EDAF55 : Design Draft - Camera Project

Due on Wednesday, November 8, 2017

Ola Johansson, Joachim Wedin, Hannes Berntsson, Erman Nurdal

Design 1

Camera - Server:

Threads:

TakePicture:
Waits for picSent to be true. Then sets picSent to false. takePicture har access to the camera monitor and uses mode to determine frequency of picture taking. Stores the image in a variable in the camera monitor. Holds a variable which determines the wait time between images in idle mode. when done, sets picTaken to true.

SendPicture:
Waits for picTaken to be true. Then sets picTaken to false. Accesses the picture attribute in the camera monitor, wraps in package for sending over network, and sends. Sets picSent to true and notifies takePicture. No logic concerning mode, sends asap.

UpdateMode:
Waits for data packet from the client to set the mode in the cameraMonitor.

MotionSensor:
A thread called from the client in order to check for motion, sends back motion data to client via http.

Monitors:

CameraMonitor:
Methods for setMode, getImage and putImage. All these methods should be synchronized. Holds private booleans takePic and picTaken which are also accessed via synchronized methods. They are used by TakePicture(), sendPicture() to determine when to take and send pictures respectively. .

Client:

Threads:

MotionCtrl: Periodically checks the MotionSensor thread and receives motion data in return. Updates mode in the ClientMonitor.

SendMode: Waits for update of mode in ClientMonitor and sends to UpdateMode thread in camera.

PictureReceiver: Waits for new picture from sedPicture thread in camera. Stores pictures from both cameras in a common buffer, in the ClientMonitor.

Display: Periodically updates the DisplayMonitor depending on the frequency of received pictures. Does not consider mode per se, instead uses time stamps of pictures to determine synchronous or asynchronous mode. There are two Display threads, one for each camera. Getting pictures from there respective buffers. While in synchronous mode sleeps the required time. While in asynchronous mode, does not sleep. Regardless of mode constant switching is prevented by a counter which allows for multiple frames in a row which do

Design 1 continued on next page...

not conform to the current mode (sync/async). Sends to DisplayMonitor immediately. The display thread notifies the GUI that a new picture is available.

Input: Sets mode regardless of motion in cameras.

Monitors:

ClientMonitor: Holds methods for setting and getting mode, and a buffer for each camera storing pictures. The mode of the ClientMonitor propagates accross the system e.g. determines mode for the cameras.

DisplayMonitor: Holds references to two pictures, one for each camera. Updated by the two display threads. Read by the GUI thread. The methods for access are synchronized. These images are shown in the GUI.

GUI:
the GUI.

Data flow:

From picture to display - data flow:
Picture is taken in the TakePicture thread. Picture is stored in the buffer of the CameraMonitor - notify SendPicture is notified and packs the picture with data (camera, timestamp, image size, image data) and sends via TCP PictureReceiver waits for picture from SendPicture. Gets image, saves in buffer in ClientMonitor. Display thread takes from the picture buffer ASAP. Depending on camera and frequency the pictures are displayed in different ways.

Updating mode - data flow:
Periodically the MotionsCtrl thread requests info from MotionSensor thread via http, which informs the system of motion. MotionCtrl sets mode in ClientMonitor, notify. SendMode is notified and sends the mode update via TCP to the updateMode thread in camera. UpdateMode updates mode in the CameraMonitor.

Packets:

Picture packet:

Time stamp	no of bits: 64
size	no of bits: 2Log(24 x [Resolution])
data	no of bits: determined by size

mode packet:
a single bit representing the mode:
1 - movie
0 - idle

http communication:
existing protocol.

Figure 1: UML diagram of proposed design.