Implementering af indstillinger

Indstilleren er en klasse som holder styr på systems indstillinger den bliver også omtalt som konfiguration fil nogle steder. Den skal holde styr på tiden i systemet og facitterne de virtuelle planter som der er i drivhuset, samt at hvilke hardware der må bruges til at regulere med.

Tiden kunne implementeres på et par måder. Der kunne laves en tæller hvor der tælles op hvert sekund. Denne løsning er der var blev dog ikke valgt fordi vi anvender en embedded linux platform, som selv kan holde styr på tiden hvis den kan indstilles rigtigt. Der er dog nogle krav til at indstille tiden på den embeddede linux platform, når man skal sætte den skal man ha super bruger rettigheder altså den skal sættes mens root kontoen anvendes.

Et snippet af koden som anvendes til at sætte tiden med ses her under.

string prepCommand;

std::string pwd = exec("pwd"); // get current path

//looks for the return char and removes it

if (!pwd.empty() && pwd[pwd.length()-1] == '\n') {

pwd.erase(pwd.length()-1);

}

//concatenate string.

prepCommand += pwd + string("/setTimeBeagleBoard.sh ") + to\_string(time.Year) + string(" ") +

to\_string(time.Month) + string(" ") + to\_string(time.Day) + string(" ") + to\_string(time.Hour) +

string(" ") + to\_string(time.Min);

const char \* command = prepCommand.c\_str();

std::string res = exec(command);

Det som koden gør oprette en pipe til linuxs commandline også kører kommandoen pwd på linux platformen som så retunere den sti som programmet køres i. Den sti bruges til at eksekvere et script med de ønskede tids parameteren så tiden sættes. Det script som koden afvikler bruger kommandoen Date som kan sætte tiden eller vise den.

Det som der gemmer sig under methoden exec er det som får platformen til at kører kommandoer i linux shellen. Her er methoden af koden der får det til at ske.

string exec(const char\* cmd) {

//Make a linux shell with read

FILE\* pipe = popen(cmd, "r");

if (!pipe) return "ERROR"; //Check if i got pipe

char buffer[256];

string result = "";

while(!feof(pipe)) { //While output read it.

if(fgets(buffer, 128, pipe) != NULL)

result += buffer;

}

pclose(pipe); //Close the shell

return result;

}

Som parameter tager den et char array som indeholder den kommando som der ønskes afviklet på systemet. Popen laver en proces ved at kalde lave pipe i read mode så vi får output tilbage fra kommando vi lige har kørt.

Når tiden kan sættes til den tid vi ønsker på platformen gør det, det meget nemmere at læse tiden ud fra systemet igen, Her er et andet kode snippet der viser hvordan tiden hentes ud fra platformen igen.

time\_t mytime = time(0);

struct tm\* tm\_ptr = localtime(&mytime);

struct Date date;

if (tm\_ptr)

{

date.Min = tm\_ptr->tm\_min;

date.Hour = tm\_ptr->tm\_hour;

date.Day = tm\_ptr->tm\_mday;

date.Month = tm\_ptr->tm\_mon+1;

date.Year = tm\_ptr->tm\_year+1900;

return date;

}

return date;

I projektet er der valgt at recensenter tid med en struct der indeholder år, måneder, dage, timer, og minutter. Der oprettes en struct tm og vores egen struct som udfyldes med den nuværene tid.