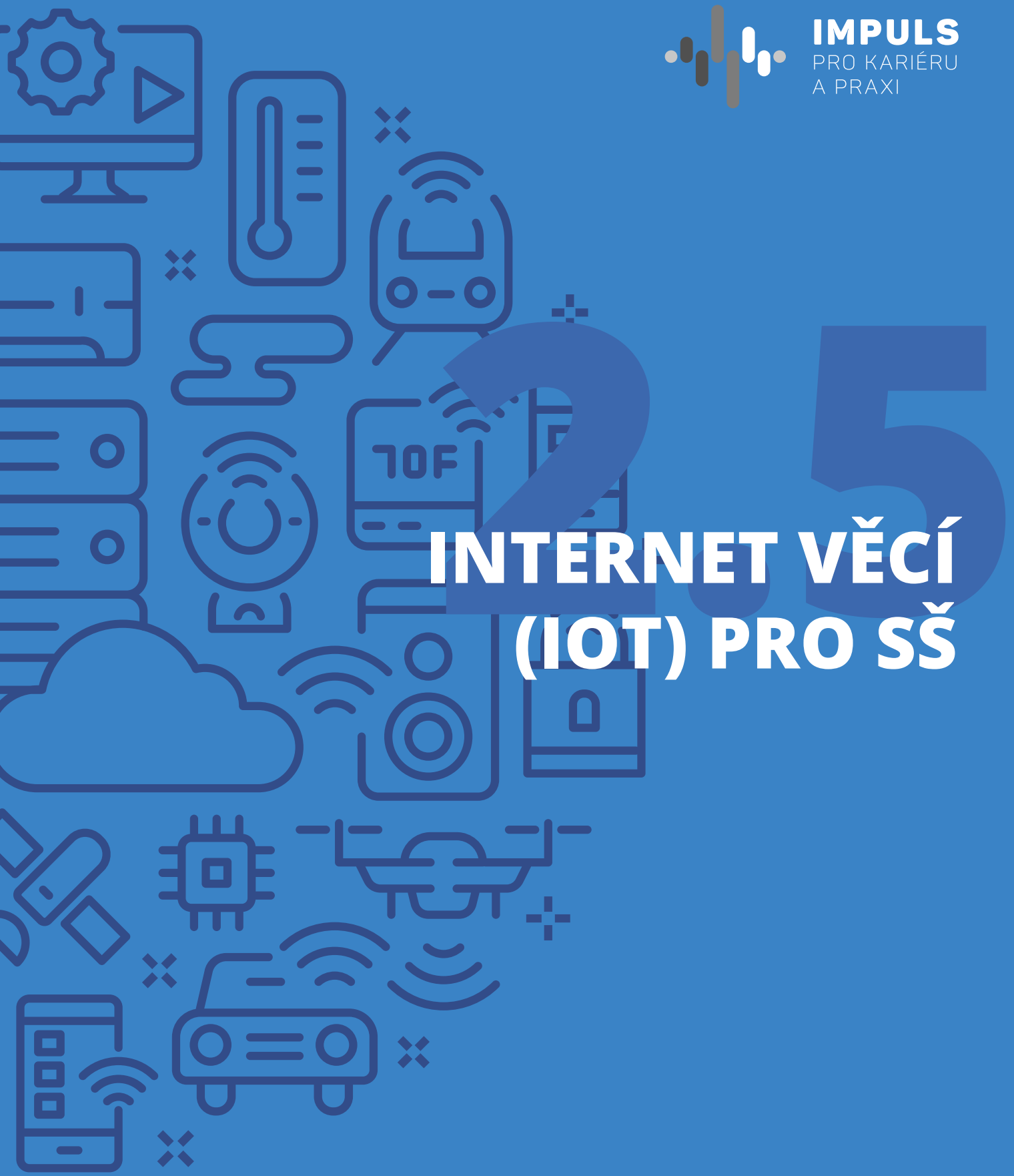
IMPULS
PRO KARIÉRU
A PRAXI




# INTERNET VĚCÍ (IOT) PRO SŠ

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Jhk.cz

JIHOČESKÁ HOSPODÁŘSKÁ KOMORA

Jihočeský kraj

Pracovní postup „Kybernetická bezpečnost pro SŠ“ je součástí publikace „Pracovní postupy pro workshopy digitalizace ve školách.“, která vznikla v rámci aktivity Asistenčního centra Impuls pro kariéru a praxi při Jihočeské hospodářské komoře díky realizaci projektu „Implementace Krajského akčního plánu Jihočeského kraje III“, který je spolufinancován Evropskou unií. Registrační číslo projektu CZ.02. 3. 68/0.0/0.0/19_078/0018246

Elektronická verze publikace je k dispozici na **www.impulsprokarieru.cz**

Autor: Ing. Jiří Jelínek, CsC, Přírodovědecká fakulta, Jihočeská univerzita v Českých Budějovicích.
Editor: doc. RNDr. Ing. Jana Kalová, Ph.D.
Publikaci připravila Přírodovědecká fakulta Jihočeské univerzity

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta
Centrum pro přírodovědné vzdělávání

Grafický design: Čestmír Sukdol – www.brandi.cz

Vydala: Jihočeská hospodářská komora

2021

# Obsah

*Cílem tohoto kurzu je představit studentům problematiku internetu věcí (IoT). Tento pojem je relativně nový, ale do budoucna má značný potenciál. Proto je potřeba s ním studenty seznámit. V rámci kurzu se seznámíme s jedním zařízením používaným v oblasti IoT, konkrétně se jedná o vývojovou desku s procesorem ESP32. Pro experimenty s ní použijeme programovací jazyk Micropython a ukážeme si příklady konkrétního použití ESP32 pro Internet věcí.*

# Základní instrukce

*Tento kurz je doporučen i pro studenty středních škol, kteří již mají nějaké základní znalosti v oblasti algoritmizace a programování. Ideální je pak znalost základu jazyka Python, nicméně i znalost základů jakéhokoliv jiného programovacího jazyka je postačující (jazyk Python bude využíván pouze na základní úrovni, která se příliš neliší od jiných programovacích jazyků)*

*Časová dotace tohoto kurzu významně závisí na dosavadních znalostech studentů v oblasti algoritmizace a programování. Je na konkrétním pedagogovi, aby na základě znalosti svých studentů rozhodl o délce kurzu.*

Celé téma lze rozdělit do několika částí, přičemž některé se částečně prolínají a navazují na sebe. Celkově je postupováno od teoretického úvodu přes zprovoznění ESP 32 až k řešení praktických úloh.

1. Seznámení s problematikou internetu věcí obecně, definice základních pojmů a popis hlavních principů a účelu celého konceptu. Toto téma je volitelné a jedná se spíše o motivační úvod do dané problematiky. Iz tohoto důvodu není v rámci tohoto materiálu zpracováno ve formě pracovního listu.
   a. Pokud toto téma nebylo dosud vyučováno, doporučuji mu věnovat jednu vyučovací hodinu.
   b. V případě, že studenti již danou oblast a její principy znají, je možné toto téma vynechat.

2. Seznámení s ESP32, jeho možnostmi a programováním. Příprava a zprovoznění zařízení a jeho programování v jazyce Micropython. Komunikace pomocí sériové linky.
   a. Vzhledem k tomu, že tato oblast bude patrně pro všechny studenty zcela nová, je nutné počítat minimálně s dvěma vyučovacími hodinami pro zvládnutí přípravy zařízení k činnosti a další hodinou pro ukázky práce s Micropythonem.

3. Komunikační možnosti ESP32 – využití vestavěné Wi-Fi v režimu stanice. Vytvoření jednoduchého webového serveru běžícího na ESP32.
   a. Toto téma je poněkud rozsáhlejší a zcela zásadní pro další oblasti. Je možné pouze odhadnout potřebný čas pro jeho zvládnutí (podle předchozích znalostí studentů). Minimální rozsah tématu tak lze odhadnout na dvě vyučovací hodiny.

4. Práce se vstupy – možnosti snímání hodnot externím čidlem a jejich publikace na webu.
   a. Problematice doporučuji věnovat minimálně jednu až dvě vyučovací hodiny.

5. Práce s výstupy – ovládání RGB LED modulu, ukázka spolupráce vstupního a výstupního modulu, vytvoření funkčního IoT řešení s plným využitím možností ESP32.
   a. Problematice doporučuji věnovat minimálně jednu až dvě vyučovací hodiny.

V rámci bodů 4 a 5 bude zahrnuto i spojení znalostí z předchozích oblastí do finálního celku reprezentujícího funkční aplikaci měřící teplotu v místnosti a reagující na ní rozsvícením kontrolní diody a informující uživatele prostřednictvím webu.

Na základě výše uvedeného lze odhadovat, že minimální doba pro realizaci celého kurzu činí cca 8–10 vyučovacích hodin (pro studenty zběhlejší v oblasti algoritmizace se programování a alespoň minimální znalostí jazyka Python). Tento údaj je však nutno považovat pouze za orientační.

Pro první část je vhodnou formou výuky výklad. Ten by měl být zastoupen i v ostatních částech, převažovat by u nich však postupně měla samostatná práce studentů. Výklad by tak měl být v posledních částech kurzu zaměřen především na zadání úlohy a případně vzorovou ukázku řešení.

Vzhledem k možným rozdílům ve znalostech jednotlivých studentů lze také doporučit práci ve dvoučlenných týmech, a to především od části 2 dále.

U částí 4 a 5 je pak v druhé polovině již možné přistoupit k projektové výuce, kdy by studentské týmy měly (například i formou soutěže) vytvořit vlastní řešení IoT, které by následně prezentovaly.

*Celý předkládaný kurz je rovněž nutno vybavit nebo podpořit příslušným technickým vybavením. V tomto případě se jedná především o zařízení ESP32 v počtu minimálně jeden kus na dva studenty. Zde je však nutné počítat i s rozdílnými úrovněmi znalostí studentů a je možné, že někteří studenti budou schopni kurz absolvovat samostatně a budou toto řešení preferovat. V takovém případě by měli mít k dispozici samostatnou sadu ESP32 a je tedy doporučené mít tyto desky k dispozici pro cca 75 % studentů.*

*Ke každému ESP32 je nutno počítat s USB kabelem (shodný s kabely pro mobilní telefony s koncovkou microUSB). Pro ukázku práce se vstupy a výstupy je dále nutné mít příslušná vstupní i výstupní zařízení. Doporučeno je pracovat s RGB LED diodou jako výstupem a čidly pro teplotu, tlak či osvětlení na vstupu (po jednom kusu pro každé ESP). Pro práci s komunikačními možnostmi ESP32 je pak velmi vhodné a preferované používat společnou Wi-Fi síť odemčenou pro přístup z neautorizovaných zařízení (bez kontroly MAC adresy, přístupové heslo je možné použít). Pro bližší kontakt studentů s danou technologií je vhodné rovněž počítat s využitím jejich mobilních telefonů. Pro práci se vstupy a výstupy je rovněž nutné mít k dispozici i nepájivé kontaktní pole (tzv. breadboard) s alespoň minimální sadou propojovacích kabelů.*

*V neposlední řadě je nutné upozornit na to, že zařízení se ve fázi programování a testování připojuje ke stolnímu počítači či notebooku ideálně s operačním systémem Linux nebo Windows. Po naprogramování pak může pracovat i samostatně. Pro ukázku samostatného provozu ESP32 bez připojení k počítači je vhodné využít napájení prostřednictvím powerbanky pro mobilní telefony.*

*V oblasti softwaru je pak situace velmi jednoduchá. Využíván bude zejména specializovaný modul v Pythonu od firmy Espressif (výrobce ESP32) umožňující komunikaci ESP32 a rovněž software Micropython (**https://micropython.org/**), který je volně dostupný. Jako operační systém pro výuku je možné využít MS Windows, ev. Linux (jednodušší instalace podpory). Obecná podpora pak spočívá v instalaci jazyka Python v.3x a vývojového prostředí Thonny (**https://thonny.org/**).*

*Pro instalaci Micropythonu lze využít např. český návod „ESP32 Návod: podpora Micropythonu“ (**https://chiptron.cz/articles.php?article_id=204**), ev. postupu na webu **https://blog.vyoralek.cz/iot/esp8266-a-ESP32-zaloha-a-nahrani-noveho-firmware-pomoci-esptool/**.*





# Teoretická část k dané problematice

Oblast internetu věcí je v posledních několika letech stále aktuálnější. O co se však jedná? Jak již název naznačuje, základem je zde využití internetu, a tedy možnost komunikace mezi jednotlivými zařízeními. Internet je pro takovou komunikaci ideální platformou a umožňuje využít na něm existující nástroje a protokoly. Mezi nimi vyniká protokol TCP a na vyšší úrovni pak především protokol HTTP používaný při provozu webu.

*Základním smyslem internetu věcí je zajistit schopnost jednotlivých koncových zařízení sbírat informace o svém provozu a svém okolí, tyto informace dále komunikovat a na základě jejich vyhodnocení a komunikace s ostatními zařízeními pak řídit jejich činnost.*

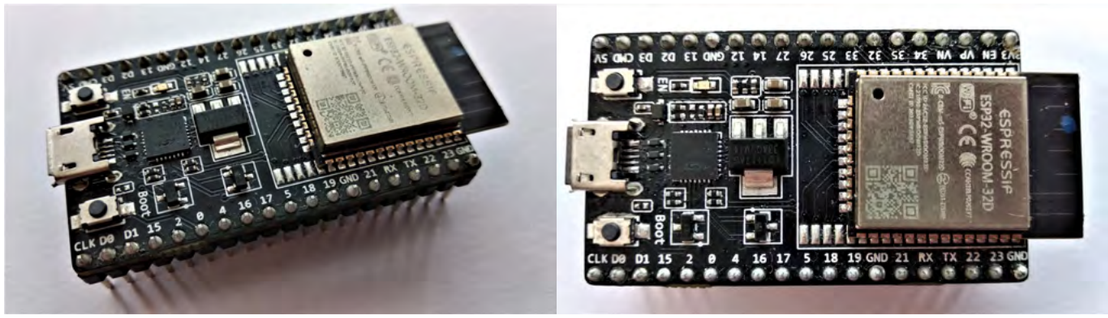
V rámci IoT je klíčová volba koncového zařízení, které by mělo umožnit výše zmíněnou komunikaci, a to jak směrem k přímo připojeným komponentám (čidla, akční prvky), tak do Internetu. Typickými představiteli jsou zařízení typu SoC (System on Chip). Mezi ně pak patří systémy typu Arduino vynikající zejména širokou škálou možných periferií, a to jak vstupních (čidla všeho druhu), taky výstupních (ovladače motoru, relé atd.). Zajištění vnější konektivity a komunikace u těchto zařízení je však poněkud problematičtější, protože žádnou jednotnou komunikační platformou a přístupem k internetu samy o sobě nedisponují. Podstatně vyšší kategorii tvoří v tomto směru zařízení typu Raspberry Pi, které již běží na standardním operačním systému typu Linux. I u něj je možné využít periferie používané například s Arduinem.

Společným rysem těchto zařízení je pak jejich využití ve formě vloženého nebo vestavěného prvku (angl. embedded system) v jiném koncovém systému, ve kterém dosud mikroprocesorová technika byla využívána málo nebo vůbec. Typicky může jít například o domácí spotřebiče.

Ideálním řešením by však bylo zařízení někde na půl cesty mezi výše zmíněnými, tedy s velmi dobrou konektivitou k jednotlivým čidlům a výkonným prvkům, ale i s možností internetové konektivity. Takovým zařízením je pak ESP32.

*SoC systém ESP32 disponuje bezdrátovým připojením Wi-Fi v pásmu 2,4 GHz, Bluetooth připojením (které je však v jazyce Micropython obtížně využitelné) a připojením pomocí USB rozhraní (takzvanou sériovou linkou). Procesor ESP32 je dvoujádrový. Kromě toho má zařízení řadu digitálních a analogových vstupů a rovněž digitálních výstupů s případnou regulací šířky pulzu (Pulse Width Modulation – PWM) používanou při řízení motorů či nastavení jasu LED osvětlení.*

ESP32 lze programovat různými způsoby. V našem kurzu využijeme programovací jazyk Micropython a jeho konkrétní implementaci na ESP32.




Obrázek 1: Vývojová deska ESP32

# Příklady z praxe

V závěru kurzu se zaměříme na ukázku praktického využití principu IoT. Jak již však bylo řečeno, nejprve musíme vytvořit komunikační prostředí mezi ESP32 a Internetem, v našem případě webovým prohlížečem. V takto vytvořené platformě je pak již možné ukazovat využití vstupu a výstupu ESP32.

My si vyzkoušíme sběr informací o teplotě a tlaku v dané místnosti pomocí čidla a předávání těchto informací uživateli připojenému k internetu. Druhým příkladem bude ukázka využití získané informace pro ovládání výstupního zařízení, v našem případě RGB LED diody. Ukázky jsou ryze praktické a ukazují, jakým způsobem je možno využít obdobnou technologii například pro řízení domácnosti a sběr informací nutných k jejímu řízení.

Absolvováním tohoto kurzu studenti získají nebo si prohloubí základní programátorské návyky. Dále se dozví informace ze světa hardware a integrace software a hardware. Zde záleží pouze na učiteli, který musí odhadnout úroveň svých žáků a jak hluboko si může dovolit ponořit se do dané problematiky.

Příklady zároveň studentům ukáží, že i na úrovni jejich znalostí lze s touto technologií experimentovat a prakticky využít ve vlastních projektech vytvářených jak pro domácí užití, tak v rámci školní výuky.

# Metodická a didaktická část

Jak bylo vysvětleno v první kapitole, kurz je rozdělen do několika částí. Nyní se pozastavíme u každé z nich.

## Seznámení s ESP32

V této části rozdá učitel studentům zařízení ESP32 a USB kabel pro jeho připojení k počítači. Žáci poté své zařízení připojí ke svému PC nebo notebooku a provedou instalaci nutných komponent, aby bylo možné se zařízením ESP32 pracovat v jazyce Micropython (popsáno v pracovním listu). Funkčnost řešení studenti následně ověří otestováním krátkého programu v jazyce Python provedeného na zařízení ESP32. Rolí učitele je v tomto případě zejména vysvětlování a kontrola činnosti jednotlivých studentů a vysvětlení jednotlivých kroků, které jsou prováděny.

## Internetové připojení ESP32

Tento bod navazuje na předchozí a studenti v něm implementují krátký program, který umožní připojení ESP32 do Wi-Fi sítě školy. Dalším krokem je pak vytvoření velmi jednoduchého webového serveru, na který bude možné se připojit z libovolného zařízení ve školní síti (detaily opět popsány v pracovním listu). Role učitele spočívá v tomto případě opět zejména ve vysvětlení jednotlivých programových kroků, které je nutné provést pro zajištění výše popsané funkcionality. Tento bod předpokládá alespoň základní znalosti studentů v oblasti webových technologií (základní znalost html, která však může být nahrazena širším výkladem učitele o této problematice).





## ESP32 jako senzor

Cílem tohoto bodu je využití výstupů bodů předchozích a rozšíření funkcionality jednoduchého webového serveru tak, aby jeho prostřednictvím bylo možné získávat údaje z teplotního čidla, které bude k ESP32 externě připojeno (detaily opět v pracovním listu). Tyto údaje pak budou využity dvěma různými cestami – pro řízení výstupu přímo na ESP32 (v našem případě RGB LED) a také pro informování uživatele přes webový server.

Náplní bodu tak bude jednak vytvoření patřičného hardwarového zapojení a rovněž rozšíření funkcionality pro indikaci teploty a rozšíření webového serveru o prezentaci teploty. Role učitele bude zaměřena na kontrolu hardwarového zapojení a vysvětlení základních programových kroků, které bylo potřeba provést pro dosažení stanovených cílů. I tento bod vyžaduje alespoň základní znalosti jazyka html, které bude v případě nutnosti potřeba nahradit širším výkladem učitele v této problematice.

# Doporučené pomůcky

Doporučenou pomůckou pro celý kurz je níže popsaná sada komponent, která tvoří základ pro všechny úkoly. Těchto sad musí být k dispozici odpovídající množství (jedna sada pro 1 – 2 studenty). Sadu tvoří:

- Vývojová deska ESP32
- Micro USB kabel
- Nepájivé kontaktní pole
- Modul RGB LED diody
- Tlakové a teplotní čídlo BMP180
- Propojovací kablíky (cca 10)

Zbytek vybavení je již pouze softwarový a vyžaduje mít k dispozici pro každé ESP32 1 PC nebo notebook s přístupem k internetu a v daném prostoru celkově dostupnou otevřenou Wi-Fi, aby bylo možné na ni ESP32 napojit.

# Pracovní list

Přílohou tohoto materiálu jsou pracovní listy. Tyto pracovní listy jsou k dispozici v editovatelné elektronické formě, aby si je každý učitel mohl upravit, např. dle toho, co má již s žáky probráno.

Pracovní listy jsou čtyři, dva jsou zaměřeny na vytvoření předpokladů pro praktické užití ESP32, další dva pak na řešení praktických úkolů. V pracovních listech je předpokládána dostupnost všech komponent popsaných výše.

# Pracovní list 1
# Seznámení s ESP32

## Co budeme potřebovat

- Počítač s nainstalovaným OS MS Windows s přístupem na internet, plnými přístupovými právy pro instalaci SW a dostatkem volné kapacity na pevném disku
- ESP32
- Propojovací USB kabel s microUSB koncovkou

## A jdeme na to

Nejprve se seznamte s vývojovou deskou zařízení ESP32. Deska obsahuje samotný čip ESP32 a související doplňkové obvody a ovládací prvky. Těmi jsou dvě tlačítka po obou stranách microUSB konektoru. Pokud se na ESP32 díváte tak, že vidíte USB konektor, pak tlačítko vlevo je tlačítko resetu a tlačítko vpravo umožňuje přejít do bootovacího režimu, kdy je možné nahrávat software. Za čipem ESP32 také najdete integrovanou (na plošném spoji) anténu pro Wi-Fi a Bluetooth a po obou stranách dvě řady tzv. pinů umožňujících připojení dalších zařízení, případně další komunikaci.

Prvním krokem, který musíme provést, je instalace programovacího jazyka Python 3 a souvisejícího nástroje pro správu balíčků a rozšíření pip. Instalaci Pythonu 3 provedete stažením aktuální verze ze stránky **www.python.org** a následnou samotnou instalací, která se nijak neliší od jiných programů instalovaných na platformě Windows. Při instalaci zadejte, že cesta k Pythonu se má zařadit do proměnné PATH Windows, a že pro program Python bude vytvořena ikona na pracovní ploše.

Další kroky již budou probíhat v samotném Pythonu, kde pomocí programu pip3 nainstalujeme rozšíření pro ESP32.

```
pip3 install esptool
```

Dalším krokem je nutné již připojit ESP32 pomocí kabelu USB kabelu k počítači. Ten by měl stáhnout potřebné ovladače a začít komunikovat s ESP32 pomocí sériového portu. Právě označení sériového rozhraní je zásadní pro další činnost, neboť právě na tomto portu musíme s ESP32 komunikovat. Čísla nebo označení aktuálních sériových portů je možné zjistit pomocí příkazu „mode“ zadaného na příkazový řádek Windows (po spuštění příkazu CMD).

```
mode
```

Port ESP32 bude označen jako COM s navazujícím číslem (např. COM3). V dnešní době bývá tento sériový port jediný, který bývá připojen.

Dalším krokem je ověření komunikace s ESP32 pomocí skriptu esptool.py, který má značné možnosti a používá se pro všechnu komunikaci s ESP32. Zásadním parametrem uvedeného příkazu je označení portu, které jsme zjistili v předchozím kroku. Výstup zadaného příkazu nám pak identifikuje konkrétní zařízení ESP32 a uvede jeho základní charakteristiku. Pokud se tak nestane, je vhodné v průběhu komunikace stisknout na ESP32 tlačítko





bootu (toto řešení problémů s inicializací komunikace je použitelné i v následujících bodech naší úlohy).

```
esptool.py --port COMx flash_id
```

Nyní tedy máme ESP32 připojeno k počítači a jsme schopni s ním komunikovat. Dalším krokem je instalace jazyka Micropython na ESP32, přičemž nejprve je nutné provést výmaz paměti ESP32. To provedeme příkazem:

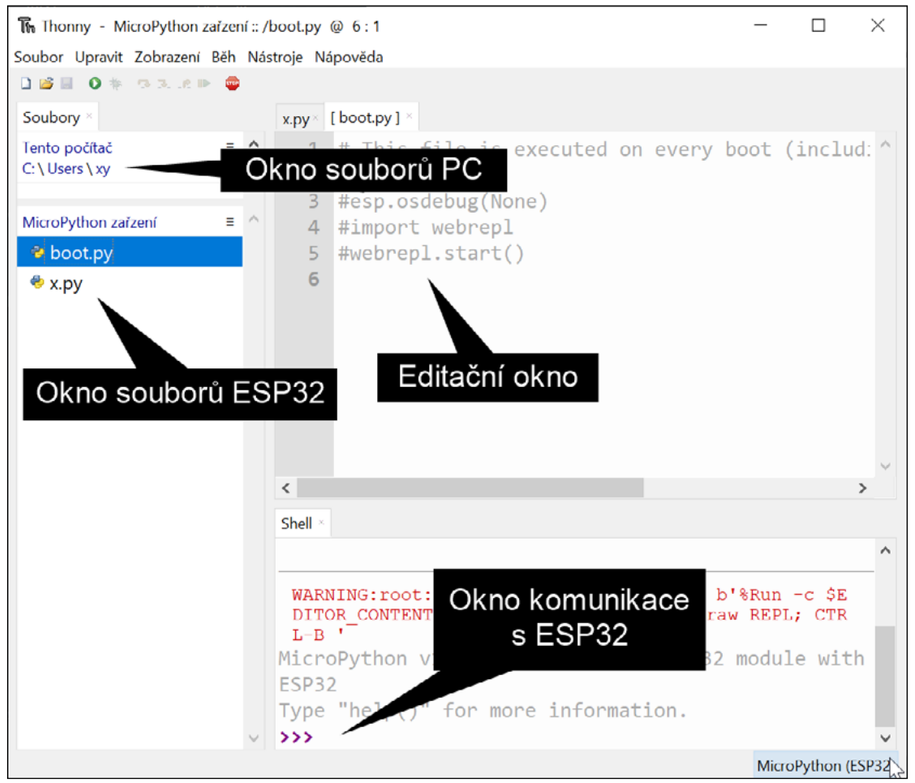
```
esptool.py --chip esp32 --port COMx erase_flash
```

Nyní k instalaci Micropythonu. Její první krok spočívá v získání vhodné verze tohoto jazyka ze stránek **www.micropython.org**. Při výběru aktuální verze nezapomeňte zvolit správný typ čipu, tedy ESP32. Soubor s koncovkou „bin“ uložte do počítače. (V době přípravy tohoto materiálu byla aktuální verze https://micropython.org/resources/firmware/ESP32-20210418-v1.15.bin.)

Nyní již převedeme získanou verzi Micropythonu přímo do paměti ESP32. To provedeme přiloženým příkazem, kde uvedete jako poslední parametr aktuální název a umístění vaší verze Micropythonu, kterou jste v předchozím kroku stáhli. Proces instalace jazyka chvíli trvá, buďte proto trpěliví.

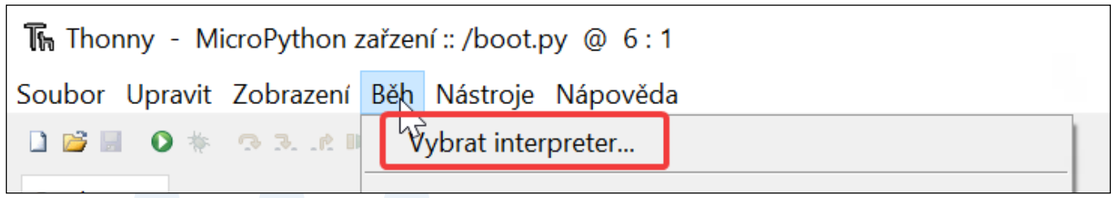
```
esptool.py --chip esp32 --port /dev/ttyUSB0 --baud 460800 write_flash -z 0x1000 /
cesta/xxxx.bin
```

Nyní tedy máme ESP32 kompletně připraveno pro použití s jazykem Micropython. Posledním krokem je tedy ještě instalace vývojového prostředí na stolní počítač. Tímto prostředím bude jednoduchý programový editor Thonny, který najdete na adrese www.thonny.org a jeho instalaci provedete stejným způsobem jako instalaci jazyka Python.



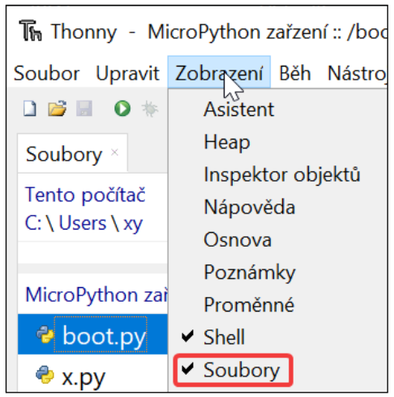

Obrázek 2: Editor Thonny (české lokalizace je experimentální)

Prostředí Thonny je nutné upravit pro použití ESP32, kde v nabídce Běh (Run) zvolíte výběr interpreteru – z dostupných možností zvolíte „Micropython (ESP32)“.



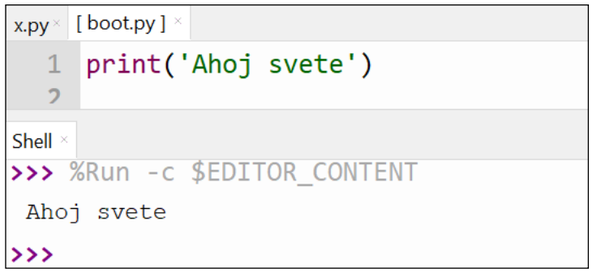

Obrázek 3: Výběr správného interpreteru (následuje volba „Micropython (ESP32)“

Vhodné je také z nabídky zobrazení zaškrtnout volbu soubory, kde v levé části uvidíte dvě sekce z nichž jedna bude váš stolní počítač a druhá sekce bude zařízení ESP32. V této sekci bude jediný soubor s názvem boot.py, který ESP32 spouští při každém startu.




Obrázek 4: Výběr zobrazení panelu souborů

Nyní již můžeme do hlavního editoru zapsat jednoduchý příkaz „print ("Hello World")“, který následně necháme vykonat pomocí spouštěcí šipky na mikrokontroleru (ESP32). Ve spodní části okna by se měl vypsat příslušný text.




Obrázek 5: Ověření funkčnosti komunikace s ESP32

## Co jste se naučili

*V této části jsme se naučili připojit zařízení ESP32 k počítači a celou sestavu připravit takovým způsobem, aby bylo možné ESP32 programovat prostřednictvím jazyka Micropython. V dalších krocích pak budeme s takto vytvořenou sestavou dále pracovat a zaměříme se již na funkcionalitu programovanou pro ESP32.*





# Pracovní list 2
# ESP32 na Internetu

## Co budeme potřebovat

- Počítač s nainstalovanými programy z Pracovního listu 1 a s přístupem na internet
- Funkční Wi-Fi síť umožňující připojení dalších zařízení (bez filtrace MAC adres, heslo není překážkou, funkční server DHCP pro přidělení IP adresy)
- ESP32 a propojovací USB kabel
- Libovolná powerbanka pro testování samostatného provozu ESP32

## A jdeme na to

Připojte k počítači zařízení ESP32. Otevřete si vývojové prostředí Thonny a jednoduchým příkazem ověřte, že jste schopni s ESP32 komunikovat (viz Pracovní list 1). V této části zajistíme, aby ESP32 komunikovalo s Internetem, a to na úrovni webových stránek.

V prvním kroku si ověříme, jaké Wi-Fi sítě jsou v daném prostoru k dispozici. Příslušný programový kód máte uvedený. Pokud chcete, aby byl spouštěn ihned po zapnutí ESP32, pište jej do souboru s názvem „boot.py“.

Jen připomeňme, že ESP32 pracuje pouze ve frekvenčním pásmu 2,4 GHz. Mezi zobrazenými sítěmi by měla být rovněž ta, kterou ve škole používáte. Programu, který zde máte zobrazený (Obrázek 6), se říká Wi-Fi scanner a umožňuje zjistit, jaké sítě jsou v daném okolí k dispozici.

```
import network


station = network.WLAN(network.STA_IF)

station.active(True)

wifis = station.scan()

print(wifis)
```

```
>>> %Run -c $EDITOR_CONTENT
[(b'[redacted]', b'\x00rcFO\xe6', 6, -43, 4, False), (b'[redacted]', b'8\xd5G\x07\xc6\x80', 3, -56, 3, False), (b'[redacted]', b'\x00rcFR/', 8, -64, 4, False), (b'[redacted]', b'\xcc\xb2Ua\xa7<', 7, -71, 4, False), (b'[redacted]', b"\x00rc\xfa':", 12, -72, 3, False), (b'[redacted]', b'\xe4\xbe\xed\x0e\x0e\x83', 4, -74, 4, False), (b'[redacted]', b'\xd8G2\xc5\xc2\xfa', 3, -77, 3, False), (b'[redacted]', b'd\xd1T\xf6\x88\xf0', 9, -78, 4, False), (b'[redacted]', b'\xc8`\x00\x90\xebT', 1, -84, 3, False), (b'[redacted]', b',V\xdc\xcfW\xe8', 12, -89, 4, False), (b'[redacted]', b"\x90\\D'\x87\r", 6, -90, 4, False), (b'[redacted]', b'\xc8\xd3\xa34\xeb\n', 7, -90, 4, False), (b'[redacted]', b'\x04\x8d8FJ\xf3', 3, -91, 4, False), (b'[redacted]', b'8C}e{W', 11, -93, 4, False), (b'[redacted]', b'\xa8^EH\x91\xa4', 1, -94, 3, False)]
```

Obrázek 6: Výpis Wi-Fi sítí v okolí – výstup (jména sítí (SSID) jsou rozmazána)

Druhým krokem je připojení k Wi-Fi, kterou jste si zvolili pomocí jejího názvu (tzv. SSID). Síť může být otevřená anebo vyžadovat ověření vaší totožnosti pomocí hesla. Tato možnost je běžnější a pokud tomu tak je, obraťte se na svého vyučujícího, aby vám heslo sdělil. Pokud do Thonny přepíšete kód uvedený níže a spustíte jej, připojíte se k vámi uvedené síti Wi-Fi. Toto bude potvrzeno níže uvedeným výpisem od ESP32, který vám zobrazí aktuální nastavení vašeho Wi-Fi. Z uvedených čtyř IP adres konfigurace sítě je zásadní zejména ta první, která udává, jaká IP adresa byla vašemu ESP32 přidělena. V případě, že výpis těchto adres neuvidíte, zkuste restart ESP32 tlačítkem RESET nebo odpojte ESP32 od počítače a následně opět připojte a celý proces opakujte.

```
ssid = 'sitxx'
passwd = 'xxxx'
import network
import time


sta_if = network.WLAN(network.STA_IF)
if not sta_if.isconnected():
    print('connecting to network...')
    sta_if.active(True)
    sta_if.connect(ssid, passwd)
    while not sta_if.isconnected():
        sleep(1)
print('network config:', sta_if.ifconfig())
```

Dalším krokem je zprovoznění jednoduchého webového serveru na ESP32. Využijeme předchozí kód pro připojení k Wi-Fi síti a doplníme jej dle Obrázku 8. Ověření funkcionality celého řešení provedete velmi snadno. Prostřednictvím webového prohlížeče na stolním počítači se připojte na IP adresu, kterou jste zjistili v předchozím bodu. Pokud vidíte odpověď vašeho webového serveru na ESP32, vše jste provedli správně.

```
try:
  import usocket as socket
except:
  import socket


html = """<html><head></head><body><h2>Vase ESP32 vas vita!</h2></body></html>"""
```





```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)
print('starting server')
while True:
    conn, addr = s.accept()
    print(Connection request from %s' % str(addr))
    response = html
    conn.send('HTTP/1.1 200 OK\n')
    conn.send('Content-Type: text/html\n')
    conn.send('Connection: close\n\n')
    conn.sendall(response)
    conn.close()
    print('Connection closed')
```

```
>>> %Run -c $EDITOR_CONTENT
 network config: ('192.160.43.194', '255.255.255.0', '192.160.43.1', '192.160.43.1')
 starting server
 Connection request from ('192.168.43.116', 49490)
 Connection closed
```

Obrázek 8: Jednoduchý web server – výstup

Uvedený program pracuje v nekonečné smyčce, činnost lze ukončit vypnutím ESP32.

Pokud jste kód vložili do souboru „boot.py“, je možné ESP32 odpojit od PC a připojit např. k powerbance a provozovat jej i takto samostatně.

## Co jste se naučili

*Při plnění tohoto vcelku náročného pracovního listu jste se naučili využít nastavené komunikace se zařízení ESP32, abyste jej mohli programovat a měnit jeho chování. Základním chováním je však komunikace ESP32 a my jsme si ukázali, jak tuto komunikaci zajistit prostřednictvím sítě Wi-Fi tak, aby se vaše ESP32 chovalo jako webový server a vy jste mohli v dalších krocích právě tento webový server obohacovat o další funkcionality.*

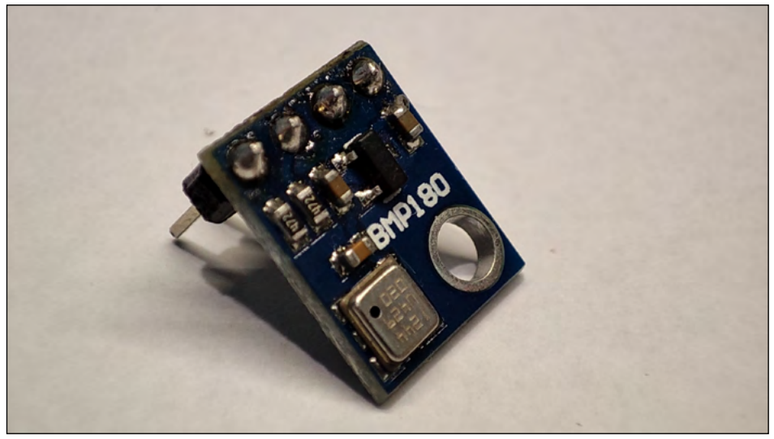
# Pracovní list 3
# ESP32 jako senzor

## Co budeme potřebovat

- Počítač s nainstalovanými programy z Pracovního listu 1 a 2 a s přístupem na internet
- Funkční Wi-Fi síť umožňující připojení dalších zařízení (bez filtrace MAC adres, heslo není překážkou, funkční server DHCP pro přidělení IP adresy)
- ESP32 a propojovací USB kabel
- Čidlo využívající čip BMP180

## A jdeme na to

Připojte k počítači zařízení ESP32. Otevřete si vývojové prostředí Thonny a jednoduchým příkazem ověřte, že jste schopni s ESP32 komunikovat (viz Pracovní list 1). V této části zajistíme, aby ESP32 snímalo informace z externího čidla teploty a tlaku a tyto informace publikovalo přes webový server.

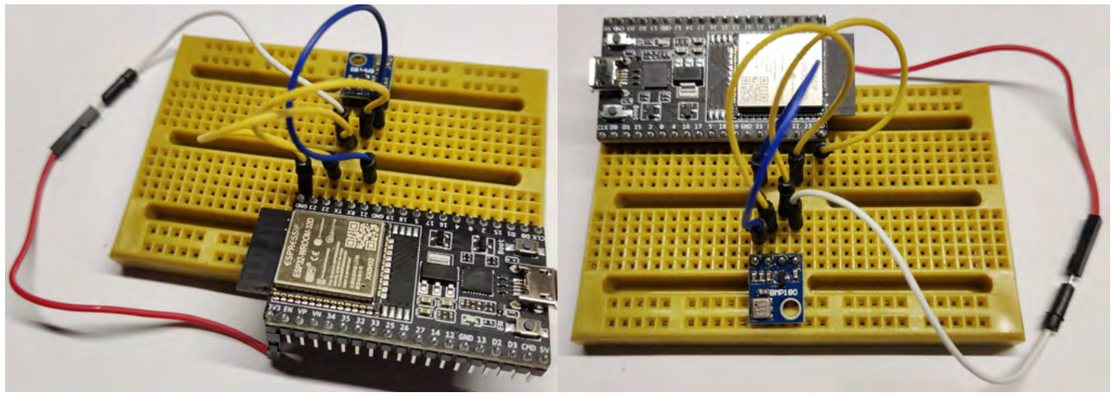



Obrázek 9: Modul čidla BMP180

V prvním kroku zprovozníme externí čidlo. K tomu budete potřebovat nepájivé kontaktní pole se zapojením dle obrázku. Čidlo BMP180 je digitální a s ESP32 komunikuje pomocí tzv. I2C sběrnice. Aby tato komunikace pracovala, je potřeba do ESP32 nahrát obslužnou knihovnu pro BMP180, kterou najdete na adrese **https://github.com/micropython-IMU/micropython-bmp180**. Soubor bmp180.py nahrajte pomocí Thonny do ESP32.







Obrázek 10: Zapojení ESP32 a čidla BMP180

Nyní již můžeme přistoupit k programování ESP32. Kód pro čtení hodnot z čidla následuje. Informace z externího čidla budou aktualizovány po 5 sekundách.

```
import time

from bmp180 import BMP180

from machine import SoftI2C, Pin


bus =  SoftI2C(scl=Pin(22), sda=Pin(21), freq=100000)

bmp180 = BMP180(bus)

bmp180.oversample_sett = 2

bmp180.baseline = 101325


while True:

    temp = bmp180.temperature

    press = bmp180.pressure

    altitude = bmp180.altitude

    print("Temperature: ", temp)

    print("Pressure: ", press)

    print("Altitude: ", altitude)

    time.sleep(5)
```

Pro zobrazování hodnot přes webový server je potřeba uvedený nekonečný cyklus zrušit a upravit kód webserveru. Ten nyní bude prohlížeč uživatele informovat o tom, že stránka se bude obnovovat každých 5 sekund a při každém obnovení se aktualizují hodnoty z čidla.

Celý kód upraveného webserveru včetně čtení z čidla následuje (kód předpokládá funkční připojení ESP32 k wifi, viz Pracovní list 2):

```
try:
  import usocket as socket
except:
  import socket

import time
from bmp180 import BMP180
from machine import SoftI2C, Pin

bus =  SoftI2C(scl=Pin(22), sda=Pin(21), freq=100000)
bmp180 = BMP180(bus)
bmp180.oversample_sett = 2
bmp180.baseline = 101325

def temp():
    temp = bmp180.temperature
    return str(temp)

def web_page():
    html = """
<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="refresh" content="2" />
</head>
<body>
    <h2>Vase ESP32 vas vita</h2>
    <p>Aktualni teplota je (C): """+temp()+"""</p>
</body>
</html>
"""
    return html

def server():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(('', 80))
    s.listen(5)

    while True:
        try:
            conn, addr = s.accept()
            conn.settimeout(3.0)
            print('Received request from %s' % str(addr))
            request = conn.recv(1024)
            conn.settimeout(None)
            request = str(request)
            print('GET Request Content = %s' % request)
            response = web_page()
            conn.send('HTTP/1.1 200 OK\n')
            conn.send('Content-Type: text/html\n')
            conn.send('Connection: close\n\n')
            conn.sendall(response)
            conn.close()
        except OSError as e:
            conn.close()
            print('Connection closed')
server()
```

Jak je vidět, z čidla se zde využívá pouze informace o teplotě, doplnění dalších informací můžete zkusit samostatně.

## Co jste se naučili

*V tomto Pracovním listu jsme navázali na Pracovní list 2 a ukázali jsme si, jak připojit externí digitální čidlo k ESP32 a jak z něj získávat hodnoty teploty, tlaku a informaci o nadmořské výšce. Také jsme se naučili, jak informaci o teplotě publikovat pomocí webového serveru.*

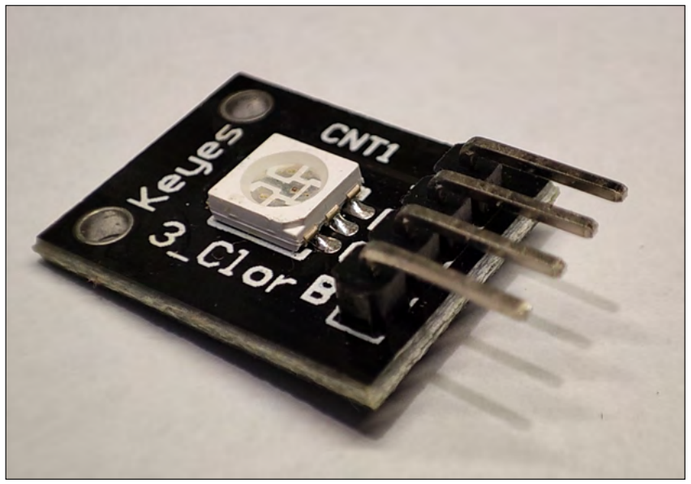
# Pracovní list 4
# ESP32 jako IoT zařízení

## Co budeme potřebovat

- Počítač s nainstalovanými programy z Pracovního listu 1 a 2 a 3 s přístupem na internet
- Funkční Wi-Fi síť umožňující připojení dalších zařízení (bez filtrace MAC adres, heslo není překážkou, funkční server DHCP pro přidělení IP adresy)
- ESP32 a propojovací USB kabel
- Čidlo využívající čip BMP180, RGB LED modul

## A jdeme na to

Připojte k počítači zařízení ESP32. Otevřete si vývojové prostředí Thonny a jednoduchým příkazem ověřte, že jste schopni s ESP32 komunikovat (viz Pracovní list 1). V této části nejprve zprovozníme RGB LED modul umožňující indikaci stavu zařízení či naměřených hodnot. V druhé části se pak podíváme, jak zkombinovat činnosti RGB modulu s výstupy Pracovního listu 3.

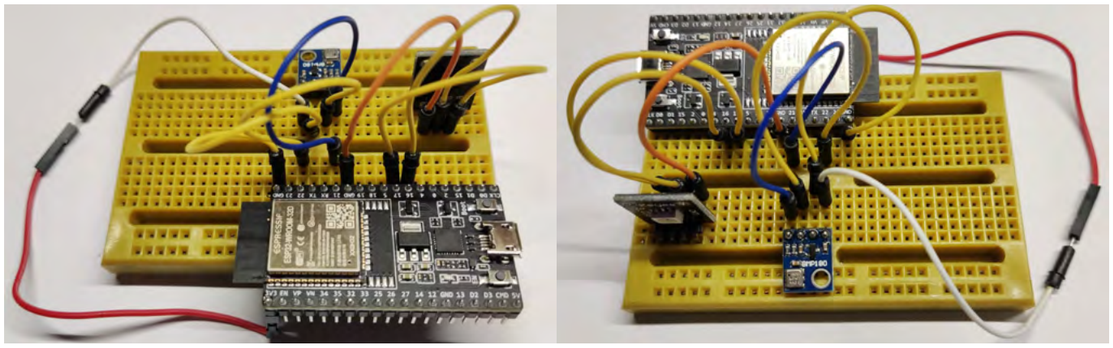



Obrázek 11: Modul RGB LED

Nejprve tedy zprovoznění RGB LED modulu. Pro to budete potřebovat nepájivé kontaktní pole se zapojením dle Obrázku 12. REGB LED modul je externí modul, na kterém je umístěna RGB dioda se společnou anodou nebo katodou.







Obrázek 12: Zapojení modulů BMP180 a RGB LED

Dále uvedený kód pak umožňuje ovládat jednotlivé barevné složky RGB.

```
import time

from machine import Pin


led_red = Pin(16,Pin.OUT)

led_green = Pin(17,Pin.OUT)

led_blue = Pin(18,Pin.OUT)


led_red.value(0)

led_green.value(0)

led_blue.value(0)


while True:

    led_red.value(1.0)

    time.sleep(0.5)

    led_red.value(0)

    time.sleep(0.5)

    led_green.value(1.0)

    time.sleep(0.5)

    led_green.value(0)

    time.sleep(0.5)

    led_blue.value(1.0)
```

```
    time.sleep(0.5)
    led_blue.value(0)
    time.sleep(0.5)
```

Druhým cílem je využít informaci o teplotě z externího modulu dle Pracovního listu 3 pro řízení barvy RGB diody. Kompletní kód tohoto úkolu následuje (je zřejmé, že je nutné doplnit i hardwarové zapojení dle Pracovního listu 3). Rozhodující teplotou je zde 30 °C, pod touto teplotou svítí LED červeně (teplota je příliš nízká), nad ní pak zeleně (teplota je dostatečná). Měření teploty probíhá každých 5 sekund.

```
import time
from machine import Pin


led_red = Pin(16,Pin.OUT)
led_green = Pin(17,Pin.OUT)
led_blue = Pin(18,Pin.OUT)


led_red.value(0)
led_green.value(0)
led_blue.value(0)


from bmp180 import BMP180
from machine import SoftI2C, Pin
bus =  SoftI2C(scl = Pin(22), sda = Pin(21), freq = 100000)
bmp180 = BMP180(bus)
bmp180.oversample_sett = 2
bmp180.baseline = 101325


while True:
    temp = bmp180.temperature
    if temp<30:
        led_red.value(1)
        led_green.value(0)
```





```
    else:
        led_green.value(1)
        led_red.value(0)
    time.sleep(5)
```

Naším finálním projektem bude spojení funkcionalit z Pracovního listu 3 a dosud uvedených v tomto pracovním listu. Zde ale narazíme na určitý problém. Chceme totiž pravidelně měřit teplotu a reagovat na ni dvěma odlišnými postupy, a to současně. Tento stav jasně vede na paralelní zpracování informace o teplotě, což se dá zajistit zprovozněním více tzv. programových vláken. Kompletní kód tohoto již pokročilého úkolu následuje (včetně částí zajišťujících připojení k síti Wi-Fi).

```
import time
from machine import Pin


led_red=Pin(16, Pin.OUT)
led_green=Pin(17, Pin.OUT)
led_red.value(0)
led_green.value(0)


from bmp180 import BMP180
from machine import SoftI2C, Pin


bus =  SoftI2C(scl=Pin(22), sda=Pin(21), freq=100000)
bmp180 = BMP180(bus)
bmp180.oversample_sett = 2
bmp180.baseline = 101325


import _thread
act_temp='0'


def update():
    global act_temp
```

```
    while True:
        temp = bmp180.temperature
        if temp<30:
            led_red.value(1)
            led_green.value(0)
        else:
            led_green.value(1)
            led_red.value(0)
        time.sleep(5)
        act_temp=str(temp)

_thread.start_new_thread(update, ())

import network

def connect(ssid,passwd):
    import network
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        print('connecting to network...')
        sta_if.active(True)
        sta_if.connect(ssid, passwd)
        while not sta_if.isconnected():
            pass
    print('network config:', sta_if.ifconfig())

connect('sitSSID,'sitPASSWD')

try:
  import usocket as socket
except:
```





```
  import socket

def web_page(temp):
    html = """
<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="refresh" content="5" />
</head>
<body>
    <h2>Vase ESP32 vas vita</h2>
    <p>Aktualni teplota je (C): """+temp+"""</p>
</body>
</html>
"""
    return html

def server():
    global act_temp
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(('', 80))
    s.listen(5)

    while True:
        try:
            conn, addr = s.accept()
            conn.settimeout(3.0)
            print('Received request from %s' % str(addr))
            request = conn.recv(1024)
            conn.settimeout(None)
            request = str(request)
```

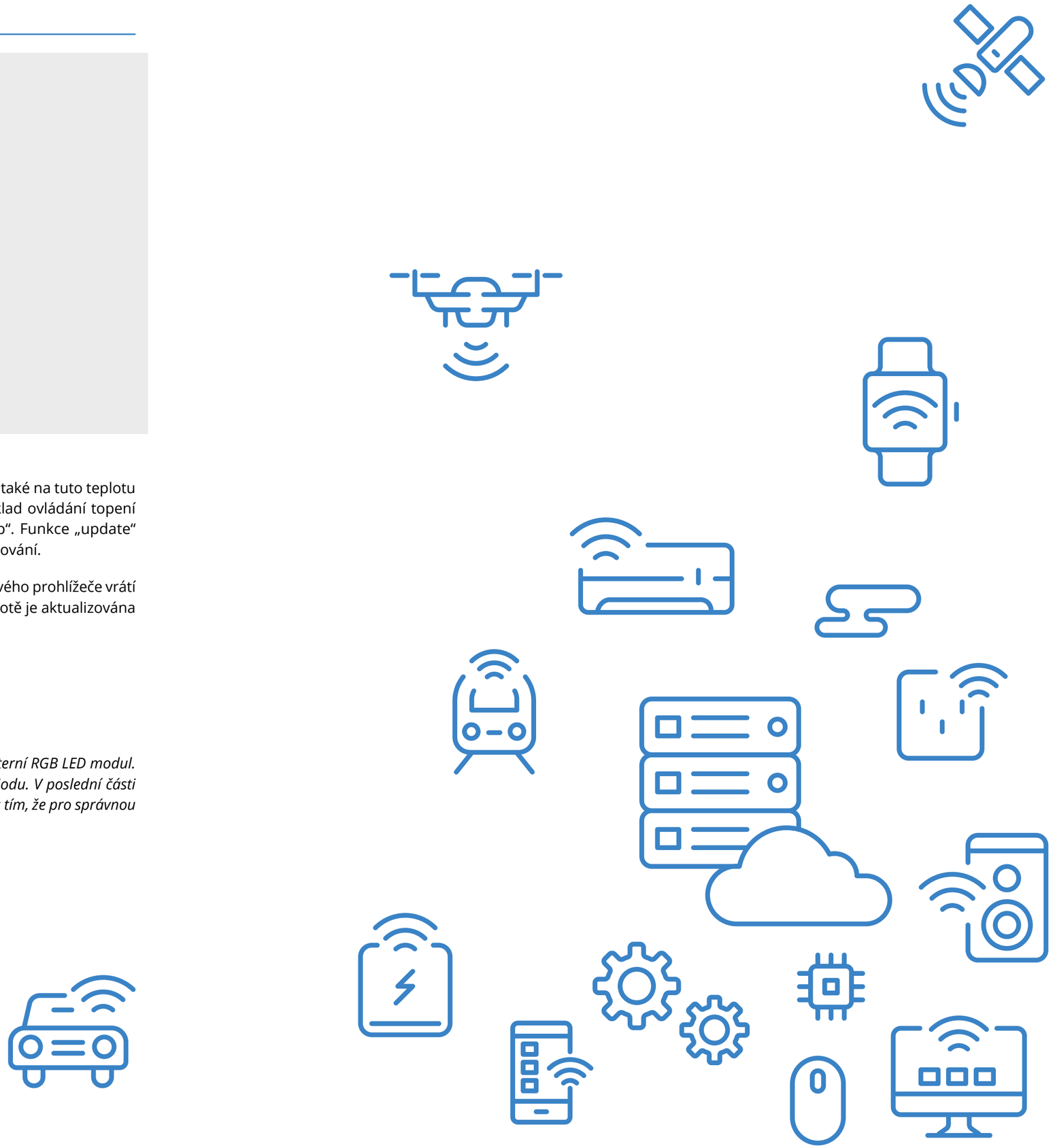
```
            print('GET Request Content = %s' % request)
            response = web_page(act_temp)
            conn.send('HTTP/1.1 200 OK\n')
            conn.send('Content-Type: text/html\n')
            conn.send('Connection: close\n\n')
            conn.sendall(response)
            conn.close()
        except OSError as e:
            conn.close()
            print('Connection closed')
server()
```

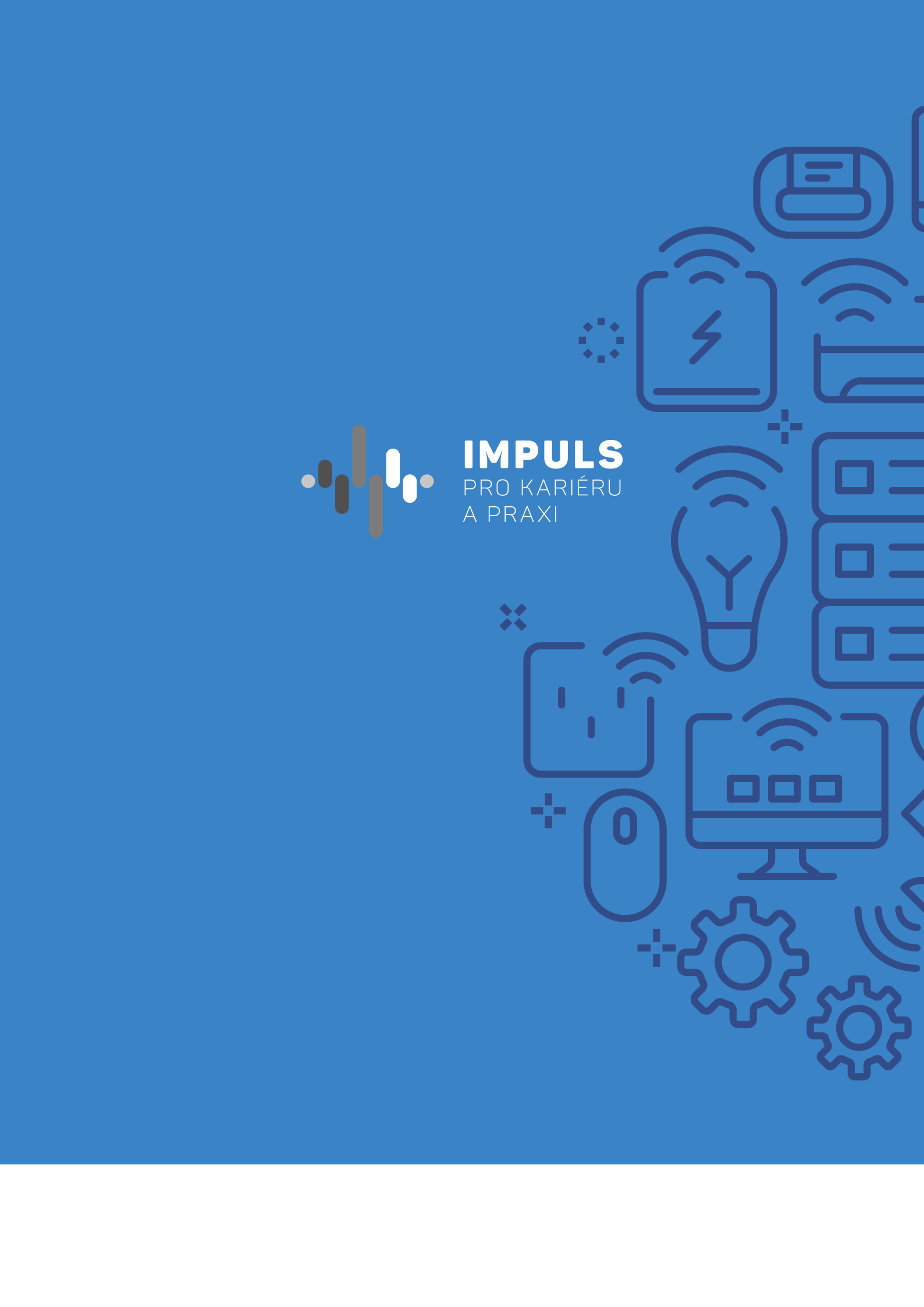
Klíčová je zde funkce „update“, která zjišťuje informaci o teplotě z čidla každých 5 sekund a také na tuto teplotu reaguje nastavením správné barvy RGB diody (místo které je možné si představit například ovládání topení v rodinném domě). V každém cyklu je také aktualizována globální proměnná „act_temp“. Funkce „update“ je spouštěna v samostatném pracovním programovém vláknu, což zajišťuje nezávislé fungování.

V hlavním vláknu našeho ESP32 pak běží webový server, který po přijetí požadavku od webového prohlížeče vrátí uživateli aktuální teplotu převzatou právě z proměnné „act_temp“. Stránka s údajem o teplotě je aktualizována opět každých 5 sekund.

## Co jste se naučili

*V Pracovním listu 4 jsme navázali na Pracovní listy 2 a 3 a ukázali jsme si, jak ovládat externí RGB LED modul. V druhé části jsme pak zajistili, že teplota zjištěná externím teplotním čidlem řídí RGB diodu. V poslední části jsme pak k již získané funkcionalitě doplnili publikaci hodnoty teploty přes webový server s tím, že pro správnou funkci bylo nutné použít vláknové programování ESP32.*




IMPULS
PRO KARIÉRU
A PRAXI