C++ Fokus på sproget #03

in C++ Fokus with 0 Comments

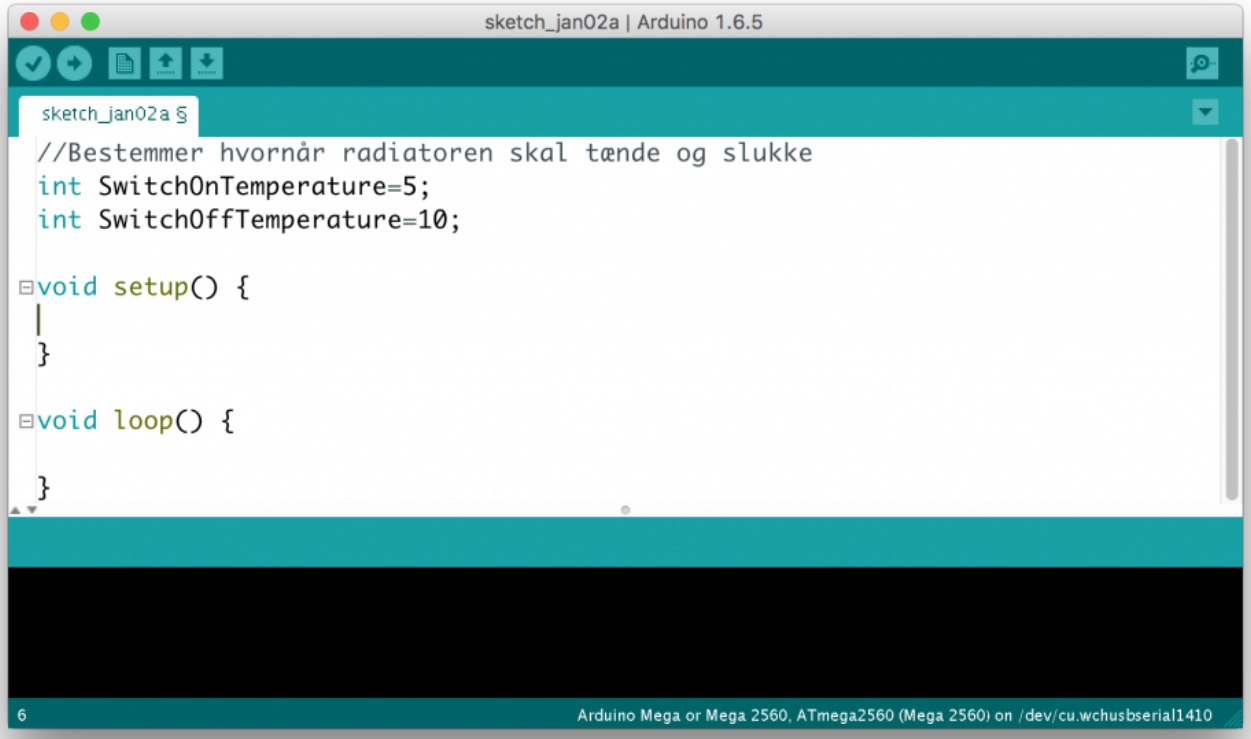
Variabler – navngivning og scope

Når man skriver kode, har man brug for noget, som kan opbevare forskellige informationer til videre forarbejdning og det er netop hvad der ligger i konstruktionen 'en variabel'. Man kan opfatte variabler som små bokse, hvori der kan gemmes en værdi, som kan være et tal eller noget tekst. Uden på hver boks sætter man en unik etiket, som skal hjælpe en til at huske, hvad netop denne boks anvendes til. Man giver med andre ord sine variabler navne og da programmet skal læses af mennesker, dig og andre som ønsker at anvende din kode, er det vigtigt at anvende meningsfulde navne. Når man opretter en variabel, skal man ud over navnet forholde sig til, hvilken type information den pågældende variabel skal lagre.

Vigtig info: Variabler er flygtige og bliver nulstillet for hver gang Arduino genstarter. Det engelske ord for det er 'volatile'. Det medfører, at man selv skal sørge for at lagre de variabler, hvis værdier man ønsker at bygge videre på ved en genstart. Det er nogen gange særdeles ønskværdigt, at ens Arduino starter op i samme tilstand som ved nedlukningen. Et eksempel på hvorledes dette kan håndteres bliver illustreret her: DDS AD9820 #01

Ønsker du eksempelvis at oprette (erklære) en variabel, som skal rumme en værdi for, hvornår der skal tændes for radiatoren, kan du vælge en integer-type. Hvis navnet skal være dækkende, kunne det f. eks. være 'SwitchOnTemperature'. Det er umiddelbart et noget langt navn, men til gengæld fortæller det hele historien. Og bruger man et IDE med auto completion, vil systemet selv foreslå netop dette navn, når blot man har skrevet de første bogstaver. Så med et passende IDE får man ikke skrivekrampe af at skulle skrive lange navne. 'SwitchOffTemperature' kunne så passende være navnet på den variabel, som rummer størrelsen af den temperatur, hvor der skal slukkes for radiatoren igen. Bemærk at C++ er case sensitive, hvilket vil sige, at der er forskel på store og små bogstaver. Hvis du skriver MyVariable og Myvariable opfatter C++ det som to forskellige variabler!

Når man opretter variable i C++ skal man altid huske at tildele dem en værdi. I visse programmeringssprog vil denne type variable altid som default få værdien 0, men sådan er det IKKE i C++. Derfor er det vigtigt, at man tildeler sine variabler en startværdi. De skal initialiseres, – det er god programmeringsstil. Hvis man undlader, får de ved fødslen tildelt en tilfældig værdi. Sker det eksempelvis, at man under opbygningen af sin sketch kommer til at kalde netop denne variabel FØR den har fået tildelt en meningsfuld værdi, kan man havne i nogle situationer, hvor det kan være vanskeligt at finde ud af, hvorfor tingene opfører sig sært. De to linjer, der skal være i sketchen, kan derfor se således ud:



Ved at benytte en blanding af store og små bogstaver bliver det endnu lettere (og hurtigere) at forstå koden. Denne måde at skrive navne på kaldes camelcase. Og brug nu engelske navne! Hvis du ikke er en ørn til engelsk, så få Google translate til at hjælpe dig. Selv et dårligt oversat navn på engelsk gør det markant lettere at søge råd hos andre programmører.

Så jeg skriver det lige igen: *Programmører taler engelsk, skriver engelsk og tænker på engelsk.*

Din Arduino er helt og aldeles ligeglad med, hvad du kalder dine variabler. Men hvis du nu f. eks. kalder disse to for henholdsvis 'x1' og 'x2' kan det sagtens tænkes, at det er rigeligt for dig til at identificere dem og huske, hvad de er for nogen. Men tro mig, hvis du af en eller anden grund kommer tilbage og skal arbejde med din kode nogle måneder senere, vil du ærgre dig over, at du ikke var omhyggelig med navngivningen. Det er også god skik at skrive kommentarer til sig selv (og andre) i sin kode. Det gør det*så* meget lettere at vende tilbage og arbejde videre. Og det gør det *så* meget lettere at få hjælp. Hvis du er god til engelsk, så skriv altid dine kommentarer på engelsk.

CamelCase i forbindelse med variabler er ikke obligatorisk, men det er en RIGTIG god ide at være systematisk i forhold til, hvorledes man navngiver og hvilken typografi, man anvender. Camelcase er mit forslag til, hvorledes man skriver variabelnavne. Hvis man ønsker at blive inspireret af andre programmører, er det blot at Google: C++ naming conventions. Vælg hvad du synes bedst om, men foretag et valg og vær konsekvent når du koder!

Læg mærke til at linjerne afsluttes med et semikolon. Hvis du glemmer det, får du en fejlmeddelelse, når du compilerer. I Arduino IDE compilerer du ved at trykke på 'Verify' og fejlmeldingen her vil lyde: *expected unqualified-id before numeric constant*

Ovenstående er en fejlmeddelelse, der ikke ligefrem er til stor hjælp i forhold til at finde den konkrete fejl. Så husk dine semikoloner!

Mange, der har beskæftiget sig med at skrive kode til Arduino, har fået en fejlmeddelelse i stil med:

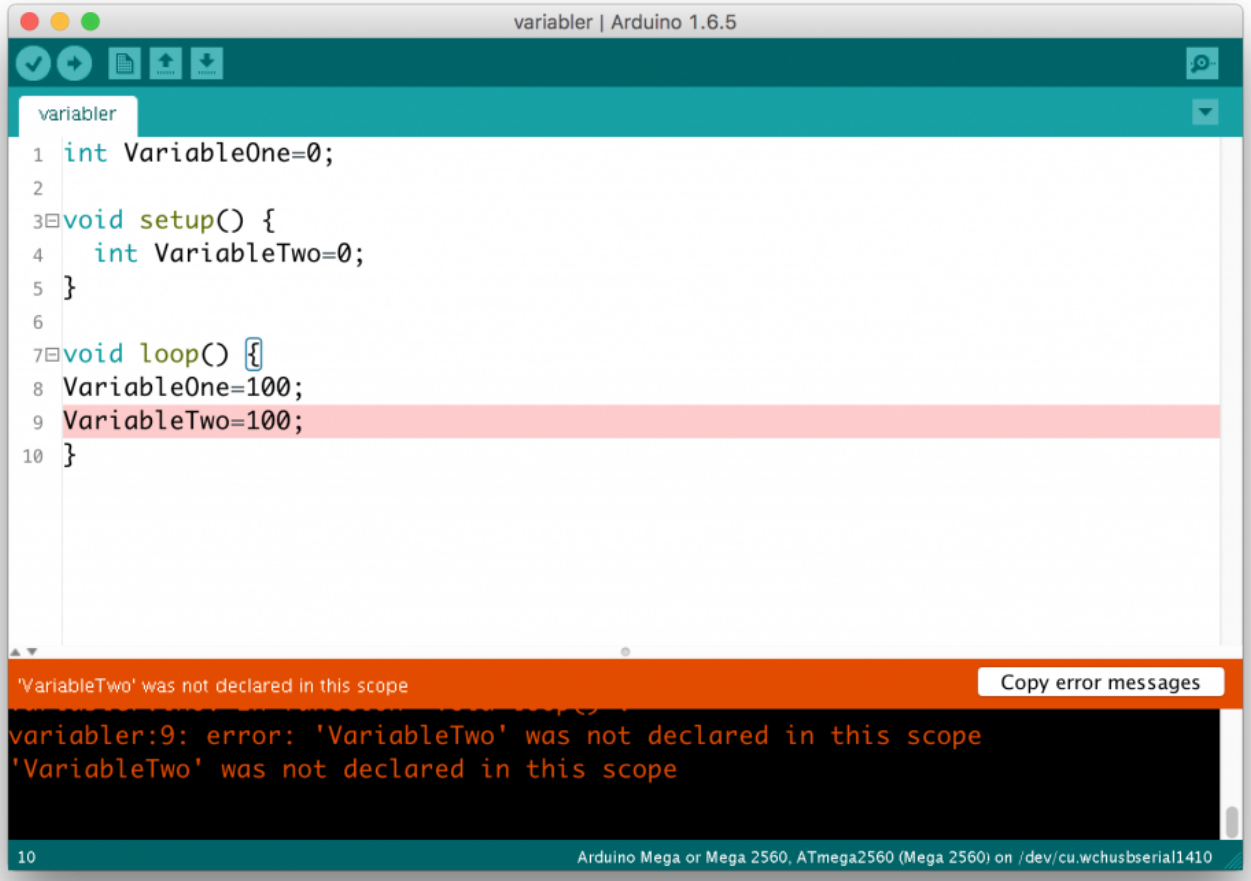
…… was not declared in this scope

Hvad er der på spil her? Oversat betyder det noget i retning af:

…… var ikke erklæret inden for dette anvendelsesområde.

Grundlæggende er det en melding om, at du har lavet et kald til en bestemt variabel på et sted i din sketch, hvor C++ ikke har kendskab til den. Den er så at sige ukendt for systemet. Det kan skyldes flere ting. Som tidligere nævnt er C++ case sensitive, så fejlmeldingen kan skyldes, at du har lavet en eller anden form for stavefejl. Så det kan altid betale sig at checke, om det er tilfældet.

Men det kan også handle om noget andet. Når du erklærer en variabel i toppen af sketchen som i ovenstående eksempel, bliver din variabel kendt i hele sketchen uanset hvor du befinder dig i koden. Vi siger at din variabel er global. Kig grundigt på de to erklæringer i nedenstående lille sketch:



VariableOne er erklæret i linje 1 som global variabel og man kan derfor godt lave et kald til den i loop().

VariableTwo derimod er erklæret i linje 4, som ligger INDEN I setup() og kan derfor kun ses INDE fra setup().

Med andre ord: Står man inde i loop() kan man IKKE se, hvad der befinder sig inde i setup(). Variablen er erklæret som lokal variabel og har dermed begrænset scope Det er god stil kun at erklære globale variable, hvor det vitterligt er nødvendigt at kunne tilgå variablen fra alle steder i sketchen.