

2/2/2022

Camera Project Report

Introduction:

I'm a computer engineer who has experience designing embedded systems. Design experience ranges from VLSI, FPGA programming, device drivers, and writing software. I've graduated from University with a masters and am looking to develop an embedded system to solve a problem with AI. One of my goals is to gain experience using neural networks.

In 2021 I spent much of my time learning UPF and getting familiar with the VLSI physical design flow. During this time I decided to get back to where I started with my computer engineering career which was writing software for embedded systems. Before I decided what application I wanted to work on, I stumbled upon the NVIDIA Jetson embedded platform and decided this would be a great solution for what my goals are for learning. I needed a compute platform that could handle AI workloads in real time.

The application I decided to work on is a security camera that can track objects. One gripe about security cameras I currently have is the range of detection and I believe AI can have a better detection range than currently used thermal sensors.

When I was getting back into embedded systems I started programming peripherals like SPI and I2C to interface with the on board sensors of the Nexys A7. I quickly realized however that for a one person team the task would be daunting to set up the camera drivers and the application to run object detection. Even if I were to complete these tasks, the FPGA may be too slow due to the slower frequency and generally slower performance of FPGAs compared to hard core processors.

Project Block Diagram:

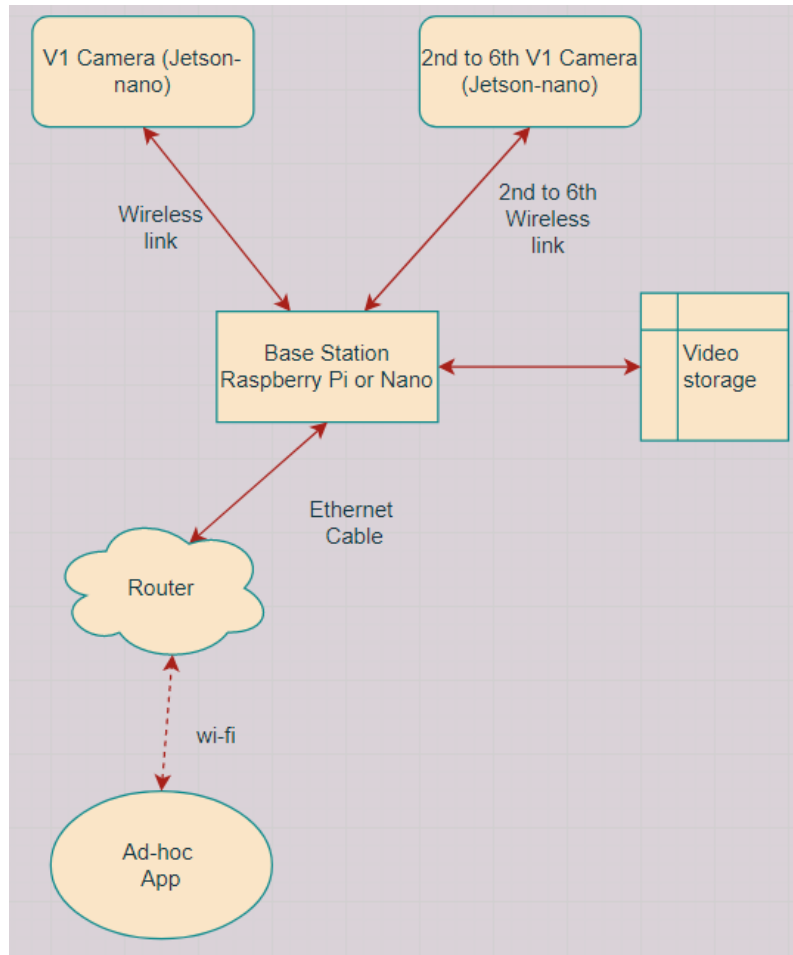


Figure 1: Diagram of initial prototype.

The block diagram for this project is shown in Figure 1. The initial prototype does not use the cloud. This will be essential as the project progresses. I believe the most secure solutions keep recorded data where it is inaccessible by intruders.

Each camera will be a Jetson Nano with a transceiver so there can be communication with the base station. The base station will save videos and be responsible for transmitting videos to the phone application. The base station implemented in my prototype is a raspberry pi 3b+.

Hardware Used:

Jetson Nano:

The Jetson Nano boasts a quad-core ARM Cortex A57 CPU and 128 CUDA core Maxwell GPU. The GPU provides 0.5 TFLOPS performance. The development kit created by NVIDIA uses an SD card for storage while the production version uses eMMC for storage. Furthermore, the developer kit provides a working carrier board for the System On Module which shows off the different peripherals that the Jetson supports. The Nano carrier board supports the peripherals MIPI CSI-2, HDMI, DisplayPort, USB, Ethernet, and a 40 pin GPIO header. All these peripherals can be added or removed to the production carrier board; it is up to the designer to implement the custom carrier board for the System On Module. NVIDIA provides the design guidelines and board files to get started. To find these resources, developers must sign up and access the Jetson Download Center.

Raspberry Pi 3b+:

The 3b+ is great for a media server. The GPU isn't as powerful as the nano so it will not be running any AI workload. The primary role of the 3b+ will be to receive and transmit videos to the mobile app. The 3b+ is similar to the nano where the peripherals can be customized and added with a carrier board. The production version also uses eMMC storage rather than an SD card. The 3b+ also boasts a quad core processor but we will not spend much time covering the 3b+ specs here.

NRF24L01+:

The NRF is a low power wireless transceiver designed for operation at 2.4GHz frequency. This transceiver handles the wireless communication and all is needed is a SPI master to command the transceiver. The frequency channel, output power, and air data rate are some of the many configurable options. For my prototype I am using the low power version with the PCB antenna. It is highly recommended to add a 10uF decoupling capacitor. The communication was faster once I added decoupling capacitance.

Arducam V2 pan-tilt:

This hardware module needs an MCU to give I2C commands to it. The printed circuit board will generate the necessary PWM signals to drive the servos. This is used with the camera only.

Camera Software

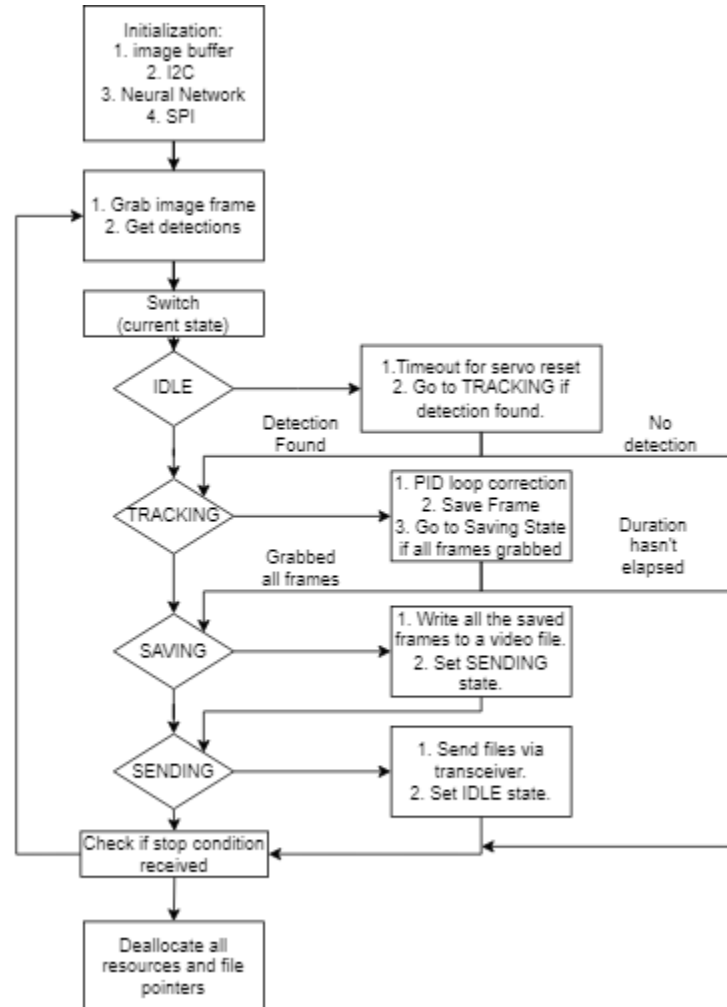


Figure 2: Block diagram of software for the camera.

The camera application is essentially a switch statement that is supposed to be an FSM. The IDLE state resets the servos to its default location if no objects are found. Once a detection is found the IDLE state transitions to the TRACKING state. The TRACKING state has a PID controller implemented to pan and tilt the camera wherever the object goes within the camera's view. If the duration hasn't elapsed the next frame is grabbed. Otherwise, the SAVING state will be set to save the video to a file. Next the SENDING state will send an image and video to the base station server via the transceiver.

Server Software

The server software is a threaded program that has two threads at the moment. The first thread is to handle communication between the app and server. Second thread handles the communication between the base station and cameras. The app uses TCP connections to communicate with the server. The cameras use the wireless transceiver to communicate with

the server. At the moment a basic handshake protocol is used to confirm requests on the TCP side. However due to the auto acknowledgement feature of the transceivers, handshake is unnecessary there.

The server uses gstreamer RTSP server for playing back video. The same method will be used to play live video as well. Callbacks needed to be added to the server to detect when a client disconnects and shutdown the server properly.

Mobile App Software

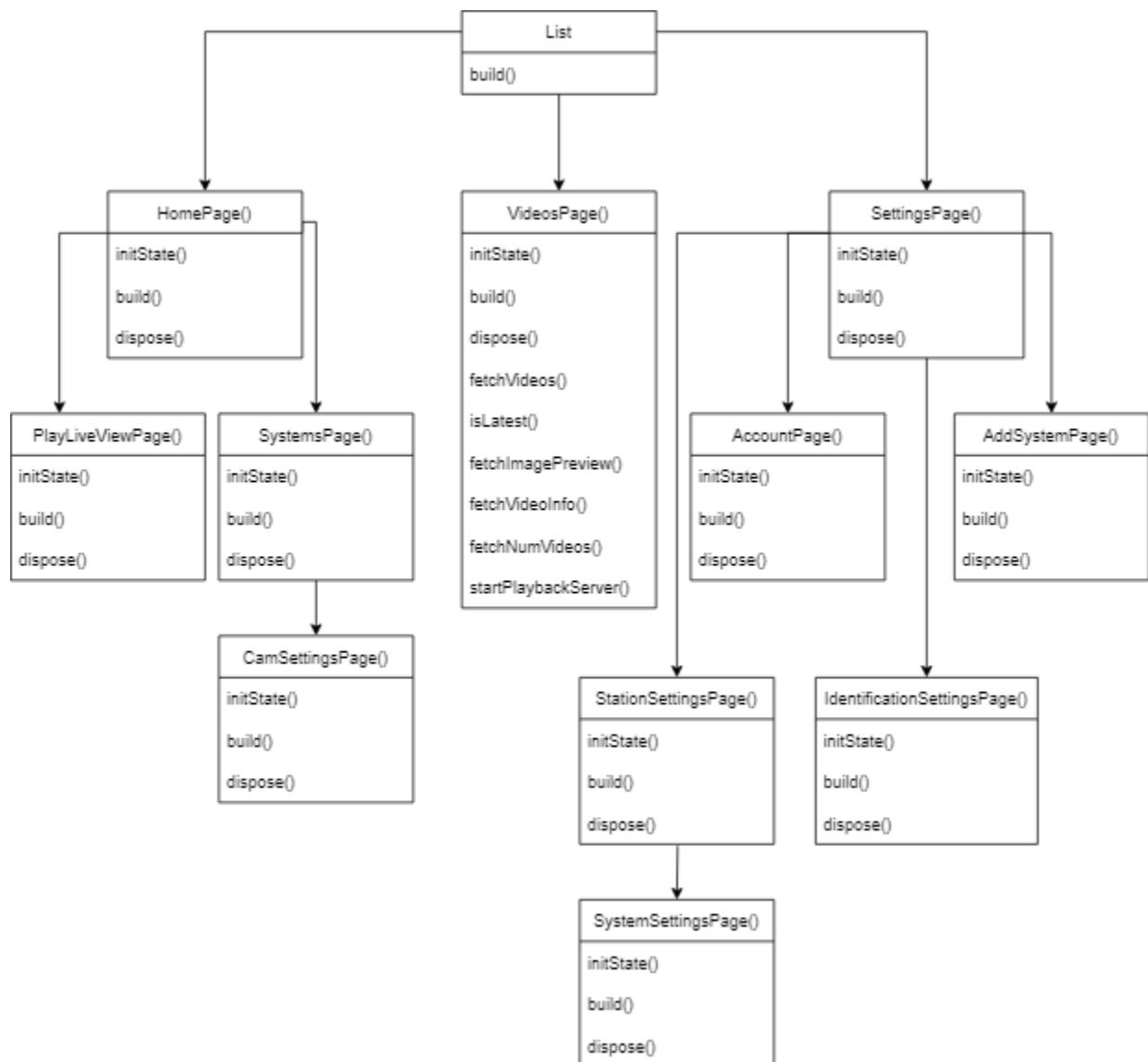


Figure 3: Activity hierarchy for mobile application.

The mobile application was written in Dart using the Flutter SDK. Flutter was chosen because apps for iOS and Android can be built with one codebase. That makes maintainability easier and shortens development time. Essentially the mobile app is split into three tabs where the first tab allows the user to watch live views, adjust camera settings, and monitor each of their cameras for each base station system on the network. The second tab allows users to load and playback their videos. The package Fijk Player is used to play RTSP streams. The third tab is for adjusting base station settings and user account information.

I2C, SPI, and GPIO:

Since the raspberry pi 3b+ and Nano are running linux, I have been able to reuse my GPIO code and SPI driver for each platform. I've been using sysfs to control the GPIO and spidev file to control the SPI driver. I encountered some issue where I was doing full duplex communication even though I didn't configure the driver to do that. That gave me some reading and writing issues I needed to solve.

The I2C driver is only used on the Nano for the servo driver. The I2C driver is controlled by the same means as the SPI driver by file I/O.

Current State of the Project:

The current state of the project is that I have a working prototype that can track objects, transmit video to a server, and playback the videos on a mobile app. This is as far as I'm willing to take the project for now. I have put in about 4-5 months of work and have written roughly 5000 lines of code. I have briefly started getting a KiCAD library together so that I can make a carrier board. However during this endeavor I learned that it is too much work and I would need a team to complete this task. It would be the first PCB I'd be designing so I have lots of learning to do. Therefore I am halting any more work on this project and will focus on finding a job in an embedded systems related job.

Extra work done which I haven't mentioned is retraining the neural network model on my laptop for better detection. I've trained the model for objects which would be encountered in a typical neighborhood setting. I've used the NVIDIA Jetson inference library to support this. NVIDIA provides lots of software to help with the AI tasks.

A task I would like to take on next is to create a neural network to remember the objects it sees frame by frame. My current implementation only identifies the objects for a single frame.

Current BOM:

Item	Quantity	Link
------	----------	------

Camera		
Jetson Nano 2GB	1	https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/education-projects/
Nano Power Supply	1	https://www.amazon.com/gp/product/B07TSDJSQH/ref=ppx_yo_dt_b_asin_title_o06_s00?ie=UTF8&psc=1
Arducam V2 pan-tilt	1	https://www.arducam.com/product/arducam-pan-tilt-platform-for-raspberry-pi-camera-2-dof-bracket-kit-with-digital-servos-and-ptz-control-broad/
Raspberry Pi V2 cam	1	https://www.raspberrypi.org/products/camera-module-v2/
Noctua 5V fan	1	https://www.amazon.com/gp/product/B071FNHVXN/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1
Jetson nano case	1	https://www.amazon.com/gp/product/B07SXJHQD1/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1
Server		
Raspberry pi 3b+	1	https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/
Transceiver		
nrf24l01+	1	https://www.amazon.com/Makerfire-Arduino-NRF24L01-Wireless-Transceiver/dp/B00O9O868G/ref=sr_1_3?crid=3SD5H155BF6IZ&keywords=nrf24l01%2B&qid=1643819703&srefix=nrf24l01%2B%2Caps%2C156&sr=8-3
Misc		
Jumper wires	1	https://www.amazon.com/gp/product/B07RN4WN2N/ref=ppx_yo_dt_b_asin_title_o03_s00?ie=UTF8&psc=1
Cat 6 Ethernet cables	2	https://www.amazon.com/Monoprice-Flexboot-Ethernet-Patch-Cable/dp/B00AJHC7BY/ref=sr_1_3?keywords=cat+6+ether+cable&qid=1643820064&srefix=cat+6+eth%2Caps%2C146&sr=8-3

10uF electrolytic capacitor	10	https://www.digikey.com/en/products/detail/nichicon/UVY2A100MDD1TD/4328564
Soldering kit	1	https://www.amazon.com/Soldering-Iron-Kit-Temperature-Desoldering/dp/B07Q2B4ZY9/ref=sr_1_4?crd=24BQQ6AWX62EX&keywords=soldering+kit&qid=1643820347&srefix=soldering+ki%2Caps%2C148&sr=8-4
Ethernet switch	1	https://www.amazon.com/NETGEAR-8-Port-Gigabit-Ethernet-Unmanaged/dp/B07PFYM5MZ/ref=sxsts_rp_s1_0?crd=16HUHUXZCLWYE&cv_ct_cx=ethernet+switch&keywords=ethernet+switch&pd_rd_i=B07PFYM5MZ&pd_rd_r=29bb8d6b-8e15-43d9-8f67-6438e5c13638&pd_rd_w=V7or5&pd_rd_wg=XzqcK&pf_rd_p=dc8286ba-5f1e-4679-adde-8b7fe66c128e&pf_rd_r=X0YE2H8RFWHZPJR4R17A&psc=1&qid=1643820486&srefix=ethernet+switch%2Caps%2C172&sr=1-1-5e1b2986-06e6-4004-a85e-73bfa3ee44fe

