LISTING PROGRAM

```
#include <iostream>
#include <cstdlib>
#include <string>
#include <sys/types.h>
#include <dirent.h>
#include <errno.h>
#include <signal.h>
#include <unistd.h>
#include <cstdarg>
#include <sys/time.h>
#include <opencv2/highgui/highgui.hpp>
#include "defs.h"
#include "FaceDetector.h"
#include "PersonRecognizer.h"
#include "VideoFaceDetector.h"
#include <pigpio.h>
#include <pigpiod_if.h>
#include "i2c.h"
#include "lcd.h"
#define I2C_FILE_NAME "/dev/i2c-1"
#define    I2C_SLAVE_ADDR    0x3f
const cv::String CASCADE_FILE("haarcascade_frontalface_default.xml");
using namespace std;
using namespace cv;
#define GPIO_SERVO_OUT  5
const int GPIO_KEYPAD_NUM[4][2] = {
   {6, 11},
```

```
    {13, 16},
    {19, 20},
    {26, 21}
};
const char KEYPAD_CHAR_MAP[16] = {
//    //col 1
//    '*', '7', '4', '1',
//
//    //col 2
//    '0', '8', '5', '2',
//
//    //col 3
//    '#', '9', '6', '3',
//
//    //col 4
//    'D', 'C', 'B', 'A'

    //col 1
    'A', 'B', 'C', 'D',

    //col 2
    '3', '6', '9', '#',

    //col 3
    '2', '5', '8', '0',

    //col 4
    '1', '4', '7', '*'
};
```

```
/* password file */
#define PASSWORD_FILE   "password.conf"
        //col
        gpioSetMode(GPIO_KEYPAD_NUM[i][1], PI_INPUT);
        gpioSetPullUpDown(GPIO_KEYPAD_NUM[i][1], PI_PUD_UP);
/* make a row out low, others as input */
void keypad_clear_row(int row_id) {
   int i;
   for (i = 0; i < 4; i++) {
      if (i == row_id) {
         gpioSetMode(GPIO_KEYPAD_NUM[i][0], PI_OUTPUT);
         gpioWrite(GPIO_KEYPAD_NUM[i][0], PI_LOW);
      } else {
         gpioSetMode(GPIO_KEYPAD_NUM[i][0], PI_INPUT);
         gpioSetPullUpDown(GPIO_KEYPAD_NUM[i][0], PI_PUD_OFF);
      }
   }
}
   /* try for max 5 times */
   for (i = 0; i < 5; i++) {
      printf("Verification #%d\n", i + 1);

      is_valid = id_lcd_buff = 0;
      memset(lcd_buff, 0, sizeof (lcd_buff));

      /* get the password */
      for (;;) {
         /* get keypad data */
```

```
dt_kpd = keypad_get_data();

/* continue on no keypressed */
if (dt_kpd == 0xff) {
   continue;
}

/* valid? */
/* number */
tmp_kpd = KEYPAD_CHAR_MAP[dt_kpd];
//printf("%c\n", tmp_kpd);
if ((tmp_kpd >= '0') && (tmp_kpd <= '9')) {
   if (id_lcd_buff < 16) {
      lcd_buff[id_lcd_buff++] = tmp_kpd;
    if (is_valid) {
      //open key door
      gpioServo(GPIO_SERVO_OUT, 1500);
      //exit_main_loop = 1;
      lcd_printf(&LCD_DEV, 1, "%-16s", "OK");
   } else {
      lcd_printf(&LCD_DEV, 1, "%-16s", "INVALID");
      id_lcd_buff = 0;
      memset(lcd_buff, 0, sizeof (lcd_buff));
   }

   printf("Verification result %s\n", is_valid ? "OK" : "INVALID");

   time_sleep(2.0);
   lcd_printf(&LCD_DEV, 1, "%-16s", " ");
```

```
            break;
          }/* clear */
          else if (tmp_kpd == 'C') {
            id_lcd_buff = 0;
            memset(lcd_buff, 0, sizeof (lcd_buff));
            lcd_printf(&LCD_DEV, 1, "%-16s", " ");
          }/* escape */
          else if ((tmp_kpd == '*') || (tmp_kpd == 'D')) {
            printf("Validation aborted\n");
            exit_main_loop = 1;
            gpioServo(GPIO_SERVO_OUT, 800);
            break;
          }/* others */
          else {
    //sent email on max retry
    if (i >= 3) {
       printf("Maximum retry reach, send warning notification...\n");
       //kirim email
       //system();
       system("curl -i -F 'filename=frame.jpg' -F 'file=@/dev/shm/frame.jpg'
http://dickiherlambang.esy.es/uploadPhotonew.php");

    }

    lcd_clear(&LCD_DEV);
    lcd_printf(&LCD_DEV, 0, "SYSTEM READY");
}CV_LOAD_IMAGE_GRAYSCALE));
```

```
            if ((cam_face_detector.face().height >= 100) &&
(cam_face_detector.face().width >= 100)) {
                face_frame = image_frame(cam_face_detector.face());
                imwrite(cv::format("%s%d_%s", dirname.c_str(), dtf,
basename(listfiles[i].c_str())), face_frame);

                printf("POS = %03d,%03d  [%03d,%03d]\n",
cam_face_detector.facePosition().x, cam_face_detector.facePosition().y,
cam_face_detector.face().width, cam_face_detector.face().height);
            }

        }

        return EXIT_SUCCESS;
    }

    /* init pigpio */
    if (gpioInitialise() < 0) {
        printf("GPIO Init failed\n");
    } else {
        printf("GPIO Init OK\n");

        if (gpioServo(GPIO_SERVO_OUT, 800) == 0) {
            printf("Servo OK\n");
        } else {
            printf("Servo failed\n");
        }
    }
```

```
    /* init lcd */
    if (lcd_initialize(&LCD_I2C_HDL, &LCD_DEV)) {
        printf("Init lcd ok\n");
        //        lcd_print(&LCD_DEV, "SYSTEM READY", 12, 0);
        lcd_printf(&LCD_DEV, 0, "SYSTEM READY");
    } else {
        printf("Init lcd failed\n");
    }

    /* init face recognizer */
    /* read training set images */
    read_training_set_folder(argv[2], training_set, training_labels);
  /* main loop */
    while (s_signal_received == 0) {
        //get frame
        cam_face_detector >> cam_frame;

        //write face frame to /dev/shm
        face_img = cam_face_detector.getFaceFrame(cam_frame);
        imwrite("/dev/shm/facex.jpg", face_img);

        //write cam frame to /dev/shm
        imwrite("/dev/shm/frame.jpg", cam_frame);

        //save to mjpg stream
        //        VideoWriter mjpg_stream("/dev/shm/stream.mjpeg", CV_FOURCC('M',
'J', 'P', 'G'), 5, Size(320, 240));
        //        if (mjpg_stream.isOpened()) {
        //            printf("Write mjpg stream\n");
```

```
//            mjpg_stream.write(cam_frame);
//        } else {
//            printf("Fail init mjpg stream\n");
//        }
//        if (mjpg_stream.isOpened()) {
//            mjpg_stream.write(cam_frame);
//        }

        //print to console
        //check for face with height and width >= 100
        if ((cam_face_detector.face().width >= 150) &&
(cam_face_detector.face().height >= 150)) {
            printf("Got valid face size\nTry to recognize...\n");

            //recognize it
            if (pr.recognize(face_img, face_confidence, face_predicted_label) != -1) {
                printf("Got match for label %d with confidence %f\n",
face_predicted_label, face_confidence);

                //different face?
                if (face_predicted_label != face_previous_predicted_label) {
                    face_previous_predicted_label = face_predicted_label;

                    //thresholding confidence level
                    if (face_confidence <= 150) {
                        /* display to lcd */
                        lcd_printf(&LCD_DEV, 0, "%-16s", "ID :");
                        lcd_printf(&LCD_DEV, 1, "%-16d", face_predicted_label);
                        time_sleep(2.0);
```

```
                    /* verify password */
                    verify_password_uid_process(face_predicted_label);
                    ///break;
                    //cam_face_detector = new
VideoFaceDetector(CASCADE_FILEX,cap);
                    //cam_face_detector(CASCADE_FILEX,cap);
                    //VideoFaceDetector cam_face_detector(CASCADE_FILEX, cap);
                }
            }
        }
    }else{
        face_previous_predicted_label=0;
    }
  }

  /* cleaning */
  printf("Close program....\n");
  cap.release();
  gpioTerminate();
  return EXIT_SUCCESS;
}
```