**Embedded System Software프로젝트**

**Webcam control App & Device**

**(결과 보고서)**

**과목명: [CSE4116] 임베디드시스템소프트웨어**

**담당교수: 서강대학교 컴퓨터공학과 박 성 용**

**학번 및 이름: 120230200, 김지섭**

**개발기간: 2023. 06. 01. - 2023. 06. 26.**

**I. 개발 목표**

한 학기 동안 습득한 리눅스 시스템 프로그램 및 안드로이드 프로그래밍 기법을 이용해 실습 보드 상에서 자유 주제로 임베디드 소프트웨어를 개발한다.

**Project 목표:**

* Embedded Board에 USB webcam을 연결하여, Android App에 해당 영상을 송출한다.
* Webcam이 원하는 위치를 촬영할 수 있도록 방향을 조절할 수 있도록 한다.
* 사진 촬영 및 저장 후 Android의 갤러리에서 해당 이미지를 확인할 수 있도록 한다.

**II. 개발 범위 및 내용**

**가. 개발 범위**

Webcam과 Motor를 JNI를 활용하여 Control하는 Android Application 구현

Motor Device driver와 Video for Linux2를 활용한 JNI 구현

Push switch와 motor의 device driver를 활용한 Motor control application 구현

1. Android Application 예시



**나. 개발 내용**

1. System Architecture

스크린샷, 텍스트, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

도표, 라인, 스크린샷, 평행이(가) 표시된 사진

자동 생성된 설명

1. Flow chart

텍스트, 도표, 라인이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

대부분의 Code가 순차적으로 수행되기 때문에 OnTouch와 OnClick의 차이점에 대해 보여주는 것으로 Flow chart를 그렸습니다.

OnTouch의 경우 Release된 경우 Motor를 멈추는 JNI 함수를 수행하지만, 사진을 촬영하는 onClick의 경우 최초 click이 확인되면, 이후의 touch 여부는 중요하지 않기때문에 기능적 차이와 flow적 차이가 존재합니다.

1. Android Application 구현

|  |
| --- |
| public class MainActivity extends Activity implements OnTouchListener, OnClickListener {  private static final String TAG = "WebCam";  private native int moveMotorRight();  private native int moveMotorLeft();  private native int stopMotor();  private native byte[] capture();  ImageView webCam;  Bitmap bitmap = null;  byte[] source = null;  Thread thread;  static{  System.loadLibrary("motor\_jni");  };  } |

Main Activity 선언부 입니다. JNI를 사용하기 위해 System.loadLibaray(“motor\_jni”)를 활용하여, libmotor\_jni.so 파일을 load합니다.

|  |
| --- |
| @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  Log.d(TAG, "Start Application");  setContentView(R.layout.fragment\_main);  View leftButton = findViewById(R.id.left\_button);  leftButton.setOnTouchListener(this);  View rightButton = findViewById(R.id.right\_button);  rightButton.setOnTouchListener(this);  View shootButton = findViewById(R.id.shoot\_button);  shootButton.setOnClickListener(this);  webCam = (ImageView) findViewById(R.id.web\_cam);    if(thread == null) {  thread = new Thread(new WebCamImage());  }  thread.start();    } |

onCreate 입니다. ImageView를 지속적으로 갱신하기 위해 uiThread를 사용했고, 이때, 별도의 thread를 구현하기 위햇 thread를 생성한 후 thread를 실행시켰습니다.

|  |
| --- |
| @Override  public boolean onTouch(View v, MotionEvent event) {  // TODO Auto-generated method stub  switch(v.getId()) {  case R.id.right\_button :  switch(event.getAction()) {  case MotionEvent.ACTION\_DOWN:  // PRESSED  Log.d(TAG, "Pressed Right");  Log.d(TAG, String.valueOf(moveMotorRight()));  return true; // if you want to handle the touch event  case MotionEvent.ACTION\_UP:  // RELEASED  Log.d(TAG, "Released Right");  Log.d(TAG, String.valueOf(stopMotor()));  return true; // if you want to handle the touch event  }  case R.id.left\_button :  switch(event.getAction()) {  case MotionEvent.ACTION\_DOWN:  // PRESSED  Log.d(TAG, "Pressed Left");  Log.d(TAG, String.valueOf(moveMotorLeft()));  return true; // if you want to handle the touch event  case MotionEvent.ACTION\_UP:  // RELEASED  Log.d(TAG, "Released Left");  Log.d(TAG, String.valueOf(stopMotor()));  return true; // if you want to handle the touch event  }  }  return false;  } |

onTouch 함수입니다. SW버튼을 누르는 동안 모터가 돌아가고, 손을 뗴는 순간 모터가 멈추는 것을 구현하기 위해 강의 시간에 배운 onClickListener가 아닌 onTouchListener를 활용하였습니다. 이를 통해, 손을 버튼에서 떼는 것 또한 event로 인식하여, 해당 event 발생시 motor를 멈추는 function을 실행할 수 있었습니다.

|  |
| --- |
| public void onClick(View v) {  // TODO Auto-generated method stub  switch(v.getId()){  case R.id.shoot\_button:  SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd-hh-mm-ss");  Log.d(TAG, formatter.format(new Date()));  OutputStream out = null;  try {  File file = new File("/sdcard/DCIM", formatter.format(new Date()) + ".png");  file.createNewFile();  out = new FileOutputStream(file);  Context c = this.getBaseContext();  c.sendBroadcast(new Intent(Intent.ACTION\_MEDIA\_SCANNER\_SCAN\_FILE, Uri.fromFile(file)));  bitmap.compress(Bitmap.CompressFormat.PNG, 80, out);  } catch (FileNotFoundException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  } |

onClick 함수입니다. 촬영버튼을 누를시, 이미지를 저장하는 로직을 만들었습니다.

저장된 이미지를 갤러리에서 확인할 수 있도록 카메라 이미지가 저장되는 경로인 DCIM에 이미지를 저장하였습니다.

이때, application 실행 중에는 이미지가 disk에 저장되지 않는 문제를 발견했고, 해당 문제가 이미지 파일을 memory에 저장하고, disk로 flush하지 않기 때문이라는 것을 파악하였습니다.

이를 해결하기 위해 context를 bradcast하는 code를 추가하였습니다.

Context c = this.getBaseContext();

c.sendBroadcast(new Intent(Intent.ACTION\_MEDIA\_SCANNER\_SCAN\_FILE, Uri.fromFile(file)));

|  |
| --- |
| class WebCamImage implements Runnable{  Handler handler = new Handler();  boolean hasImageAlready = (webCam.getDrawable() != null);  @Override  public void run() {  // TODO Auto-generated method stub  while(true) {  // get Image  Log.d(TAG, "Get Image");  source = capture();  if(source != null) {  Log.d(TAG, "Image Not NULL : ");  // decode Image  ByteArrayOutputStream out = new ByteArrayOutputStream();  YuvImage yuvImage = new YuvImage(source, ImageFormat.YUY2, 640, 480, null);  yuvImage.compressToJpeg(new Rect(0, 0, 480, 360), 100, out);  byte[] imageBytes = out.toByteArray();    bitmap = BitmapFactory.decodeByteArray(imageBytes, 0, imageBytes.length);  Log.d(TAG, bitmap.getConfig().name());    if(bitmap != null) {  Log.d(TAG, "Bitmap Not NULL");  runOnUiThread(new Runnable() {  @Override  public void run() {  // TODO Auto-generated method stub  webCam.setImageBitmap(bitmap);  Log.d(TAG, "Set Image");  }  });  }  }  }  }  } |

1. JNI구현

|  |
| --- |
| #define LOG\_TAG "jniTag"  #define LOGD(...) \_\_android\_log\_print(ANDROID\_LOG\_VERBOSE, LOG\_TAG, \_\_VA\_ARGS\_\_)  #define FPGA\_STEP\_MOTOR\_DEVICE "/dev/fpga\_step\_motor"  #define VIDEO\_DEVICE "/dev/video2"  #define VIDEO\_WIDTH 640  #define VIDEO\_HEIGHT 480  #define VIDEO\_FORMAT V4L2\_PIX\_FMT\_YUYV  jint JNICALL Java\_com\_example\_androidex\_MainActivity\_moveMotorRight(JNIEnv \*env, jobject this){  int dev;  unsigned char motor\_state[3] = {1, 0, 50};  LOGD("Start SUCCESS");  dev = open(FPGA\_STEP\_MOTOR\_DEVICE, O\_WRONLY);  LOGD("DEV : %d", dev);  if (dev<0) {  return -1;  }  write(dev,motor\_state,3);  close(dev);    return 1;  } |

Motor를 구동하는 JNI코드입니다. Right, left, stop 총 3가지 함수가 존재하며, motr\_stat를 각 기능별로 다르게 하여, 모터가 지정된 동작을 수행할 수 있도록 하였습니다.

|  |
| --- |
| JNIEXPORT jbyteArray JNICALL Java\_com\_example\_androidex\_MainActivity\_capture(JNIEnv \*env, jobject this){  int video\_fd;  struct v4l2\_format format;  struct v4l2\_requestbuffers reqbuf;  struct v4l2\_buffer buf;  video\_fd = open(VIDEO\_DEVICE, O\_RDWR);  if (video\_fd == -1) {  LOGD("Failed to open video device");  return NULL;  }  // Set video format  format.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;  format.fmt.pix.width = VIDEO\_WIDTH;  format.fmt.pix.height = VIDEO\_HEIGHT;  format.fmt.pix.pixelformat = VIDEO\_FORMAT;  format.fmt.pix.field = V4L2\_FIELD\_NONE;  if (ioctl(video\_fd, VIDIOC\_S\_FMT, &format) == -1) {  LOGD("Failed to set video format");  close(video\_fd);  return NULL;  }  // Request video buffers  reqbuf.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;  reqbuf.memory = V4L2\_MEMORY\_MMAP;  reqbuf.count = 1;  if (ioctl(video\_fd, VIDIOC\_REQBUFS, &reqbuf) == -1) {  LOGD("Failed to request video buffers");  close(video\_fd);  return NULL;  }  // Map video buffers  struct v4l2\_buffer buffer\_info;  buffer\_info.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;  buffer\_info.memory = V4L2\_MEMORY\_MMAP;  buffer\_info.index = 0;  if (ioctl(video\_fd, VIDIOC\_QUERYBUF, &buffer\_info) == -1) {  LOGD("Failed to query video buffer");  close(video\_fd);  return NULL;  }  void \*buffer\_start = mmap(NULL, buffer\_info.length, PROT\_READ | PROT\_WRITE, MAP\_SHARED, video\_fd, buffer\_info.m.offset);  if (buffer\_start == MAP\_FAILED) {  LOGD("Failed to map video buffer");  close(video\_fd);  return NULL;  }  // Start video streaming  enum v4l2\_buf\_type type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;  if (ioctl(video\_fd, VIDIOC\_STREAMON, &type) == -1) {  LOGD("Failed to start video streaming");  close(video\_fd);  return NULL;  }  // Capture image  if (ioctl(video\_fd, VIDIOC\_QBUF, &buffer\_info) == -1) {  LOGD("Failed to enqueue video buffer");  munmap(buffer\_start, buffer\_info.length);  close(video\_fd);  return NULL;  }  if (ioctl(video\_fd, VIDIOC\_DQBUF, &buffer\_info) == -1) {  perror("Failed to dequeue video buffer");  munmap(buffer\_start, buffer\_info.length);  close(video\_fd);  return NULL;  }  jbyteArray bytes = (\*env)->NewByteArray(env, buffer\_info.length);  (\*env)->SetByteArrayRegion(env, bytes, 0, buffer\_info.length, buffer\_start);  munmap(buffer\_start, buffer\_info.length);  close(video\_fd);  return bytes;  } |

Webcam의 이미지를 java로 보내기 위한 JNI입니다. 삼성 Webcam으로, General한 환경에선 삼성에서 자체적으로 개발한 device driver가 제공되기 때문에, 이미지 처리 format이나 다른 문서가 존재하지 않았습니다. 관련 정보를 찾던 중 Video 4 Linux를 지원한다는 자료를 확인하여, 해당 driver로 구현하였습니다.

1. C Code 구현

|  |
| --- |
| #define MAX\_BUTTON 9  #define MOTOR\_ATTRIBUTE\_ERROR\_RANGE 4  #define FPGA\_STEP\_MOTOR\_DEVICE "/dev/fpga\_step\_motor"  int move(int move, int right\_left)  {  int dev;  unsigned char motor\_state[3] = {move, right\_left, 50};  dev = open(FPGA\_STEP\_MOTOR\_DEVICE, O\_WRONLY);  if (dev<0) {  return -1;  }  write(dev,motor\_state,3);  close(dev);    return 1;  } |

Motor를 움직이는 코드로, push switch 버튼의 입력 유무에 따라 동작되는 함수입니다.

|  |
| --- |
| int main(void)  {  int i;  int dev;  int buff\_size;  int moved = 0;  unsigned char push\_sw\_buff[MAX\_BUTTON];  dev = open("/dev/fpga\_push\_switch", O\_RDWR);  if (dev<0){  printf("Device Open Error\n");  close(dev);  return -1;  }  (void)signal(SIGINT, user\_signal1);  buff\_size=sizeof(push\_sw\_buff);  printf("Press <ctrl+c> to quit. \n");  while(!quit){  usleep(400000);  read(dev, &push\_sw\_buff, buff\_size);  if(push\_sw\_buff[3] == 1 && push\_sw\_buff[5] == 0)  {  moved = 1;  move(1, 1);  }  else if(push\_sw\_buff[3] == 0 && push\_sw\_buff[5] == 1)  {  moved = 1;  move(1, 0);  }  else if(moved ==1)  {  move(0,0);  moved = 0;  }  }  close(dev);  } |

Push switch 버튼의 입력을 받아 모터를 조작하는 코드입니다.

지속적으로 push switch의 stat를 읽어오고, 4,6번 버튼에서 손이 떨어지는 경우 모터를 멈추도록 구현했습니다.

**III. 추진 일정 및 개발 방법**

1. **추진 일정**

**텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명**

1. **개발 방법**

개발해야 하는 기능들을 구분하여 waterfall 방식으로 개발하였습니다. 우선 프로젝트의 가능여부 판단을 위해 webcam에서 image를 캡쳐하는 code를 구현한 후, 해당 이미지 파일을 여러 format에 맞춰 확인했습니다.

전자제품, 텍스트, 전자 기기, 케이블이(가) 표시된 사진

자동 생성된 설명

이를 통해 해당 webcam에서 이미지를 처리하는 format을 확인할 수 있었습니다.

이후 Android Application의 목업을 구현하고, JNI와 물리버튼을 통해 Motor를 조작하는 코드를 구현하였습니다. 마지막으로 웹캠 이미지를 해당 format에 맞춰 android에서 제공하는 JPG 형태로 변환하였고, 이를 android의 imageView로 보내 화면이 나오도록 구현하였습니다.

**IV. 연구 결과**

Linux의 기본 Device driver인 Video for Linux2를 활용하여, Webcam 영상을 화면에 송출하고, 수업시간에 배운 내용들을 활용하여, 모터와 버튼을 통해 Webcam을 조작하는 Application과 Embedded Device를 구현하였습니다.

Webcam Device의 Memory 구조, Data 처리 방식 등 Device에 대한 정보가 없어, 정확한 Format을 찾고, 해당 format을 android에서 지원하는 format으로 변환하는 과정에서 많은 Overhead가 있었습니다.

향후 외부 Device에서의 Control이나, Device Driver Custom 등 다양한 것들을 추가적으로 진행하면 좋은 공부가 될 것 같습니다.