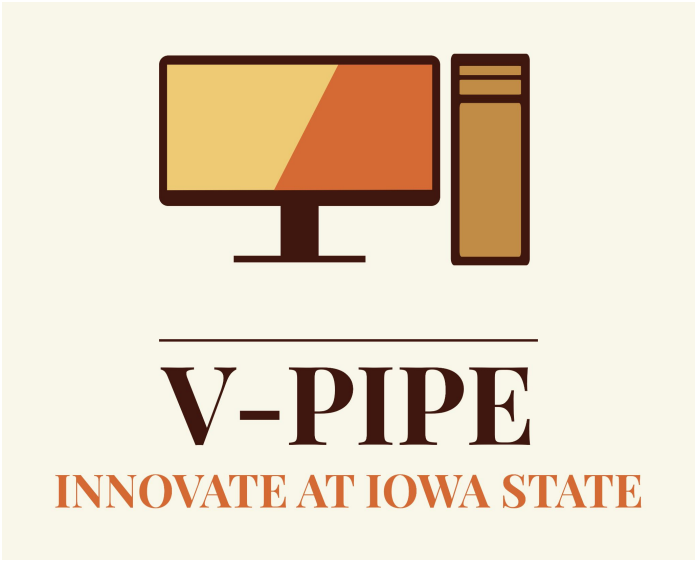
Video Pipeline for Computer Vision

sddec24-06 V-PIPE

Team Members: Liam Janda, Taylor Johnson, Ritwesh Kumar, Deniz Tazegul
Client: JR Spidell
Advisor: Dr. Phillip Jones

Problem Statement



- Video pipeline prototype for computer vision
- Compatible with ML algorithms
- Off the shelf components & open source software
- Multi-team, multi-university, multi-year project
- Long-term goal of improving the quality of life for wheelchair bound individuals with limited mobility through eye tracking algorithms
- Broad range of applications: medical imaging, self-driving cars, surveillance drones

Users & Stakeholders

Characteristics

- Disability
- Wheelchair bound
- Poor fine & gross motor skills
- Risk of seizures

Hears

- “There are certain tasks that you can’t do”

Says

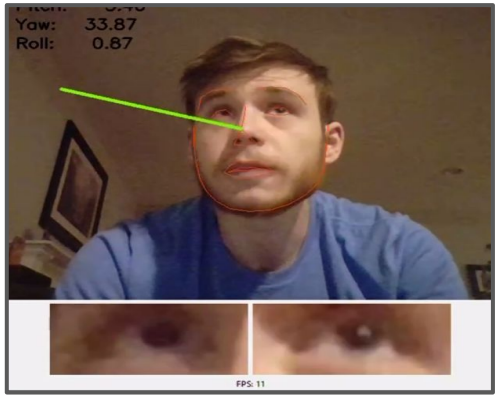
- “I want to be able to navigate on my own”

Thinks & Feels

- Desire for independence

Pain

- Medical problems limit the ability to perform daily tasks



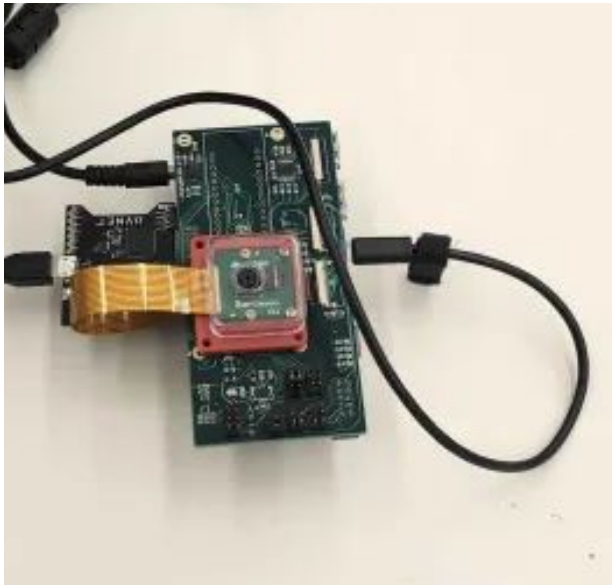
Eye-tracking algorithm by a previous SD team

User

Market Research

Producer	Pros	Cons
V-PIPE (Ours)	Off-the-shelf hardware	Proper configuration
Intel	Efficient	Specialized components with longer development time
LUCI	Attachable to existing wheelchairs	Niche market
EyesOnIt	Easily configurable software	Requires own hardware

Requirements & Assumptions



Functional

- Output live video to a DisplayPort **monitor**
- Route data using a **Linux image** with C and/or Python
- Execute using an **Ultra96-v2** FPGA board
- Use a **Sony IMX219QH5-C** image sensor

Non-Functional

- **640x480p** resolution at a minimum of **15 fps**

Assumption

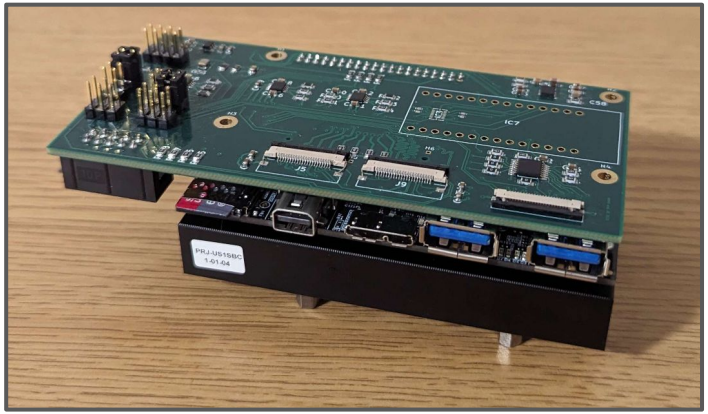
- User's face is well lit by a light source

Potential Risks and Mitigation

- File/OS corruption
  - Backup SD cards
  - Shutdown Ultra96-v2 using terminal
- Hardware damage
  - Proper handling and storing of hardware
- Loss of important code
  - GitHub for code version control

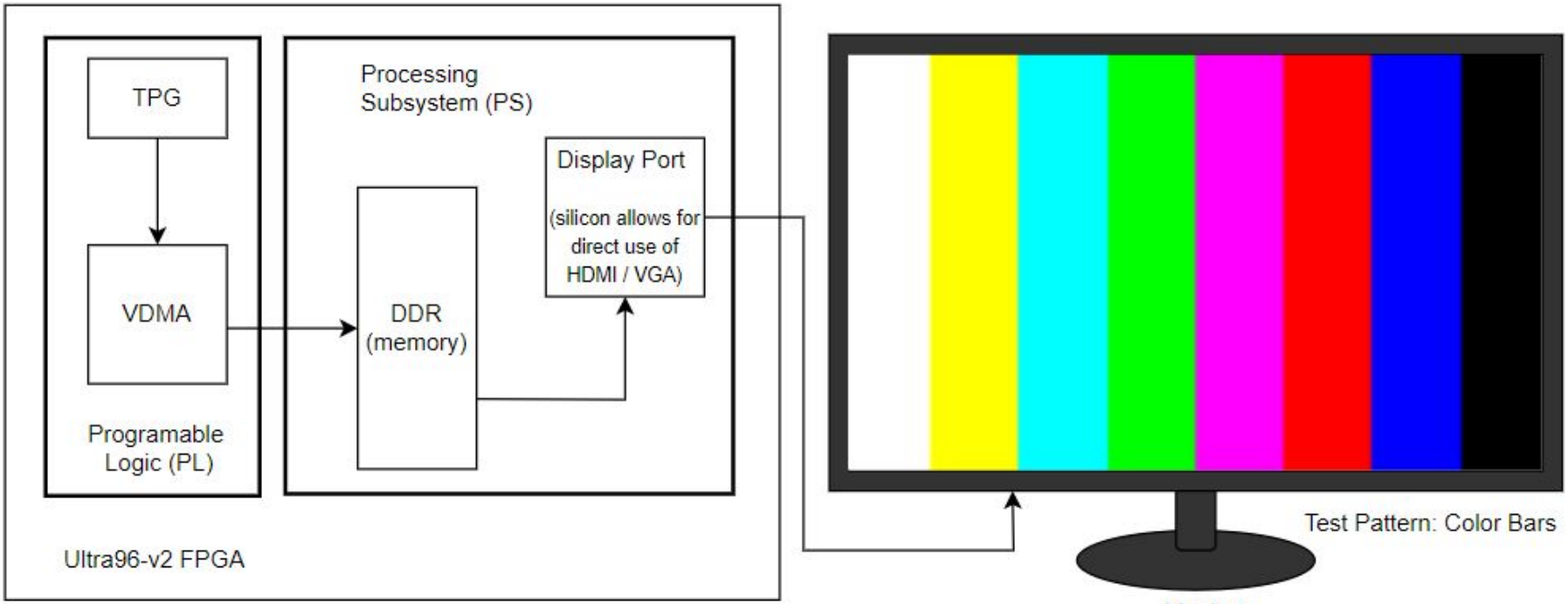
Project Vocabulary

- **TPG:** test pattern generator
- **MIPI:** mobile industry processor interface
- **VDMA:** video direct memory access
- **Overlay:** can be loaded onto the Ultra96-v2 to configure the hardware
- **PYNQ:** python productivity for Zynq software environment for embedded systems



Ultra96-v2 FPGA Board

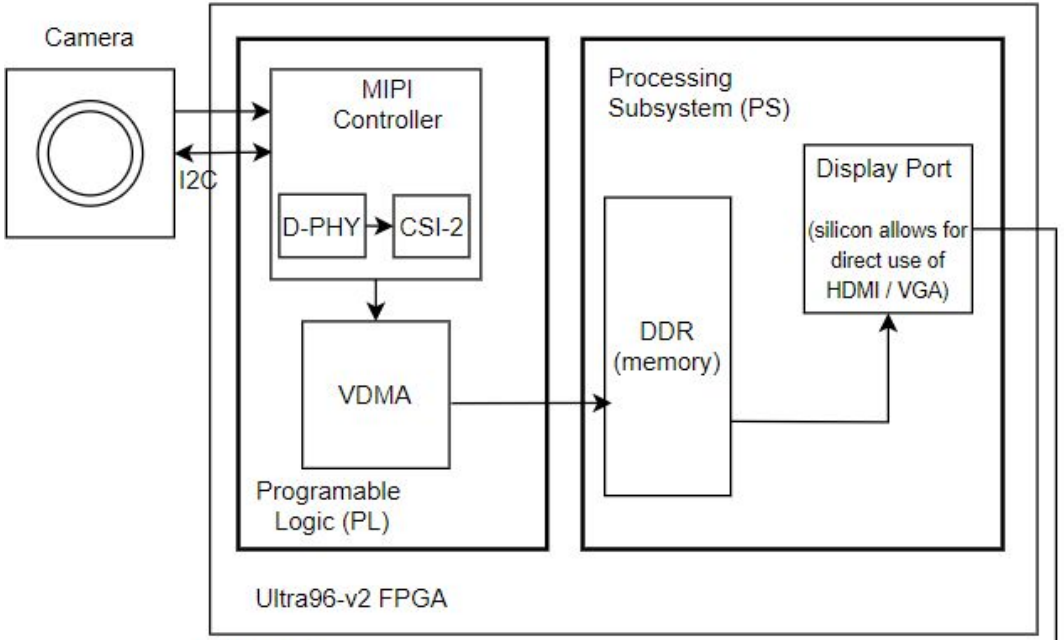
Block Diagram: TPG to VDMA



Protocols:
AXI-4 Lite controls register configurations
AXI-4 Stream carries the data between IP blocks

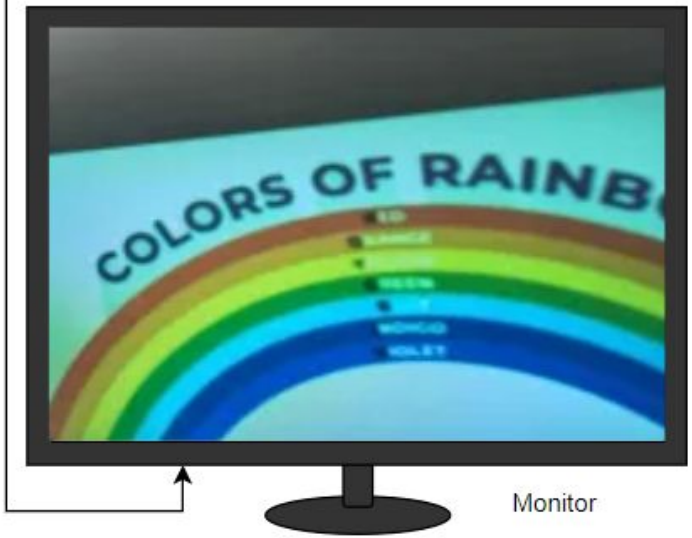
Resolution: 640x480

Block Diagram: Camera to Monitor



Protocols:
AXI-4 Lite controls register configurations
AXI-4 Stream carries the data between IP blocks
MIPI D-PHY physical communication layer
MIPI CSI-2 high speed camera serial interface
I2C configures the camera

Camera is pointed at the image below



Resolution: 640 x 480 at 20 fps

Evaluating Results Using Frame Rate

- Calculated by capturing a set number of new frames iteratively over a measured period of time
- Measured at the VDMA after software processing performing the color transformation
- Frame rate = Number of new frames received / time measured
- The measured frame rate is always lower than the configured frame rate based on the frame rate equation
  - Speed of Python code
  - Color transformation software processing

Evaluating Performance

- Resolution vs Frame Rate Tradeoff
- Max Theoretical Frame Rate formula
- Line length and Frame length determine frame rate
  - Line Length = 640 + 2808 = 3448
  - Frame Length = 480 + 2248 = 2728
- 640p x 480p results in 24 fps (calculated)
  - Base configuration for pipeline
  - Resolution and frame rate are configurable
- **Measured frame rate is 20 fps**

Frame Rate Calculation Equation

Frame rate in all-pixel scan mode is calculated by the followings.

$$\text{Frame\_Rate[frame/s]} = \frac{1}{\text{Time\_per\_Line[sec]} \times (\text{Frame\_Length})}$$

$$\text{Time\_Per\_Line\_[sec]} = \frac{\text{Line\_Length\_pck[pix]}}{2 \times \text{Pix\_Clock\_Freq[MHz]}}$$

Min Line Length = 3448

Min Frame Length = Vertical size + 32

Max Pix_Clock_Frequency = 114.5 MHz

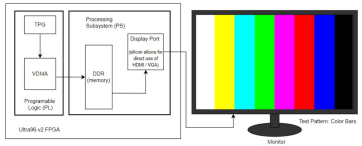
Development Approach

- Modular pipeline implementation
- Reading from read only registers
- Read and interpret status register values
  - MIPI Controller
  - VDMA
- Visual verification with known patterns/images
- Measure pipeline framerate
- PYNQ software environment

PYNQ

- Open-source Python libraries that abstract low-level code for the TPG, MIPI controller, VDMA, and DisplayPort

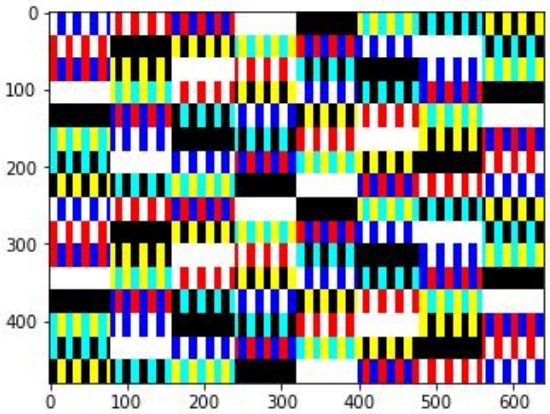
Block Diagram: TPG to VDMA



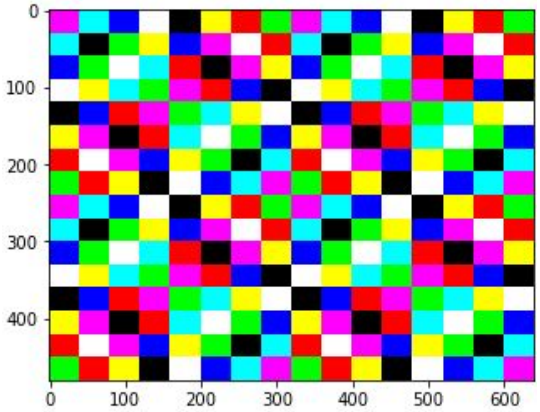
Testing TPG to VDMA Pipeline

Test Pattern - Tartan Bars

Incorrect pattern:

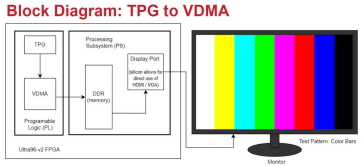


Expected pattern:

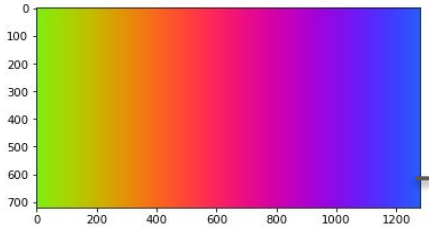


Data width and **frame synchronization** did not conform with AXI4 protocol

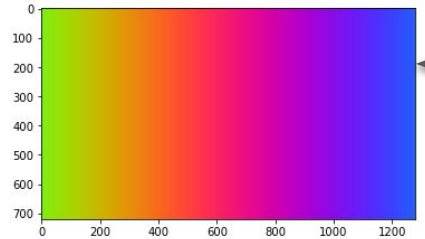
Testing TPG to VDMA Pipeline



Expected at 1280x720p:



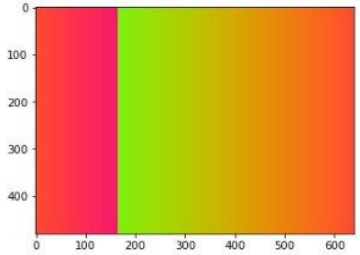
Output at 1280x720p:



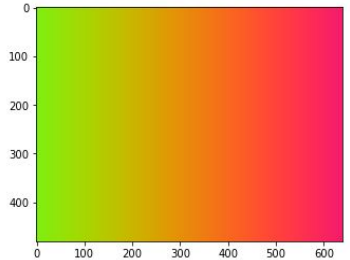
Frame in sync

Test Pattern - Color Sweep

Test Results at 640x480p:

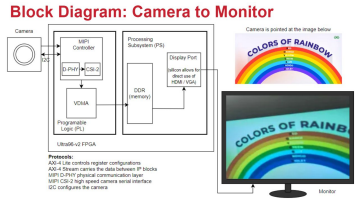


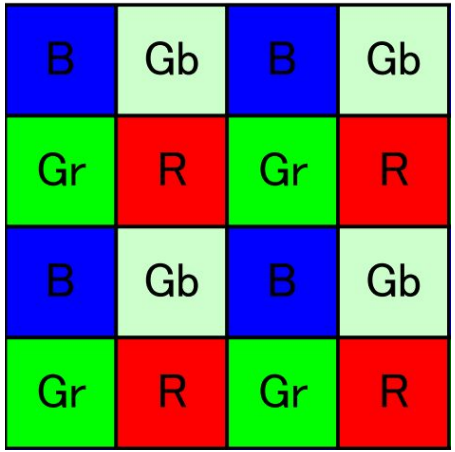
Frame not in sync



Frame in sync

Color Transformation

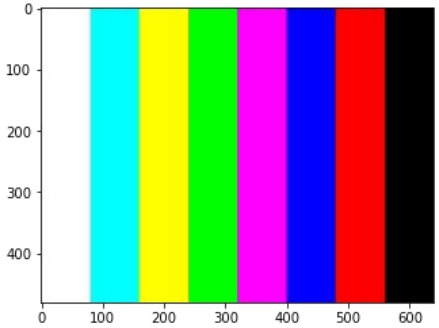




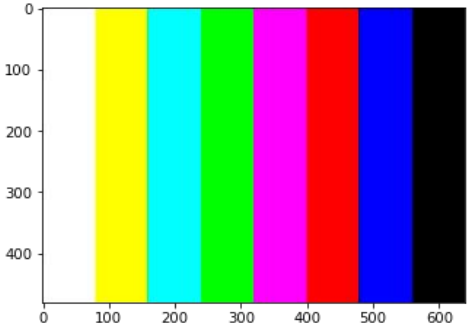
BayerBGGR Pattern

Transformation Used - Bayer BGGR

BG to RGB:

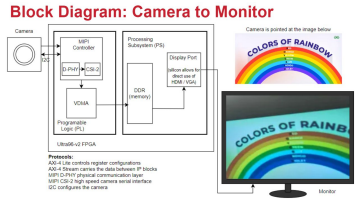


RG to RGB:



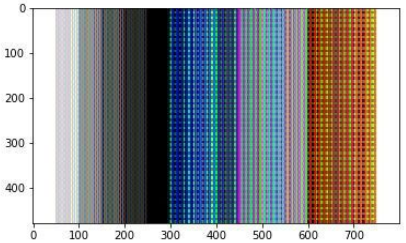
Data is converted from a Bayer pattern to RGB color

Testing Full Pipeline

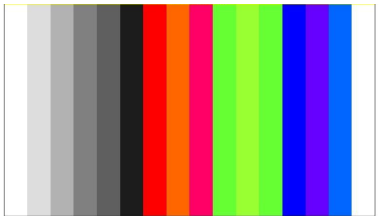


IMX219 Video Output

Incorrect Test Pattern:

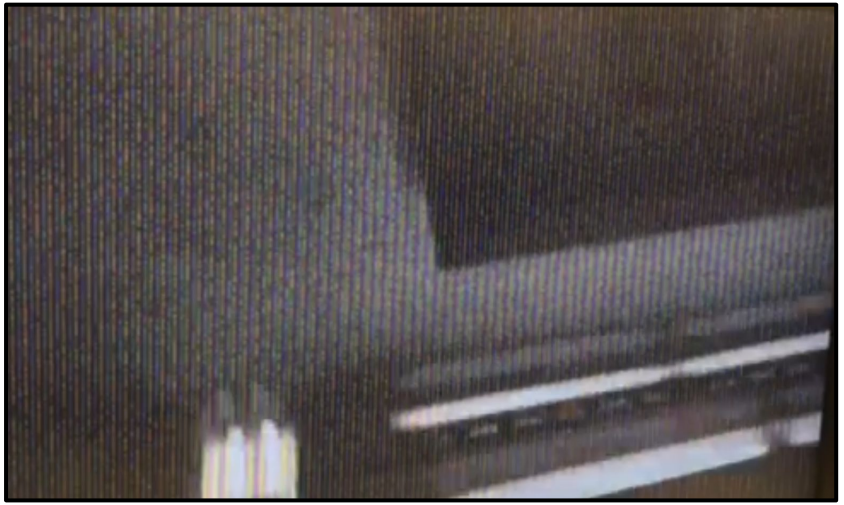


Expected Test Pattern:



Incorrect data format (RAW 10 to RAW 8)

Incorrect Monitor Output:



Current Video Output

Resolution: 200x75p at 200 fps

Resolution: 640x480p at 20 fps

Lessons Learned

- Read and understand the datasheets
- Debugging process
  - Making observations
  - Developing theories
  - Testing theories
- Correct hardware and software configurations necessary
- Data must comply with corresponding protocols between components
- Technical information on video transmission

Conclusion

Accomplishments
Sent video data from TPG to monitor
Sent video data from image sensor to monitor at 640x480p and 20 fps
Custom configuration code for IMX219 image sensor to set a desired resolution and frame rate based on the frame rate equation

Future Work
Convert to C Code ○ Faster configuration
Integrate into the larger project

Questions?

Accomplishments
Sent video data from TPG to monitor
Sent video data from image sensor to monitor at 640x480p and 20 fps
Custom configuration code for IMX219 image sensor to set a desired resolution and frame rate based on the frame rate equation

Future Work
Convert to C Code ○ Faster configuration
Integrate into the larger project

Project Resource/Cost Estimate

Resource / Component	Cost
Budget	$3,000
2 Ultra96-v2 FPGA Boards	2 * ($300 to $600) = $600 to $1200
IMX219/OV5647 Image Sensors	$10 + $20 = $30
2 SD cards: 1 PYNQ and 1 non-PYNQ	$15 + $20 = $35
(1 DisplayPort Cable + 2 USB Mini-to-USB A Cables + 1 Power Supply) * 2 sets	2 * ($15 + $10 * 2 + $25) = $120
(JTAG board + Ultra96-v2 Daughter Card) * 2 sets	2 * ($40 + $125) = $330
Total	**$1115 to $1715**