

Python med M5Stack Core 2

Af: Michael Hansen, Coding Pirates Furesø, 2021, version 1.00

Dokumentet ligger her: <https://github.com/mhfalken/m5stack/blob/main/m5stack-python.pdf>



Formålet er at lære at programmere i Python, hvor man bruger en M5Stack Core 2 som platform. Selve dokumentet er baseret på 'learning by doing', dvs der er ikke meget Python undervisning i selve dokumentet. Jeg har lavet et præsentationssæt som gennemgår de vigtigste ting i Python og som også indeholder lidt generelt viden omkring embedded systemer.

Det er meningen at eleven (piraten) får dette dokument og så selv arbejder med det i sit eget tempo.

Før første brug er det vigtigt at man (de voksne) har installeret den rigtige firmware på M5Stack'en. Se kapitel 12.

1	Installation af editor på PC	3
1.1	Visual Studio Code editor	3
2	Hello world	4
3	Tekst og grafik.....	5
3.1	Tekst	5
3.2	Grafik	5
3.3	Touchskærm	6
4	Gyro	6
5	SD kort, lyd og billeder	7
6	Ekstern hardware (Grove)	8
7	Vaterpas.....	8
8	Terning.....	8
8.1	Tegn terningen	9
8.2	Kast terningen.....	9
8.3	To terninger	9
9	Labyrint spil	9
9.1	Lav labyrint	9
9.2	Flyt kuglen	10
9.3	Selve spillet.....	10
9.4	Flere features.....	10
10	Ideer.....	10
11	M5Stack hardware overblik.....	11
11.1	M5Stack Core 2.....	11
12	Installering af Python på M5stack Core 2.....	11
12.1	M5Burner	12
13	API reference.....	12
13.1	Generelle tips	12
13.2	Problemer.....	12
13.3	Funktions overview	12
14	Løsninger	14
15	Links.....	17

1 Installation af editor på PC

For at programmere M5stack'en i Python er det nødvendigt at installere Visual Studio Code (VSC) editor på sin computer.

1.1 Visual Studio Code editor

Windows & Mac

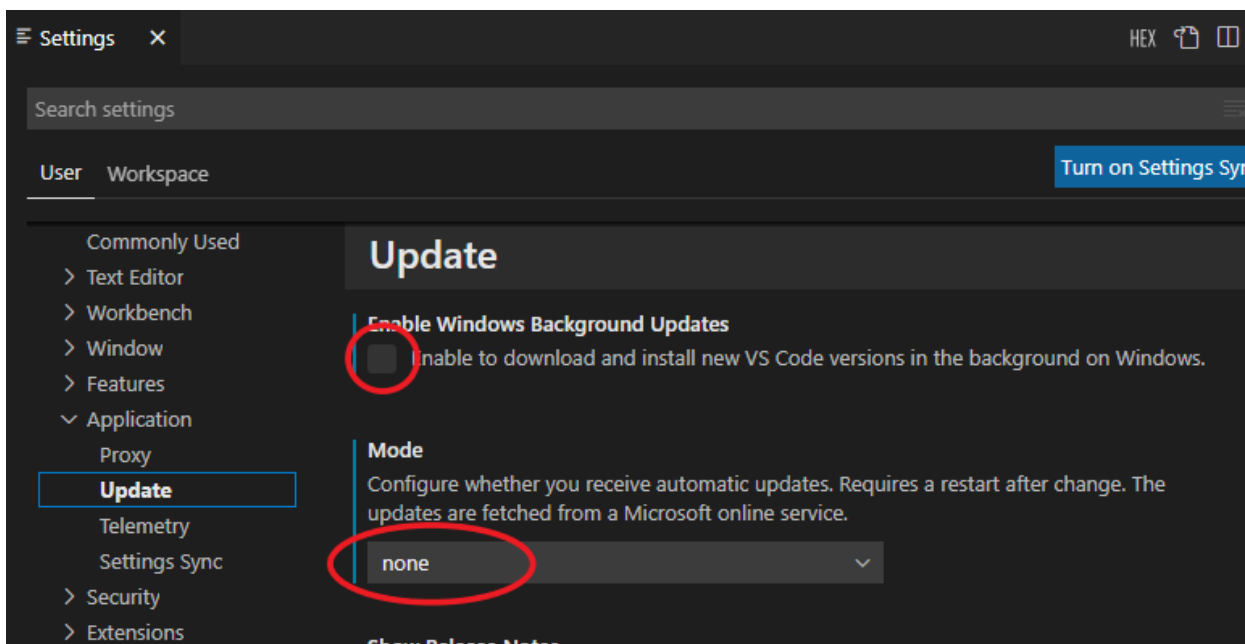
Vi skal bruge en editor som hedder Visual Studio Code og en M5Stack plug-in. Problemet er at denne plug-in desværre ikke virker på nyeste version af VSC!

Det er derfor nødvendigt at installere en ældre version af VSC. Dette gøres på følgende måde:

Man skal installere version 1.52.1: https://code.visualstudio.com/updates/v1_52 (User)

Derefter er det vigtigt at man forhindrer den i at opdatere sig selv. Start VSC og tryk på **File->Preferences->Settings**. Vælg: **Application->Update**.



Fjern 'Enable Windows Background Update' og set Mode til 'None' (se billede).



Hvis VSC når at opdatere sig selv inden man får disabled auto-update, så skal man bare installere versionen igen, da den husker instillingerne og derfor ikke når det næste gang 😊

1.1.1 M5Stack udvidelsen

Efter installation af VSC skal man installere en M5Stack plugin som hedder `vscode-m5stack-mpy`.

Start VSC editoren tryk på  ude til venstre og skriv **M5Stack** og installer plug-in'en. Efter installeringen, så kommer der en knap frem i bunden af editoren: **Add M5Stack**. Tilslut M5Stack til USB porten og tryk på knappen **Add M5Stack** og vælg USB porten (COM eller /dev/ttyUSBx). Herefter kan man se filerne på enheden ved vælge Explorer (CTRL+SHIFT+E) eller .

Nu er der forbindelse og vi kan oprette filer på enheden og flytte filer til enheden.

1.1.2 USB drivers (kun Windows)

For at kommunikere med M5stack er det muligvis nødvendigt at installere de rigtige USB-drivere.

Windows kan have disse installeret allerede, men ellers kan de findes her:

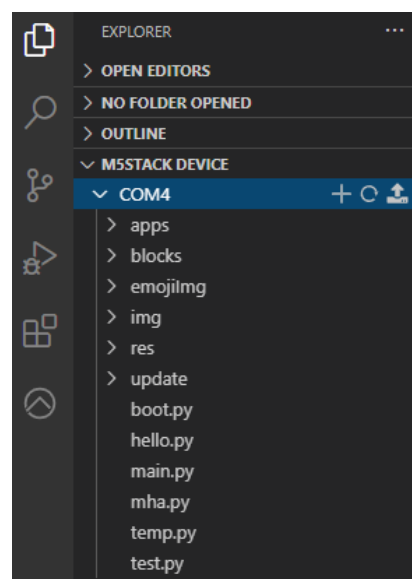
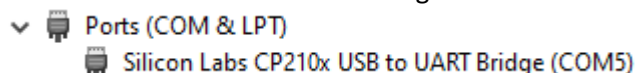
USB driver:

http://www.ftdichip.com/Drivers/CDM/CDM21228_Setup.zip. Filen pakkes ud og programmet køres for at installere driveren.

USB bridge:

https://www.silabs.com/documents/public/software/CP210x_Universal_Windows_Driver.zip. Files pakkes ud og filen CP210xVCPInstaller_x64.exe køres for at installere bridgen.

Det skal se sådan ud i Device Manageren:



1.1.3 UIFlow

Dette program er ikke nødvendigt, men kun omtalt, da der kan være godt til at 'gætte' hvordan man programmere hardwaren. Det er ikke yderligere omtalt i dette dokument.

UIFlow desktop IDE (til visuel programmering)

UIFlow desktop er et program til at programmere M5stack på en mere intuitiv måde med visual blocks programmering, som det er kendt fra Scratch og Microbit. Det hentes her:

<https://m5stack.com/pages/download>. Det er en zip-fil som blot pakkes ud, og så kører man programmet som det er, altså ingen egentlig installation så der kræves ikke super-bruger privilegier.

(Den nuværende version af UIFlow er ikke opdateret til den nyeste Python version som findes til Core 2! – og de er desværre ikke helt kompatible...!)

Der findes også en online version, som er mere opdateret: <https://flow.m5stack.com/>

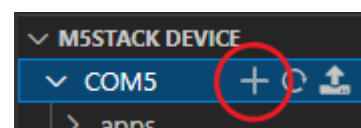
2 Hello world

Start VSC editoren og sørg for at der er forbindelse til M5Stack'en.


Tryk på + (ud for M5Stack-forbindelsens COM port) for at tilføje en fil på enheden og kald den `my_demo.py`.

(Når man har oprettet en ny fil på enheden, så har den en tendens til at 'falde af USB forbindelsen'. Så skal man bare re-konnekte den igen ved at trykke på Add M5Stack i bunden af VSC, og derefter åbne filen.)

Skriv følgende i filen:



```
from m5stack import lcd
lcd.print('Hello world!', 50, 50, 0xff0000)
```

Kør det ved at trykke på  (den hvide) i toppen til højre af editoren. Enheden skulle nu gerne skrive "Hello World!" på skærmen.

Tip: Hvis Python koden ikke 'afslutter', så kan man ikke køre koden igen, før man har resat enheden. Se 13.3.2.

Når man gemmer sin kode, så gemmes den på selve M5Stack Core 2 enheden, dvs, at hvis man næste gang får en anden enhed, eller hvis den er blevet slettet, så er al ens kode væk. Det er derfor vigtigt at man også gemmer sine filer på sin egen laptop!

3 Tekst og grafik

3.1 Tekst

Displayet er 320x240 pixels, med (0, 0) i øverste venstre hjørne (hvilket er helt normalt).

Man skriver en tekst på følgende funktion (se eksemplet ovenfor):

```
lcd.print("Test", x, y, textColor)
```

Man kan ændre font'en på følgende måde:

```
lcd.font(lcd.FONT_DejaVu18)
```

hvor '18' angiver størrelsen på bogstaverne og kan være 18, 24, 40, 56 og 72.

Farven styres generelt med en 'farvekode' med følgende format: 0xRRGGBB, hvor RR er rød, GG er grøn og BB er blå, og 0x angiver at det er angivet i HEX. Grunden til at der er to bogstaver er, at det skal skrives i HEX hvor 2 tegn angiver en værdi mellem 0 og 255 (0-FF).

```
lcd.print('Test', 50, 50, 0xff8000)
```

'Gode' HEX værdier er: 00, 40, 80, c0, f0, ff. (Det er frit valg om man bruger store eller små bogstaver)

Man kan slette hele skærmen med følgende kommando:

```
lcd.clear()
```

Man kan ikke skrive 'æøå'.

Hvis man skriver noget som M5Stack'en ikke kan forstå skriver den typisk 'Invalid syntax' Man kan så trykke på skærmen (detail) for at se hvad den ikke kan lide. Her vil typisk være en fejl kode og et linjenummer, hvor det er gået galt.

Opgave

1. Skriv tekster med forskellige størrelser og farver

Gem som: tekst-size-color.py.

3.2 Grafik

Følgende funktioner laver streger og figurer på skærmen:

```

lcd.line(x0, y0, x1, y1, lineColor)      # Tegn linje
lcd.rect(x, y, dx, dy, lineColor, fillColor)  # Tegn firkant
lcd.circle(x, y, r, lineColor, fillColor)    # Tegn cirkel

```

Parametrene er lidt ulogiske. Linjen angives med de på endepunkter, mens firkanten angives med et punkt og en afstand (dx, dy). Cirklen angives med centrum og en radius.

Opgave

1. Tegn nogle figurer på skærmen

Gem som grafik-test.py.

3.3 Touchskærm

M5Stack'en har ingen fysiske knapper, men en touchskærm. Man kan aflæse hvor man har trykket sidst med følgende funktion.

```

from m5stack import touch
(x, y) = touch.read()

```

Hvis man vil vide om der er trykket på skærmen, bruges følgende funktion:

```
touch.status()
```

Opgave

1. Lav et program som skriver positionen ud når man trykker på skærmen

Hint

Man kan lave en loop på 10 iteration på følgende måde:

```

import time

for i in range(10):      # Loop på 10 iterationer
    time.sleep(0.5)      # Lille pause

```

`range(10)` laver reelt en liste med 10 elementer fra 0-9. Vi skal dog ikke her bruge selve værdien.

Man kan printe variable ud på følgende måde:

```

lcd.print("Value= %i" %x)      # Print værdien af x
lcd.print("V1= %i, V2= %i" %(x1, x2))  # Print 2 værdier

```

hvor `%i` betyder integer, `%f`: float, `%s`: string og `%c`: char (bogstav). Hvis man har flere parametre, så puttes de ind i en parentes.

Gem som touch-pos.py.

4 Gyro

M5Stack'en har en 3 akse gyro og en 3 akse accelerometer. Gyroen kan måle retningsforandringer, mens accelerometeret kan måle "kraften" i forhold til jorden (tyngdekraften) og bruges til at måle vinklen med. De kan aflæses på følgende måde:

```
import imu
imu0 = imu.IMU() # Skal kun kaldes een gang

(gx, gy, gz) = imu0.gyro
(ax, ay, az) = imu0.acceleration # [-1;1]
```

Opgave

1. Start med at printe værdien af accelerometeret ud for at se hvordan den virker
2. Tegn en cirkel midt på skærmen, med en radius på 10.
3. Flyt cirklen rundt på skærmen vha. accelerometeret, så den tegner en streg.
4. Ændre koden så den flytter cirklen, uden at der kommer en streg efter den.

Hint

1. Hvis programmet ikke afslutter, så kan man ikke download et nyt uden at resette M5Stacken først. Det er lidt besværligt, så i stedet for `while True` kan man skrive

```
while not touch.status():
```

2. Accelerometeret returnerer en float (kommatal). Den kan derfor ikke bruges i tegnefunktionerne som forventer en integer (heltal). Man kan konvertere en float til en integer med `i = int(f)`.
3. Husk at man skal slette den 'gamle' cirkel inden man tegner den 'nye'. (Slet cirkel, flyt punkt, tegn ny cirkel).

Gem som cirkel-gyro.py.

5 SD kort, lyd og billeder

M5Stacken har et micro SD kort slot i siden, som kan bruges til mediafiler som lyd og billeder. For at det virker skal SD kortet side i når den starter op, enten efter en power-on eller reset. Set fra CPU'en ligger filerne i `/sd/`. Vi vil ikke bruge SD kortet lige nu.

M5Stacken har en lille højttaler indbygget, som man kan få til at spille noget lyd. Det kan gøres på to forskellige måder:

En simpel tone:

```
speaker.playTone(220, 2) # Tone: frekvens (Hz), tid (s)
```

Den kan også afspille en lydfil i WAV format med 44100 Hz og 16 bit format.

```
speaker.playWAV('/sd/tadar.wav', 44100, speaker.F16B) # WAV file
```

Den har også en vibrator, som kan bruges til haptic feedback. Den er beskrevet i referencen længere nede i dokumentet, kapitel 13.3

Opgave

1. Få den til at spille en serie af toner. Brug en liste af tupler.

Hint

1. En liste af tupler ser sådan ud: `tones = [(220, 1), (400, 2), ...]`, hvor det første tal er tonen og det andet tal er tiden den skal spilles.
2. Man tilgår listen på følgende måde: `for (t, s) in tones:`

Gem som tones.py.

M5Stacken kan vise billeder i jpg format, men i praksis så resetter den hvis man gør det direkte fra SD kortet. Man skal derfor først kopiere billederne ind på flash'en og derefter vise dem. Det er derfor ikke beskrevet her hvordan man gør, men selve API funktionen findes i referencen, kapitel 13.3.

6 Ekstern hardware (Grove)

M5Stack'en har et stik på siden (GROVE), hvor man kan sætte eksterne sensorer til den. Man tilslutter sensorerne vha. et lille kable.

Vi har forskellige sensorer at vælge i mellem og deres API er lidt forskellige alt afhængig af hvad de kan.

Lyssensor [LIGHT] (kan måle hvor lyst det er).

```
import unit
light = unit.get(unit.LIGHT, (32,33))
light.analogValue # 0-1023
```

Farvesensor [COLOR] (kan måle hvor meget rød/grøn/blå farve lyset har)

```
import unit
color = unit.get(unit.COLOR, (32,33))
color.red # 0-255
color.green # 0-255
color.blue # 0-255
```

Barometer [BPS] (kan måle lufttrykket og temperaturen. Den er meget følsom og kan derfor måle hvor højt den er oppe).

```
import unit
bps0 = unit.get(unit.BPS, unit.PORTA)
bps0.pressure() # hPa
bps0.temperature() # C
```

Opgave

1. Vælg en sensor og lav et program som kan skrive værdien/erne ud på skærmen.
2. Prøv at få værdien til at forandre sig, ved at påvirke sensoren.
3. Får farvesensoren til at vise den farve den ser på skærmen. Brug bit shift til at få farverne 'på plads'.

farve = red << 16 | green<<8 | blue.

Gem som color-test.py

7 Vatterpas

Lav et vatterpas som kan bruges til at se om flader er vandrette.

Lav en funktion som kan tegne et vatterpas med libelle (luftboblen) som kan flytte sig som funktion af en parameter.

```
def VPDraw(level): # level: [-1;1]
    <code>
```

Lav en loop som aflæser accelerometeret og kalder VPDraw().

```
import imu
imu0 = imu.IMU()
(ax, ay, az) = imu0.acceleration
```

Vær opmærksom på at accelerometeret ikke er kalibreret, så nul punktet ikke nødvendigvis er helt præcist.

Gem filen som vatterpas.py.

8 Terning

Lav en terning med M5Stack'en.

8.1 Tegn terningen

Lav en funktion som kan tegne en terning. Fx på denne måde.

```
def DieDraw(x, y, no):      # (x, y)= centrum, no= antal øjne
```

Sørg for at terningen ikke bliver større end at der er plads til to.

Hint

Det kan være en fordel at bruge en dobbelt liste af tupler til at angive hvor øjnene på terningen sidder. Her er vist hvordan det kan se ud for de to første værdier.

```
dotPos = [[(0, 0)], [(-1, -1), (1, 1)], [...]]
```

Kald `DieDraw()` funktionen med alle værdier (1-6) for at se om den virker.

8.2 Kast terningen

Lav så man skal ryste boksen for at ændre værdien. Brug gyroen – en værdi over 80 virker godt.

```
import imu
imu0 = imu.IMU()
(x, y, z) = imu0.gyro
```

Gyroen returnerer både positive og negative værdier, så funktionen `abs()` kan bruges til at finde den numeriske værdi af et tal.

Random nummer:

```
from random import *
v1 = randint(1, 6)
```

Det vil se godt ud hvis man kan se at værdierne ændrer sig, mens man ryster boksen.

8.3 To terninger

Lav så man har to terninger i stedet for én.

Hint

Kald `DieDraw()` to gange.

Gem som `terning.py`.

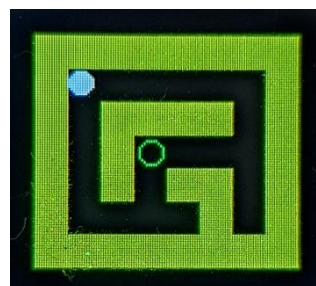
9 Labyrint spil

Lav et simpelt kugle labyrint spil.

9.1 Lav labyrint

Lav en datastruktur til en labyrint. Den skal helst opfylde følgende krav:

- Let at kode i Python
- Let at printe på skærmen
- Let at finde rundt i



Hint

Det kan se sådan ud:

```
labyrint = [
    "XXXXXXXXX",
    "X        X",
    "X XXXXX X",
    "X XE    X",
    "X X XX  X",
    "X      X X",
    "XXXXXXXXX"]
```

Dette er en simpelt dobbelt liste (liste af linjer og liste af tegn). Det er nemt at tegne i Python, let at printe og let at finde rundt i.)

Tilgang til listen er: `labyrint[y][x]`

Tegn labyrinten på skærmen, og brug flere pixels pr. felt, 14x14 (tegn som et rektangel).

Hint

Man kan læse listen på følgende måde:

```
for line in labyrint:
    for c in line:
```

9.2 Flyt kuglen

Tegn en kugle (cirkel) på skærmen med en radius på 5. Aflæs accelerometeret og flyt kuglen ligesom i kapitel 4.

(Tip: hvis kuglen blinker for meget, så husk at man skal slette kugle lige inden den tegnes igen, og have et lille delay efter den er tegnet. Slet gammel kugle, flyt position, tegn ny kugle, delay)

9.3 Selve spillet

Sæt kuglen ind i labyrinten, så den starter i øverste venstre hjørne.

Flyt kuglen som før, men check om den rammer væggen eller hullet inden den flyttes. Dette er lidt tricky, da man skal checke alle 4 'hjørner' af kuglen (se figur). Lav en funktion som returnerer hvad den rammer, dvs. 'X', 'E' eller ' '.

Hint

Husk at når man kalder funktionen med centrum af kuglen, så skal man 'fjerne' labyrintoffsettet inden (se figur).

Labyrinten er 14 pixels bred og kuglen har en radius på 5. Så man finder indekset i labyrinten på følgende måde for det 1. hjørne:

```
x = int((kugleX-5)/14)
y = int((kugleY-5)/14)
```

Opdater funktionen så den checker alle fire hjørner.

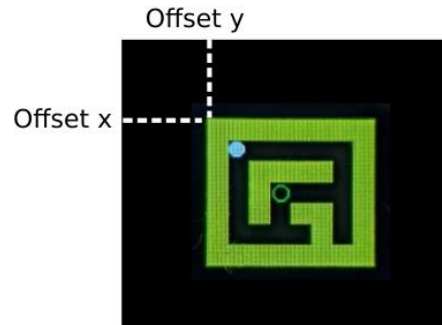
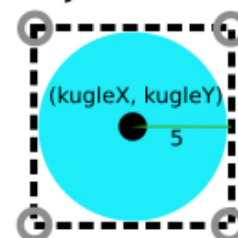
Hvis bare et af 'hjørnerne' rammer X, så må kuglen ikke flyttes!

Opdater funktionen så den også checker om man rammer 'E'.

Ændre koden så hvis man 'rammer' 'E', så er man færdig med spillet.

Gem som labyrint.py.

1. hjørne



9.4 Flere features

Spillet kan udvides med flere features, som fx:

- Større bane
- Splash skærm (start skærm med spil navn, udvikler mv.)
- Tidtagning (brug `time.time()`)
- Ekstra 'huller' som man ikke må ramme
- Flere baner efter hinanden
- Lydeffekter
- Inerti i kuglen og bounc på vægen (svært)

10 Ideer

Ping-pong spil – spil 'bold' op af 'mur' og bevæg 'bat' med gyro eller touch.

Slange der spiser prikker og bliver længere og længere

11 M5Stack hardware overblik

M5Stack findes i flere varianter, som har mange ligheder, men også en del forskelle. De fleste er baseret på en CPU som hedder ESP32, som er en dual core 32 bit 240MHz processor med indbygget WiFi og bluetooth. De har USB-C interface og indbygget batteri, samt en generisk 4 pin connector kaldet Grove.



Her er nogle eksempler:

- De mindste hedder M5StickC og M5StickC+ og har en 0.96" skærm, 1 knap, gyro og en extension connector på enden.
- Den næste i rækken er M5Stack core basic, som har en 2" skærm og 3 knapper, SD kort, højttaler og en extension connector på siderne.
- Den mest avanceret er M5Stack core 2, som har en 2" touch skærm, gyro, SD kort, mikrofon, højttaler og en extension connector på undersiden.



Der ud over så findes der også nogle, som er til specifikke formål og derfor er noget anderledes og derfor ikke yderligere omtalt her.

Vær opmærksom på at M5Stack basic og m5stack core 2 IKKE har de samme tilføjelse muligheder. På trods af samme udseende og dimensioner, så er det kun via de porte (Sd-kort, 4 pin connector) som er i siden, at de to moduler kan arbejde med samme hardware og sensorer. De moduler som kan "stackes" er ikke kompatible imellem de to versioner, og køber du derfor et stackmodul, så virker det kun til den ene version!

Der findes mange eksterne enheder som kan tilsluttes den fælles 4 pin konnektor, såsom temperatur sensorer, neopixels mv. Her er de fysiske ben desværre heller ikke de samme og det er ikke alle sensorer som virker på M5Stack basic, da de har valgt de 'forkerte' ben. Det er blevet løst på fx. M5Stack core 2. Koden kan derfor ikke flyttes mellem de forskellige enheder uden at den bliver rettet en smule. Dette dokument forudsætter at man bruger en M5Stack Core 2.

11.1 M5Stack Core 2

Det er M5Stack Core 2 som bliver brugt i dette dokument. Der er en beskrivelse af hardwaren på undersiden af enheden.

- ESP32 (2x 240MHz, 16MB flash, 520 kB RAM, bluetooth, WiFi)
- 2" touch farveskærm (320 x 240)
- 6x Gyro, højttaler, mikrofon, SD kort
- Grove (pin 32, 33)
- Extension konnektor (som er brugt)
- USB C stik



Power-on	Tryk på Power knappen, eller tilslut USB
Power-off	Tryk på Power knappen i 6 sekunder
Reset	Tryk på Reset knappen – der går lidt tid inden der 'sker' noget.

12 Installering af Python på M5stack Core 2

For at køre programmer skrevet i Python er der brug for at der er installeret en Python fortolker på M5Stack'en. På M5Stack'en ligger der typisk noget software i forvejen. Det er normalt enten noget demo software eller UIFlow. UIFlow bruges til Python og UIFlow programmering.

12.1 M5Burner

M5Burner er et stykke software som bruges til at “brænde” software over på M5stack-hardwaren. Det hentes her: <https://m5stack.com/pages/download>. Det er en zip-fil som blot pakkes ud, og så kører man programmet som det er, altså ingen egentlig installation så der kræves ikke super-bruger privilegier.

Windows

Udpak zip-filen et passende sted.

Herefter startes programmet som `M5Burner.exe`.

Linux

Udpak filerne med

```
$ unzip -d M5Burner_Linux{, .zip}
```

Herefter startes programmet som `M5Burner_Linux/bin/m5burner`.

Med enheden sat i USB-porten og ved at starte programmet `M5Burner` kan man se en stribe muligheder for at installere allerede udviklede firmware. Firmware er specifik for en bestemt enhed, så man skal være opmærksom på hvilken enhed man har. Her tager vi udgangspunkt i **M5Stack Core 2**.

Man vælger enhed i venstre side, vi vælger **Core**.

Både til Python programmering og til visuel programmering bruger man den firmware som hedder UIFlow. Se billede.

Efter man har valgt version (i billedets tilfælde 1.8.1-en, så klikkes der “Download”, hvorefter firmwaren hentes til din PC og der fremkommer 3 nye knapper.

Så vælger man “Burn” og til Wifi-billedet som kommer frem klikker man blot “Start”. Så går der noget tid (ca. 30 sekunder), hvorefter firmware er brændt på enheden. Nu er enheden klar til at fortolke Python-programmer.

M5Stack’en skal sættes i USB mode, før den kan bruges. Dette gøres lidt forskellige afhængig af M5Stack type. Her er Core 2 beskrevet:

Tænd for enheden (eller sæt USB stikket i). Tryk på **Flow** og vælg **USB**.

Enheden er nu i USB mode og husker selv denne tilstand.



13 API reference

13.1 Generelle tips

Der er nærmest ingen dokumentation på hvordan Python interfacet til M5Stack hardwaren ser ud. Det lettest er at bruge UIFlow, vælge de blokke man vil vide noget om og så trykke på Python knappen for at se hvordan Python koden ser ud. Det kan man så kopiere direkte ind i sin egen kode.

Tip: når man vælger en specifik blok, så kommer der lidt yderlige hjælp frem ude til højre på skærmen.

De enkelte Python releases er generelt helt u-testet – det skal man lige tænke over inden man opdaterer. De er heller ikke nødvendigvis helt bagud-kompatible!

13.2 Problemer

Hvis man skifter ekstern sensor (Grove) ‘on the fly’, så er det ikke sikkert at koden virker - det har de ikke support for. (Dette er ellers brugbart, hvis man laver et Escaperoom spil - se senere) Det er et rent software problem, da hardwaren ikke har noget problem.

13.3 Funktions overview

Her er en list over de mest gængse Python API funktioner til at interface med M5Stack hardwaren.

I de følgende eksempler, skal der bruges diverse imports. Det er ikke sikkert alle er nødvendige hver gang, men her er en kort list over hvilke man kan prøve (eller bare inkludere altid).

```
from m5stack import *
from m5ui import *
from uiflow import *
```

Tekst & grafik

```
from m5stack import lcd
color = 0xffffffff # RRGGBB

lcd.clear()
lcd.font(lcd.FONT_DejaVu18) #18, 24, 40, 56, 72
# lcd.FONT_Ubuntu, lcd.FONT_7seg, lcd.FONT_Comic
lcd.print("Test", x, y, textColor)

lcd.line(x0, y0, x1, y1, lineColor) # Tegn linje
lcd.rect(x, y, dx, dy, lineColor, fillColor) # Tegn firkant
lcd.circle(x, y, r, lineColor, fillColor) # Tegn cirkel

lcd.image(x0, y0, filename) # Vis billede
```

Touch UIFlow metode:

```
from m5stack_ui import *

touch_button0 = M5Btn(text='Button', x=60, y=50, w=70, h=30,
bg_c=0xFFFFFFFF, text_c=0x000000, font=FONT_MONT_14, parent=None)
touch_button0.set_pos(50, 50)
touch_button0.get_state()

def touch_button0_pressed():
    # Event pressed
    pass
touch_button0.pressed(touch_button0_pressed)
```

Touch raw metode:

```
from m5stack import touch
(x, y) = touch.read()
touch.status() # Touched=True
```

Lyd:

```
speaker.playTone(220, 2) # Tone: frekvens (Hz), tid (s)
speaker.playWAV('/sd/tadar.wav', 44100, speaker.F16B) # WAV file
# 44100 Hz, 16 bit format
```

Vibrator:

```
power.setVibrationIntensity(100) # 0-100
power.setVibrationEnable(True)
power.setVibrationEnable(False)
```

Gyros

```
import imu
imu0 = imu.IMU()
(x, y, z) = imu0.acceleration # [-1;-1]
(x, y, z) = imu0.gyro
```

Batteri

```
v = power.getBatVoltage()
```

13.3.1 Eksterne sensorer (Grove)

Lys sensor

```
import unit
light = unit.get(unit.LIGHT, (32,33))
light.analogValue # 0-1023
```

Farve sensor (kan måle hvor meget rød/grøn/blå farve lyset har)

```
import unit
color = unit.get(unit.COLOR, (32,33))
color.red # 0-255
color.green # 0-255
color.blue # 0-255
```

Barometer (kan måle lufttrykket og temperaturen. Den er meget følsom og kan derfor måle hvor højt den er oppe).

```
import unit
bps0 = unit.get(unit.BPS, unit.PORTA)
bps0.pressure() # hPa
bps0.temperature() # C
```

13.3.2 Koden afslutter

Det er vigtigt at koden afslutter, da man ellers ikke kan downloade en ny version. Så hvis man ikke vil trykke på reset hele tiden og vente på at enheden booter, så kan man med fordel sørge for at en 'knap' får koden til at afslutte.

I stedet for

```
while True:
```

så kan man skrive:

```
while not touch.status(): # Debug
```

Så når man trykker på skærmen, så kommer man ud af løkken.

14 Løsninger

text-size-color.py

```
from m5stack import lcd

lcd.clear()

lcd.font(lcd.FONT_DejaVu18)
lcd.print("Test", 10, 10, 0xff0000)
lcd.font(lcd.FONT_DejaVu24)
lcd.print("Hej", 10, 50, 0x00ff00)
lcd.font(lcd.FONT_DejaVu40)
lcd.print("Dav", 10, 100, 0x0000ff)
```

grafik-test.py

```

from m5stack import lcd

lcd.clear()

lcd.line(10, 10, 100, 20, 0xff0000)
lcd.rect(150, 100, 80, 25, 0xffff00, 0x0000ff)
lcd.circle(100, 150, 40, 0x00ffff, 0x8040c0)

```

touch-pos.py

```

from m5stack import lcd
from m5stack import touch
import time

lcd.font(lcd.FONT_DejaVu24)

for i in range(10):
    lcd.clear()
    (x, y) = touch.read()
    lcd.print("Tryk (%i, %i)" %(x, y), 10, 10, 0xff0000)
    time.sleep(0.5)

```

cirkel-gyro.py

```

from m5stack import lcd
import imu
import time

imu0 = imu.IMU()
lcd.clear()

x = 160
y = 120

while not touch.status():
    (ax, ay, az) = imu0.acceleration
    x = x - int(ax*10)
    y = y + int(ay*10)
    lcd.circle(x, y, 10, 0x00ffff, 0x8040c0)
    time.sleep(0.1)
    lcd.circle(x, y, 11, 0x0, 0x0)

```

tones.py

```

from m5stack import lcd

tones = [(200, 2), (400, 4), (800, 1)]

for (t, s) in tones:
    speaker.playTone(t, s)

```

color-test.py

```

from m5stack import *
from m5stack_ui import *
from uiflow import *
from m5stack import touch
import time
import unit

lcd.font(lcd.FONT_DejaVu24)
lcd.clear()

while not touch.status():
    color = unit.get(unit.COLOR, (32,33))
    farve = color.red<<16 | color.green<<8 | color.blue
    lcd.circle(150, 100, 40, 0xffffffff, farve)
    time.sleep(0.2)

```

vaterpas.py

```

from m5stack import *
from m5stack import touch
import time
import imu

fillColor = 0x00e000
lastLevel = 160
def VPDraw(level):
    global lastLevel
    dx = 18
    lcd.circle(160+int(lastLevel*150), 120, 16, fillColor, fillColor)
    lcd.circle(160+int(level*150), 120, 15, 0xf0f0fff, 0xfffff00)
    lcd.line(160-dx, 105, 160-dx, 135, 0x0)
    lcd.line(160+dx, 105, 160+dx, 135, 0x0)
    lastLevel = level

imu0 = imu.IMU()

lcd.font(lcd.FONT_DejaVu56)
lcd.clear()
lcd.print("Vaterpas", 30, 30, 0xffc0c0)
lcd.rect(50, 100, 220, 40, 0xfffff00, fillColor)
while not touch.status(): # Debug exit loop
    (ax, ay, az) = imu0.acceleration
    VPDraw(ax)
    time.sleep(0.1)
lcd.clear()

```

terning.py


```

from m5stack import *
from m5stack import touch
import time
import imu
from random import *

def DieDraw(x, y, no):
    dotPos = [[(0, 0)], [(-1, -1), (1, 1)], [(-1, -1), (0, 0), (1, 1)],
               [(-1, -1), (1, 1), (-1, 1), (1, -1)],
               [(-1, -1), (1, 1), (-1, 1), (1, -1), (0, 0)],
               [(-1, -1), (1, 1), (-1, 1), (1, -1), (-1, 0), (1, 0)]]
    d = 80
    lcd.rect(int(x-d/2), int(y-d/2), d, d, 0xffff00, 0xffff00)
    dotP = dotPos[no-1]
    for (dx, dy) in dotP:
        lcd.circle(x+dx*20, y+dy*20, 7, 0, 0)

imu0 = imu.IMU()

lcd.font(lcd.FONT_DejaVu56)
lcd.print("Ryst", 80, 80, 0xff00ff)
while not touch.status(): # Debug exit loop
    (gx, gy, gz) = imu0.gyro
    if abs(gx) > 80 or abs(gy) > 80 or abs(gz) > 80:
        lcd.clear()
        v1 = randint(1, 6)
        DieDraw(100, 100, v1)
        v2 = randint(1, 6)
        DieDraw(220, 100, v2)
        time.sleep(0.1)
    lcd.clear()

```

labyrinth.py

Koden kan findes her: <https://github.com/mhfalken/m5stack/blob/main/labyrinth-simple.py>

Her er en version med flere features: <https://github.com/mhfalken/m5stack/blob/main/labyrinth.py>

15 Links

Her er lidt brugbare links:

Officiel M5Stack hjemmeside:

<https://m5stack.com/>

Firmware download:

<https://m5stack.com/pages/download>

Guide på dansk til M5StickC

<https://m5guide.readthedocs.io/>