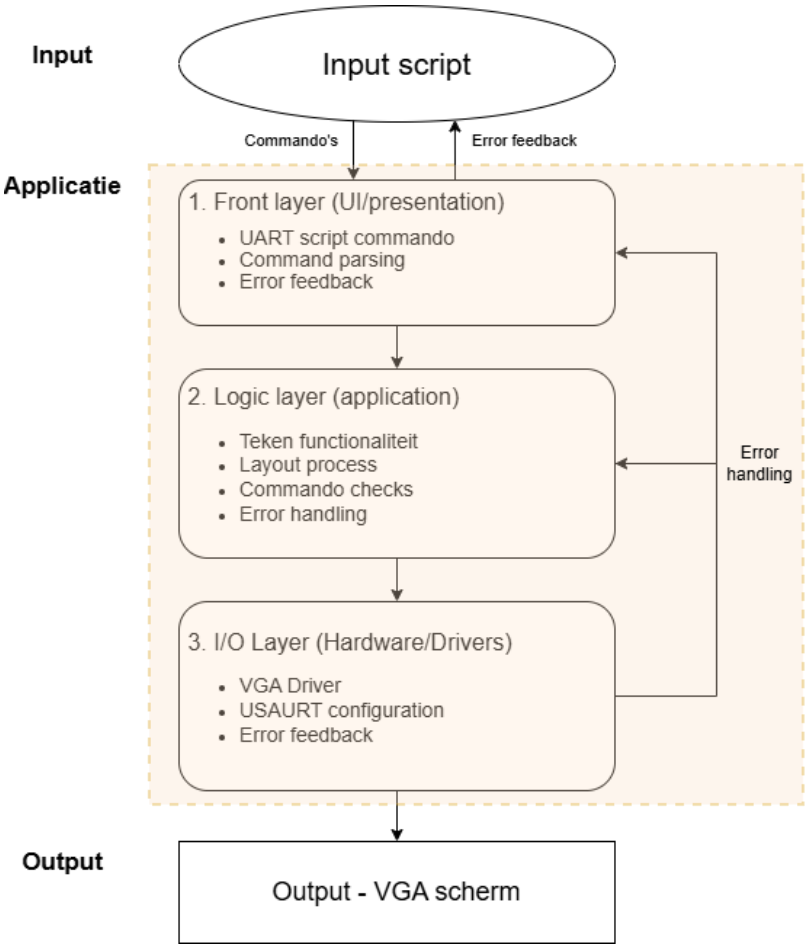
Handleiding: VGA-aansturing

VESOFTON

Naam en studentnummer:	Janiek de Bruijne, 1862484
	Max Untersalmberger, 1852619
	Joris Mullink, 1852990
Klas:	EV5A
Docenten:	Franc van der Bent, Michiel Scager
Datum:	12-1-2026
Versie:	1.0

Grafisch Overzicht en Architectuur

Hieronder is een grafisch overzicht van de software architectuur te zien.



De applicatie is ontworpen volgens het 3-lagenmodel. Dit model scheidt de applicatie in drie logische lagen:

1. **Front-laag:** Verantwoordelijk voor de gebruikersinteractie. In dit project is dit de laag die de seriele communicatie afhandelt en de binnenkomende commando's doorgeeft aan de logica-laag.
2. **Logica-laag:** Het hart van de applicatie. Deze laag bevat de businesslogica en vertaalt de commando's van de front-laag naar teken-operaties voor de I/O-laag. Het valideert de parameters van de commando's.
3. **I/O-laag:** De laag die direct met de hardware communiceert. In dit project is dit de stm32_ub_vga_screen library die de VGA-controller aanstuurt om de gewenste vormen op het scherm te tekenen.

De lagen zijn zoveel mogelijk onafhankelijk van elkaar. De front-laag praat met de logica-laag, en de logica-laag praat met de I/O-laag. In dit project is er een uitzondering waarbij de front-laag ook direct de I/O-laag kan aanroepen voor simpele taken.

Beschrijving + onderbouwing van 3-lagenmodel

Het 3-lagenmodel is een vorm van een multitier-architectuur. Bij een standaard 3-lagenmodel scheidt laag 2 (vaak business of logic layer genoemd) de lagen 1 en 3, dat wil zeggen: laag 3 mag niet met laag 1 praten en omgekeerd, ze praten alleen met laag 2, die de 'mediator' is.

Voor onze embedded toepassing is iets meer vrijheid nodig; wij willen dat de frontlayer toch direct functies uit de I/O-layer aanroept. Hieronder zie je een aangepast 'embedded' 3-tier-model, waarin je ziet dat ook de front layer functies uit de API-LIB mag gebruiken. Wat bljift is dat het software design strikt volgens 3 lagen is opgebouwd.

Wat levert het lagenmodel op?

Veel. Door de scheiding in lagen krijg je een transparanter en stabieler ontwerp, en is het makkelijker producten verder te ontwikkelen. Want de lagen zijn in principe onafhankelijk van elkaar. Zo kun je gemakkelijk hardware- en platformonafhankelijke applicaties bouwen en eerder gemaakte sources en libraries steeds hergebruiken. Bijvoorbeeld: functies voor de datalaag (in de data-library) kunnen binnen verschillende applicaties gebruikt worden. Of, per hardwaretype (processor, screen, sensoren etc.) bouw je een aparte datalaag, maar wel met exact dezelfde functiecalls. De business- of logic-laag gebruikt dan steeds dezelfde calls naar de data layer, maar gaat hier mogelijk per platform anders mee om. De user-interface blijft ongewijzigd.

Niet alleen zijn de lagen zoveel mogelijk onafhankelijk van elkaar (en bevinden ze zich mogelijk fysiek op andere locaties), ook praten ze zoveel mogelijk alleen met hun buurman. De user-interface praat dus in principe alleen met de business laag. Sterker nog, het interesseert de user-interface (als het goed is) geen zier wat 'eronder' hangt voor hardware. De programmeurs kunnen met de beschikbare user-interface-functies dan 'blind' programmeren. Als de datalaag ook al klaar is, wordt alleen de business- of logic-layer aangepast. Zolang de data layer niet klaar is, worden simulatiefuncties gecreëerd.

Script commando’s

lijn

Functie: lijn(x, y, x2, y2, kleur, dikte)

Variabelen:

- x, y: Startpunt.
- x2, y2: Eindpunt.
- kleur: Naam van de kleur.
- dikte: Dikte in pixels.

Voorbeeld: lijn,0,0,50,50,rood,1

rechthoek

Functie: rechthoek(x_lup, y_lup, breedte, hoogte, kleur, gevuld)

Variabelen:

- x_lup, y_lup: Linker-bovenhoek positie.
- breedte, hoogte: Afmetingen van de rechthoek.
- kleur: Naam van de kleur.
- gevuld: 1 voor gevuld, 0 voor alleen een rand.

Voorbeeld: rechthoek,10,10,100,50,blauw,0

tekst

Functie: tekst(x, y, kleur, tekst, fontnaam, fontgrootte, fontstijl)

Variabelen:

- x, y: Positie op het scherm.
- kleur: Naam van de kleur.
- tekst: De string met de tekstinhoud.
- fontnaam: "arial" of "consolas".
- fontgrootte: 1 of 2.
- fontstijl: "normaal", "vet" of "cursief".

Voorbeeld: tekst,20,20,wit,Hallo,arial,1,normaal

cirkel

Functie: cirkel(x, y, radius, kleur)

Variabelen:

- x, y: Middelpunt van de cirkel.
- radius: De straal van de cirkel.
- kleur: Naam van de kleur.

Voorbeeld: cirkel,150,150,30,geel

figuur

Functie: figuur(x1, y1, x2, y2, x3, y3, x4, y4, x5, y5, kleur)

Variabelen:

- x1, y1 t/m x5, y5: Vijf afzonderlijke coördinatenpunten.
- kleur: Naam van de kleur.

Voorbeeld: figuur,0,0,10,0,10,10,0,10,5,5,groen

bitmap

Functie: bitmap(nr, x_lup, y_lup)

Variabelen:

- nr: Bitmap ID (0-3 voor pijlen, 4-5 voor smileys).
- x_lup, y_lup: Positie op het scherm.

Voorbeeld: bitmap,5,50,50

clearscherm

Functie: clearscherm(kleur)

Variabele:

- kleur: De kleur waarmee het hele scherm gevuld wordt.

Voorbeeld: clearscherm,zwart

wacht

Functie: wacht(msecs)

Variabele:

msecs: Aantal milliseconden om te wachten.

Voorbeeld: wacht,500

herhaal

Functie: herhaal(aantal, hoevaak)

Variabelen:

- aantal: Aantal voorgaande commando's om te herhalen.
- hoevaak: Hoe vaak deze reeks herhaald moet worden.

Voorbeeld: herhaal,2,10