin_bit_mask: uint32;  // Pin mask of pins to configure
    mode: integer;         // 0 = disabled, 1 = input, 2 = output, 6 = output
open drain
    pull_up_en: integer;   // 0 = disabled, 1 = enabled
    pull_down_en: integer; // 0 = disabled, 1 = enabled
    intr_type: integer;    // 0 = disabled, 1 = positive edge, 2 = negative
edge, 3 = any edge, 4 = low level, 5 = high level
  end;

const
  // GPIO pin number for pin connected to LED
  LED = 2;  // NodeMCU LED on ESP-12E module

// Return value is error code, 0 = success
function gpio_config(constref gpio_cfg: Tgpio_config): integer; external;

// mode: GPIO_MODE_DISABLE = 0, GPIO_MODE_INPUT = 1, GPIO_MODE_OUTPUT = 2,
GPIO_MODE_OUTPUT_OD = 6
function gpio_set_direction(gpio_num: integer; mode: integer): integer;
  external;

// level: Low = 0, High = 1
function gpio_set_level(gpio_num: integer; level: uint32): integer; external;

procedure vTaskDelay(xTicksToDelay: uint32); external;

var
  cfg: Tgpio_config;

begin
  cfg.pin_bit_mask := 1 shl ord(LED);
  cfg.mode := 1;
  cfg.pull_up_en := 0;
  cfg.pull_down_en := 0;
  cfg.intr_type := 0;
  gpio_config(cfg);

  gpio_set_direction(LED, 2);
  repeat
    writeln('.');
    gpio_set_level(LED, 0);
    vTaskDelay(10);
    writeln('*');
    gpio_set_level(LED, 1);
    vTaskDelay(10);
  until false;
end.
```

Das Projekt in einem eigenen Verzeichnis abspeichern.

Wenn beim ersten Projekt „Hello World“ der Haken „Diese Compilereinst. als Vorgabe“ gesetzt wurde hat Lazarus diese Einstellungen ins neue Projekt übernommen. Ansonsten alle Einstellungen wie oben beschrieben neu eingeben. Da im Flash Befehl der Projektname bisher hart Codiert wurde sollte man dafür besser das IDE Makro TargetFile verwenden. Man muss dann den Befehl nicht jedesmal anpassen:

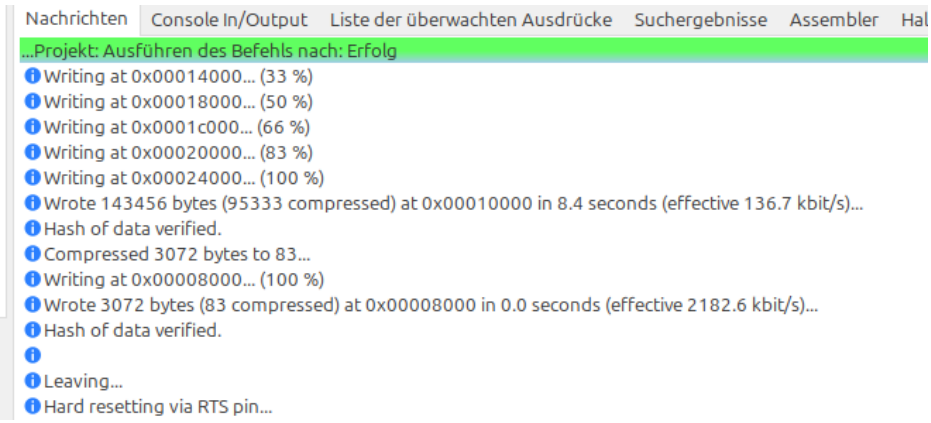
/home/bernd/Programme/Laz-ESP8266/cross/bin/xtensa-freertos/esp-rtos-3.4/components/esptool_py/esptool/esptool.py --chip esp8266 --port "/dev/ttyUSB0" --baud 115200 --before "default_reset" --after "hard_reset" write_flash -z --flash_mode "dio" --flash_freq "40m" --flash_size "4MB" 0x0 /home/bernd/Programme/Laz-ESP8266/cross/lib/xtensa-freertos/lx106/bootloader.bin 0x10000 $(TargetFile)**.bin** 0x8000 /home/bernd/Programme/Laz-ESP8266/cross/lib/xtensa-freertos/lx106/partitions_singleapp.bin



Wenn man möchte Haken bei „Diese Compilereinst. als Vorgabe“

ESP8266 anstecken und dann kompillieren [Strg+F9]

So sollte es dann in Lazarus aussehen:

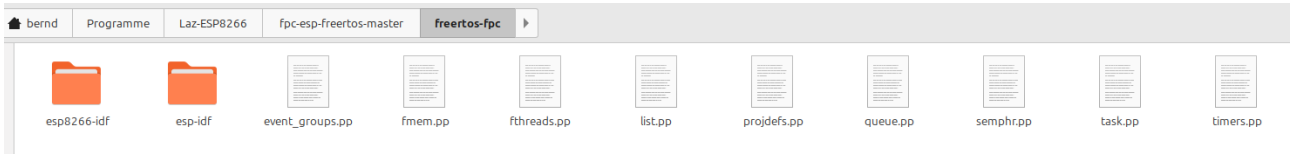


Die integrierte blaue LED des ESP8266 sollte nun blinken!

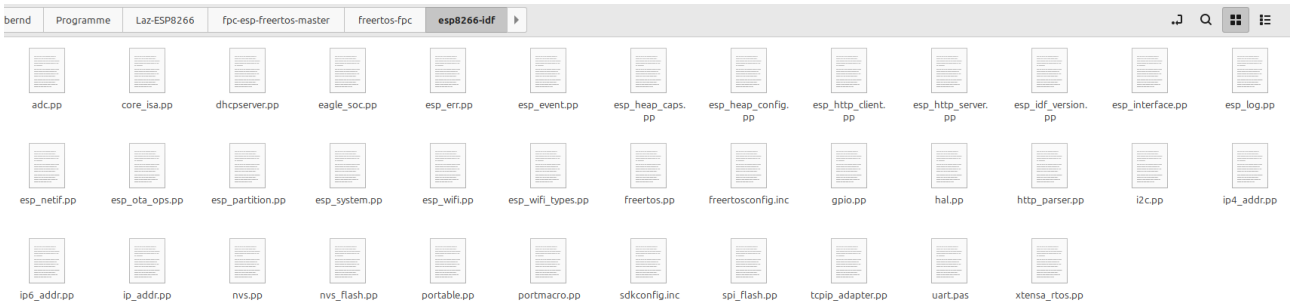
Um den ESP8266 richtig nutzen zu können benötigt man aber noch die passenden Pascal Header (Bindings). Auf der GitHub Seite von ccrause findet man diese. Da er sie unter die Mozilla Public License Version 2.0 gestellt hat sollte es meiner Meinung nach kein Problem sein diese zu benutzen. Also am besten hierhin:
https://github.com/ccrause/fpc-esp-freertos/tree/master
springen und das ganze Repo Klonen oder als zip downloaden und einen Stern dort lassen.
Direkter Download:
https://github.com/ccrause/fpc-esp-freertos/archive/refs/heads/master.zip

Dann im Lazarus-Verzeichnis einen neuen Ordner mit dem Namen units anlegen. Ich habe es unter home/bernd/Programme/Laz-ESP8266/cross/units gemacht.

Nun folgende Header-Dateien in diesen units Ordner kopieren:



Und noch alle Header im Ordner esp8266-idf:



Dann die Datei fpc.cfg öffnen und an der Stelle den Suchpfad für units ergänzen:
(Mein Pfad zur fpc.cfg: /home/bernd/Programme/Laz-ESP8266/fpc/bin/x86_64-linux)

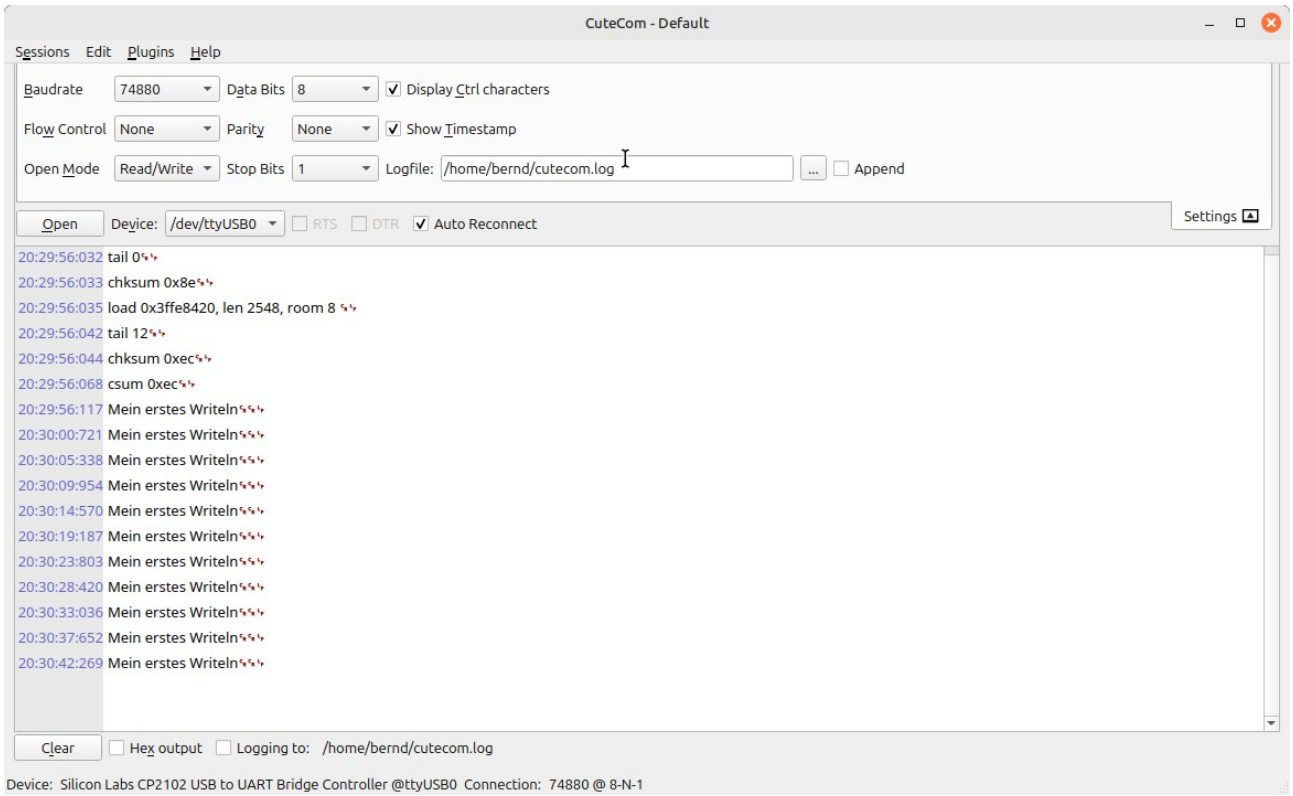
searchpath for units and other system dependent things
-Fu/home/bernd/Programme/Laz-ESP8266/fpc/units/$fpctarget
-Fu/home/bernd/Programme/Laz-ESP8266/fpc/units/$fpctarget/*
-Fu/home/bernd/Programme/Laz-ESP8266/fpc/units/$fpctarget/rtl
-Fu/home/bernd/Programme/Laz-ESP8266/cross/units

Nun können die Header in uses eingebunden werden und werden gefunden!

Das erste Programm mit Ausgabe auf die serielle Schnittstelle:

```
program project1;
uses uart, esp_err;

procedure vTaskDelay(xTicksToDelay: uint32); external;

const
 UART_PORT: Tuart_port = UART_NUM_1;

var
  uart_cfg: Tuart_config;

begin
 uart_cfg.baud_rate  := (115200*40) div 26;
 uart_cfg.data_bits  := UART_DATA_8_BITS;
 uart_cfg.parity     := UART_PARITY_DISABLE;
 uart_cfg.stop_bits  := UART_STOP_BITS_1;
 uart_cfg.flow_ctrl  := UART_HW_FLOWCTRL_DISABLE;
 EspErrorCheck(uart_param_config(UART_PORT, @uart_cfg));
 EspErrorCheck(uart_driver_install(UART_PORT, 256, 0, 0, nil, 0));
 repeat
  writeln('Mein erstes Writeln');
  vTaskDelay(300);
 until false;
end.
```

Ausgabe über CutCom:



Ausgabe über das Terminal:

Unter /home/bernd/Programme/Laz-ESP8266/cross/bin/xtensa-freertos/esp-rtos-3.4/tools findet man die Datei idf_monitor.py. Diese mit einem Texteditor öffnen.
In der ersten Zeile stand bei mir:
#!/usr/bin/env python
Das so ändern das die Python Version unter env verwendet wird:
#!/home/bernd/Programme/Laz-ESP8266/cross/bin/xtensa-freertos/esp-rtos-3.4/venv/bin/python3
(*Ich hab das gleich mit allen Dateien in dem Ordner gemacht ist aber glaube ich nicht nötig*)

Terminal öffnen und folgendes eingeben:

```
(source ~/.bash_profile
pyenv global 3.7.17)

cd /home/bernd/Programme/Laz-ESP8266/cross/bin/xtensa-freertos/esp-rtos-3.4/
tools
./idf_monitor.py --port /dev/ttyUSB0 --baud 74880  /home/bernd/Programme/Laz-
ESP8266/MeineProjekte/03_Writeln/project1.elf
```