```
  1: /*
  2:  * Server.cpp
  3:  *
  4:  * Created on: 22-feb.-2016
  5:  *     Author: david12
  6:  */
  7:
  8: #include "Server.h"
  9: #include <iostream>
 10: #include <stdio.h>
 11: #include <stdlib.h>
 12: #include <string.h>
 13: #include <unistd.h>
 14: #include <sys/types.h>
 15: #include <sys/socket.h>
 16: #include <netinet/in.h>
 17: #include <sstream>
 18: #include <string>
 19:
 20:
 21: int sockfd, newsockfd, portno;
 22:  socklen_t clilen;
 23:  char buffer[16];
 24:  struct sockaddr_in serv_addr, cli_addr;
 25:  int n;
 26:
 27: void error(const char *msg)
 28: {
 29:             std::cout <<   "ERROR: NETWERK VERBINDING"
 30:     perror(msg);
 31: }
 32:
 33: Server::Server(int port) {
 34:         sockfd = socket(AF_INET, SOCK_STREAM, 0);
 35:              if (sockfd < 0)
 36:              error(             "ERROR opening socket");
 37:         bzero((        char *) &serv_addr,     sizeof(serv_addr));
 38:         portno = atoi(          "2259");
 39:         serv_addr.sin_family = AF_INET;
 40:         serv_addr.sin_addr.s_addr = INADDR_ANY;
 41:         serv_addr.sin_port = htons(portno);
 42:              if (bind(sockfd, (     struct sockaddr *) &serv_addr,
 43:                                      sizeof(serv_addr)) < 0)
 44:                      error(                  "ERROR on binding");
 45:         listen(sockfd,5);
 46:         clilen =           sizeof(cli_addr);
 47:             std::cout <<   "INFO:Server started @ "  << port << std::endl;
 48:     Loop();
 49:         newsockfd = accept(sockfd,
 50:                     (                 struct sockaddr *) &cli_addr,
 51:                     &clilen);
 52: }
 53:
 54: void* Server::Loop(){ /*
 55:       for(;;)for(;;){
 56:             newsockfd = accept(sockfd,
 57:                              (struct sockaddr *) &cli_addr,
 58:                              &clilen);
 59:             if (newsockfd < 0)
 60:                   error("ERROR on accept");
 61:             bzero(buffer,16);
 62:             n = read(newsockfd,buffer,15);
 63:             std::cout << n;
 64:             if(n<0){
 65:                   error("Voledig berig is niet doorgekomen");
 66:             }
 67:             for(int a=0;a!=16;a++){
 68:                   std::cout << buffer[a] << std::endl;
 69:                   std::cout << ((int)buffer[a])<< std::endl;
 70:             }
 71:       }*/
 72: }
 73:
 74: Inst*   Server::Continue(Inst* out,InputData* in){      /*
 75:     bzero(buffer,16);
 76:       n = read(newsockfd,buffer,15);
 77:
 78:       if(buffer[0]==1){ // recuest memory bit
 79:             switch(buffer[1]){ // select memory bit
 80:                   case 1:{
 81:                   char sendBuff[1024];
 82:                     memset(sendBuff, '\0', sizeof(sendBuff));
 83:                     sprintf(sendBuff, "This is line ");
 84:                     std::cout << send(newsockfd,sendBuff,strlen(sendBuff),0) << std::endl;
 85:
 86:                   }
 87:             }
 88:       }
 89:
 90:     //n = write(newsockfd,"I got your message\n",18);
 91:
 92: */
 93:             return out;
 94: }
 95:
 96: Server::~Server() {
 97:     close(newsockfd);
 98:     close(sockfd);
 99: }
100:
```

```
 1: /*
 2:  * Menu.cpp
 3:  *
 4:  *  Created on: 2-jan.-2016
 5:  *      Author: david12
 6:  */
 7:
 8: #include "Menu.h"
 9:
10: int oldbutton=0;
11: int posi=0;
12:
13: int submenu=0;
14: /*
15:  * 0:
16:  */
17:
18: Menu::Menu() {
19:         // TODO Auto-generated constructor stub
20:
21: }
22:
23: int Menu::Loop(int buttonPos){
24:         if (buttonPos!=oldbutton){
25:                                                 if (buttonPos==3){
26:                                 posi++;
27:                                                         if (posi>2)
28:                                         posi=0;
29:                                 }                       else if (buttonPos==4){
30:                                         posi--;
31:                                                         if (posi<0)
32:                                         posi=2;
33:                                 }                       else if (buttonPos==5){
34:                                                         if (posi==0)
35:                                                                 return -2;
36:                                                         if (posi==2)
37:                                                                 return -1;
38:                                 }
39:                         }
40:     oldbutton=buttonPos;
41:         return posi;
42: }
43:
44: Menu::~Menu() {
45:         // TODO Auto-generated destructor stub
46: }
47:
```

```
 1: /*
 2:  * Functions.cpp
 3:  *
 4:  *  Created on: 11-mrt.-2016
 5:  *      Author: david12
 6:  */
 7:
 8: #include  "Functions.h"
 9:
10: Functions::Functions () {
11:             // TODO Auto-generated constructor stub
12:
13: }
14:
15: std::string Functions::GetIp(){
16:
17: }
18:
19: Functions::~Functions () {
20:             // TODO Auto-generated destructor stub
21: }
22:
```

```
 1: /*
 2:  * Inst.cpp
 3:  *
 4:  *  Created on: 1-jan.-2016
 5:  *      Author: david12
 6:  */
 7:
 8: #include "Inst.h"
 9:
10: int ScreenPos;
11: bool NeetsArming=false;
12:
13: int MotorPos[]={0,0,0,0,0,0,0,0};
14:
15: Inst::Inst () {
16:             // TODO Auto-generated constructor stub
17:
18: }
19:
20: void Inst::SetScreenPos(int i){
21:         ScreenPos=i;
22: }
23:
24: int Inst::GetScreenPos(){
25:             return ScreenPos;
26: }
27:
28: bool Inst::GetNeedsArming(){
29:             return NeetsArming;
30: }
31:
32: void Inst::SetNeedsArming(bool a){
33:         NeetsArming=a;
34: }
35:
36: void Inst::SetMotorPosition (int Motor, int Pos){
37:         MotorPos[Motor]=Pos;
38: }
39:
40: int   Inst::GetMotorPosition (int Motor){
41:             return MotorPos[Motor];
42: }
43:
44: Inst::~Inst () {
45:             // TODO Auto-generated destructor stub
46: }
47:
```

```
 1: /*
 2:  * Verwerking.cpp
 3:  *
 4:  *  Created on: 26-dec.-2015
 5:  *      Author: david12
 6:  */
 7:
 8: #include <iostream>
 9: #include "Verwerking.h"
10: #include "../Input/Data/InputData.h"
11: #include "Instructies/Inst.h"
12: #include "Menu/Menu.h"
13: #include "Server/Server.h"
14:
15: Menu*   menu;
16: Server* server;
17:
18: int tmp;
19: int state=1;
20: /*
21:  * 1:menu
22:  * 2:in progam
23:  * 3:manual mode
24:  */
25: int countdown=100;
26:
27: bool running=true;
28:
29: Verwerking::Verwerking() {
30:       menu=   new Menu();
31:       server=     new Server(1234);
32: }
33:
34: bool Verwerking::Running(){
35:           return running;
36: }
37:
38: Inst*  Verwerking::loop(InputData* data){
39:       Inst* instrukties=          new Inst();
40:           switch (state){
41:                   case 1: // in menu
42:                   tmp=menu->Loop(data->GetButtonState());
43:                           if (tmp==-1)
44:                       running=false;
45:                           else if (tmp==-2)
46:                       state=2;
47:                           else
48:                       instrukties->SetScreenPos(tmp);
49:
50:                           if (data->GetArm() && data->GetCon()){
51:                       countdown--;
52:                               if (countdown<0){
53:                           countdown=100;
54:                           state=3;
55:                                   std::cout <<  "INFO:Systeem word NU BEWAPEND (Door JoyStick)" << std::endl;
56:                       }
57:                   }
58:                           else
59:                       countdown=100;                  // 5 seconden
60:
61:                   instrukties->SetScreenPos(4);
62:                   instrukties->SetNeedsArming(false);
63:                     break;
64:                     case 2:
65:                   instrukties->SetScreenPos(3);                   // schow running progam emblem
66:                     break;
67:                     case 3:
68:                           if (data->GetArm()){
69:                       instrukties->SetScreenPos(5);
70:                       instrukties->SetNeedsArming(true);
71:                       instrukties->SetMotorPosition(1,90-data->GetJoystickState1().x*30+data->GetJoystickState1().y*30);
72:                       instrukties->SetMotorPosition(0,90+data->GetJoystickState1().x*30+data->GetJoystickState1().y*30);
73:                       instrukties->SetMotorPosition(2,90-data->GetJoystickState1().x*30-data->GetJoystickState1().y*30);
74:                       instrukties->SetMotorPosition(3,90+data->GetJoystickState1().x*30-data->GetJoystickState1().y*30);
75:                   }           else{
76:                       state=1;
77:                       instrukties->SetScreenPos(4);
78:                       instrukties->SetNeedsArming(false);
79:                               std::cout <<  "INFO:Systeem word NU ONTWAPEND (Door JoyStick)" std::endl;
80:                   }
81:                     break;
82:       }
83:
84:       instrukties=server->Continue(instrukties,data);
85:
86:           return instrukties;
87: }
88:
89: Verwerking::~Verwerking() {
90:           delete menu;
91:           delete server;
92: }
93:
```

```
1: /*
2:  * MoveMentCalcuator.cpp
3:  *
4:  *  Created on: 28-feb.-2016
5:  *      Author: david12
6:  */
7:
8: #include "MoveMentCalcuator.h"
9:
10: MoveMentCalcuator::MoveMentCalcuator(){
11:         // TODO Auto-generated constructor stub
12:
13: }
14: MoPs MoveMentCalcuator::GetNextState(double time, float XRot, float YRot, float ZRot, float yaw){
15:
16:     MoPs motorPos;
17:
18:         for(bool a=false;a;a=true){
19:
20:                 float sinust=sin(time*M_PI);
21:
22:                 if (sinust>0){
23:                 motorPos.MO[0+a*2]=sinust*120+yaw*20*a;
24:                 motorPos.MO[1+a*2]=sinust*120+yaw*20*a;
25:
26:           }
27:                 else{
28:                 motorPos.MO[0+a*2]=0;
29:                 motorPos.MO[1+a*2]=0;
30:
31:           }
32:
33:                 if (sinust<0){
34:                 motorPos.MO[0+a*2]=-sinust*120+yaw*20*a;
35:                 motorPos.MO[1+a*2]=-sinust*120+yaw*20*a;
36:
37:           }
38:                 else{
39:                 motorPos.MO[0+a*2]=0;
40:                 motorPos.MO[1+a*2]=0;
41:
42:           }
43:     }
44:         return motorPos;
45:
46: }
47:
48: MoveMentCalcuator::~MoveMentCalcuator(){
49:         // TODO Auto-generated destructor stub
50: }
51:
```

```
 1: //============================================================================
 2: // Name        : K9.cpp
 3: // Author      : David Gorrebeeck
 4: // Version     : 0.1
 5: // Copyright   : GPL
 6: // Description : K9 software om hond te controleeren
 7: //============================================================================
 8:
 9: #include <iostream>                    // Importeer benoigtheden schrijf naar log
10: #include <sstream>                     // Importeer benodigtheden om van int naar string te gaan
11: #include <stdlib.h>
12: #include <unistd.h>
13: #include <cstdio>
14: #include <ctime>
15:
16: #include "Input/Input.h"               // importeer benodigtheden voor input
17: #include "Input/Data/InputData.h"      // importeer infohouder inputdata
18: #include "Output/Output.h"             // importeer benodigtheden voor output
19: #include "Verwerkingseenhijd/Verwerking.h"  // importeer benodigtheden voor verwerking
20: #include "Verwerkingseenhijd/Instructies/Inst.h"// importeer infohouder verwerkingseenhijd
21:
22: using namespace std;                                   // gebruik de standaart bibliotheek
23:
24: std::clock_t start;
25:
26: void wait(){
27:     double duration = (( std::clock() - start ) / ( double) CLOCKS_PER_SEC)*1000;
28:     if (duration>=50){
29:       cout<< "warning: tick took to long : " << duration << "ms"<< endl;
30:     }
31:     else{
32:       usleep((50.0-duration)*1000.0);
33:     }
34:     start=std::clock();
35:     return;
36: }
37:
38: int main() {                                           // START
39:     cout << "INFO:starten" << endl;                    // Shrijf naar log dat we starten
40:
41:     // aanmaken van merkers
42:
43:     Input* input;                                      // Maak merker aan waar de input naar gescreven moet worden
44:     Verwerking* verwerker;                             // Maak merker aan waar de verwerker in opgeslagen moet worden
45:     Output* output;                                    // maak merker aan waar de output moet opgeslagen worden
46:
47:
48:     InputData* Idata;
49:     Inst* Udata;
50:
51:     try {                                              // Vang op bij erinstige fout fout
52:       output= new Output();                            // Start verbinding met output
53:       input= new Input();                              // Start De inputgenerator op
54:       verwerker= new Verwerking();                     // laad een verwerkingseenhijd in
55:
56:       output->OnStartup();                             // Execute on startup
57:
58:       usleep(5000000);
59:
60:           std::clock_t start;                          // START hier
61:           while(verwerker->Running()){                 // blijf herhaalen tot running false word
62:
63:
64:          Idata=input->LoopIt();                        // openen van gegevens
65:          Udata=verwerker->loop(Idata);
66:          output->LoopIt(Udata);
67:
68:          wait();
69:       }
70:
71:       output->OnStop();                                // Zet alle output in stop posietie
72:
73:
74:     } catch ( int Error) {                             // Dit word uitgevoert vanaf dat er een Erensige fout is gevonden
75:       ostringstream Errorc;                            // maak merker aan waar foutcode als text in weergeven kan wordee
76:       Errorc << Error;                                 // zet code om in text
77:       cout << "FATAL : Fout opgetreden met code : " +Errorc.str();
78:     }                                                  // Zeg dat er een fout was en schrijf de code er
bij
79:
80:
81:     delete Idata;
82:     delete Udata;
83:
84:
85:     delete input;                                      // stop input module
86:     delete verwerker;                                  // stop verwerkings module
87:     delete output;                                     // stop output module
88:
89:     return 0;                                          // Stop progama met code 0 ( Gestopt geen probleem )
90: }                                                      // STOP
91:
```

```
 1: /*
 2:  * Input.cpp
 3:  *
 4:  *  Created on: 26-dec.-2015
 5:  *      Author: david12
 6:  */
 7:
 8: #include <iostream>
 9: #include <list>
10: #include "Input.h"
11:
12: #include "Joystick/Joystick.h"
13: #include "Data/InputData.h"
14: #include "d/cJoystick1.h"
15:
16: Joystick* joystick;
17: cJoystick* cj;
18:
19:
20: Input::Input() {                                                    // Start bij aanmaken
21:         std::cout <<  "INFO:Starten van Sensoren"
22:                 << std::endl;                                               // zeg dat we gaan beginen
23:
24:     joystick=       new Joystick();
25:     cj =        new cJoystick();
26:
27:         std::cout <<  "INFO:Sensoren gestart."
28:                 << std::endl;                                       // zeg dat de sensoren gestart zijn.
29:         return;
30: }                                                                               // STOP
31:
32: InputData*  Input::LoopIt(){
33:     InputData* ID=      new InputData();
34:
35:     ID->SetButtonState(joystick->GetPos());
36:
37:     ID->SetJoystickState(cj->joystickPosition(0).x,cj->joystickPosition(0).y,
38:                                     cj->joystickPosition(1).x,cj->joystickPosition(1).y,
39:                                     cj->buttonPressed(0),cj->buttonPressed(1),cj->buttonPressed(2),
40:                                     cj->buttonPressed(3),cj->buttonPressed(4));
41:
42:         return ID;
43: }
44:
45:
46:
47: Input::~Input() {
48:         delete joystick;
49:         delete cj;
50: }
51:
```

```
 1: /*
 2:  * cJoystick1.cpp
 3:  *
 4:  * Created on: 17-feb.-2016
 5:  *     Author: david12
 6:  */
 7:
 8: #include "cJoystick1.h"                                    // Haal te doen lijst op
 9:
10:     #include <pthread.h>                                                    // Zorg ervoor dat je meerdere dingen teglijk kan
11:     #include <string.h>                                                     // Werk met stukken tekst
12:     #include <stdio.h>                                                      // Gebruik standaart I/O bibliotheek
13:     #include <stdlib.h>                                                     // Gebruik statndaart bibliotheek
14:     #include <unistd.h>                                                     // Maak gebruik van Unsignt INT's
15:     #include <errno.h>                                                      // Maak weg vrij voor fautschrijven
16:     #include <ctype.h>                                                      // Werk ook in C
17:
18: cJoystick::cJoystick () {                                                  // Bij opstarten
19:     active = false;                                                                 // zet actief op false
20:     joystick_fd = 0;                                                                // zet dev nummer op null
21:     joystick_ev =     new js_event();                                       // sla joystick zaken op
22:     joystick_st =     new joystick_statee();                                // Maak joystick input klaar
23:     joystick_fd = open(JOYSTICK_DEV, O_RDONLY | O_NONBLOCK);    // Verbind Joystick
24:         if (joystick_fd > 0) {                                              // kontroleer op fauten.
25:           ioctl(joystick_fd, JSIOCGNAME(256), name);                        // Haal joystick naam op
26:           ioctl(joystick_fd, JSIOCGVERSION, &version);                  // Haal jaystick versie op
27:           ioctl(joystick_fd, JSIOCGAXES, &axes);                            // Haal hoeveelhijd assen op
28:           ioctl(joystick_fd, JSIOCGBUTTONS, &buttons);                  // haal hoeveelhijd knoppen op
29:                 std::cout <<  "   Name: "  << name << std::endl;          // zeg naam
30:                 std::cout <<  "Version: "  << version << std::endl;       // zeg vertie
31:                 std::cout <<  "   Axes: "  << (int )axes << std::endl;    // zeg aantal assen
32:                 std::cout <<  "Buttons: "  << (int )buttons << std::endl;  // zeg knoppen
33:           joystick_st->axis.reserve(axes);                                  // koppel assen aan
34:           joystick_st->button.reserve(buttons);                             // koppel knoppen aan
35:           active = true;                                                            // zet actief aan
36:           pthread_create(&thread, 0, &cJoystick::loop, this );          // start controle draat op
37:     }
38: }
39:
40: cJoystick::~cJoystick () {                                                  // als er afgesloten wordt
41:         if (joystick_fd > 0) {                                              // als joystick aangesloten is
42:           active = false;                                                           // zet actief uit
43:           pthread_join(thread, 0);                                          // wacht tot draad klaar is
44:           close(joystick_fd);                                                       // sluit verbing joystick
45:     }
46:         delete joystick_st;                                                         // verwijder stuk geheugen voor joystick
47:         delete joystick_ev;                                                         // verwijder stuk geheugen voor joystick
48:     joystick_fd = 0;                                                                // zet geheugen hier op null
49: }
50:
51: void* cJoystick::loop (void *obj) {                                        // doe op aparte draad
52:         while (reinterpret_cast<cJoystick *>(obj)->active)          // blijf doen tot afsluiten gevraagt
53:           reinterpret_cast<cJoystick *>(obj)->readEv();             // haal nieuwe bewegingen op
54: }
55:
56: void cJoystick::readEv () {
57:         int bytes = read(joystick_fd, joystick_ev,      sizeof (*joystick_ev));     // lees gebeurtenis
58:         if (bytes > 0) {                                                                        // als gebeurtenis echt is
59:           joystick_ev->type &= ~JS_EVENT_INIT;                                  // krijg soort gebeurtenis
60:                 if (joystick_ev->type & JS_EVENT_BUTTON) {                      // is knop?
61:               joystick_st->button[joystick_ev->number] = joystick_ev->value;
62:           }                                                                                             // / / Zet knop juist
63:                 if (joystick_ev->type & JS_EVENT_AXIS) {                        // is as?
64:               joystick_st->axis[joystick_ev->number] = joystick_ev->value;      // Zet as juist
65:           }
66:     }
67: }
68:
69: joystick_positione      cJoystick::joystickPosition   (int n) {       // vraag positie joystick - as
70:     joystick_positione pos;                                                     // maak geheugen vrij
71:
72:         if (n > -1 && n < axes) {                                               // als as niet te klijn is
73:                 int i0 = n*2, i1 = n*2+1;                                       // doe wiskunde om te bepalen waar
74:                 float x0 = joystick_st->axis[i0]/32767.0f,                      // de joystick is
75:                     y0 = -joystick_st->axis[i1]/32767.0f;                       //
76:                 float x  = x0 * sqrt(1 - pow(y0, 2)/2.0f),
77:                     y  = y0 * sqrt(1 - pow(x0, 2)/2.0f);
78:
79:           pos.x = x0;
80:           pos.y = y0;
81:
82:           pos.theta = atan2(y, x);
83:           pos.r = sqrt(pow(y, 2) + pow(x, 2));
84:     }    else {
85:           pos.theta = pos.r = pos.x = pos.y = 0.0f;                          // bevestig vars
86:     }
87:         return pos;                                                                     // zet terug
88: }
89:
90: bool cJoystick::buttonPressed (int n) {                                    //vraag knop op
91:         return n > -1 && n < buttons ? joystick_st->button[n] : 0;      //geef knop terug
92: }
```

```
 1: /*
 2:  * InputData.cpp
 3:  *
 4:  *  Created on: 26-dec.-2015
 5:  *      Author: david12
 6:  */
 7:
 8: #include "InputData.h"              // haal te doen lijst op
 9:
10: int ButtonPos;                                  // maak byte vrij voor knop posietie
11:
12: double Temperature;                             // maak coma getal vrij
13: double Vogtighijd;                              // maak coma getal vrij
14: double luchtdruk;                               // maak coma getal vrij
15:
16: joystick_positions PosJoy1;                     // sla joystick positie op
17: joystick_positions PosJoy2;                     // sla joystick positie op
18:
19: bool arming;                                    // maak bits aan
20: bool sta;                                       //
21: bool zit;                                       //
22: bool lig;                                       //
23: bool continu;                                   //
24:
25: InputData::InputData() {
26:
27: }
28:
29: InputData::~InputData() {
30: }
31:
32: // Getters and setters
33: int  InputData::GetButtonState(){
34:         return ButtonPos;
35: }
36:
37: void  InputData::SetButtonState(int  a){
38:       ButtonPos=a;
39: }
40:
41:
42: joystick_positions      InputData::GetJoystickState1(){
43:         return PosJoy1;
44: }
45:
46: joystick_positions      InputData::GetJoystickState2(){
47:         return PosJoy2;
48: }
49:
50: bool  InputData::GetArm(){
51:         return arming;
52: }
53:
54: bool  InputData::GetCon(){
55:         return continu;
56: }
57:
58: void  InputData::SetJoystickState (float  a1, float  a2,
59:                                                                 float  b1, float  b2,
60:                                                                 bool c1, bool c2, bool c3, bool c4, bool c5){
61:       joystick_positions d;
62:       d.x=a1;
63:       d.y=a2;
64:       PosJoy1=d;
65:
66:       d.x=b1;
67:       d.y=b2;
68:       PosJoy2=d;
69:
70:       arming=c1;
71:       sta=c2;
72:       zit=c3;
73:       lig=c4;
74:       continu=c5;
75: }
```

```
1: //============================================================================
2: // Name        : Joystick.cpp
3: // Author      : David Gorrebeeck
4: // Version     : 0.1
5: // Copyright   : GPL
6: // Description : K9 software om hond te controleeren
7: //============================================================================
8:
9: #include "Joystick.h"
10: #include <iostream>
11:
12: #include <stdio.h>
13: #include <stdlib.h>
14: #include <string.h>
15: #include <unistd.h>
16: #include <errno.h>
17: #include <fcntl.h>
18: #include <dirent.h>
19: #include <linux/input.h>
20: #include <sys/types.h>
21: #include <sys/stat.h>
22: #include <sys/select.h>
23: #include <sys/time.h>
24: #include <termios.h>
25: #include <signal.h>
26: #include <thread>
27:
28: int pos;
29: int ne=100;
30: // niets   0
31: // up      1
32: // down    2
33: // lings   3
34: // rechts  4
35: // enter   5
36:
37: struct input_event ev[64];
38: int fevdev = -1;
39: int result = 0;
40: int size =  sizeof (struct input_event);
41: int rd;
42: int value;
43: char name[256] = "Unknown";
44: char *device =  "/dev/input/event0";
45:
46:
47: void Scan(){
48:          for(;;){
49:            rd = read(fevdev, ev, size * 64);
50:
51:                  switch(ev[0].code){                                    // selecteer toetscode
52:                        case 106:
53:                    pos = 1;
54:                        break;
55:                        case 105:
56:                    pos =  2;
57:                        break;
58:                        case 108:
59:                    pos =  3;
60:                        break;
61:                        case 103:
62:                    pos =  4;
63:                        break;
64:                        case 28:
65:                    pos =  5;
66:                        break;
67:                        default :
68:                    pos =  0;
69:            }
70:            ne=100;
71:      }
72: }
73:
74: void setZero(){
75:          for(;;){
76:            usleep(2500);
77:            ne--;
78:                  if (ne<0)
79:                pos=0;
80:      }
81: }
82:
83: Joystick::Joystick () {
84:          std::cout <<  "INFO:aankoppelen van joystick"
85:                  << std::endl;                                          // zeg dat we gaan beginen
86:
87:    fevdev = open(device, O_RDONLY);
88:       if (fevdev == -1) {
89:        printf(       "ERROR:Failed to open event device.\n");
90:          throw 102;
91:    }
92:
93:    result = ioctl(fevdev, EVIOCGNAME(       sizeof (name)), name);
94:    printf (      "DEBUG:Reading From : %s (%s)\n", device, name);
95:
96:    printf(     "DEBUG:Getting exclusive access: " );
97:    result = ioctl(fevdev, EVIOCGRAB, 1);
98:    printf(     "%s\n", (result == 0) ?       "SUCCESS" :  "FAILURE");
99:
100:          std::thread first(Scan);
101:      first.detach();
102:
103:          std::thread Zero(setZero);
104:      Zero.detach();
105:
106:
107:          std::cout <<  "INFO:joystick aangekopelt."
```

```
108:                    << std::endl;                    // zeg dat de sensoren gestart zijn.
109:            return;
110:
111: }
112:
113: int  Joystick::GetPos (){
114:            return pos;
115: }
116:
117: Joystick::~Joystick  () {
118:        close(fevdev);
119:            delete device;
120: }
```

```
 1: /*
 2:  * Output.cpp
 3:  *
 4:  *  Created on: 26-dec.-2015
 5:  *      Author: david12
 6:  */
 7:
 8: #include "Output.h"
 9: #include <iostream>
10:
11: #include "Display/Display.h"
12: #include "Sound/Sound.h"
13: #include "ArduinoCom/ArduinoCom.h"
14: #include "../Verwerkingseenhijd/Instructies/Inst.h"
15:
16: Display*              display;
17: Sound*                sound;
18: Arduino_Com*     ArduinoC;
19:
20: int  i=0;
21:
22: Output::Output() {
23:         std::cout <<  "INFO:Starten van output aparatuur"
24:                 << std::endl;                                   // schrijf naar log dat we met output begginen
25:
26:     display=                new Display();                      // Start het display op
27:     sound=                  new Sound();                        // Start geluid systeem
28:     ArduinoC=               new Arduino_Com();                  // Start Comunication whit arduino
29:
30:         std::cout <<  "INFO:output aparatuur gestart"
31:                 << std::endl;                                   //schrijf naar log dat alles gestart is
32: }
33:
34: void  Output::LoopIt(Inst* data) {
35:         switch(data->GetScreenPos()){
36:                 case 0:
37:                 display->SetTerug();
38:                   break;
39:                   case 1:
40:                 display->SetHome();
41:                   break;
42:                   case 2:
43:                 display->SetStop();
44:                   break;
45:                   case 3:
46:                   //      display->SetLoop();
47:                   break;
48:                   case 4:
49:                 display->green();
50:                   break;
51:                   case 5:
52:                 display->red();
53:                   break;
54:     }
55:
56:         for(int  a=0;a!=8;a++)
57:           ArduinoC->SetMotor(a,data->GetMotorPosition(a));
58:
59:         if (data->GetNeedsArming()!=ArduinoC->GetArmd()){
60:                 if (data->GetNeedsArming()){
61:                 ArduinoC->SetArmd(true);
62:
63:           }
64:                 else{
65:                 ArduinoC->SetArmd(false);
66:           }
67:     }
68: }
69:
70: void  Output::OnStartup(){
71:     sound->Play(1);                                             // Play startup sound
72:     display->SetHome();
73: }
74:
75: void  Output::OnStop(){
76:     sound->Play(2);
77:     display->Clear();
78: }
79:
80: Output::~Output() {
81:         std::cout <<  "INFO:Starten van output aparatuur"
82:                 << std::endl;                                   // schrijf naar log dat we met output stoppen
83:
84:         delete display;                                         // Ontkoppel scherm van progama
85:         delete sound;                                           // Ontkoppel geluid
86:         delete ArduinoC;                                        // Ontkoppel arduino
87:
88:         std::cout <<  "INFO:output aparatuur gestopt"
89:                 << std::endl;                                   //schrijf naar log dat alles gestart is
90: }
91:
```

```
/*
 * ArduinoCom.cpp
 *
 * Created on: 18-feb.-2016
 *     Author: david12
 */

#include "ArduinoCom.h"                                 // Haal te doen lijst op

FILE *arduino;                                                                  // Maak Geheugenplaats vrij voor arduino aan te k
oppelen
bool Block=false;                                                               // Maag bit vrij voor onderbreken loop

bool Armd=false;                                                                // Maak bit vrij om te weten of de arduino (BEWAP
END) is
pthread_t thread;                                                               // Geheugen voor draat in te laten lopen

int MotorPoses[]={-1,-1,-1,-1,-1,-1,-1,-1};                     // Sla alle statusen van de motor's op

Arduino_Com::Arduino_Com(){                                                     // Voet uit bij aanmaak
      arduino = fopen(      "/dev/ttyUSB0", "w+");                    // Koppel arduino aan (Shrijven + lezen)
          if (arduino==NULL)                                                    // Check of koppelen geslaagt is
            arduino = fopen(          "/dev/ttyUSB1", "w+");          // Probeer aparaat op ander kontact te starten
          if (arduino==NULL)                                                    // Check of koppelen geslaagt is
            arduino = fopen(          "/dev/ttyUSB2", "w+");          // Probeer aparaat op ander kontact te starten
          if (arduino==NULL){                                                   // Check of koppelen geslaagt is
            std::cout <<  "PANIC:Kan arduino niet verbinden\n";  // Schrijf faut naar log
            std::cout <<  "@ARDUINOCOM.CPP LINE:20\n"           //
       fputs (      "ARDUINO NIET GEVONDEN\n",stderr);          //
       exit (-1);                                                           // NOOTSTOP (Motor's kunnen onkotroleerbaar zijn)
      }
      pthread_create(&thread, 0, &Arduino_Com::Looper,          // Start loop die input nakijkt en die nakijkt of
                     this );                                                                    // de arduino nog aangekopeld is.
}

void Arduino_Com::SetMotor(int Motor, int Value){               // Voer uit om een motor naar positie te laten gaan
          if (MotorPoses[Motor]!=Value)                                 // als motor staat al niet juist staat
            fprintf(arduino,            "M %d %d \n",Motor,Value);      // zent signaal in vorm (M1 50)
      fflush (arduino);
      MotorPoses[Motor]=Value;
}

char* Arduino_Com::GetInput(FILE* f){                                   // voer uit om input te lezen van arduino
          char * buffer;                                                        // maak geheugen vrij voor verwerking
          long lSize=1;                                                         // maak geheugen vrij voor verwerking
       size_t result;                                                           // maak geheugen vrij voor verwerking


       buffer = (      char*) malloc (   sizeof (char)*lSize);          // Zeg hoeveelhijd gehuegen nodig is
          if (buffer == NULL) {                                                 // FOUT : niet genoeg geheugen vrij
                  std::cout <<  "PANIC:Kan geheugen niet sta\nn";  // Schrijf faut naar log
              std::cout <<  "@ARDUINOCOM.CPP LINE:38\n"          //
           fputs (          "Memory error\n",stderr);                  //
           exit (-1);                                                           // NOOTSTOP (waarshijnlijk geheugen aan het lekke
n)
       }


          // copy the file into the buffer:
       result = fread (buffer,1,lSize,arduino);
          //if (result != lSize) {fputs ("Reading error",stderr); exit (3);}
          //free (buffer);

          //cout << buffer <<result;
          return buffer;
}

bool Arduino_Com::GetArmd(){                                            // Vraag of het systeem (BEWAPEND) is
          return Armd;                                                          // Geef bit terug
}

void Arduino_Com::SetArmd(bool armd){                                   // Zet (BEWAPEND) aan of uit
      Armd=armd;                                                                // Zet geheugen in juiste staat

          if (armd)                                                             // Check of het aan of uit gezet moet wor
den
            fprintf(arduino,            "A1 \n");                       // Zet uit
          else                                                                  // Anders
            fprintf(arduino,            "A0 \n");                       // Zet aan
      fflush (arduino);                                                         // Stuur door
}

bool Arduino_Com::Reconect(){                                           // Doe een poging om de arduino opnieuw aan te koppelen

          if (arduino!=NULL)                                                    // Als arduino nog aangesloten is
            fclose (arduino);                                                   // kopel hem dan nu los

      arduino = fopen(      "/dev/ttyUSB0", "w+");                    // Koppel arduino aan (Shrijven + lezen)
          if (arduino==NULL)                                                    // Check of koppelen geslaagt is
            arduino = fopen(          "/dev/ttyUSB1", "w+");          // Probeer aparaat op ander kontact te starten
          if (arduino==NULL)                                                    // Check of koppelen geslaagt is
            arduino = fopen(          "/dev/ttyUSB2", "w+");          // Probeer aparaat op ander kontact te starten
          if (arduino==NULL){                                                   // Check of koppelen geslaagt is
            std::cout <<  "PANIC:Verbing met arduino krijt\n";  // Schrijf faut naar log
            std::cout <<  "@ARDUINOCOM.CPP LINE:83\n"           //
       fputs (      "ARDUINO LOSGEKOPeLD\n",stderr);            //
       exit (-5);                                                           // NOOTSTOP (Motor's kunnen onkotroleerbaar zijn)
      }
return true;
}

void* Arduino_Com::Looper(void *obj){                                   // Lus die blijft kijken naar opkomende oproepen
          for (;;){                                                             // doe voor ewig
                  /*if(access( arduino, F_OK ) != -1){                    // kontroleer of arduino aangekopeld is
              arduino = fopen("/dev/ttyUSB0", "w+");            // Koppel arduino aan (Shrijven + lezen)
              if (arduino==NULL)                                        // Check of koppelen geslaagt is
                  arduino = fopen("/dev/ttyUSB1", "w+");        // Probeer aparaat op ander kontact te starten
```

```
104:                 if (arduino==NULL)                                  // Check of koppelen geslaagt is
105:                     arduino = fopen("/dev/ttyUSB2", "w+");          // Probeer aparaat op ander kontact te starten
106:                 if (arduino==NULL){                                 // Check of koppelen geslaagt is
107:                   std::cout << "PANIC:Verbing met arduino krijt\n"; // Schrijf faut naar log
108:                   std::cout << "@ARDUINOCOM.CPP LINE:83\n";          //
109:                   fputs ("ARDUINO LOSGEKOPeLD\n",stderr);           //
110:                   exit (-5);                                        // NOOTSTOP (Motor's kunnen onkot
roleerbaar zijn)
111:                 }
112:             }*/
113:                                                                                                  //TODO MAKE FUNKTION
114:                 usleep(1000);                                                                    // WACHT 1MS
115:                     while(Block)                                                                 // tot blocken uit gaat
116:                     usleep(1000);                                                                // WACHT 1MS
117:         }
118: }
119:
120: int  Arduino_Com::GetMotor(int Motor){                              // kijk motor staat na
121:         Block=true;
122:         fprintf(arduino,        "M %d -1 \n",Motor);
123:         fflush (arduino);
124:
125:             //for(int a=0;a!=100;a++){
126:         usleep(1000000);
127:               GetInput(arduino);
128:             //}
129:         Block=false;
130:
131:             return -1;
132: }
133:
134: Arduino_Com::~Arduino_Com(){
135:         fclose (arduino);
136: }
137:
```

```
 1: /*
 2:  * Sound.cpp
 3:  *
 4:  *  Created on: 26-dec.-2015
 5:  *      Author: david12
 6:  */
 7:
 8: #include "Sound.h"
 9:
10: #include <string>
11: #include <stdlib.h>
12:
13: Sound::Sound() {
14:         // TODO Auto-generated constructor stub
15:
16: }
17:
18: void Sound::Play(int sound){
19:         switch(0){ //TODO repair
20:                 case 1:
21:                 system(         "screen -S K9_Play_sound -d -m aplay ./sound/start.wav");
22:                   break;
23:                 case 2:
24:                 system(         "screen -S K9_Play_sound -d -m aplay ./sound/stop.wav");
25:                   break;
26:     }
27: }
28:
29: Sound::~Sound() {
30:         // TODO Auto-generated destructor stub
31: }
32:
```

```
 1: /*
 2:  * Motor.cpp
 3:  *
 4:  *  Created on: 26-dec.-2015
 5:  *      Author: david12
 6:  */
 7:
 8: #include  "Motor.h"
 9:
10: Motor::Motor() {
11:             // TODO Auto-generated constructor stub
12:
13: }
14:
15: Motor::~Motor() {
16:             // TODO Auto-generated destructor stub
17: }
18:
```

```
 1: /*
 2:  * Motoren.cpp
 3:  *
 4:  *  Created on: 26-dec.-2015
 5:  *      Author: david12
 6:  */
 7:
 8: #include "Motoren.h"
 9:
10: Motoren::Motoren() {
11:         // TODO Auto-generated constructor stub
12:
13: }
14:
15: Motoren::~Motoren() {
16:         // TODO Auto-generated destructor stub
17: }
18:
```

```
  1: /*
  2:  * Display.cpp
  3:  *
  4:  * Created on: 26-dec.-2015
  5:  *     Author: david12
  6:  */
  7:
  8: #define DEV_FB "/dev"
  9: #define FB_DEV_NAME "fb"
 10:
 11: #include <stdio.h>
 12: #include <stdlib.h>
 13: #include <stdint.h>
 14: #include <fcntl.h>
 15: #include <linux/fb.h>
 16: #include <sys/mman.h>
 17: #include <sys/ioctl.h>
 18: #include <time.h>
 19: #include <poll.h>
 20: #include <dirent.h>
 21: #include <unistd.h>
 22: #include <string.h>
 23: #include <iostream>
 24:
 25: #include <linux/input.h>
 26: #include <linux/fb.h>
 27:
 28: #include "Display.h"
 29:
 30:
 31: int fbfd = 0;
 32: int ret = 0;
 33:
 34: struct pollfd evpoll;
 35:
 36: struct fb_t {                                                    // maak framebuffer aan
 37:     uint16_t pixel[8][8];                                       // maak een scherm van 8X8 aan
 38: };
 39:
 40: struct fb_t *fb;                                                 //maak merker aan waar framebuffer word opgeslagen
 41:
 42: void SetStartupScreen(){                                        //Zet pixels op groen voor K
 43:     fb->pixel[1][6]=0x0F0F;
 44:     fb->pixel[1][5]=0x0F0F;
 45:     fb->pixel[1][4]=0x0F0F;
 46:     fb->pixel[1][3]=0x0F0F;
 47:     fb->pixel[1][2]=0x0F0F;
 48:     fb->pixel[2][5]=0x0F0F;
 49:     fb->pixel[2][4]=0x0F0F;
 50:     fb->pixel[3][3]=0x0F0F;
 51:     fb->pixel[3][6]=0x0F0F;
 52:     fb->pixel[3][2]=0x0F0F;
 53:
 54:     fb->pixel[5][6]=0xF00F;                                     //Zet pixels op groen voor 9
 55:     fb->pixel[5][5]=0xF00F;
 56:     fb->pixel[5][4]=0xF00F;
 57:     fb->pixel[5][2]=0xF00F;
 58:     fb->pixel[6][6]=0xF00F;
 59:     fb->pixel[6][4]=0xF00F;
 60:     fb->pixel[6][2]=0xF00F;
 61:     fb->pixel[7][6]=0xF00F;
 62:     fb->pixel[7][5]=0xF00F;
 63:     fb->pixel[7][4]=0xF00F;
 64:     fb->pixel[7][3]=0xF00F;
 65:     fb->pixel[7][2]=0xF00F;
 66:
 67:
 68:     fb->pixel[0][0]=0x00FF;                                     //start laad balk
 69: }
 70:
 71: fb_t* CreateMMap(int fbfd){
 72:         return static_cast<fb_t*>(mmap(0, 128, PROT_READ | PROT_WRITE, MAP_SHARED, fbfd, 0));
 73: }
 74:
 75: static int is_framebuffer_device(const struct dirent *dir)
 76: {
 77:         return strncmp(FB_DEV_NAME, dir->d_name,
 78:                 strlen(FB_DEV_NAME)-1) == 0;
 79: }
 80:
 81: static int open_fbdev(const char *dev_name)
 82: {
 83:         struct dirent **namelist;
 84:         int i, ndev;
 85:         int fd = -1;
 86:         struct fb_fix_screeninfo fix_info;
 87:
 88:     ndev = scandir(DEV_FB, &namelist, is_framebuffer_device, versionsort);
 89:         if (ndev <= 0)
 90:                 return ndev;
 91:
 92:         for (i = 0; i < ndev; i++)
 93:     {
 94:                 char fname[64];
 95:                 char name[256];
 96:
 97:           snprintf(fname,             sizeof (fname),
 98:                            "%s/%s", DEV_FB, namelist[i]->d_name);
 99:           fd = open(fname, O_RDWR);
100:                 if (fd < 0)
101:                         continue;
102:           ioctl(fd, FBIOGET_FSCREENINFO, &fix_info);
103:                 if (strcmp(dev_name, fix_info.id) == 0)
104:                         break;
105:           close(fd);
106:           fd = -1;
107:     }
```

```
108:            for (i = 0; i < ndev; i++)
109:              free(namelist[i]);
110:
111:            return fd;
112: }
113:
114:
115: Display::Display () {
116:        fbfd = 0;
117:
118:        srand(time(NULL));
119:
120:        fbfd = open_fbdev(       "RPi-Sense FB");
121:            if (fbfd <= 0) {
122:              ret = fbfd;
123:              printf(           "Error: cannot open framebuffer device.\n"  );
124:                      goto err_ev;
125:
126:        }
127:
128:        fb = CreateMMap(fbfd);
129:            if (!fb) {
130:              ret = EXIT_FAILURE;
131:              printf(           "Failed to mmap.\n");
132:                      goto err_fb;
133:        }
134:        memset(fb, 0, 128);
135:
136:        SetStartupScreen();                                                    // show startupscreen
137:
138:            err_fb:
139:              close(fbfd);
140:            err_ev:
141:              close(evpoll.fd);
142:                      return;
143: }
144:
145: void Display::SetPixel (int x, int y,uint16_t collor){
146:        fb->pixel[x][y]=collor;
147: }
148:
149: void Display::SetHome(){
150:        memset(fb,0,128);
151:        SetPixel(2,1,0xF00F);
152:        SetPixel(3,1,0xF00F);
153:        SetPixel(4,1,0xF00F);
154:        SetPixel(5,1,0xF00F);
155:        SetPixel(2,2,0xF00F);
156:        SetPixel(5,2,0xF00F);
157:        SetPixel(2,3,0xF00F);
158:        SetPixel(5,3,0xF00F);
159:        SetPixel(2,4,0xF00F);
160:        SetPixel(5,4,0xF00F);
161:        SetPixel(2,5,0xF00F);
162:        SetPixel(5,5,0xF00F);
163: }
164:
165: void Display::SetStop (){
166:        memset(fb,0,128);
167:        SetPixel(0,0,0xF00F);
168:        SetPixel(1,1,0xF00F);
169:        SetPixel(2,2,0xF00F);
170:        SetPixel(3,3,0xF00F);
171:        SetPixel(4,4,0xF00F);
172:        SetPixel(5,5,0xF00F);
173:        SetPixel(6,6,0xF00F);
174:        SetPixel(7,7,0xF00F);          // STOP icoon
175:
176:        SetPixel(0,7,0xF00F);          // #     #
177:        SetPixel(1,6,0xF00F);          //  #   #
178:        SetPixel(2,5,0xF00F);          //   # #
179:        SetPixel(3,4,0xF00F);          //    ##
180:        SetPixel(4,3,0xF00F);          //    ##
181:        SetPixel(5,2,0xF00F);          //   # #
182:        SetPixel(6,1,0xF00F);          //  #   #
183:        SetPixel(7,0,0xF00F);          // #     #
184: }
185:
186: void Display::SetTerug(){
187:        memset(fb,0,128);
188:        SetPixel(3,0,0x0B0F);
189:        SetPixel(4,0,0x0B0F);
190:        SetPixel(2,1,0x0B0F);
191:        SetPixel(3,1,0x0B0F);
192:        SetPixel(4,1,0x0B0F);
193:        SetPixel(5,1,0x0B0F);          // Terug icoon
194:        SetPixel(1,2,0x0B0F);
195:        SetPixel(2,2,0x0B0F);          //
196:        SetPixel(3,2,0x0B0F);          //   #
197:        SetPixel(4,2,0x0B0F);          //  ##
198:        SetPixel(5,2,0x0B0F);          // ########
199:        SetPixel(6,2,0x0B0F);          // ########
200:        SetPixel(3,3,0x0B0F);          //  ##
201:        SetPixel(4,3,0x0B0F);          //   #
202:        SetPixel(3,4,0x0B0F);          //
203:        SetPixel(4,4,0x0B0F);
204:        SetPixel(3,5,0x0B0F);
205:        SetPixel(4,5,0x0B0F);
206:        SetPixel(3,6,0x0B0F);
207:        SetPixel(4,6,0x0B0F);
208:        SetPixel(3,7,0x0B0F);
209:        SetPixel(4,7,0x0B0F);
210: }
211:
212: void Display::green(){
213:            for(int a=0;a!=8;a++)
214:                    for(int b=0;b!=8;b++)
```

```
215:                    SetPixel(a,b,0x0F0F);
216: }
217:
218: void  Display::red (){
219:             for(int  a=0;a!=8;a++)
220:                      for(int  b=0;b!=8;b++)
221:                    SetPixel(a,b,0xF00F);
222: }
223:
224: void  Display::Clear (){
225:        memset(fb,0,128);
226: }
227:
228: Display::~Display () {
229:        memset(fb, 0, 128);                                                    // cleare display
230:             delete fb;                                                                  // stop framebuffer
231:
232: }
233:
```

```
  1: /*
  2:  * TSS.cpp
  3:  *
  4:  *  Created on: 10-mrt.-2016
  5:  *      Author: david12
  6:  */
  7:
  8: #include  "TSS.h"
  9:
 10: bool toset[][][]={
 11:             {       {false,false,false,false,false,false,false,false},
 12:                     {false,false,true ,true ,true ,false,false,false},
 13:                     {false,false,false,true ,true ,false,false,false},
 14:                     {false,false,false,true ,true ,false,false,false},
 15:                     {false,false,false,true ,true ,false,false,false},
 16:                     {false,false,false,true ,true ,false,false,false},
 17:                     {false,false,true ,true ,true ,true ,false,false},
 18:                     {false,false,false,false,false,false,false,false},},
 19:
 20:             {       {false,false,false,false,false,false,false,false},
 21:                     {false,true ,true ,true ,true ,true ,true ,false},
 22:                     {false,true ,false,false,false,false,true ,false},
 23:                     {false,false,false,false,true ,true ,true ,false},
 24:                     {false,true ,true ,true ,false,false,false,false},
 25:                     {false,true ,false,false,false,false,false,false},
 26:                     {false,true ,true ,true ,true ,true ,true ,false},
 27:                     {false,false,false,false,false,false,false,false},},
 28:
 29:             {       {false,false,false,false,false,false,false,false},
 30:                     {false,false,false,false,false,false,false,false},
 31:                     {false,false,false,false,false,false,false,false},
 32:                     {false,false,false,false,false,false,false,false},
 33:                     {false,false,false,false,false,false,false,false},
 34:                     {false,false,false,false,false,false,false,false},
 35:                     {false,false,false,false,false,false,false,false},
 36:                     {false,false,false,false,false,false,false,false},},
 37:
 38:             {       {false,false,false,false,false,false,false,false},
 39:                     {false,false,false,false,false,false,false,false},
 40:                     {false,false,false,false,false,false,false,false},
 41:                     {false,false,false,false,false,false,false,false},
 42:                     {false,false,false,false,false,false,false,false},
 43:                     {false,false,false,false,false,false,false,false},
 44:                     {false,false,false,false,false,false,false,false},
 45:                     {false,false,false,false,false,false,false,false},},
 46:
 47:             {       {false,false,false,false,false,false,false,false},
 48:                     {false,false,false,false,false,false,false,false},
 49:                     {false,false,false,false,false,false,false,false},
 50:                     {false,false,false,false,false,false,false,false},
 51:                     {false,false,false,false,false,false,false,false},
 52:                     {false,false,false,false,false,false,false,false},
 53:                     {false,false,false,false,false,false,false,false},
 54:                     {false,false,false,false,false,false,false,false},},
 55:
 56:             {       {false,false,false,false,false,false,false,false},
 57:                     {false,false,false,false,false,false,false,false},
 58:                     {false,false,false,false,false,false,false,false},
 59:                     {false,false,false,false,false,false,false,false},
 60:                     {false,false,false,false,false,false,false,false},
 61:                     {false,false,false,false,false,false,false,false},
 62:                     {false,false,false,false,false,false,false,false},
 63:                     {false,false,false,false,false,false,false,false},},
 64:
 65:             {       {false,false,false,false,false,false,false,false},
 66:                     {false,false,false,false,false,false,false,false},
 67:                     {false,false,false,false,false,false,false,false},
 68:                     {false,false,false,false,false,false,false,false},
 69:                     {false,false,false,false,false,false,false,false},
 70:                     {false,false,false,false,false,false,false,false},
 71:                     {false,false,false,false,false,false,false,false},
 72:                     {false,false,false,false,false,false,false,false},},
 73:
 74:             {       {false,false,false,false,false,false,false,false},
 75:                     {false,false,false,false,false,false,false,false},
 76:                     {false,false,false,false,false,false,false,false},
 77:                     {false,false,false,false,false,false,false,false},
 78:                     {false,false,false,false,false,false,false,false},
 79:                     {false,false,false,false,false,false,false,false},
 80:                     {false,false,false,false,false,false,false,false},
 81:                     {false,false,false,false,false,false,false,false},},
 82:
 83:             {       {false,false,false,false,false,false,false,false},
 84:                     {false,false,false,false,false,false,false,false},
 85:                     {false,false,false,false,false,false,false,false},
 86:                     {false,false,false,false,false,false,false,false},
 87:                     {false,false,false,false,false,false,false,false},
 88:                     {false,false,false,false,false,false,false,false},
 89:                     {false,false,false,false,false,false,false,false},
 90:                     {false,false,false,false,false,false,false,false},},
 91:
 92:             {       {false,false,false,false,false,false,false,false},
 93:                     {false,false,false,false,false,false,false,false},
 94:                     {false,false,false,false,false,false,false,false},
 95:                     {false,false,false,false,false,false,false,false},
 96:                     {false,false,false,false,false,false,false,false},
 97:                     {false,false,false,false,false,false,false,false},
 98:                     {false,false,false,false,false,false,false,false},
 99:                     {false,false,false,false,false,false,false,false},}
100: };
101:
102: TSS::TSS() {
103:             // TODO Auto-generated constructor stub
104:
105: }
106:
107: void  TSS::SetInt( int  teken,uint16_t collor,uint16_t* pixel[8][8]){
```

```
108:            for(int a=0;a!=8;a++)
109:                    for(int b=0;b!=8;b++)
110:                            if (toset[teken][a][b])
111:                        pixel[a][b]=collor;
112: }
113:
114: TSS::~TSS() {
115:            // TODO Auto-generated destructor stub
116: }
117:
```