esp

John Doe

May 28, 2022

Contents

1 TODO separate project agnositc stuff 2

2 TODO fix user wheel group problem 2

3 TODO ideas 2

3.1 try to let gopro (cam) interface with esp 2

4 file-gen 2

5 hardware 4

5.1 esp32 4

5.1.1 pinout 4

5.1.2 block-diagram 4

5.1.3 hacks 4

6 API references 4

6.1 micropython implementation of espressif 4

6.2 Microdot 4

7 $\textbf{api}_{\textbf{test.py}}$ 4

8 curl (test API) 5

8.0.1 Root / Doc 5

8.0.2 Digital 6

8.0.3 Analog 6

8.0.4 ds18b20 6

9 tools **6**
9.1 **TODO** document img flashin process 6
9.2 **TODO** cli$_{\text{socket.py}}$ 6
9.3 doc extractor (docer.py) 6
9.4 rshell (filesystem) 8
9.5 webrepl$_{\text{cli}}$ (hotswap code) 8
9.6 screen (connect to console/repl) 8
9.7 webrepl.html 9

10 specification **9**
10.1 **TODO** ULP API 9
10.2 I/Os 9
10.3 Control-Loops 9
10.4 UI features 9
10.5 API 9
10.6 sensors 10
10.6.1 pt100 10
10.6.2 OEM DS18B20 10

11 merge org files **10**

[NAME] is a web based dashboard running on a esp32, with it, you can control all I/O peripherals plus read some metadata from the esp

1 TODO separate project agnositc stuff

2 TODO fix user wheel group problem

3 TODO ideas

3.1 try to let gopro (cam) interface with esp

4 file-gen

```
from os import popen
"""this script outputs an emacs-org table (simple ascii table with '-' and '|'),
    which is programmatically aggregated.
    Since this is meant to be used in an org-file,
    there is a return statement in the global scope"""

src = "./src/"              # src directory from where files are grabbed
```

```
url = "192.168.43.31:/"     # ws domain where files are transferred [col 'esp']
files = popen(f"ls {src}").read().split("\n")[:-1]

webrepl_cli_bin = "/home/$USER/py/esp/webrepl/webrepl_cli.py" # [col 'esp']

pop = lambda c: popen(c).read()[:-1]  # because it sucks to always read, especially wit
tab = lambda l: "|" + "|".join(l) + "|"   # outputs the table rows -> |x|y|z|
emacs_link = lambda t,l,n: f"[[{t}:{l}][{n}]]"
f_link = lambda l,n: emacs_link("file", l, n)
s_link = lambda l,n: emacs_link("shell", l.split("&")[0] + " &", n)

rows = (
    {"head": "files",
     "body": lambda f: f_link(src + f, f),
     "tail": f"[[file:{src}][{src}]]"},
    {"head": "esp",
     "body": lambda f: s_link(f"{webrepl_cli_bin} {src}{f} {url}{f}", "->"),
     "tail": ""},
    {"head": "du",
     "body": lambda f: pop(f"du -h {src}{f}").split()[0],
     "tail": pop(f"du -S -h {src}").split()[0]},
    {"head": "wc",
     "body": lambda f: ", ".join(pop(f"cat {src}{f} |  wc ").split()),
     "tail": ", ".join(pop(f"cat {src}* |  wc ").split())})

def table_as_string():
    return "\n".join(
        (tab(c["head"] for c in rows),
        "|-",
        "\n".join(tab([c["body"](f) for c in rows]) for f in files),
        tab(c["tail"] for c in rows)))

if __name__ == "__main__":
    return table_as_string()  # for emacs-org src_blocks
    # print(table_as_string() # to not get en error otherwise

import uasyncio; from api import coro
```

5 hardware

5.1 esp32

5.1.1 pinout

left-side right-side discription

1. legacy pinouts pinout pinout_

5.1.2 block-diagram

block-diagram

5.1.3 hacks

make adc resolve 12bit `adc.atten(ADC.ATTN_11DB) #3.3V full range of voltage`

6 API references

6.1 micropython implementation of espressif

uasyncio

6.2 Microdot

Tut of Microdot

7 $\text{api}_{\text{test.py}}$

```
#!/usr/bin/env python3

"""this script for the [NAME] api"""

import requests
from time import perf_counter

ROOT_URL = "http://192.168.43.31:5000/"

get_gen = lambda calls: (requests.get(ROOT_URL + call["route"], params=call["payload"])
                         for call in calls)
```

```
def benchmark(gen, count):
    t1 = perf_counter()
    return {"results": [[requests.get(ROOT_URL + call["route"], params=call["payload"])
                                .json()
                        for call in api_calls] for i in range(count)],
            "time": (t := perf_counter() - t1),
            "count": count,
            "average_ms": t / count,
            "request/s": 1 / ( t / count )}

def pop_keys(keys, *dicts):
    """removes all keys in dicts and returns a list of the resulting dicts"""
    res_dict = []
    for b in dicts:
        [b.pop(k) for k in keys]
        res_dict.append(b)
    return res_dict

api_calls = ({"route": "api/gpio/set",
             "payload": {"pin":26}}, )

# gpio_bench = pop_keys((), benchmark(get_gen(api_calls), 1))
gpio_bench = benchmark(get_gen(api_calls), 1)

for report in gpio_bench:
    for k in report:
        print(k, " -> ", report[k])
```

8 curl (test API)

api spec

8.0.1 Root / Doc

/ <- /gpio <-

8.0.2 Digital

-> /gpio/get 2 -> `curl-G-d"pin=33"192.168.43.31:5000/api/gpio/get&` -> /gpio/get type error str instead of int -> `curl-G-d"pin=26"192.168.43.31:5000/api/gpio/set&` -> /gpio/set 2 toggle -> /gpio/set 2 OFF -> /gpio/set 2 ON -> /gpio/set 2 type error str instead of int

8.0.3 Analog

/gpio/read 33 <- /gpio/read 33 type error str instead of int <-

8.0.4 ds18b20

/gpio/ds18 4 <-

9 tools

9.1 TODO document img flashin process

9.2 TODO cli$_{\text{socket.py}}$

```
import websockets
#!/usr/bin/env python

import asyncio
import websockets

uri = "ws://192.168.43.31:8266"

async def hello():
    msg = ""
    async with websockets.connect(uri) as websocket:
        await websocket.send(msg)
        msg = input(">>>" + await websocket.recv())

asyncio.run(hello())
```

9.3 doc extractor (docer.py)

```
#!/usr/bin/env python
"""this python script extracts signature and docstring, of given functions.
Recogniseing decoratores is yet not implemented.
```

```
The data is then outputed to a json or org file,
later is especially usefull, if this script is
implemented in a org file. In that case, make
sure that the src_block has the ':results raw' option added
"""

from os import popen
from json import dump
from datetime import datetime as dt

SRC_PATH = "./src/"
OUT_FILE = "./docs.json"

files = [SRC_PATH + f for f in popen("ls ./src").read().split("\n") if ".py" in f]

def extract_docs(files):
    """outputs a dict with given filenames as keys,
    whose values are dicts with function signature
    and docstring key value pairs"""
    docs = {}
    for file in files:
        functions = {}
        for f in "".join(open(file, "r").readlines()).split("def "):
            try:
                doc = f.split('"""')[1]
            except IndexError:
                doc = "N/A"
            functions.update({f.split("\n")[0]: doc})
        docs.update({file: functions})
    return docs

def dict_to_org(d, indent=1):
    """recursivly transforms a dict to an org-file.
    The heading level is controled by the indent parameter
    e.g.
    >>> dict_to_org({"foo":"bar","bar"{"oof":"baz"}}, indent=2)"""
    # ** foo
    #    bar
    # ** bar
    # *** off
    #     baz
    return "\n".join([
        "\n".join((
            "*"*(indent) + " " + str(k),
            dict_to_org(d[k], indent+1)
            if isinstance(d[k], dict)
            else str(d[k])))
        for k in d])

return "** -> docstrings (" \
        + dt.now().strftime("%Y-%m-%d - %H:%M:%S") \
        + ")\n" \
        + dict_to_org(extract_docs(files), indent=3)

# dump(docs, open(OUT_FILE, "w"), indent=2)
```

9.4 rshell (filesystem)

rshell provides a shell which mounts the filesystem of the microcontroller (eps32) under the path /pyboard `rshell -p /dev/ttyUSB0 -b 115200`

rshell also supports scripts (-f option) edit and tangel this rshell script block,

```
cp /home/nls/py/esp/src/boot.py /pyboard
```

and execute it `sudorshell-p/dev/ttyUSB0-b115200-f~/py/esp/dist.rshell&` <-

9.5 webrepl$_{cli}$ (hotswap code)

transfer a file with webrepl (don't forget to save the file beforhand) -> gpio -> main

9.6 screen (connect to console/repl)

C-c to abort the programm and access `screen /dev/ttyUSB0 115200`

9.7 webrepl.html

10 specification

10.1 TODO ULP API

10.2 I/Os

type	location	porpuse	pin
INPUTS			
tcp/ip ui	portable		
dht22	lung		
dht22	bloom		
dht22			
humid	soil / bloom		
co2	bloom		
pt100	outdoor		
OUTPUTS			
light	bloom		
light	veg		
circ	bloom		
circ	vig		
exhaust			
mister	lung		
dosage-punp	feed		

10.3 Control-Loops

actor	sensor
light	time
mister	humid, gro
exhaust	temperature

10.4 UI features

- E-Stop (Mute)
- abstract mapping

10.5 API

test the api with curl read <- docs.json generate docs -> docs.json

```
return "\n".join([l for l in open("./src/api.py", "r").readlines() if l[0] == "@"])
```

10.6 sensors

10.6.1 pt100

desired resistor for pt100: ~2.57kOhm

10.6.2 OEM DS18B20

tutorial for esp32

11 merge org files

```
FILE=~/base.org
echo "* -> "$FILE
cat $FILE | sed 's,[*] ,** ,'
```