# Struktureret tekst

TCP

TCP-programmet er vores modtage-program/TCPserver, der over en internet eller LAN forbindelse kan modtage et array fra en anden enhed. Dette array bliver sendt som en streng, men bliver lavet om så det modtages som et array (input) med 8 bit på hver plads. Selve koden er genbrug fra en opgave B&R har lavet med os. Dette betyder også, at selvom der er en del af koden der kan sende data, benyttes det ikke. Når strengen bliver modtaget af PLC’en lægges der et U, D, N eller Q ind på hver plads i arrayet, men da det er et USINT array lægges disse ind som decimaltal. Karaktererne i strengen bliver til nedenstående ascii kode.

Tallene bliver selvfølgelig sat ind i arrayet fra plads 0. Dette gør TCP’en for de pakker der modtages. Der er så det problem, at som programmet virker i version 5 kan der kun modtages 1 pakke på 64kb eller mindre, hvilket betyder hvis vores billedefil bliver større end 64kb vil den starte på pakke nummer 2, som den så lægger ind i arrayet fra plads 0 igen derfor begrænses arrayet til 65536 hvilket er en plads mere ned hvad en TCP pakke skal kunne håndtere. Dette gøres for at være sikker på, at det er hele pakke 1 der kommer med.

Af de variabler der bruges i denne del af programmet kan der ses en liste nedenfor:

|  |  |  |  |
| --- | --- | --- | --- |
| **Lokale variable** | **Datatype** | **Globale variable** | **Datatype** |
| tcp1 | TcpOpen | input | USINT[0..65535] |
| tcp2 | TcpServer |  |  |
| tcp3 | TcpRecv |  |  |
| tcp4 | TcpSend |  |  |
| tcp5 | TcpClose |  |  |
| state | UINT |  |  |

TcpOpen, TcpServer, TcpRecv, TcpSend og TcoClose er alle funktionsblokke fra biblioteket AsTCP.

Til sidst i programmet lukkes der for serveren for at der senere kan modtages ny data, uden at PLC’en skal genstartes. Dette kan gøres da serveren åbnes igen med det samme, men venter på at modtage ny data inden den lukker serveren igen.

Serveren bruger ip-adressen som PLC’en er blevet tildelt i konfigurationen. I dette setup bruges der en router så der er mulighed for at tilslutte flere enheder. Dette betyder så også, at der bliver brugt en ip ud fra routerens ip-range, i dette tilfælde bruges 192.168.0.103. der ud over bruges en port, som bliver defineret i programmet under variabel tcp1. Her bruges port 12345. Der kunne bruges en hel række andre porte, dog ikke porte der allerede er i brug som 5900 som bruges af visu-delen(Interfacet).

DRAW

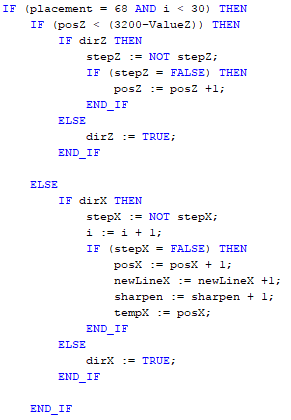
Draw programmet er den del der læser arrayet, og ud fra hvad der sker i arrayet ”input" sørger for at robotten bevæger sig.

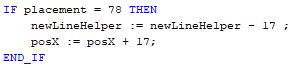
I programmet bruger vi en lang række variabler, som kan ses neden for.

|  |  |  |  |
| --- | --- | --- | --- |
| **Lokale variable** | **Datatype** | **Globale variable** | **Datatype** |
| Activator | BOOL | dirX | BOOL |
| tempX | UDINT | dirY | BOOL |
| Timer | TON | dirZ | BOOL |
| sharpenderLen | UINT | enabX | BOOL |
| I | UDINT | enabY | BOOL |
|  |  | enabZ | BOOL |
|  |  | input | USINT[0..65535] |
|  |  | posX | UDINT |
|  |  | posY | UDINT |
|  |  | posZ | UDINT |
|  |  | reset | BOOL |
|  |  | stepX | BOOL |
|  |  | stepY | BOOL |
|  |  | stepZ | BOOL |
|  |  | switch\_X | BOOL |
|  |  | sharpen | UINT |
|  |  | status | USINT |
|  |  | switch\_Z | BOOL |
|  |  | switch\_Y | BOOL |
|  |  | newLineX | UDINT |
|  |  | ValueZ | UINT |
|  |  | penLength | REAL |
|  |  | placeInArray | UDINT |
|  |  | placement | UINT |
|  |  | newLineHelper | UDINT |
|  |  | testEnab | BOOL |

Først i koden testes der, om robotten er blevet nulstillet ”reset”, og om der er blevet sat en længde på blyanten ”penLength”. Den tester også om variablen ”testEnab” er falsk, hvis alle disse er sande i forhold til det if-statement de ligger i, vil den gå ind og teste, om der er nogle af knapperne ude i akserne der er blevet trykket ned. Hvis dette er tilfældet stoppes robotten, da dette er et tegn på at der er en akse der er kørt for langt i en retning den ikke skulle køre i. Herefter sættes variablen ”status” da denne viser, i visuen, at programmet køre.

Efter alt dette bliver der testet om blyanten skal spidses på variablen ”sharpen”. Hvis variablen er nået til 30000 vil den flytte blyanten over og spidse denne. Dette gør den ved først at flytte blyanten op af z-aksen. Hver gang den tegner sætter den variablen ”tempX” som bruges for at robotten kan huske, hvor den er noget til på papiret. Dette udnyttes i den næste del, hvor robotten bevæger sig hen af X-aksen mod blyantspidseren. Da blyantspidseren er placeret under nulpunktet ved robotten også, hvor den skal køre hen af x-aksen for at spidse blyanten. Nå den så er nået til nulpunktet bliver blyanten sænket ned i spidserne. Hvor langt blyanten bliver sænket ned, kommer an på, hvor lang blyanten er sat i ”penLength”. Her bruges formlen som giver antallet af steps den skal gå ned, plus en værdi der er sat til 1500 steps fra 0 på z-aksen. Denne værdi er sat ud fra at det er den længst mulige blyant der er vurderet til at kunne blive spidset. Når blyanten har været nede i spidseren i 6 sekunder, hvilket er bestemt af variablen ”timer” vil den blive taget op af spidseren og bevæge sig tilbage til der, hvor den stoppede.

Hvis variablen ”sharpen” ikke er nået til 30000, vil den begynde at tegne. Her bruger den så arrayet ”input”. For at holde styr på hvor den er nået til i arrayet bruges variablen ”placeInArray” hvor den så lægger, hvad der er på den plads i arrayet, ind i variablen ”placement”. ”placeInArray” ændres hver gang variablen ”i” når 30, hvilket den gør at når der er taget 15 steps på enten x- eller y-aksen hvilket sker mens den tegner. Dette gøres for at kunne bestemme størrelsen af de pixels der tegnes. Grunden til hver pixel kun er 15 step store selvom ”i” er 30, er fordi stepper motorerne reagerer på et højt signal, altså når variablen for step bliver boolsk lav, på grund af transistorende som invertere signalet. I programmet bruges ”i” kun sammen med variablerne ”stepX” og ”stepY” da der tegnes i 2 dimensioner.

For at tegne pixels bruges ”U” og ”D” som i ascii er 85 og 68. ”U” betyder at der ikke skal tegnes, mens ”D” betyder at der skal tegnes. Så det første der sker er, at der bliver testet om blyanten er nede eller oppe. Hvis højden ikke stemmer overens med hvad den plads i array den er noget til siger den skal være, altså ”U” eller ”D” vil den først bevæge sig af z-aksen, enten op eller ned, hvorefter den vil bevæge sig af x-aksen for en pixel altså 15 steps.

Y-aksen bevæger sig kun når ”placement” er blevet ascii for ”N”, altså 78, hvilket betyder, at der skal laves en ny linje. Den starter med at z-aksen køre op til 2900 minus variablen ”ValueZ” (ValueZ er sat ud fra penLength igennem ligningen denne virker desværre ikke med alle længder så der skal findes en anden konstant.) efter blyanten er rykket op bevæger robotten sig tilbage af x-aksen ud fra variablerne ”newLineX” og ”newLineHelper” disse 2 variabler sørger for, at hver linje starter det samme sted. Dette gøres bl.a. ved at trække 17 steps fra ”newLineHelper”, hver gang der laves en ny linje da motorerne springer ca. 17 steps over ved hver ny linje derfor lægges der 17 til variablen ”posX” igen for at kompensere for motorerne. På grund af alt dette starter både ”newLineX” og ”newLineHelper” meget højt, så der ikke laves et overflow.

Nå den så er flyttet tilbage af x-aksen rykker den y-aksen 1 pixel altså 15 steps ved hjælp af ”i”.