Gesture Detection

Camera Based Gesture Tracking System Controlling Audio Output

A camera is used to detect gestures made in the camera's field of view using LEDs. Detected gestures will be used to control a music player.

Final Report

Winter 2016

Group 7

Patrick Kuczera
Andrew Zhong
Shahzeb Asif

Abstract

The goal of this project is to detect controlled movement of LEDs on a flashlight as specific gestures. A camera will be used to track the location of LEDs on the flashlight. Incoming frames will be continuously processed using a VHDL component to find the position of the illuminated point created by the LEDs. The entire image frame will be reduced to a simple x,y coordinate representation of the position of the LEDs. The detected coordinates will be compared to stored gesture sequences. When a gesture is recognized, the name of the recognized gestures will be displayed on the LCD screen on the DE2. Recognized gestures will be used to control a music player on the DE2 that can be played or paused using the correct gestures. The volume of the music player can also be controlled using gestures. Video output will show the LED position tracking in real time.

Table of Contents

Functional Requirements

The purpose of this project is to recognize gestures using a camera to track LEDs provided by an LED flashlight. A digital camera, in our case a Nikon S8200, is used to provide video-in for the DE2. The incoming video frames are then processed to find the position of the LED. This position information is compared to stored gesture sequences. Recognized gestures are used to control a music player on the DE2. Video-out to a LCD monitor in 640x480 resolution is also supported to show live LED detection.

Position of LED Detection in Video Frame
The DE2's video-in line and the onboard TV decoder chip is used to decode the incoming data from a camera. The frame will be analysed to detect a brightly lit point from the LEDs in the frame. This is done using a hardware component to provide the best performance. The position of the LEDs in the image frame are exported from the component. The component must be capable of processing >10 frames/second.

The component was created and can process >45 frames/second. But it has interference issues in areas with other bright lights. A possible extension to fix this issue is in Appendix D3.

Gesture Detection
The LEDs' position information is analysed in a uCOS task to detect gestures. Multiple frames need to be analysed over a certain period of time to be able to detect gestures. The task is capable of detecting gestures using a constant stream of position information with no clear beginning or end. Similarly, the program does not require the flashlight to be turned off and on to separate different gestures. Gestures will be detected according to predefined gesture sequences. The detected gestures will be displayed on the LCD screen of the DE2.

Gesture detection was completed as expected. The current implementation is best at recognizing simple gestures such as straight lines. It has issues with complex gestures. A more complicated approach may be necessary to improve recognition of complex gestures.

Music Control
WAV music files can be played from an SD card to speakers connected to the DE2 board. Specific gestures are assigned to specific controls on the music player allowing the music track to be controlled using gestures to play, pause, increase/decrease volume.

Music control was successfully implemented apart from some minor static issues with the audio playback. These could be fixed with one or two more weeks of work.

Real-Time Video
A real-time output feed of input video is displayed on a monitor. The location of the LED is also displayed on the video feed.

The video-out was mostly implemented. The location of the LED is not explicitly labeled but it can be clearly seen if the threshold has been set correctly. Explicit labeling of the LED location could be an extension for the future.

Design & Description of Operation

In order to describe the design and operation of our project, a hardware block diagram demonstrating the required components will first be analysed.

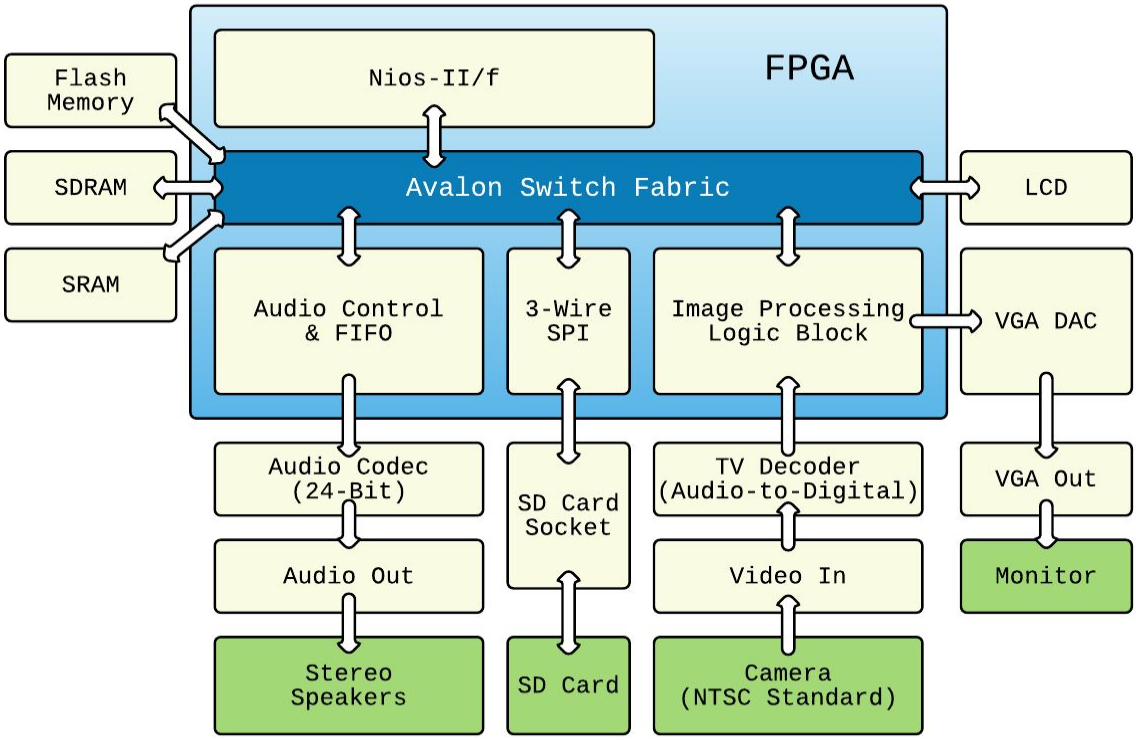


Figure 1: Hardware Block Diagram

This block diagram shows a majority of the hardware required in order to implement the functionality of our project. The image processing logic block is abstracted and can be seen in fuller detail in Figure 2. All interfaces required are proprietary ports directly on the Terasic Altera DE2-115 development board. External devices required

are shown in green and include stereo speakers, an SD card, a camera, and a 4:3 ratio monitor.

The user initially interacts with the camera by waving an LED flashlight in specific patterns creating physical gestures within the camera's field of view. NTSC standard video is then streamed from the camera to the RCA video in port where it is further passed to the ADV7181 TV Decoder chip on the DE2 board. A collection of transformations are then applied to the video stream as shown in Figure 2.

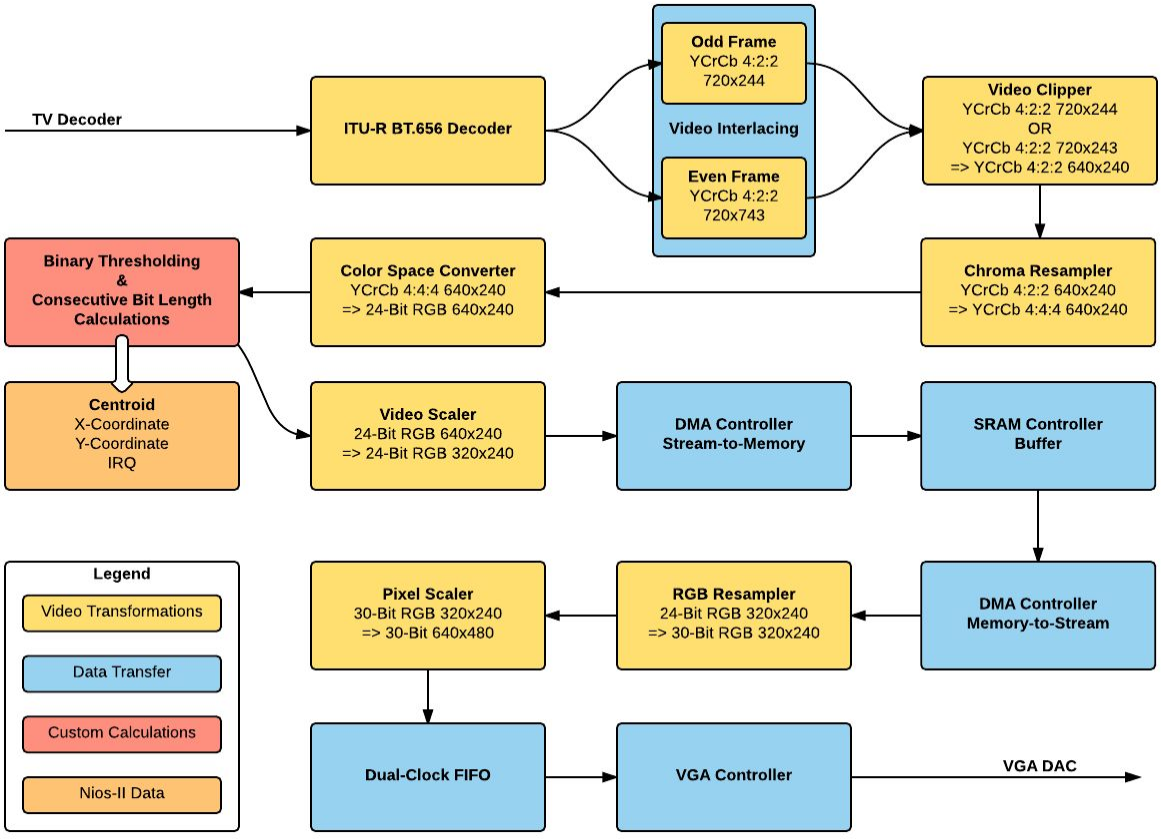


Figure 2: Image Processing Logic Block

The Altera Video IP Cores[1] were utilized in applying transformations onto the video stream. Originally, the digital frames obtained from the decoder are interlaced at 720x244 (Odd) and 720x243 (Even) sizes, and formated to YCrCb 4:2:2. Being a non-ideal size ratio, these frames are then clipped down to 640x240 frames. In order to eventually display live video on the monitor, the video frames need to be formatted to RGB which requires upscaling the data bus size. This is initially done using a chroma resampler which converts the 640x240 YCrCb 4:2:2 frames to YCrCb 4:4:4 frames. Now that the data bus has been scaled up to 24-bit it can be reformatted to the desired RGB format using a color space converter. At this point **Binary Thresholding** and

Consecutive Bit Length Comparisons are applied to each frame in order to determine the centroids of our gesture tracking sequences. The frames are then scaled down to 320x240 in order to fit into SRAM. After temporary buffering each frame, resampling occurs bringing the data up to 30-bit RGB which is required by the VGA ADV7123 DAC chip. Prior passing frames to the VGA controller, the frames are also scaled back up to 640x480 and are passed through a Dual Clock FIFO since our system clock runs at 75Mhz and the VGA requires a video stream at 25 Mhz Clock. Video can then be streamed from the VGA port to a monitor for viewing live stream video.

Once centroids are calculated by the VHDL component in the video stream, they are passed off to the Nios-II/f processor using a memory mapped slave in conjunction with a custom interrupt. A gesture task running under uCOS will then cross reference the change in position of the centroids with Trie Data structures[5], which are further described in our Software Design section. These data structures store the gesture sequences required for matching to specific instructions. When a gesture is hit, the associated instruction will be displayed on the LCD screen by the LCD task in order to ensure proper mapping has occurred.

Instructions, including the ability to pause/play music and increase/decrease volume of audio output are applied against the audio task. In order to pause/play, the functions OSTaskSuspend()/OSTaskResume() are respectively used. This works effectively since the audio task interacts with the audio FIFO in hardware by first reading .WAV bytes from the SD card via SPI protocol and then writing those bytes to the FIFO. If the FIFO is empty, then audio playback is effectively paused until more bytes are loaded. The volume is handled in a slightly different way, instead of interacting with the audio task, it interacts directly with the Wolfson WM8731 audio codec[17]. By modifying the LHPVOL and RHPVOL values in the Left Headphone Out and Right Headphone Out registers respectively, the volume of audio playback can be controlled.

Binary Thresholding

The frames will further be reduced to a binary image by checking the intensity of the pixels and either assigning 0 (black) or 1 (white) depending where the pixel sits relative to a threshold. This will give a simple data form in which the centroid can be calculated from.

$$out(x, y) = 1 \; if \; in(x, y) > Threshold; \; else \; 0.$$

Consecutive Bit-Length Comparison

In order to calculate the centroid in an efficient manner that uses minimal hardware, a process that looks for the longest length of consecutive white pixels (or 1's) was used. As a frame is streamed through the process line by line, a counter will keep track of the line (y coordinate), and a counter paired with a register will keep track of the horizontal position and length (x coordinate). Whenever a larger length is achieved for

any given frame, the x position and length will be updated. At the end of the frame, the absolute x position will be calculated based on the offset from the original x coordinate and half the maximum length recorded. This absolute (x,y) pair represents the centroid that is passed to the Nios-II processor using an IRQ and a memory mapped slave.

The **Data Diagram** can be seen in the Software Design section.

Bill of Materials

Altera/Terasic DE2 Development Board
Part number: P0301
Unit cost: $495.00 USD
Quantity: 1
Total cost: $495.00 USD
Supplier: http://www.terasic.com.tw/cgi-bin/page/archive.pl?No=30
Datasheet:
http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=53&No=30&PartNo=4

Nikon COOLPIX S8200
Part number: 26288
Unit cost: $499.99 USD
Quantity: 1
Total cost: $499.99 USD
Supplier:
http://www.amazon.com/Nikon-COOLPIX-Digital-Discontinued-Manufacturer/dp/B005IGVXO8
Datasheet:
http://cdn-10.nikon-cdn.com/pdf/manuals/kie88335f7869dfuejdl=-cww2/S8200EN.pdf

Generic LED Flashlight (Example)
Part number: Cree XM-L Q5
Unit Cost: $6.11 CAD
Quantity: 1
Total cost: $6.11 CAD
Supplier:
https://www.amazon.ca/UltraFire-Zoomable-Flashlight-Adjustable-Waterproof/dp/B00WTV2AEC/ref=sr_1_1?ie=UTF8&qid=1459872134&sr=8-1&keywords=led+flashlight
Datasheet:
https://www.amazon.ca/UltraFire-Zoomable-Flashlight-Adjustable-Waterproof/dp/B00

WTV2AEC/ref=sr_1_1?ie=UTF8&qid=1459872134&sr=8-1&keywords=led+flashlight#productDetails

Apacer 2GB SD Card
Part number: AP-ISD02GIS2B-T
Unit cost: $7.98 CAD
Quantity: 1
Total cost: $7.98 CAD
Supplier:
http://ca.mouser.com/ProductDetail/Apacer/AP-ISD02GIS2B-T/?qs=sGAEpiMZZMv2YtEh%252bLCbYkVh9x8Erp%2feN0DQCeWKVTo%3d
Datasheet: http://www.mouser.com/ds/2/24/Industrial-SD_20120217-779539.pdf

Lenovo ThinkVision Monitor
Part number: 9419-HC2
Unit cost: N/A
Quantity: 1
Total cost: N/A
Supplier: ECE Department
Datasheet: N/A

Available Sources

Altera Video IP Cores[1]
The video IP cores can be used to help process video data on the Altera DE2.

Altera Audio IP Cores[2]
The audio IP cores can be used to help process audio data on the Altera DE2.

NTSC to VGA using Altera University Program IP Cores[7]
Appnotes from 2013 referenced for constructing video input & output components in SOPC Builder.

G4 Coloured Object Tracking Camera[10]
This project from 2015 will be referenced in order to help with image processing.

SD Card Interfacing[9]
These app notes from 2013 will be helpful with SD card interfacing.

Embedded Filesystem Library Library[15]

Library used to interface SD card.

G12 Network-Controllable Embedded MP3 Player[16]
Project from 2015 referenced in order to help with audio.

Data Sheet

Performance Evaluation
LED Detection
The performance of the LED detection component can be discussed with respect to two qualities: frames per second, detection rate.

The LED detection component operates on a stream of video frames and can process >45 frames per second easily satisfying the >10 frames per second requirement specified at the start of the project.

The detection rate of the LED heavily depends on the environment. The component searches for the largest, bright blob in the frame so it can be easily confused if more than one very bright light is in the frame. The detection rate is poor in environments with many bright lights. But if thresholding is correctly adjusted and the environment has even lighting, the detection rate is close to perfect.

Gesture Recognition
The gesture recognition can be evaluated in terms of the complexity of gestures it can recognize and the detection rate of gestures.

The gesture recognition algorithm can only handle fairly simple gestures such as straight lines. Any fine movements, such as an S shape, or movements that are too quick cannot be reliably detected.

The detection rate of gesture is about 80% assuming the position information coming in is fairly accurate. The detection rate for complex gestures is much lower.

Music Control
The performance of the music control can be talked about in terms of the quality of the audio and the effectiveness of the commands.

The audio can be played with fair quality but the audio output can have some audio static.
The music commands operate well and as expected when they are recognized.

Real-Time Video
The video is an accurate video produced by the camera. It is outputted at >45 frames per second and updates to match different threshold values almost instantly.

User Perspective Block Diagram

Operating Conditions
For satisfactory operation of this project, a dark environment is recommended. This is to avoid unnecessary sources of light that we wouldn't want to pick up when we threshold our video. The operating temperature is restricted to 15 ℃ to 32 ℃, from the DE2 board. It is recommended to avoid rapid changes in temperature, to prevent condensation build-up on the devices. It is also important to note to not

point strong sources of light at the lens of the camera for an extended period of time. This may cause the degradation of the image sensor of the camera.

I/O Signals

The interfacing for this projects adheres to components found on the DE2-115 board for all of the signal connections. As such, information regarding all required pin connections was obtained from the User Manual [8].

Clock, Switch, & LED Signals

Signal Name	FPGA Pin No.	Description
CLOCK_27	PIN_D13	27 Mhz Clock
CLOCK_50	PIN_N2	50 Mhz Clock
SW[0-17]	Multiple Pins	Thresholding Switches
LEDR[0-17]	Multiple Pins	Red LEDs
KEY[0]	PIN_G26	System Reset Button

SD Card Signal

Signal Name	FPGA Pin No.	Description
SD_DAT	PIN_AD24	MISO to SPI
SD_DAT3	PIN_AC23	SS from SPI
SD_CLK	PIN_AD25	Sclk from SPI
SD_CMD	PIN_Y21	MOSI from SPI

Video In Signal

Signal Name	FPGA Pin No.	Description
TD_D0	PIN_39	Video Pixel Output D0
TD_D1	PIN_E8	Video Pixel Output D1
TD_D2	PIN_H8	Video Pixel Output D2
TD_D3	PIN_H10	Video Pixel Output D3
TD_D4	PIN_G9	Video Pixel Output D4
TD_D5	PIN_F9	Video Pixel Output D5

TD_D6	PIN_D7	Video Pixel Output D6
TD_D7	PIN_C7	Video Pixel Output D7
TD_VS	PIN_K9	Vertical Synchronization Output Signal
TD_HS	PIN_D5	Horizontal Synchronization Output Signal
TD_RESET	PIN_C4	System Reset Input
I2C_SCLK	PIN_A6	I2C Serial Clock Input
I2C_SDAT	PIN_B6	I2C Serial Data Input/Output
TD_CLK	PIN_C16	Video Clock

Video Out Signal

Signal Name	**FPGA Pin No.**	**Description**
VGA_R[0-9]	Multiple Pins	VGA Red[0-9]
VGA_G[0-9]	Multiple Pins	VGA Green[0-9]
VGA_B[0-9]	Multiple Pins	VGA Blue[0-9]
VGA_CLK	PIN_B8	VGA Clock
VGA_BLANK	PIN_D6	VGA BLANK
VGA_HS	PIN_A7	VGA H_SYNC
VGA_VS	PIN_D8	VGA V_SYNC
VGA_SYNC	PIN_B7	VGA SYNC

Audio Out Signal

Signal Name	**FPGA Pin No.**	**Description**
AUD_DACLRCK	PIN_C6	Audio CODEC DAC LR Clock
AUD_DACDAT	PIN_A4	Audio CODEC DAC Data
AUD_XCK	PIN_A5	Audio CODEC Chip Clock
AUD_BCLK	PIN_B4	Audio CODEC Bit-Stream Clock
I2C_SCLK	PIN_A6	I2C Data

I2C_SDAT	PIN_B6	I2C Clock

LCD Signal

Signal Name	FPGA Pin No.	Description
LCD_DATA[0-7]	Multiple Pins	LCD Data[0-7]
LCD_RW	PIN_K4	LCD Read/Write Select, 0/1
LCD_EN	PIN_K3	LCD Enable
LCD_RS	PIN_K1	LCD Command/Data Select, 0/1
LCD_ON	PIN_L4	LCD Power ON/OFF
LCD_BLON	PIN_K2	LCD Backlight ON/OFF

SDRAM Signal

Signal Name	FPGA Pin No.	Description
DRAM_ADDR[0-11]	Multiple Pins	SDRAM Address[0-11]
DRAM_DQ[0-15]	Multiple Pins	SDRAM Data[0-15]
DRAM_BA_0	PIN_AE2	SDRAM Bank Address[0]
DRAM_BA_1	PIN_AE3	SDRAM Bank Address[1]
DRAM_LDQM	PIN_AD2	SDRAM Low-Byte Data Mask
DRAM_UDQM	PIN_Y5	SDRAM High-Byte Data Mask
DRAM_RAS_N	PIN_AB4	SDRAM Row Address Strobe
DRAM_CAS_N	PIN_AB3	SDRAM Column Address Strobe
DRAM_CKE	PIN_AA6	SDRAM Clock Enable
DRAM_CLK	PIN_AA7	SDRAM Clock
DRAM_WE_N	PIN_AD3	SDRAM Write Enable
DRAM_CS_N	PIN_AC3	SDRAM Chip Select

SRAM Signal

Signal Name	FPGA Pin No.	Description
SRAM_ADDR[0-11]	Multiple Pins	SRAM Address[0-17]
SRAM_DQ[0-15]	Multiple Pins	SRAM Data[0-15]
SRAM_WE_N	PIN_AE10	SRAM Write Enable
SRAM_OE_N	PIN_AD10	SRAM Output Enable
SRAM_UB_N	PIN_AF9	SRAM High-Byte Data Mask
SRAM_LB_N	PIN_AE9	SRAM Low-Byte Data Mask
SRAM_CE_N	PIN_AC11	SRAM Chip Enable

Flash Memory Signal

Signal Name	FPGA Pin No.	Description
FL_ADDR[0-21]	Multiple Pins	FLASH Address[0-21]
FL_DQ[0-7]	Multiple Pins	FLASH Address[0-7]
FL_CE_N	PIN_V17	FLASH Chip Enable
FL_OE_N	PIN_W17	FLASH Output Enable
FL_RST_N	PIN_AA18	FLASH Reset
FL_WE_N	PIN_AA17	FLASH Write Enable

Power Consumptions

Source	Voltage [V]	Current [mA]	Consumption [W]
DE2	9.06	552	5
Camera	3.7	1.05	3.885e-3
Monitor			40
Total			45

Background Research

Background research for this project was started by looking at the **Colored Object Tracking Camera (G4-2015) project** [10]. That project solved some similar problems to ours. That project appears to process frames as a stream of data rather than storing to memory and rereading. This is the same course we take. Our exact implementation of reducing image frames down to just a position value is influenced by this project.

To detect an LED in an image depends on the ability to remove unnecessary information from the image. We use binary thresholding in such a way that we are left with just the LED and very little noise in the image. A paper called **Face Recognition Using Binary Thresholding for Features Extraction** [4] was published in 1999. It details using binary thresholding to find facial features. The paper uses a more complicated process than we may need to but the basic process of subtracting frames and then using binary thresholding to find a feature was considered for our project.

To detect gestures, we can use a data structure called a **Trie** to do all the heavy lifting for us [5]. A trie stores key-value pairings using prefixes to organize itself. We can use this idea and store gesture sequences in a trie which will allow us to store gestures as a series of data points. This can be seen in Figure 5 in Software Design. It will also make it easier to process a stream of data with no clear beginning or end because it is in some sense a directed graph as well which lets us quickly invalidate invalid gestures compared to something like a string comparison approach.

Software Design

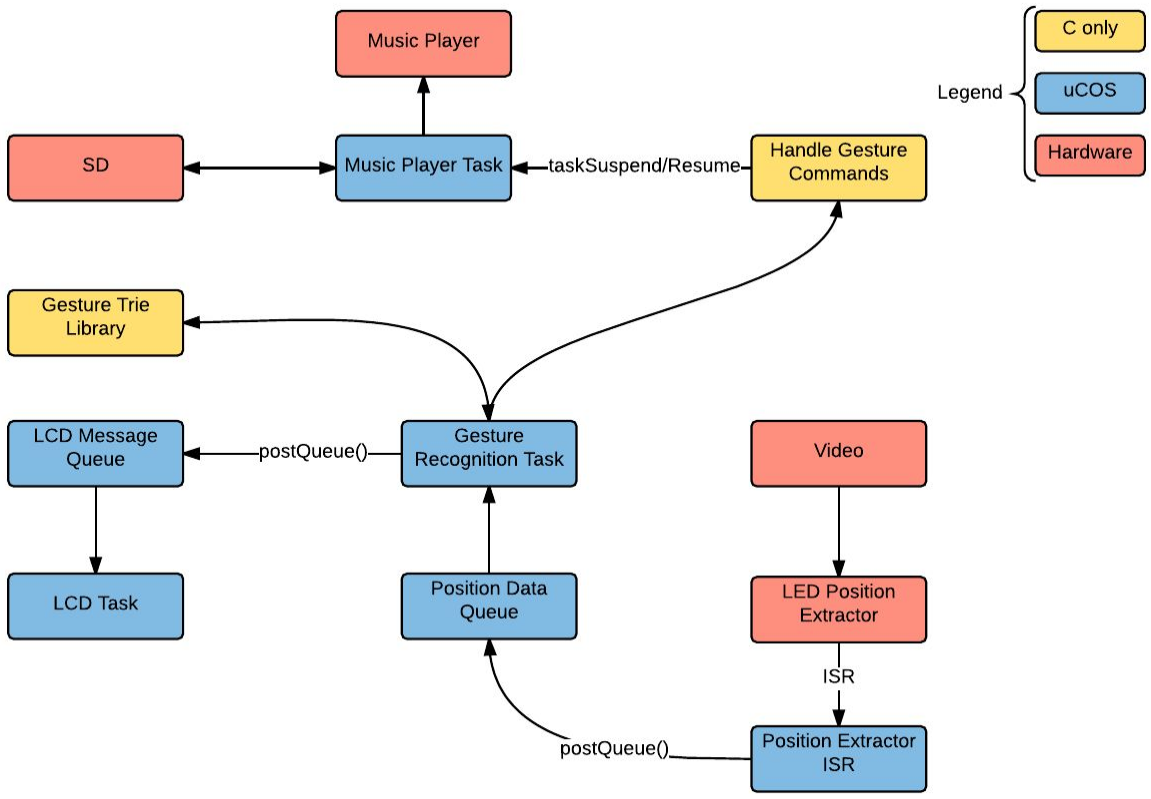


Figure 4: Software Flow/Data Diagram

The software flow can be seen above. Note that this program uses **uCOS**.

The LED position extractor will be done in a VHDL component. The purpose of the component is to take a stream of incoming frames and for each frame, perform a series of operations to remove the background and enhance the LED. The end result will be a position vector that describes the LED position in the frame. The VHDL component will then trigger an ISR to let the software know that it's processed a frame.

The ISR will post to a semaphore when the exception is raised. The Position Retrieve Task will pend() on the semaphore and extract the position from the VHDL component and put it on the Position Data Queue.

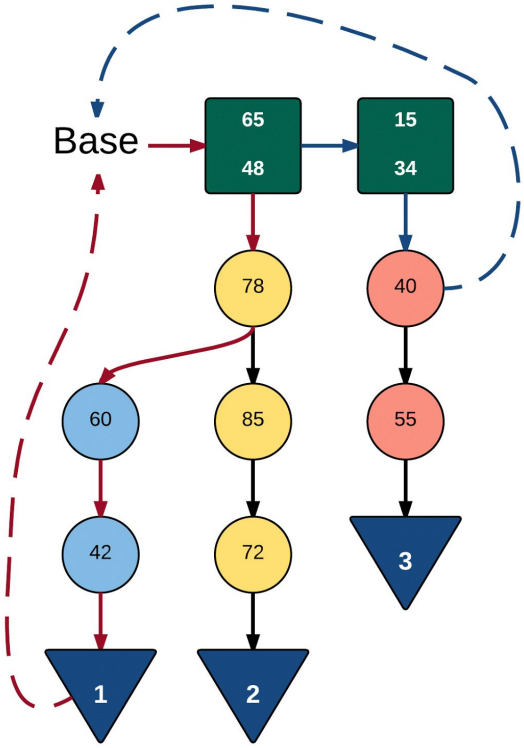


Figure 5: Gesture Trie

The Gesture Recognition Task will make up a large bulk of the software. The incoming position is converted to a grid number from the original image coordinates. The grid numbers are obtained by dividing up the image space into a grid with each grid location being assigned a grid number. The grid number is then stored or compared to the stored data.

The data structure is best illustrated with an example. Note the *Base* of the structure above. At the start of the program, a pointer will point at the base and wait for position information to come in. The red path indicates the path a pointer takes for a recognized gesture. At first, the program will compare a movement, defined by pairs of grid numbers, that the user makes to the list of pairs stored at the top of the data structure. These are basically the start points of many gestures.

For the red path, the user makes a movement that is roughly the same angle as the stored (65, 48) grid number pair. This is considered a hit and the current pointer is moved to the (65, 48) node. The user once again makes a movement that is roughly the same angle as (48, 78) and is registered as a hit. The pointer is now at the (78) node. Note that, once again, pairs of grid numbers are compared.

Next, the user could theoretically make two different movements that could both be hits because gesture #1 and #2 shared a common prefix sequence. The user makes a movement that matches the (78, 60) pair and the pointer is moved once more to (60) and eventually, assuming another hit, to (42).

At this point, the program will recognize gesture #1 and reset back to the base and start detecting gestures from scratch once more.

Gesture misses occur in a somewhat similar way. Note the blue path in the diagram above. The (65, 48) node does not match the user's movement so the program compares the next node. The (15, 34) does match so the pointer moves to that node. The next movement also matches so the pointer is moved to the (40) node. But now the user makes a movement that does not match the stored gesture. The program will now reset the pointer and once again start detecting from the Base.

When a gesture has been detected, it will be posted to the LCD Message Queue which will be extracted by the LCD Task and posted to the LCD.

The music player task runs on its own playing a file off the SD card continuously. The task initializes all the components it needs and then suspends itself. The task is resumed when a specific assigned gesture is recognized and the music is played again. Similarly, the music is paused by suspending the task once more. To raise and lower the volume, a function provided by Group 12 Network-Controllable Embedded MP3 Player from 2013 will be used.

Test Plan

The testing is difficult to do on the actual DE2 board. Most of the testing occurred separately with individual modules and staged data rather than all together testing.

There are three major components to test for this project:

- Position extraction
- Gesture detection
- Music player

The position extraction will be done using a VHDL component. It can be tricky to test a VHDL component because there are a lot of problems that could occur that may be related to hardware and could be irrelevant to the actual position recognizing function. Testing for this function has been done in a proof-of-concept prototype made in MATLAB that can be found in the Appendix B. The algorithm has been checked for

accuracy using the prototype. The tests largely involve whether the LED is detected and if the position reported by the algorithm is accurate.

Once the component was created, it was tested for accuracy and reliability and verified to be working accurately. Video out was tested in conjunction with the detector component and verified to be working correctly.

The gesture analysis was also, largely, tested independent of the other modules. The position extractor prototype can be used to stage data in CSV format. This data was fed to the gesture analysis module and checked to see if the gesture detection was being done correctly. The gesture detection module was tested for memory leaks and accuracy. The gesture detection was also be tested for false positives and false negatives and the effect of speed and sampling rates on detection rates.

Once the LED detector component was working, the gesture module was combined with the component on the DE2 and tested with live data. Adjustments were made to the component to improve detection of live data.

The music player was, similarly, tested by itself. Music was read off an SD card and played out the speakers. The audio was checked for accuracy and noise.

Once the music module was working, it was combined with the other two modules. Gestures were assigned to given music player commands and verified to be functioning.

Overall, incremental testing was performed. When the LED position extraction and the gesture detection modules were confirmed to be working independently, they were connected and tested as a group. Similarly, the music player was also connected together and tested in conjunction with the other two modules.

Results of Experimentation & Characterization

The MATLAB LED detector prototype was tested using both complicated built-in algorithms in the MATLAB image processing toolbox and a simple pixel-by-pixel approach. The built-in algorithms like the Hough transform, using the imfindcircles() function, are much faster in MATLAB than the pixel-by-pixel approach. The pixel-by-pixel approach can only get around ~2 frames/second on a desktop computer. The VHDL implementation of the same algorithm gave ~45 frames/second.

Gesture recognition has also been attempted in a few different ways. The first approach was to change the incoming data points into discrete predefined directions.

The advantage of this approach was the simplicity but there were problems with edge cases. The next approach was to store and compare an angle for a data point compared to the previous data point. This works better than the last approach but it can be improved by storing relative length as well.

Eventually, three different approaches were compared using the mocked data and the computer prototype.

Two gestures were stored from a long sequence of raw position data. The two gestures were assigned to be #10 and #20. #20 was expected to be more difficult to detect.

The results of the testing are below:

	r=1, n=1		**r=3, n=5**		**r=5, n=5**		**r=8, n=5**	
	#10	#20	#10	#20	#10	#20	#10	#20
queue (ang, len)								
5,5	1	1	4	1	2	0	0	0
10,10	1	1	3	5	3	4(+1)	1	0
20,5	1	1	4	2	3	2	3	0
grid (size, thresh)								
5,3	1	1	4	2	4	3	0	1
10,2	1	1	5	4	3	3	4	0
20,2	1(+1)	0	5(+7)	abandoned				
20,1	1	1	0	2	3	0	1	0
master (ang, len)								
5,5	1	1	3	4	4	2	1	0
10,10	0	1	3	2	2(+1)	3	0	1
20,5	1	1	5	5	4	0	0	0

r = randomizer parameter (higher makes data more erratic)
n = number of trials
(+x) = false positives detected

The left column contains settings for each approach.

The testing showed that the best approach was the grid system at those settings. This result was carried forward to incremental testing and the grid system was chosen to be used for the real project.

Safety

A hazard with the operation of this project would be to keep the power outlet clear of other cords, cables or anything that may be conductive for safe operation.
Another hazard would be the operation of the battery. It must be kept in a cool, dry area. In the case that the battery ruptures, flush exposed skin with lukewarm water for 15 minutes and seek medical attention for exposure with eyes or internal digestive system. In the case of fire, use a self-contained breathing apparatus to avoid inhalation of hazardous products/vapor. [6]

Max Voltage: 9v from the operation of the DE2 board

Max Power: 11.7W from the DE2 board

Stored Energy: 1500mAh* 1.5V * 3600s/h = 8100J from a battery to power LED

Operating Temperature: 15 ℃ to 32 ℃ by the DE2 board

Regulatory & Society

With this project, there would be no violations of privacy or storage of personal data, unless intended to by the operator. Within the present goals of the project, we create the gestures to control music. When custom gestures is implemented, the operator may choose gestures that may be sexual, racist, or deemed inappropriate by the majority of society. The music can also be selected by the user, as it will be a music file stored on a SD card, the user would be free to choose their own music that others may find inappropriate. There would be no worries of the board or camera being remotely hacked as there will be no connection to any servers being made through either device.

Environmental Impact

The batteries in this project contain manganese dioxide which may be harmful if inhaled or swallowed. An alternative to using batteries would be to power the LED through a power outlet. However, it is ROHS compliant as it does not contain lead, mercury, cadmium, hexavalent chromium, polybrominated biphenyls, polybrominated diphenyl ethers, bis phthalate, benzyl butyl phthalate, dibutyl phthalate or diisobutyl phthalate.
Our other components of this project are also ROHS compliant and they include the:

- Altera/Terasic DE2 Development Board
- Nikon COOLPIX S8200
- LEDs
- Apacer 2GB SD Card

Sustainability

DE2 Power Consumption

$P_{DE2} = I_{DE2} \times V_{DE2} = 0.552A \times 9.06V = 5W$

Camera Power Consumption

$P_{camera} = I_{camera} \times V_{camera} = 1.05mA \times 3.7V = 3.885W \sim negligible$

Monitor Power Consumption

$P_{monitor} = 40W$ [11]

Total Power Consumption

$P_{total} = 5W + 40W = 45W$

Yearly Power Consumptions

Assuming 2 hours per day:

$P_{total} = 45W \times 3600s/h \times 2h \times 365days = 118260000\ J/year = 32.85\ kWh/year$

Yearly CO2 Production

$Production_{CO_2} = 32.85\ kWh \times 909\frac{g}{kWh} = 29.86\ kg$ of CO_2 [12]

Area of Solar Cells Required

$A_{solar} = \frac{(45 \times 3600 \times 2)0.0172\ kWh)}{6\ hrs} \div \frac{80}{1000} \times 0.856\frac{m^2}{panel} = 0.0267m^2$ [13][14]

References

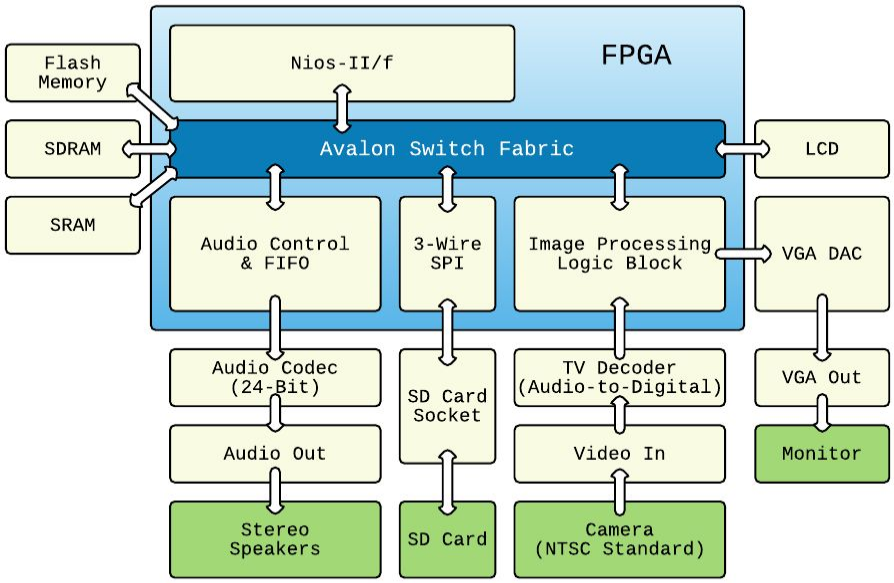
[1]	Altera Corporation. (August 2012) “Video IP Cores” [Online] Last accessed: Feb 28, 2016 ftp://ftp.altera.com/up/pub/Altera_Material/12.1/University_Program_IP_Cores/Audio_Video/Video.pdf
[2]	Altera Corporation. (May 2012) “Audio Cores” [Online] Last accessed: Feb 16, 2016 ftp://ftp.altera.com/up/pub/Altera_Material/12.1/University_Program_IP_Cores/Audio_Video/Audio.pdf
[3]	Altera Corporation. (October 2012) “Secure Data Card IP Core” [Online] Last accessed: Feb 16, 2016 ftp://ftp.altera.com/up/pub/Altera_Material/12.1/University_Program_IP_Cores/Memory/SD_Card_Interface_for_SoPC_Builder.pdf
[4]	IEEE Xplore. (September 1999) “Face Recognition Using Binary Thresholding for Features Extraction” [Online] Last accessed: Feb 19, 2016 http://ieeexplore.ieee.org.login.ezproxy.library.ualberta.ca/xpls/abs_all.jsp?arnumber=797742&tag=1
[5]	Boston University. (1993-2000) “Trie” [Online] Last accessed: Feb 18, 2016 https://www.cs.bu.edu/teaching/c/tree/trie/
[6]	Spectrum Brands, Inc. Rayovac Division (2014) “Rayovac Safety Data Sheet” [Online] Last accessed: Feb 29, 2015 http://cdn.spectrumbrands.com/~/media/Rayovac/Rayovac%20US/Files/Material%20Safety%20Data%20Sheet%20PDFs/Manganese%20Dioxide%20Battery%20Mercury%20and%20Lead%20Free.ashx
[7]	Gerry Finlay. (2013, March 27) “NTSC to VGA using Altera University Program IP Cores” [Online] Last accessed: Feb 19, 2015 https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/G15_Video_Out/G15_SOPC_Video
[8]	Altera Corporation. (2006) “DE2 Development and Education Board - User Manual v1.4” [Online] Last accessed: Jan 29, 2015 ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf
[9]	Jason Brown, Brady Thornton. (2013, February 21) “SD Card Interfacing” [Online] Last accessed: Feb 18, 2015 https://www.ualberta.ca/~delliott/local/ece492/appnotes/2013w/SD_card_interfacing/
[10]	Ryan Corpuz, Hang Peng, Jingjing Liang. (2015) “Coloured-Object Tracking Camera” [Online] Last accessed: Feb 25, 2015

	https://www.ualberta.ca/~delliott/local/ece492/projects/2015w/G4_Coloured_Object_Tracking_Camera/G4_final_Coloured_Tracking_Camera.pdf
[11]	Newegg. (2016) “lenovo Thinkvision Tilt, swivel & height adjustable standing L192p 19” 20 ms LCD Monitor 250 cd/m2 1000:1” [Online] Last accessed: Feb 29, 2016 http://www.newegg.com/Product/Product.aspx?Item=N82E16824146049
[12]	BlueSkyModel. (2016) “1 kilowatt-hour” [Online] Last accessed: Feb 29, 2016 http://blueskymodel.org/kilowatt-hour
[13]	Newegg. (2016) “Sunforce 39810 80 Watt Solar Panel” [Online] Last accessed: Feb 29, 2016 http://www.newegg.ca/Product/Product.aspx?Item=N82E16882260011&nm_mc=KNC-GoogleAdwordsCA-PC&cm_mmc=KNC-GoogleAdwordsCA-PC-_-pla-_-Solar-_-N82E16882260011&gclid=COz599aKnssCFQdqfgodsUcLIA
[14]	Solar Power Is The Future (2013) “How Many Solar Panels Does It Take to Make one Kilowatt? Calculating the Number of Panels Required for Your Home Energy Needs” [Online] Last accessed: Feb 29, 2016 http://www.solarpoweristhefuture.com/how-many-solar-panels-to-make-one-kilowatt.shtml
[15]	Lennart Ysboodt, Michael De Nil. “Embedded Filesystems Library” [Online] Last accessed: Mar 31, 2016 https://sourceforge.net/projects/efsl/
[16]	Brady Thornton, Jason Brown. “Network-Controllable Embedded MP3 Player” [Online] Last accessed: April 8, 2016 https://www.ualberta.ca/~delliott/local/ece492/projects/2013w/g12_EmbeddedMP3/g12_EmbeddedMP3Player_FinalReport_Web.pdf
[17]	Wolfson Microelectronics. WM8731/WM8731L Datasheet. [Online] Last accessed: April 5, 2016 http://www.cs.columbia.edu/~sedwards/classes/2008/4840/Wolfson-WM8731-audio-CODEC.pdf

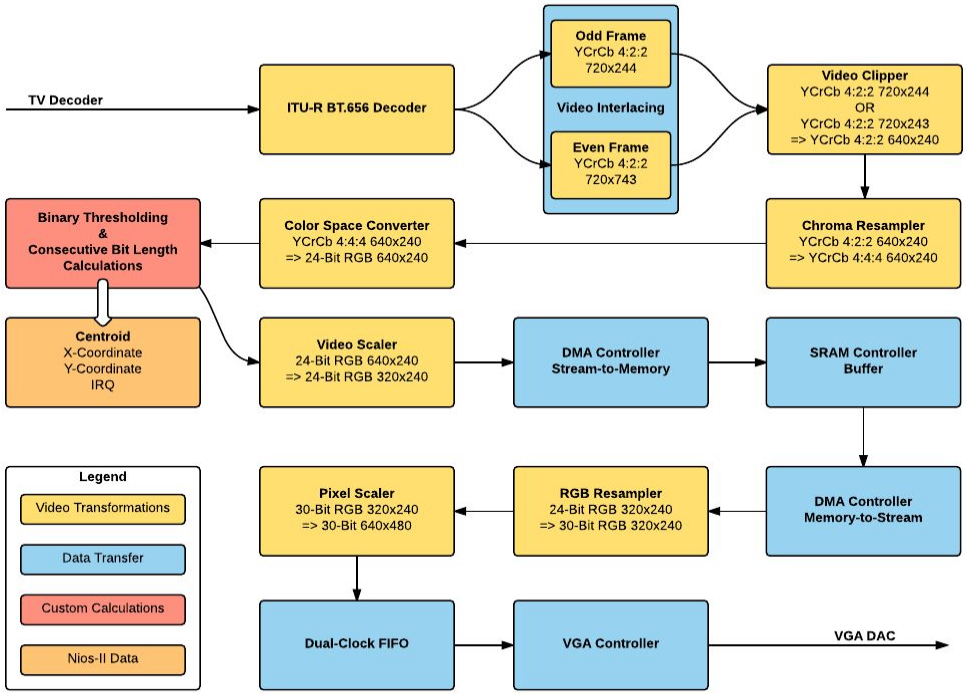
Appendix

A: Hardware Documentation

A1: Hardware Block Diagram



A2: Image Processing Logic Block



B: Source Code

All source code located at https://github.com/ece492-w16-g7/ece492_w16.git.

B1: LED Position MATLAB Prototype & Tools

https://github.com/ece492-w16-g7/ece492_w16/tree/master/image_position_extractor

B2: Gesture Detection Prototype

https://github.com/ece492-w16-g7/ece492_w16/tree/master/gesture_detection/prototype

B3: LED Centroid Detection (VHDL)

https://github.com/ece492-w16-g7/ece492_w16/blob/master/niosII_gesture_detection/vhdl/LED_RGB_detector.vhd

B4: uCOS Directory

https://github.com/ece492-w16-g7/ece492_w16/tree/master/niosII_gesture_detection/software/g7_gesture_detection_nv

B5: main.c

https://github.com/ece492-w16-g7/ece492_w16/blob/master/niosII_gesture_detection/software/g7_gesture_detection_nv/main.c

B6: Non-Volatile Project

https://github.com/ece492-w16-g7/ece492_w16/tree/master/niosII_gesture_detection

B6: Project Video

https://www.youtube.com/watch?v=zju_IC2Ss6c&feature=youtu.be

C: Quick Start Manual

Hardware setup:

1. Connect the power to the DE2.
2. Connect the USB to the USB Blaster port and the computer USB port.
3. Power a monitor and connect to DE2 via VGA port.
4. Turn on camera and connect to the DE2 via the video-in port.
5. Obtain a flashlight.

Steps to recreate software project:

1. Download and open the Quartus project from Appendix B6.
2. Open QSys and generate the .qsys file.
3. Compile the project.
4. Program the board.
    a. Ensure that interference from light sources apart from the flashlight are minimized.
    b. Flip the left-most switch on the DE2 to go from normal mode to thresholding mode.
    c. Use the switches on the DE2 to set the thresholding value such that all external light sources are minimized.
5. Create a folder called “software” inside the quartus project folder.
6. Extract the software zip from Appendix B3 into “software”.
7. Open Eclipse from Quartus.
8. Import the projects that was unzipped into “software”.
9. Generate the BSP.
10. Go into system.h and edit LEGACY_INTERRUPT_API to ENHANCED_INTERRUPT_API.
11. Build the project.
12. Run the project as hardware.

D: Future Work

D1: Custom Gestures

The ability to record and store custom gestures can be added. Custom gestures can be added in addition of the preset gestures. Pressing a button should trigger custom gesture recording mode. The LCD will direct the user to perform a gesture within a given time frame so that it can be recorded as a new custom gesture.

D2: Multiple Audio Playback

Be able to playback multiple audio tracks overtop of one another. New gestures should be used to change the song.

D3: Color or Object Tracking

The VHDL component could be modified to track a specific color or object shape rather than simple binary thresholding. This may make it more resistant to interference from other light sources.

D4: More Audio Control

Add the ability to control the tempo of the music. Add the ability to control the pitch and other music properties. New gestures should be assigned to control these new controls.