Lab 5. WebCam and System Integration

I. Purpose

The purpose of this lab is to design and implement a remote video viewer using off-the-shelf WebCam, and perform system integration of RoboCam.

II. Problem Statement

Problem 5. WebCam and System integration
Implement ***Video functionality.***
Also perform ***system integration*** of Video and control functionalities.

We start from the simplest and perform step-by-step improvements.

Problem 5A. Test Capture WebCam on Beaglebone.
Test capture image from WebCam with V4L2 (Video for Linux 2) on Beaglebone.

Problem 5B. Learn SDL2 via Tutorials.
Learn SDL 2 (Simple DirectMedia Layer 2) via Tutorials.

Problem 5C. Implement Video functionality.
Implement Video functionality composed of

- Camera on Beaglebone (Capture from WebCam and send video to network) and
- Viewer on PC (Receive video from network and display video to user) using SDL2.

Problem 5D. System Integration.
Perform system integration of Video and control functionalities:
A. Video functionality composed of

- Camera on Beaglebone (Capture from WebCam and send video to network) and
- Viewer on PC (Receive video from network and display video to user).

B. Control functionality composed of

- Commander on PC (Get key input from user and send command packet to network) and
- Controller on Beaglebone (Receive command packet from network and actuate servos and lights on the robot).

For recording photos and videos, Commander on PC sends user command to Viewer, which records photos and videos to storage device.

Data Flow is summarized as follows:

User	PC	Network	Beaglebone	Robot
Eye ←	Viewer ←	Video ←	Camera ←	WebCam
Storage ←	(Record command)			
Finger →	Commander ^→	Command →	Controller →	Servos & Lights

III. Technical Backgrounds

First Week

A. Test Capture WebCam on Bone

1. Choosing a Webcam device driver in Linux

["Webcam", https://help.ubuntu.com/community/Webcam**]**

Webcam support in Linux is mainly provided by the Linux UVC Project's *UVC driver.* This aims to provide a universal driver in the same way that a generic driver handles USB storage devices (memory sticks, hard drives, etc.). However, other drivers also exist that may allow more devices to be used.

When looking to purchase a webcam for use with Ubuntu, you should look for a *UVC compatible camera*. The Linux-UVC project has a good list of UVC compatible webcams as well as The Quickcam Team for Logitech cameras.

2. UVC compatible cameras

["Linux UVC (USB Video Class) driver and tools", http://www.ideasonboard.org/uvc/]

Welcome to the USB Video Class Linux device driver home.
The goal of this project is to provide all necessary software components to fully support UVC compliant devices in Linux. This include a V4L2 kernel device driver and patches for user-space tools.

The *USB Device Class Definition for Video Devices*, or USB Video Class, defines video streaming functionality on the Universal Serial Bus. Much like nearly all mass storage devices (USB flash disks, external SATA disk enclosures, ...) can be managed by a single driver because they conform to the USB Mass Storage specification, UVC compliant peripherals only need a generic driver.

The UVC specification covers webcams, digital camcorders, analog video converters, analog and digital television tuners, and still-image cameras that support video streaming for both video input and output.

Supported devices

Device ID	Name	Manufacturer	Status
046d:0994	Logitech Quickcam Orbit/Sphere AF	Logitech	✓
046d:0805	Logitech Webcam C300	Logitech	✓
046d:0819	Logitech Webcam C210	Logitech	✓

Can't find Webcam C110, but we proceed.

3. Linux WebCam device drivers and applications

Device driver:

Uvcvideo.ko in kernel

Application programs [For graphic user interface only: Not for Beaglebone, but for PC]:

VLC media player, Mplayer, streamer, etc.

4. Video for Linux

["Beaglebone Images, Video and OpenCV", http://derekmolloy.ie/beaglebone-images-video-and-opencv/]

Using v4l2 (Video for Linux version 2), you can capture images. Video4Linux or V4L is a video capture application programming interface for Linux, supporting many USB webcams, TV tuners, and other devices. Video4Linux is closely integrated with the Linux kernel.

What is V4L2?

Video4Linux

https://en.wikipedia.org/wiki/Video4Linux

Video4Linux or **V4L** is a video capture[1] and output device API and driver framework for the Linux kernel, supporting many USB webcams, TV tuners, and other devices. Video4Linux is closely integrated with the Linux kernel. Video4Linux was named after Video for Windows (which is sometimes abbreviated "V4W"), but is not technically related to it.[2][3]

Version 1

V4L had been introduced late into the 2.1.X development cycle of the Linux kernel. V4L1 support was dropped in kernel 2.6.38.

Version 2

V4L2 is the second version of V4L. Video4Linux2 fixed some design bugs and started appearing in the 2.5.X kernels. Video4Linux2 drivers include a compatibility mode for Video4Linux1 applications, though the support can be incomplete and it is recommended to use Video4Linux1 devices in V4L2 mode. The project DVB-Wiki is now hosted on LinuxTV web site.

Software supporting Video4Linux

Ffmpeg, Gstreamer, Motion, Mplayer, OpenCV, Skype, VLC media player, etc.

V4L2 Usage

Getting help

```
# v4l2-ctl --help
```

Show driver info
-D, --info   show driver info [VIDIOC_QUERYCAP]
```
        # v4l2-ctl -D
```

Video capture info
```
        # v4l2-ctl --help-vidcap
```

List supported video formats
```
        # v4l2-ctl --list-formats
        ioctl: VIDIOC_ENUM_FMT
                Index       : 0
                Type        : Video Capture
                Pixel Format: 'YUYV'
                Name        : YUV 4:2:2 (YUYV)

                Index       : 1
                Type        : Video Capture
                Pixel Format: 'MJPG' (compressed)
                Name        : MJPEG
```

Notice that C110 supports two formats: YUYV and MJPG.

Check current video format
```
        # v4l2-ctl -V
```

Set video format
  -v, --set-fmt-video=width=<w>,height=<h>,pixelformat=<f>
                              set the video capture format [VIDIOC_S_FMT]
                              pixelformat is either the format index as reported by
                              --list-formats, or the fourcc value as a string

To YUYV: (Note: Two '-' before set-fmt-video)
```
        # v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=0
```
To MJPG:
```
        # v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=1
```

5. What is YUYV?

[“**YUYV Format”,** http://linuxtv.org/downloads/v4l-dvb-apis/V4L2-PIX-FMT-YUYV.html]

Name

V4L2_PIX_FMT_YUYV — Packed format with ½ horizontal chroma resolution, also known as YUV 4:2:2

Description

In this format each four bytes is two pixels. Each four bytes is two Y's, a Cb and a Cr. Each Y goes to one of the pixels, and the Cb and Cr belong to both pixels. As you can see, the Cr and Cb components have half the horizontal resolution of the Y component. V4L2_PIX_FMT_YUYV is known in the Windows environment as

YUY2.

Y'UV422 can also be expressed in YUY2 FourCC format code. That means 2 pixels will be defined in each macropixel (four bytes) treated in the image. [5]

Y0 U0 Y1 V0 Y2 U1 Y3 V1 ...

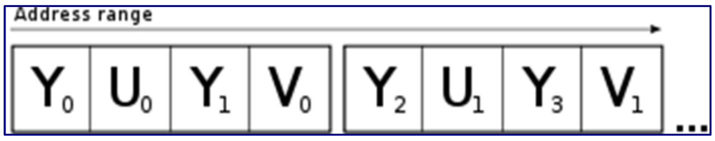

Fig. 5.1 YUYV format

6. What is JPEG?

https://en.wikipedia.org/wiki/JPEG

JPEG (/ˈdʒeɪpɛɡ/ ***JAY**-peg*)[1] is a commonly used method of lossy compression for digital images, particularly for those images produced by digital photography. The degree of compression can be adjusted, allowing a selectable tradeoff between storage size and image quality. JPEG typically achieves 10:1 compression with little perceptible loss in image quality.[2]

JPEG compression is used in a number of image file formats. JPEG/Exif is the most common image format used by digital cameras and other photographic image capture devices; along with JPEG/JFIF, it is the most common format for storing and transmitting photographic images on the World Wide Web.[3] These format variations are often not distinguished, and are simply called JPEG.

The term "JPEG" is an acronym for the Joint Photographic Experts Group, which created the standard. The MIME media type for JPEG is *image/jpeg*, except in older Internet Explorer versions, which provides a MIME type of *image/pjpeg* when uploading JPEG images.[4] JPEG files usually have a filename extension of *.jpg* or *.jpeg*.

Checking Jpeg image can be donw with the following function:

```
// Check_Jpeg(image): Check SOI/EOI and Returns 1 if Jpeg image, 0 otherwise.
int Check_Jpeg(unsigned char *image, int size)
{
        // Check SOI 0xffd8 at the head of Jpeg image
        // Also Check EOI 0xffd9 at the end of Jpeg image
        if (image[0] == 0xff && image[1] ==  0xd8
                && image[size-2] == 0xff && image[size-1] == 0xd9 ) return 1;
        else return 0;
}
```

7. What is MJPG?

https://en.wikipedia.org/wiki/Motion_JPEG

In multimedia, **Motion JPEG** (**M-JPEG** or **MJPEG**) is a video compression format in which each video frame or interlaced field of a digital video sequence is compressed separately as a JPEG image. Originally developed for multimedia PC applications, M-JPEG is now used by video-capture devices such as digital cameras, IP cameras, and webcams; and by non-linear video editing systems. It is natively supported by the QuickTime Player, the PlayStation console, and web browsers such as Safari, Google Chrome, Mozilla Firefox and Microsoft Edge.

8. Image capture program

Capture.c

The source video capture code is from:
file: media/v4l/capture.c
https://www.linuxtv.org/downloads/v4l-dvb-apis-new/uapi/v4l/capture.c.html
Download to a_CaptureBone/Capture_LinuxTV.c.
This is a generic example program to capture from WebCam using V4L2.

Use this program to capture images with some modifications.

B. Learn SDL via Tutorials.

9. Simple DirectMedia Layer

https://en.wikipedia.org/wiki/Simple_DirectMedia_Layer

Simple DirectMedia Layer (**SDL**) is a cross-platform software development library designed to provide a low level hardware abstraction layer to computer multimedia hardware components. Software developers can use it to write high-performance computer games and other multimedia applications that can run on many operating systems such as Android, iOS, Linux, Mac OS X, Windows and other platforms.

SDL manages video, audio, input devices, CD-ROM, threads, shared object loading, networking and timers.[5] For 3D graphics it can handle an OpenGL or Direct3D context.

The library is internally written in C and Objective-C and provides the application programming interface in C, with bindings to other languages available.[6] It is free and open-source software subject to the requirements of the zlib License since version 2.0 and with prior versions subject to the GNU Lesser General Public License. Under the zlib License, SDL 2.0 is freely available for static linking in closed-source projects, unlike SDL 1.2.

SDL is extensively used in the industry in both large and small projects. Over 700 games, 180 applications, and 120 demos have also been posted on the library website.

A common misconception is that SDL is a game engine, but this is not true. However, the library is well-suited for building an engine on top of it.

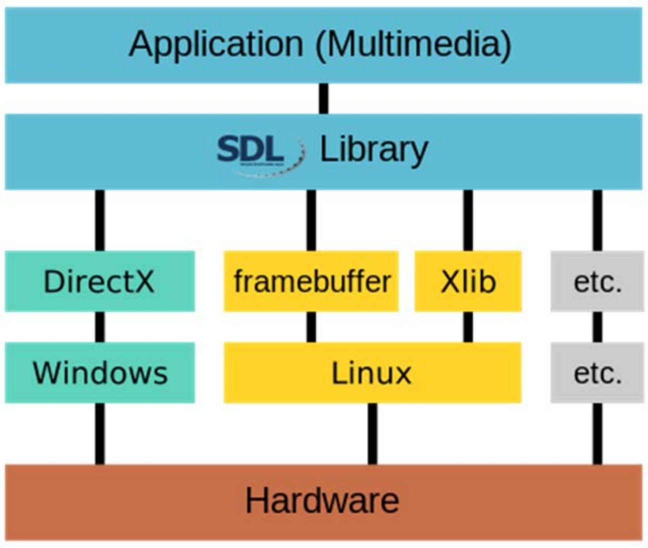

Fig 5.2 Abstraction layers of several SDL platforms

SDL is divided into several subsystems:[

Video
    Display and Window Management, surface functions, rendering acceleration, etc.
Input Events
    Event handling, Support for Keyboard, Mouse, Joystick and Game controller
Threads
    multi-threading: Thread Management, Thread Synchronization Primitives, Atomic Operations
……

There also are a few separate official libraries that provide some more functions. These comprise the "standard library", and are provided on the official website and included in the official documentation:

10. SDL_image — support for multiple image formats

IMAGE in SDL: Lesson 06. Extension Libraries and Loading Other Image Formats
http://lazyfoo.net/tutorials/SDL/06_extension_libraries_and_loading_other_image_formats/index.php

SDL extension libraries allow you do things like load image files besides BMP, render TTF fonts, and play music. You can set up SDL_image to load PNG files, which can save you a lot of disk space. In this tutorial we'll be covering how to install SDL_image.

IMG_LoadJPG_RW
SDL_Surface ***IMG_LoadJPG_RW**(SDL_RWops **src*)

src
The source SDL_RWops as a pointer. The JPG image is loaded from this.

Load *src* as a JPG image for use as a surface, if JPG support is compiled into the SDL_image library.
Note: If the image format loader requires initialization, it will attempt to do that the first time it is needed if you have not already called IMG_Init to load support for your image format.

Returns: a pointer to the image as a new SDL_Surface. **NULL** is returned on errors, like if JPG is not supported, or a read error.

```
// load sample.jpg into image
SDL_Surface *image;
SDL_RWops *rwop;
rwop=SDL_RWFromFile("sample.jpg", "rb");
image=IMG_LoadJPG_RW(rwop);
if (!image) {
      printf("IMG_LoadJPG_RW: %s₩n", IMG_GetError());
      // handle error
}
```

Checked that SDL can load PNG and Jpeg images.

11. SDL Event Handling

SDL event handling system [Lesson 3]

```
Input (SDL_Eevent)              Event queue                          Check
        Press key                             SDL_KeyboardEvent
        SDL_PollEvent
        Move mouse                            SDL_MouseMotionEvent
        Joystick                     SDL_JoyButtonEvent
        Touch a touch screen
```

1) SDL_Event (union)
A SDL event is something like a key press, mouse motion, joy button press, etc.

SDL_Event
    Uint32
        SDL_Common/Window/QuitEvent
        SDL_KeyboardEvent
        SDL_MouseMotion/Button/WheelEvent
        SDL_JoyAxis/Ball/Hat/Button/DeviceEvent
        ……

Event queue
The event queue will then store them in the order the events occured waiting for you to process them.

SDL_PollEvent
When you want to find out what events occured so you can process them, you poll the event queue to get the most recent event by calling SDL_PollEvent. What SDL_PollEvent does is take the most recent event from the event queue and puts the data from the event into the SDL_Event we passed into the function.

SDL_PollEvent will keep taking events off the queue until it is empty. When the queue is empty, SDL_PollEvent will return 0. So what this piece of code does is keep polling events off the event queue until it's empty. If an event from the event queue is an SDL_QUIT event (which is the event when the user Xs out the window), we set the quit flag to true so we can exit the application.

```
//Handle events on queue
while ( SDL_PollEvent( &e ) != 0 ) {
    //User requests quit
    if ( e.type == SDL_QUIT ) {
        quit = true;
    }
}
```

2) SDL_KeyboardEvent [Lesson4]
https://wiki.libsdl.org/SDL_KeyboardEvent

Inside of the SDL Event is an SDL Keyboard event which contains the information for the key event.

Data Fields

Uint32	**type**	the event type; SDL_KEYDOWN or SDL_KEYUP
Uint32	**timestamp**	timestamp of the event
Uint32	**windowID**	the window with keyboard focus, if any
Uint8	**state**	the state of the key; SDL_PRESSED or SDL_RELEASED
Uint8	**repeat**	non-zero if this is a key repeat
SDL_Keysym	**keysym**	the SDL_Keysym representing the key that was pressed or released

3) SDL_Keysym
Inside of SDL_KeyboardEvent is a SDL Keysym which contains the information about the key that was pressed. That Keysym contains the SDL Keycode which identifies the key that was pressed.

Data Fields

SDL_Scancode	**scancode**	SDL physical key code; see SDL_Scancode for details
SDL_Keycode	**sym**	SDL virtual key code; see SDL_Keycode for details
Uint16	**mod**	current key modifiers; see SDL_Keymod for details
Uint32	**unused**	

SDL_Scancode & SDL_Keycode are defined for each key.
Keymod: None, L/Rshift, L/Rctrl, L/Ralt, …

4) SDL_Keycode

SDL_Event	= SDL_KeyboardEvent	> SDL_Keysym	> SDL_Keycode
	window/time/up or down	ScanCode/Keycode/Keymod	

SDL_Keycode Value

SDLK_0-9, a-z, F1-24,
SDLK_UP/DOWM/LEFT/RIGHT (Four arrow keys)

SDL Events Summary

SDL_Event

Uint32
- SDL_Common/Window/QuitEvent
- SDL_KeyboardEvent
- SDL_MouseMotion/Button/WheelEvent
- SDL_JoyAxis/Ball/Hat/Button/DeviceEvent
- **……**

SDL_KeyboardEvent

https://wiki.libsdl.org/SDL_KeyboardEvent

Uint32	type	// SDL_KEYDOWN, SDL_KEYUP
Uint32 timestamp		
Uint32	windowID	// The window with keyboard focus
Uint8	state	// SDL_PRESSED, SDL_RELEASED.
Uint8	repeat	// Nonzero of this is a key repeat
SDL_Keysym	keysym	// The key that was pressed/released

https://wiki.libsdl.org/SDL_Keysym

SDL_Scancode	scancode	// SDL_SCANCODE_0-9/A-Z/a-z
SDL_Keycode	sym	// SDLK_0-9/A-Z/a-z
Uint16	mod	// **SDL_Keymod** https://wiki.libsdl.org/SDL_Keymod // KMOD_L/R SHIFT/CTRL/ALT etc.
Uint32	unused	

Struct SDL_keysym

https://www.libsdl.org/release/SDL-1.2.15/docs/html/guideinputkeyboard.html

```
typedef struct{
  Uint8 scancode;
  SDLKey sym;
  SDLMod mod;
  Uint16 unicode;
} SDL_keysym;
```

C. Implement Video functionality.

See V. Design.

D. System Integration

See V. Design.

IV. Equipment and Parts

1. Lab equipment

Router
IBM PC with Windows and Linux (Dual boot)
Embedded board Beaglebone with cables and 4GB SD

2. Electronic parts

ID	Part No	Description	Qty/ group	Unit price
51	C110	Logitech WebCam	1	16,860
52	XH 400	Unicon 4-port USB Hub or equivalent	1	4,300

Note. In order to test with batteries, you need to charge batteries fully beforehand, using LiPo charger and NiMH charger.

V. Design

Pre-report for first week

1. **Search internet for C110 specifications. You can visit www.logitech.com..**

2. **Design SDL program for Key Values (Problem 5B)**

a. Design Get_Key_Var_SDL.cpp

Objective
Prints SDL_Scancode, SDL_Keycode, and SDL_Keymod (in hexadecimal) for each key pressed.
Find which data is useful.

Algorithm
```
1. init()
        SDL_Init()
        SDL_CreateWindow()
        SDL_GetWindowSurface()          // Get window surface
3. Event loop in main()
        A. SDL_PollEvent()
                Print Key Vars
9. close()
```

SDL_DestroyWindow()
SDL_Quit()

Edit
From the example Key_event program in Lesson 3 of SDL tutorial, insert:
```
        printf("Key scancode %xh, keycode %xh, keymod %xh.\n",
               e.key.keysym.scancode, e.key.keysym.sym, e.key.keysym.mod);
```

Find which variable is useful for key input: SDL_Scancode, SDL_Keycode, and SDL_Keymod.

b. Design Key_Value_SDL.cpp

Objective
Keycode is the same as ASCII value?
Get key input without Enter key (similar to raw key input)
for left 4x3 keys:
1, 2, 3;   q, w, e;   a, s, d;   z, x, c.
Coding

May Use symbols for Keycode afterwards.
https://wiki.libsdl.org/SDL_Keycode

Print keycode value as %02xh and %c & fflush(stdout)!
```
        kcode = e.key.keysym.sym;

        printf("%02xh ‘%c’; ", kcode, kcode);
        fflush(stdout);
```

Pre-report for second week

3. Design Video functionality (Problem 5C).

Video functionality is composed of

- Camera on Beaglebone (Capture from WebCam and send video to network) and
- Viewer on PC (Receive video from network and display video to user) using SDL2.

Data flow in detail

User	**PC**	**Network**	**Beaglebone**	**Robot**
SW:	**Viewer**		**Camera**	
			Capture images ←	← WebCam
			←Send Jpeg images	
		←UDP packets		
	Recv Jpeg images ←			
See video	←Display video SDL			

Camera.c on Bone
Use X.c for cross-gcc compatibility!
Modify Capture2.c to add UDP send functionality
Additional file: Send_UDP.c for UDP-related routines.

Jpeg format: Use 320x240 Jpeg images!

Video stream packet

Sequence of Jpeg images (Mjpg video from WebCam)
Variable size.

Viewer.cpp using SDL2 on PC

Main loop used for polling events.
Require a thread to listen from socket and display.

Additional file: Recv_UDP.cpp for UDP-related routines.

Algorithm for Camera.c on Bone

```
0. Get argument of Capture: CPORT (4960) to send
        Including -p for ports & -a for ip_addr.
1. Init UDP packet
5. Loop
        Capture Webcam to Jpeg image
        Sendto Jpeg image to UDP datagram
9. Close UDP packet
```

Algorithm for Viewer.c on PC (using SDL2)

Main

```
1. init()           // Init SDL2
2. Fill the surface with light grey & update the surface
3. Init UDP port with any IP and CPORT (4960) to listen
4. Run a thread RecvDispThread to listen to datagram and display Jpeg image
5. Key event loop
        Just print input key value.
```

RecvDispThread

```
Loop
        A. recvfrom() socket datagram
            print the number of received bytes
        B. Check if Jpeg image (Header & Trailer)
        C. Display Jpeg image using SDL2
```

Note. Two programs (one on PC, another on Bone) are going to be tested with simultaneous operation. Bugs may exist on both programs, which make debugging difficult.

One good method is debug program on Bone first (capture and send) with a small program on PC with receive only (without viewer with SDL): Modified from listener.c to simply check and print received packet size.

4. Design for System Integration (Problem 5D)

Allocate multi-tasking with thread

HW	PC	Beaglebone
SW: tasks	***Viewer: Thread*** Receive video from network and display video to user (SDL2). Viewer, which records photos and videos to storage device.	***Camera: Task*** Capture from WebCam and send video to network.
	Commander: Task Get key input from user and send command packet to network (SDL2). For recording photos and videos, Commander on PC sends user command to Viewer.	***Controller: Task*** Receive command packet from network and actuate servos and lights on the robot.

Why two tasks on Beaglebone?
Since two tasks of Camera and Controller are independent.

How to run two tasks on Beaglebone?
Use two terminals for Bone: One with minicom, and the orher using ssh.
Using two terminals, outputs of each program is not inter-mixed.

Why Thread for Viewer?
Since Viewer requires user key input (photo and video), it should be inside the same memory space. Hence we use Task for Commander, Thread for Viewer, in order to share variables. Note that both commander/viewer can be controlled by a single key input process inside SDL.

Directory

e_SystemI-Integration

Subdirectory to store photos and videos

Photo
Video

Files in Directory

Camera on Beaglebone

Camera.c, Send_UDP.c

Commander/Viewer using SDL2 on PC

Viewer.cpp (containing Viewer thread), Recv_UDP.cpp,
Commander.cpp, Send_Cmd.cpp.

Controller on Beaglebone

Control_TMR.c, Recv_Cmd.c, gpio_control.c, gpio_control.h,
Init_PWMSS.sh, Acquire_Triple_PWMs.sh, Release-Triple_PWMs.sh

Makefile

Dataflow with multi-task in detail

User	PC	Bone	Robot
			← See with WebCam
		Camera task	
		Capture WebCam ←	
		← Ssend img datagrams	
	Viewer thread		
	Recv img datagrams ←		
	← Display with SDL2.		
	Record photo/video.		
Watches display ←			
Issue key commands →			
(Servos, Lights, Photo, and Video)			
	Commander task		
	Get user key command in raw mode with SDL2		
	Classify internal/external command		
	Handle internal command (Photo & Video) to Viewer ^		
	Send external command to cmd datagram →		
		Control task	
		→ Recv cmd datagram	
		Control Servos and lights →	
			Move and illuminate

Network packets

Image packet

Single Jpeg image, variable size.

Command packet

ASCII string composed of Id, ti,=me, vx, vy, wr, RL, and LL.

Algorithm

Camera on Bone & Viewer thread are the same as Video functionality in Problem 5C.

Commander task is essentially the same as Keyboard_Commander_PC in Lab 4.
Control task is the same as WiFi_Control_TMR in Lab 4.

You need to add
Catch 'p' photo and 'v' video keys from user (without Enter).
It should be transferred to Viewer thread.

In the viewer thread, record to a file in either Photo or Video directory, with filename "photoNNN.jpg" or "videoNNN.mjpg", where NNN can range from 000 to 999 to store up to 1000 files. After storing file photo999.jpg, the next photo will be over-written to photo000.jpg. Using this scheme, error of "hard disk full" can be avoided..

VI. Lab Procedures

First Week

A. Capture WebCam on Bone

Test capture image from WebCam with V4L2 (Video for Linux 2) on Beaglebone.

11. Check for WebCam device and UVC device driver

First issue "lsusb" with connecting WebCam C110 to Bone via USB hub.

```
$ lsusb
Bus 001 Device 004: ID 046d:0829 Logitech, Inc.
…...
```

Device with ID 046d:0829 is detected, but cannot identify its name.

Check with "dmesg":

```
$ dmesg | grep Webcam
[    1.823628] usb 1-1.2: Product: Webcam C110
[   13.838397] uvcvideo: Found UVC 1.00 device Webcam C110 (046d:0829)
[   14.029731] input: Webcam C110 as /devices/ocp.3/47400000.usb/musb-hdrc.1.aut
o/usb1/1-1/1-1.2/1-1.2:1.0/input/input1
```

Logitech WebCam C110 is detected by Linux!

Check UVC device driver.
Linux 2.6.26 and newer includes the Linux UVC driver natively. You will not need to download the driver sources manually unless you want to test a newer version or help with development.

```
# lsmod | grep uvc
uvcvideo              57013  0
videobuf2_vmalloc       2490  1 uvcvideo
```

uvcvideo device driver module is already installed.

When you do "ls /dev", you should see video0 at the bottom:

```
# ls /dev
/dev/audio1
…...
/dev/video0
```

…...

12. Install Video4Linux2 and Test

Refer Image Capture with WebCam on Bone

"Beaglebone Images, Video and OpenCV", http://derekmolloy.ie/beaglebone-images-video-and-opencv/

Video4Linux or V4L is a video capture application programming interface for Linux, supporting many USB webcams, TV tuners, and other devices. Video4Linux is closely integrated with the Linux kernel.

Install v4l2 on Bone

```
# sudo apt-get install v4l-utils
# sudo apt-get install libv4l-dev
```

To list the devices:

```
# v4l2-ctl --list-devices
Webcam C110 (usb-musb-hdrc.1.auto-1.2):
        /dev/video0
```

To list the formats available:

```
# v4l2-ctl --list-formats
ioctl: VIDIOC_ENUM_FMT
        Index       : 0
        Type        : Video Capture
        Pixel Format: 'YUYV'
        Name        : YUV 4:2:2 (YUYV)

        Index       : 1
        Type        : Video Capture
        Pixel Format: 'MJPG' (compressed)
        Name        : MJPEG
```

Notice that C110 supports output format

YUYV

MJPG

Get the driver detail:

```
# v4l2-ctl --get-priority
Priority: 2

# v4l2-ctl -D
Driver Info (not using libv4l2):
        Driver name   : uvcvideo
        Card type     : Webcam C110
        Bus info      : usb-musb-hdrc.1.auto-1.1
```

```
	Driver version: 3.8.13
	Capabilities  : 0x84000001
		Video Capture
		Streaming
		Device Capabilities
	Device Caps   : 0x04000001
		Video Capture
		Streaming
```

We can list the controls available:

```
# v4l2-ctl -L
                     brightness (int)    : min=-64 max=64 step=1 default=0 value=0
                       contrast (int)    : min=0 max=30 step=1 default=13 value=13
                     saturation (int)    : min=0 max=127 step=1 default=38 value=38
                            hue (int)    : min=-16000 max=16000 step=1 default=0 value=0
 white_balance_temperature_auto (bool)   : default=1 value=1
                          gamma (int)    : min=20 max=250 step=1 default=100 value=100
           power_line_frequency (menu)   : min=0 max=2 default=1 value=1
                                0: Disabled
                                1: 50 Hz
                                2: 60 Hz
      white_balance_temperature (int)    : min=2800 max=6500 step=1 default=5000 value=5000 flags=inactive
                      sharpness (int)    : min=0 max=100 step=1 default=35 value=35
         backlight_compensation (int)    : min=0 max=1 step=1 default=0 value=0
                  exposure_auto (menu)   : min=0 max=3 default=3 value=3
                                1: Manual Mode
                                3: Aperture Priority Mode
              exposure_absolute (int)    : min=2 max=5000 step=1 default=312 value=312 flags=inactive
         exposure_auto_priority (bool)   : default=0 value=1
                   pan_absolute (int)    : min=-72000 max=72000 step=3600 default=0 value=0
                  tilt_absolute (int)    : min=-54000 max=54000 step=3600 default=0 value=0
                  zoom_absolute (int)    : min=1 max=2 step=1 default=1 value=1
```

Get all of the information available for the camera:

```
# v4l2-ctl --all
Driver Info (not using libv4l2):
	Driver name   : uvcvideo
	Card type     : Webcam C110
	Bus info      : usb-musb-hdrc.1.auto-1.1
	Driver version: 3.8.13
	Capabilities  : 0x84000001
		Video Capture
		Streaming
		Device Capabilities
	Device Caps   : 0x04000001
		Video Capture
		Streaming
Priority: 2
Video input : 0 (Camera 1: ok)
Format Video Capture:
	Width/Height  : 640/480
	Pixel Format  : 'YUYV'
	Field         : None
	Bytes per Line: 1280
	Size Image    : 614400
```

```
       Colorspace    : SRGB
Crop Capability Video Capture:
       Bounds      : Left 0, Top 0, Width 640, Height 480
       Default     : Left 0, Top 0, Width 640, Height 480
       Pixel Aspect: 1/1
Streaming Parameters Video Capture:
       Capabilities     : timeperframe
       Frames per second: 30.000 (30/1)
       Read buffers     : 0
                   brightness (int)    : min=-64 max=64 step=1 default=0 value=0
                     contrast (int)    : min=0 max=30 step=1 default=13 value=13
                   saturation (int)    : min=0 max=127 step=1 default=38 value=38
                          hue (int)    : min=-16000 max=16000 step=1 default=0 value=0
 white_balance_temperature_auto (bool)   : default=1 value=1
                        gamma (int)    : min=20 max=250 step=1 default=100 value=100
          power_line_frequency (menu)   : min=0 max=2 default=1 value=1
     white_balance_temperature (int)    : min=2800 max=6500 step=1 default=5000
value=5000 flags=inactive
                    sharpness (int)    : min=0 max=100 step=1 default=35 value=35
        backlight_compensation (int)    : min=0 max=1 step=1 default=0 value=0
                exposure_auto (menu)   : min=0 max=3 default=3 value=3
            exposure_absolute (int)    : min=2 max=5000 step=1 default=312 value=312
flags=inactive
        exposure_auto_priority (bool)   : default=0 value=1
                 pan_absolute (int)    : min=-72000 max=72000 step=3600 default=0
value=0
                tilt_absolute (int)    : min=-54000 max=54000 step=3600 default=0
value=0
                zoom_absolute (int)    : min=1 max=2 step=1 default=1 value=1
```

13. Control with v4l2

We can then set an individual value. For example to set the brightness:

```
# v4l2-ctl -L
                   brightness (int)    : min=0 max=255 step=1 default=128 value=128

# v4l2-ctl --set-ctrl brightness=200
# v4l2-ctl -L
                   brightness (int)    : min=0 max=255 step=1 default=128 valu e=200
…...
```

If we wish to modify the resolution:

```
# v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=1
```

Pixelformat: as shown in "v4l2-ctl –list-formats"

0:	YUYV
1:	MJPG

Check current format

```
# v4l2-ctl -V
```

```
Format Video Capture:
        Width/Height  : 640/480
        Pixel Format  : 'MJPG'
        Field        : None
        Bytes per Line: 0
        Size Image    : 921600
        Colorspace    : SRGB
        Custom Info   : feedcafe
```

14. Get capture.c program

The source video capture code is from:
file: media/v4l/capture.c
https://www.linuxtv.org/downloads/v4l-dvb-apis-new/uapi/v4l/capture.c.html
Download to a_CaptureBone/Capture_LinuxTV.c.
This is a generic example program to capture from WebCam using V4L2.

Copy this file to Capture2.c (to be modified).

15. Edit and cross-compile on PC

The original program captures 'frame_count' images, which is preset as 70.

```
        static int frame_count = 70;          // About L50. Use Menu: Search – Find

        static void mainloop(void)                              // About L170.
        {
                unsigned int count;

                count = frame_count;

                while (count-- > 0) {
```

Change mainloop() as

```
        static void mainloop(void)
        {
            unsigned int count;
            unsigned int loopIsInfinite = 0;         //+ Added for infinite loop

            if (frame_count == 0) loopIsInfinite = 1;       //+ Set infinite loop

            count = frame_count;

            //-while (count-- > 0) {
            while ((count-- > 0) || loopIsInfinite) {       //c Added loopIsInfinite
```

In the program, search for printf statements. Many "'₩n'"s are not correct. Change "₩₩n" or "n" to "₩n"..

Edit makefile to include:
Capture2: Capture2.c

Arm-linux-gnueabhif-gcc -g -o Capture2 Capture2.c

Build

```
$ make
```

16. Test Capture2 on Bone as root

a. See help message:

```
# ./Capture2 --help
Usage: ./Capture2 [options]
Version 1.3
Options:
-d | --device name   Video device name [/dev/video0]
-h | --help         Print this message
-m | --mmap         Use memory mapped buffers [default]
-r | --read         Use read() calls
-u | --userp        Use application allocated buffers
-o | --output       Outputs stream to stdout
-f | --format       Force format to 640x480 YUYV
-c | --count        Number of frames to grab [70]
```

Notice that the number of frames to grab is preset to "70": This is why we added infinite number of captures for video.

b. Capture YUYV as root

Run Capture2   to get a 640x480 YUYV image on Bone

```
# v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=0
# ./Capture2 -c 1 -o > cap640a.yuv
select timeout

# ls -la cap*
-rw-r--r-- 1 root   root        0 Feb 10 02:29 cap640a.yuv
```

It seems that 640x480 YUYV is too heavy to capture on Bone.

Run Capture2_PC to get a 320x240 YUYV image on Bone

```
# v4l2-ctl --set-fmt-video=width=320,height=240,pixelformat=0
# ./Capture2 -c 1 -o > cap320a.yuv
.
# ls -la cap*
-rw-r--r-- 1 root   root    153600 Feb 10 02:33 cap320a.yuv
-rw-r--r-- 1 root   root         0 Feb 10 02:29 cap640a.yuv
```

File size?

It contains 320x240 YUYV image data (2 bytes per pixel):

240 rows * 320 columns * 2 bytes = 153,600 bytes.

One more capture 320x240 YUYV image on Bone

```
# v4l2-ctl --set-fmt-video=width=320,height=240,pixelformat=0
# ./Capture2 -c 1 -o > cap320b.yuv
.
# ls -la cap*
-rw-r--r-- 1 root   root    153600 Feb 10 02:33 cap320a.yuv
-rw-r--r-- 1 root   root    153600 Feb 10 02:36 cap320b.yuv
-rw-r--r-- 1 root   root         0 Feb 10 02:29 cap640a.yuv
```

Notice that the file size of YUYV image is always a constant, depending on resolution.

See raw image? Can't.

You can see the raw data with hd (Hexadecimal dump):

```
 # hd cap320a.yuv | more
00000000  33 84 32 78 30 81 2f 79  32 80 34 7b 3b 80 3c 7a  |3.2x0./y2.4{;.<z|
00000010  3e 80 43 7a 43 80 44 78  49 85 47 76 40 87 33 7c  |>.CzC.DxI.Gv@.3||
00000020  46 80 76 83 e2 79 ff 81  ff 7b ff 7f ff 80 ff 80  |F.v..y...{......|
00000030  ff 80 ff 7f ff 85 ff 77  e2 91 ad 69 9d 95 92 63  |.......w...i...c|
00000040  92 91 8c 65 8c 90 8f 64  8f 94 7c 65 61 93 3b 6d  |...e...d..|ea.;m|
00000050  4c 89 61 76 6b 84 75 74  74 86 77 6e 77 87 7d 6e  |L.avk.utt.wnw.}n|
00000060  7e 87 80 71 82 88 84 72  85 87 88 73 89 87 8c 73  |~..q...r...s...s|
00000070  8a 86 8b 73 8b 84 8d 72  90 85 98 70 93 88 93 70  |...s...r...p...p|
00000080  94 86 96 6e 97 83 8d 6c  8f 83 9f 6f c7 83 ce 72  |...n...l...o...r|
```

Notice that four-byte patterns are repeated.

c. Capture JPEG and see

Run Capture2 to get a 640x480 jpeg image on Bone

```
# v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=1
# ./Capture2 -c 1 -o > cap640a.jpg
.
# ls -la cap*.jpg
-rw-r--r-- 1 root root 6948 Feb 10 02:46 cap640a.jpg
```

Cannot see the image by clicking the file in File browser on PC…

Seems that it is an invalid file.

Try mjpeg first!

```
# ./Capture2 -c 100 -o > cap640a.mjpg
................................................................................
..................…
```

```
# ls -la cap*.mjpg
-rw-r--r-- 1 root   root   6331668 Feb 10 02:49 cap640a.mjpg
```

Can see the mjpg video by clicking the file in File browser on PC!

Run Capture2 again to get a 640x480 jpeg image on Bone

```
# ./Capture2 -c 1 -o > cap640b.jpg
.
# ./Capture2 -c 1 -o > cap640c.jpg          // Aim at other view
.
# ls -la cap*.jpg
-rw-r--r-- 1 root root  6948 Feb 10 02:46 cap640a.jpg
-rw-r--r-- 1 root root 59792 Feb 10 02:51 cap640b.jpg
-rw-r--r-- 1 root root 23444 Feb 10 02:54 cap640c.jpg
```

Can see cap640b.jpg image by clicking the file in File browser on PC!
Notice that JPEG file size is variable even for the same resolution: Dependent on image complexity.

Question. Why capture Jpeg correctly only after capture mjpg?

d. Capture (record) MJPG video and see

Note. Fresh start: Disconnect and reconnect USB connector of WebCam.

Capture video 10 s twice.

```
# v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=1
 # ./Capture2 -c 300 -o > cap640b.mjpg
 # ./Capture2 -c 300 -o > cap640c.mjpg
```

Check file size

```
$ ls -la cap*.mjpg
-rw-r--r-- 1 root root 15629532  2월 10 12:07 cap640b.mjpg
-rw-r--r-- 1 root root 19904596  2월 10 12:08 cap640c.mjpg
```

capture2_bone_2.mjpg is 5,027,696 bytes or similar.

Can you see a MJpeg image?
Double click mjpg image cap640c.mjpg shows still image only!
Double click mjpg image cap640b.mjpg opens Gedit: Cancel it!!
Try this on PC:

```
 $ sudo apt-get install mplayer
 $ mplayer cap640c.mjpg
```

Can you see a video?

B. Learn SDL via Tutorial

20. Install SDL on PC

Refer: Install - Setting up SDL 2 on Linux
http://lazyfoo.net/tutorials/SDL/01_hello_SDL/linux/index.php

21. Create SDL2 window

Refer
Setting Up SDL2 on g++
http://lazyfoo.net/tutorials/SDL/01_hello_SDL/linux/cli/index.php

Hello SDL: Your First Graphics Window
http://lazyfoo.net/tutorials/SDL/01_hello_SDL/index2.php

Algorithm

```
1. Define SDL_Window
2. Define SDL_Surface
3. SDL_Init()
3.1 SDL_CreateWindow()          // Actually Step 4.
3.2 SDL_GetWindowSurface()      // Get window surface
3.3 SDL_FillRect()              // Fill window with specified RGB color
3.4 SDL_UpdateWindowSurface()
3.5 SDL_Delay()
8. SDL_DestroyWindow()
9. SDL_Quit()
```

View what kind of SDL functions are used.

Compile

```
$ g++ 01_hello_SDL.cpp -w -lSDL2 -o 01_hello_SDL
```

22. View a BMP image on screen

Refer: Tutorial 02 - Lesson 2. Getting an Image on the Screen
http://lazyfoo.net/tutorials/SDL/02_getting_an_image_on_the_screen/index.php

Algorithm

```
1. init()
 SDL_Init()
 SDL_CreateWindow()
 SDL_GetWindowSurface()  // Get window surface
2. loadMedia()
 SDL_LoadBMP()           // hello_world.bmp image file is loaded to a buffer.

3. SDL_BlitSurface()                  // Apply the image
4. SDL_UpdateWindowSurface()
```

5. SDL_Delay()   // 2s

9. close()
 SDL_FreeSurface()
 SDL_DestroyWindow()
 SDL_Quit()

23. Event driven programming

Here we'll start handling user input by allow the user to X out the window.

Refer Tutorial 03 – Lesson 03. Event Driven Programming
http://lazyfoo.net/tutorials/SDL/03_event_driven_programming/index.php

24. Key press

Here we'll learn to handle keyboard input.

Refer: Tutorial 04 – Lesson 04. Key Presses
http://lazyfoo.net/tutorials/SDL/04_key_presses/index.php

25. Image stretching

Now that we know how to load and blit surfaces, it's time to make our blits faster. We'll also take a smaller image and stretch it to fit the screen.

Refer: Tutorial 05 - Lesson 05. Optimized Surface Loading and Soft Stretching
http://lazyfoo.net/tutorials/SDL/05_optimized_surface_loading_and_soft_stretching/index.php

26. Load PNG and Jpeg image

Here we'll be using the SDL_image extension library to load png images.

Refer: Tutorial 06 - Lesson 06. Extension Libraries and Loading Other Image Formats
http://lazyfoo.net/tutorials/SDL/06_extension_libraries_and_loading_other_image_formats/index.php

Install

```
$ sudo apt-get install libsdl2-image-dev
```

Includes

Now you may get an error saying it can't find SDL_image.h. For linux, we'll have to include the SDL headers like this:

```
#include<SDL2/SDL_image.h>
```

For SDL_ttf and SDL_mixer, we have to include them like this:

```
#include<SDL2/SDL_ttf.h>
#include<SDL2/SDL_mixer.h>
```

Change this program to load Jpeg image.
Test with a 320x240 captured Jpeg image, and view stretched image.
You can use v4l2-ctl to change the image size, and then capture a Jpeg image.
Can you see the captured Jpeg image?

27. Thread

Multithreading allows your program to do things simultaneously. Here we'll make things print to the console from outside our main thread.

Refer Lesson 46. Multithreading
http://lazyfoo.net/tutorials/SDL/46_multithreading/index.php

This thread will be used to implement Viewer on PC.

28. Test Get_Key_Var_SDL.

Test prepared program.

29. Test Key_Value_SDL.

Test prepared program.
These key values will be used in Problem 5C.

C. Test Video Functionality

31. Make a working directory

Use a suitable direcory name, e.g.,

```
        # mkdir -p 5_WebCam_SI/c_VideoFunctionality
```

32. Edit source files

Edit the following source files

Camera on Bone
- Camera.c
- Send_UDP.c

Viewer using SDL2 on PC
- Viewer.cpp
- Recv_UDP.cpp

Note that we use SDL2 in Viewer, which should be compiled with g++.
Hence we name source files as X.cpp.

33. Edit Makefile

Since we need to cross-compile for Bone, and native compile with g++ for PC,
Makefile can be constructed as follows:

```
## 5_WebCam_SI / c_VideoFunctionality_SDL_320  Makefile

all: Camera  Viewer
```

```
## CC specifies which compiler we're using
CC = g++

# COMPILER_FLAGS specifies the additional compilation options we're using
# -w suppresses all warnings
COMPILER_FLAGS = -g -w

# LINKER_FLAGS specifies the libraries we're linking against
LINKER_FLAGS = -lSDL2
LINKER_FLAGS2 = -lSDL2 -lSDL2_image

## Camera on Bone
Camera: Camera.c Send_UDP.c
	arm-linux-gnueabihf-gcc -g -o Camera  Camera.c Send_UDP.c

## Viewer using SDL2 on PC
Viewer: Viewer.cpp Recv_UDP.cpp
	$(CC) -o Viewer Viewer.cpp Recv_UDP.cpp $(COMPILER_FLAGS) $(LINKER_FLAGS2)


clean:
	rm *~ Camera Viewer
```

34. Build

Type

```
 $ make clean
 $ make
arm-linux-gnueabihf-gcc -g -o Camera  Camera.c Send_UDP.c
g++  -o Viewer Viewer.cpp Recv_UDP.cpp -g -w  -lSDL2 -lSDL2_image
```

35. Test

Camera on Bone as root

Set WebCam format and resolution as 320x240 Jpeg using v4l2:

```
 # v4l2-ctl --set-fmt-video=width=320,height=240,pixelformat=1
```

Confirm the Webcam format

```
# v4l2-ctl -V
Format Video Capture:
       Width/Height  : 320/240
       Pixel Format  : 'MJPG'
       Field        : None
       Bytes per Line: 0
       Size Image    : 230400
       Colorspace    : SRGB
```

Run camera to send with destion IP of PC and port of 4960:

```
 # ./Camera -a 192.168.100.12 -p 4960
```

```
Args: Dev /dev/video0, fmt 2, io 1, fc 0, IP 192.168.100.12, port 4960
Init socket to IP 192.168.100.12 with port 4960
……
```

Viewer using SDL on PC

Simply, type

```
$ ./Viewer
Got a socket 6 with 4960.
Running RecvDisp Thread with data = 2
Recv_UDPs 0 24424, done 24424B.  Display done.
Recv_UDPs 1 24452, done 24452B.  Display done.
……
```

Success?

36. Test 2

Set WebCam format and resolution as 640x480 Jpeg using v4l2:

```
# v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=1
```

Repeat the test.
Can you send Jpeg image with size larger than 64K?

D. Test System Integration

Working directory
DesignLab/5_WebCam_SI/d_SystemIntegration.

Perform test on System Integration.

If successful, demonstrate to TA!

VII. Final Report

Discussion for the following question should be included in the report.

1) Compare formats: YUYV and Jpeg.

2) Search image storage format in commercial digital still camera and video camera.

3) Can you suggest another multitasking method for RoboCam? Can you do multitasking on PC without thread?

4) Is there any suitable graphic user interface other than SDL2? If yes, compare this method with SDL2.

5) Can one RoboCam on Bone support multiple simultaneous users?

6) Set your own topic to discuss related to Lab 5, and explain summary of your search result.

VIII. References

[1] Beaglebone Rev A6 System Reference Manual, http://beagleboard.org/static/beaglebone/latest/Docs/Hardware/BONE_SRM.pdf

[2] AM3359 Datasheet, AM335x ARM Cortex-A8 Microprocessors (MPUs) (Rev. J), http://www.ti.com/lit/ds/symlink/am3359.pdf

[3] Technical Reference Manual - AM335x ARM Cortex-A8 Microprocessors (MPUs) Technical Reference Manual (Rev. O), http://www.ti.com/lit/ug/spruh73o/spruh73o.pdf