Hand in number 2 – Integration Test

Af Gruppe 8:

Max Barly Jørgensen	201401694
Nicholas Ladefoged	201500609

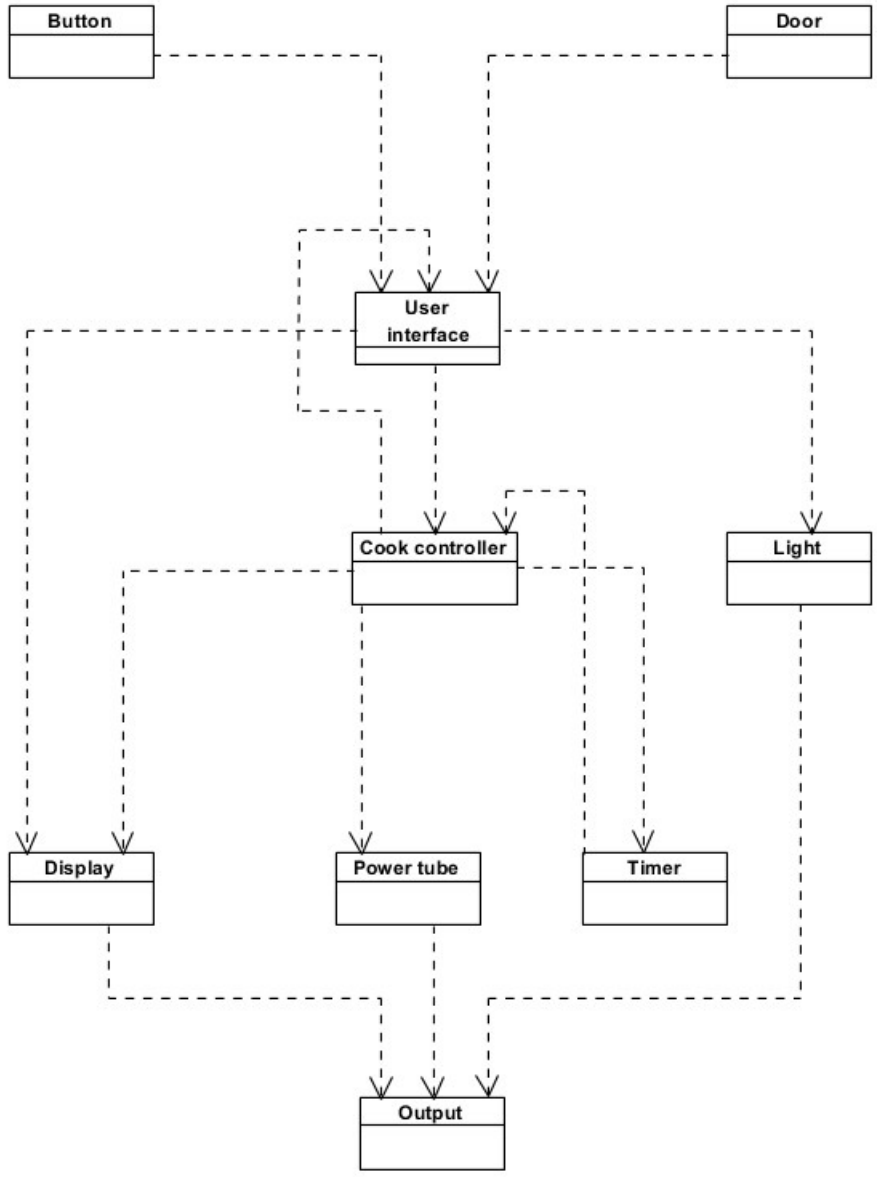
Jenkins Link: ci3.ase.au.dk:8080/job/TEAM8Handin2/

Github: https://github.com/nicho1991/SWTMicrowave

Indhold

Indledning:

Indledende vil vi se på dette dependency diagram som har til formål at give et overblik over hvordan microWave systemet er sat sammen, herefter vil vi beskrive hvordan vi har valgt at teste det. Vi har ud fra sekvensdiagrammet i opgaven analyseret os frem til dependency diagrammet i Figur 1.



Figur 1 Dependencys

Figur 1 Diagrammet har til formål at give et overblik over hvilke veje der skal testes når vi laver integrationstest. Dette diagram ligger altså grundlaget for hvilke strategier vi vælger når vi integrations tester.

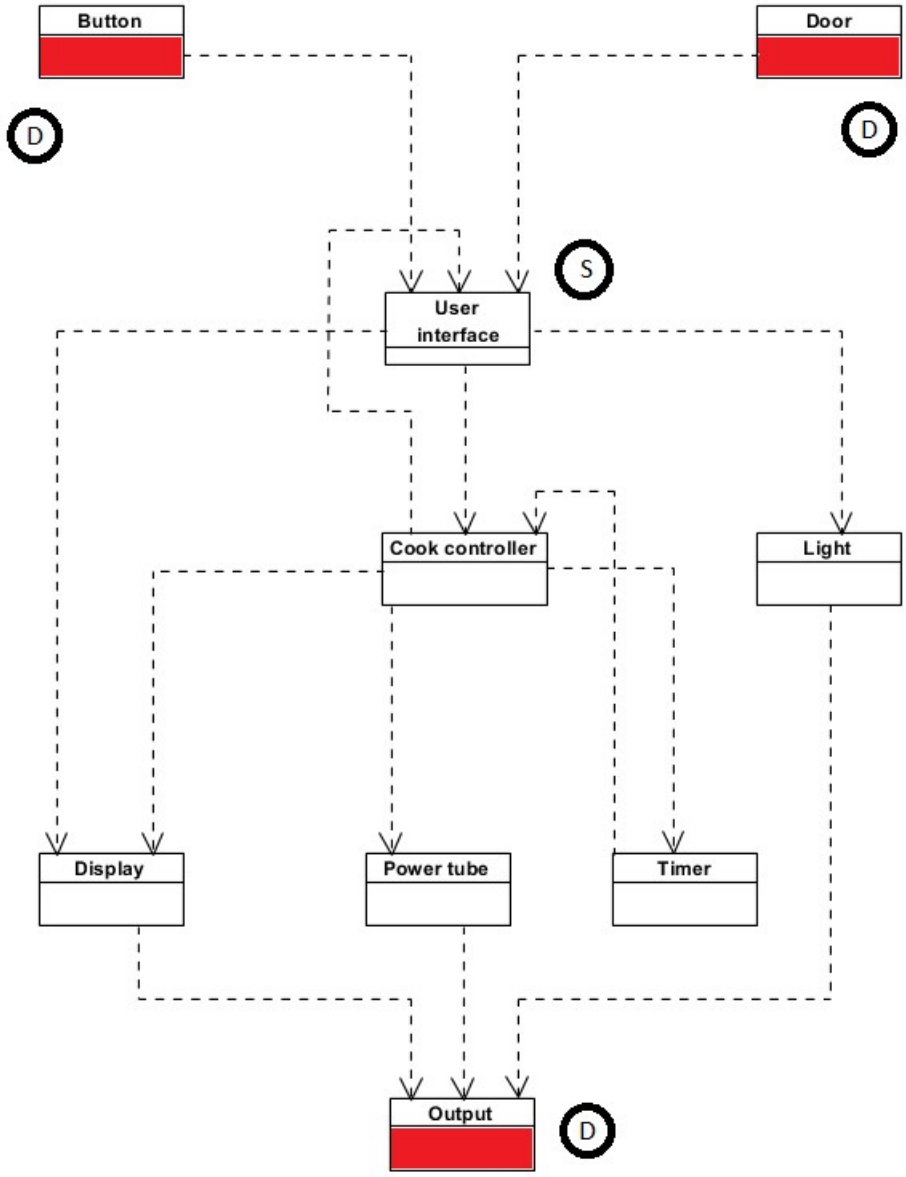
Integrations test:

Efter analyse af sekvensdiagrammet samt vores eget dependency diagram, blev der enighed om at en top-down model ville give mest mening for os i denne test situation.

Herfra har vi lavet test scenarier der skal følge disse diagrammer. Dvs følgende af hvert diagram, er grundlæggende for et testfixture, som er overordnet tag for en test klasse. Hver klasse er hermed navngivet i rækkefølge af diagrammerne (integrationsTest1, IntegrationsTest2 og IntegrationsTest3)

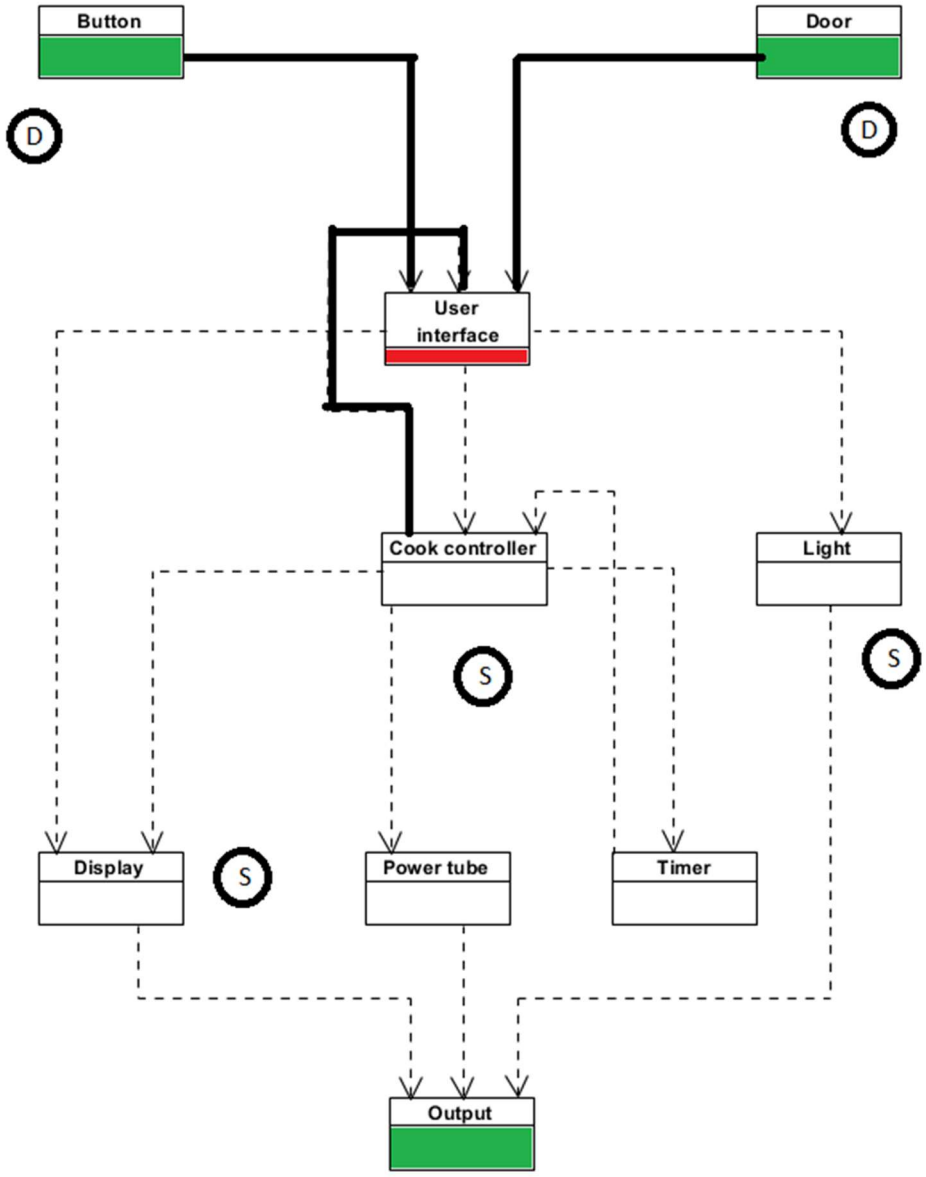
Til sidst skal det nævnes vi har fulgt navngivning af Stubs, drivers og IUT ifølge diagrammer fra vores undervisning. Følgende:

1. Røde blokke markere klasser under test
2. Grønne blokke markere klasser der er testet
3. D markere driveren af testen
4. S markere stub i testen, her er brugt Nsubstitute til at stubbe.
5. Tykke streger markere hvilket niveau der testes på i forhold til røde blokke



Figur 2 – Unit test, viser hvilke dele der allerede er testet

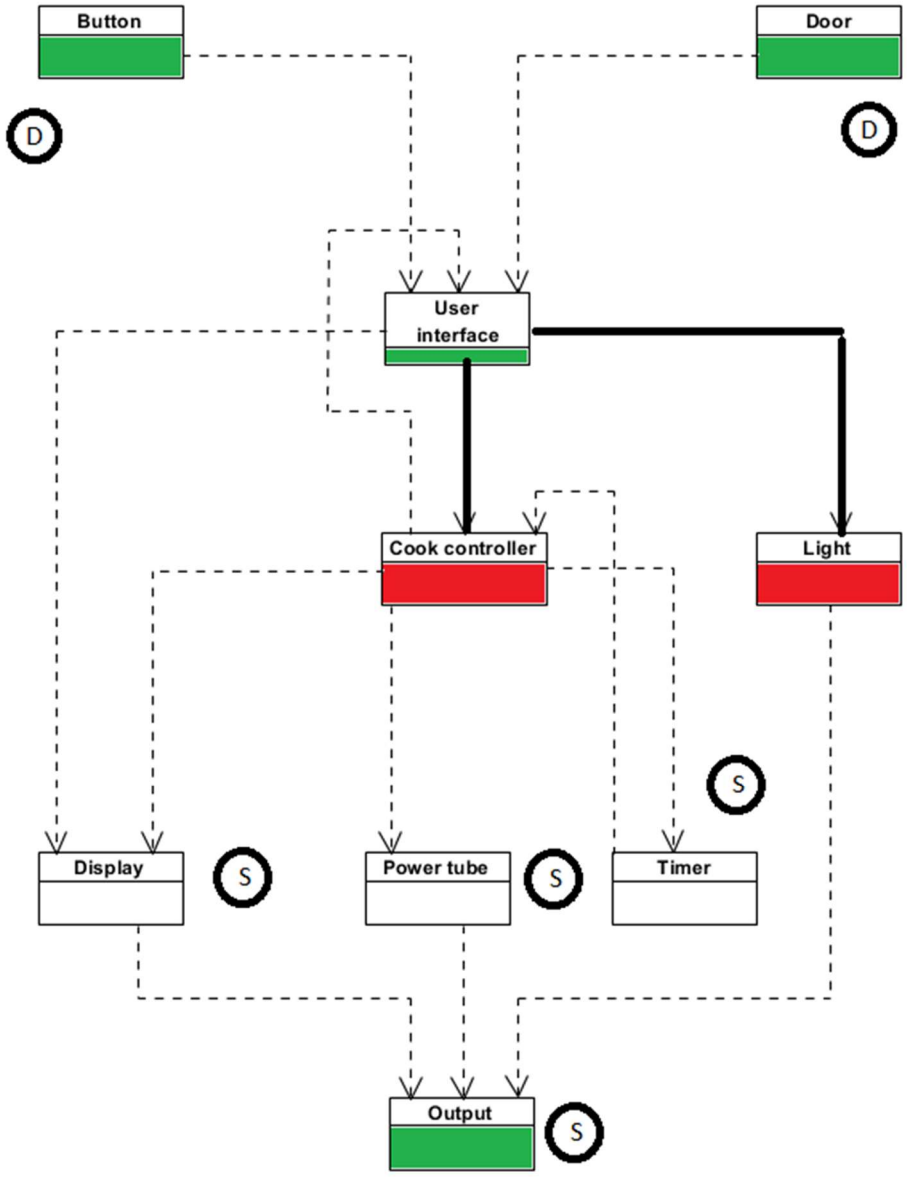
Figur 2 er ikke en del af integrations tests, men giver et overblik over hvad der allerede er testet, i vores unit tests, og derfor ikke skal være en del af integrations testene. Dette kommer bedre til udtryk når vi starter fra Figur 3, som beskriver den første integrations test



Figur 3-Integration test 1

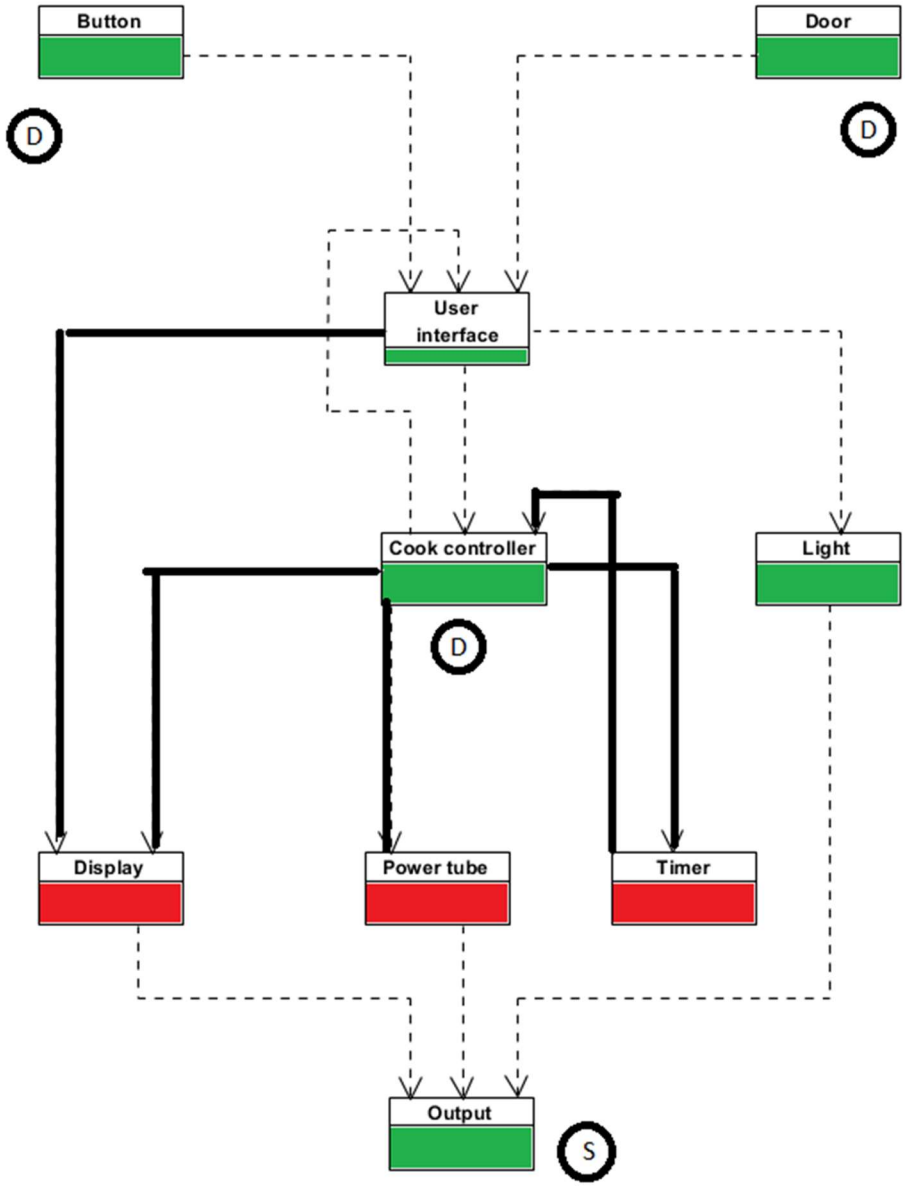
Figur 3 viser vores første integrations test. Da vi følger top-down startes der med at teste fra user interface (under test) vi starter her da både button og door allerede er unit testet.

Vejen vi tester er fra button og door ned til User interface, igen gældende da vi følger top-down. Det interresante her er nok at cookcontroller, som egentlig ligger i et lavere lag end UI, også testes her. Dette skyldes at UI og CookController har dependencies begge veje, og dermed, skal i dette scenarie er user interface afhængig af cook controller.



Figur 4- Integration test 2

Figur 4 er vores næste integrations test. Her ses forskellen fra før at cook controller nu er afhængig af user interfacet, vi tester altså den anden vej, og et lag længere nede. Light er også under test her , da dens afhængighed ligger i samme lag som cook controller. Bemærk her at vi allerede nu begynder at teste mod Output, selvom den ligger i nederste lag, dette skyldes outputs direkte afhængihed af Light(under test).



Figur 5 - Integration Test 3

Figur 5 er vores sidste integrations test, eftersom output ikke har har afhængiheder af den længere nede.

Denne test var omfattende da timers dobbelt afhængihed kan give vanskeligheder i forhold til at teste i et enkelt lag. Timers test, resulterede også i meget langsomme tests, da vi er nødt til at ”vente” på at timer er færdig i de konkrete scenarier vi tester.

Integrations Test plan:

Diagrammerne herunder

D: This module is included, and the one driven

X: This module is included

S: This modules is stubbed or mocked

Testing for Figur 3

Step #	Door	Button	UserInterface	CookControler	Timer	Light	Display	Power Tube	Output
1	D					X			
2		D	X				S		
3		D	X			S			
4		D	X	S		S	S		

Testing for Figur 4

Step #	Door	Button	UserInterface	CookControler	Timer	Light	Display	Power Tube	Output
1	D					X			S
2		D				X			S
3		D		X	S				
4		D		X	S			S	
5	D	D		X	S				
6	D	D		X				S	

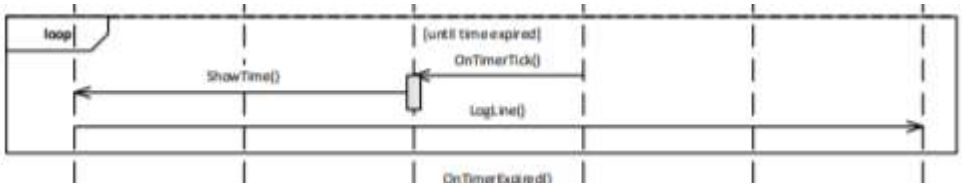
Testing for Figur 5

Step #	Door	Button	UserInterface	CookControler	Timer	Light	Display	Power Tube	Output
1				D	X				S
2	D	D			X		X		S
3	D	D			X				
4		D	X				X		S
5				D				X	S
6				D	X			X	S
7	D	D						X	S
8		D						X	S

Problemer:

Vores 3. unit test (se Figur 5) har givet os problemer i forbindelse med tråde der rendte ind over hinanden, derfor fejlede vores test tilfældigt. Vi har dog løst dette problem ved hjælp af mutex for hver test. Dette gør vores test lidt langsomere når vi tester vores Timer og behandling as responses. Tilgengæld er vi sluppet uden om at en test fejler.

Yderligere har vi haft problemer med 1 specifik test omkring timer i forhold til sekvensdiagrammet (se Figur 6) Testen i dette scenarie fejler. Tallene der skal vises på display hvert sekund, vises ikke korrekt. Vi tror det er et problem der ligger i koden, da outputtet simpelthen viser garbage.



Figur 6 problemet i timer test