PhotonVision

PhotonVision

# Overview

# Terminology

| **Term** | **Definition** |
| --- | --- |
| Telemetry | The process of remotely measuring, collecting, and transmitting data from a system to a receiving point for monitoring, analysis, or control. |
| Mutable | A property that can be changed after it has been created |
|  |  |

# WPILib for Java

This document describes PhotonVision using VSCode. The VSCode pre-requisites and FRC environment can be installed from here:

<https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-2/wpilib-setup.html>

* Open the ‘FRC VSCode’ editor from the desktop icon
* Open a command terminal and obtain the development code
  + eg… git clone https://github.com/Smittysports/ReefscapeJavaTest.git
* Open the folder where the code from the previous step was saved inside the VSCode editor

# Components

| **Name** | **Description** |
| --- | --- |
| Orange Pi 5 Pro | Installed Ubuntu 24.04 via PhotonVision |
| SanDisk MicroSD | 16 GB Industrial version, flashed with RaspberryPi imager |
| Arducam Camera | QV9281 |
|  |  |

### **Orange Pi 5 Pro Product Parameters**

### 

| **Main Control Chip** | Rockchip RK3588S (8nm LP process) |
| --- | --- |
| **CPU** | • 8-core 64-bit processor •big.LITTLE Architecture: 4-core Cortex-A76 and 4-core Cortex-A55, big core cluster is 2.4GHz, and little core cluster is 1.8GHz frequency. |
| **GPU** | • Arm Mali-G610 • Built-in 3D GPU • Compatible with OpenGL ES1.1/2.0/3.2, OpenCL 2.2 and Vulkan 1.2 |
| **NPU** | Embedded NPU supports INT4/INT8/INT16 mixed operation, with up to 6TOPS computing power |
| **RAM** | LPDDR5 496PIN：4GB/8GB/16GB (optional) |
| **Memory** | • eMMC socket: supports eMMC module (optional) • SPI Flash (default empty) Note:either eMMC socket or on-board SPI Flash is preferred. • MicroSD (TF) slot • M.2 M-KEY slot: NVMe SSD or SATA SSD |
| **USB** | USB3.1 Gen1 \* 1;USB2.0 \*1； USB HUB:USB2.0 \*2（TYPE A）+UAB2.0\*2 row of pins |
| **Video Output** | • HDMI2.1 up to 8K @60Hz • HDMI2.0, up to 4K @60Hz • MIPI DSI 4 Lane, up to 4K @60Hz |
| **Camera** | 2\*MIPI 4 Lane |
| **Audio** | CODEC: ES8388 • 3.5mm headphone jack audio input/output • Input: onboard MIC • HDMI 2.1 eARC |
| **Ethernet&Wi-Fi+BT Module** | 10/100/1000Mbps Ethernet with PoE+ support (PoE+ HAT required); Onboard Wi-Fi5+BT 5.0/BLE module: AP6256 |
| **Expansion port** | • Dual-row pin: 2.54mm 40Pin • Supports DC 5V and 3.3V power outputs • Configurable UART, PWM, I2C, SPI, CAN, GPIO and other functional interfaces |
| **RTC** | 3V 2PIN 1.25mm socket |
| **FAN** | 5V 2PIN 1.25mm socket |
| **LED** | RGB LED tri-color indicator |
| **Supported Operating Systems** | Orangepi OS（Droid）、Orangepi OS（Arch）、Ubuntu、Debian、Android12 |

# Getting Started

OrangePi 5 Pro

1) Install RaspberryPi imager from here:

https://www.raspberrypi.com/software/

2) Obtain the image from PhotonVision here

https://docs.photonvision.org/en/latest/docs/quick-start/quick-install.html#install-the-latest-image-of-photonvision-for-your-coprocessor

3) Image the MicroSD card

4) Insert it in the OrangePi and boot

5) Un: pi

pw: raspberry

6) OS is Ubuntu 24.04

7) Connect the ethernet cable to an accessible network (10.0.0.x for me)

- Bring the network down and then up

- sudo iplink set enP4p65so down

- sudo iplink set enP4p65so up

- I obtained 10.0.0.183

- ping the orangepi from another PC on the same network

8) Connect the ArduCam to a USB port on the OrangePi

* ArduCam OV9281 USB Camera

9) <http://photonvision.local:5800>

or

<http://10.0.0.183:5800>

10) Go to the Camera Matching tab and activate the camera

11) Settings:

* Team Number: 10.0.0.163 (only use 1721 if working with a RoboRIO)
* IP Assignment: DHCP (Change this to static in the lab or on a Robot)
* Hostname: photonvision-1721-lab

12) The PhotonLib webpage, shown in step 9, contains documentation on how to use it

* Install the photonlib library from the VSCode dependencies tab

13) Start the Simulator, in Java, and the PhotonVision should connect to the NetworkTables

# mDNS & PhotonVision Dashboard

mDNS stands for Multicast Domain Name System. It's a protocol used to resolve hostnames to IP addresses within small networks that do not have a local DNS server, such as home or office LANs.

**Here's how mDNS works:**

* Instead of querying a centralized DNS server, mDNS uses multicast to send DNS-like queries to all devices on the local network.
* Devices respond with their own IP address if the hostname matches.
* It typically operates on UDP port 5353
* Hostnames in mDNS usually end with .local (e.g., printer.local).

PhotonVision uses mDNS (Multicast DNS) to simplify network access to its dashboard and services, especially during initial setup. Here's how it works and what you should be aware of:

**How PhotonVision Uses mDNS**

* Hostname Access: Devices running PhotonVision (like a Raspberry Pi) advertise themselves on the local network using mDNS with a hostname like photonvision.local. This allows users to access the dashboard via a browser without needing to know the device's IP address.
* Zero-Config Networking: mDNS enables "plug-and-play" style networking, which is ideal for teams who may not have advanced networking setups or DHCP servers.

**Common Use Cases:**

* Zero-configuration networking (zeroconf): mDNS is a key part of technologies like Apple's Bonjour and Avahi on Linux.
* IoT devices: Many smart home devices use mDNS to advertise their services.
* Service discovery: Often paired with DNS-SD (DNS Service Discovery) to find services like printers or file shares on the network.

**Notes:**

* PhotonVision uses mDNS to allow easy access via photonvision.local without needing to know the IP address.
* It relies on multicast DNS traffic sent to 224.0.0.251:5353, which is standard for mDNS.
* If mDNS fails, users often resort to setting a static IP or using fallback mechanisms for easier access
  + You can configure a static IP by editing the /boot/photonvision.txt file like this:

network:

ip: 10.0.0.2

netmask: 255.255.255.0

gateway: 10.0.0.1

Team: 1721

* mDNS can be flaky in certain network setups — for example, when the Pi is directly connected to a computer or when there's no router managing the network.

## Daemon

PhotonVision runs on an OrangePi running Ubuntu 24.04. The mDNS is managed using the avahi-daemon service. This can be verified by typing the following from a command line:

systemctl status avahi-daemon

This will show that the daemon is running and the hostname used by PhotonVision.

Install additional tools with:

sudo apt install avahi-utils

I then typed:

avahi-resolve -n photonvision-1721-lab.local

and obtained:

photonvision-1721-lab.local 10.0.0.183

## Copilot AI Overview

**🔍 Step-by-Step: Browser Access via mDNS**

1. Device Advertises Itself:
   * The PhotonVision device (e.g., Raspberry Pi) runs an mDNS responder.
   * It announces its hostname (typically photonvision.local) to the local network using multicast packets on UDP port 5353.
2. Browser Resolves Hostname:
   * When you type http://photonvision.local into your browser, the browser asks the operating system to resolve the hostname.
   * The OS uses mDNS to send a multicast query to the network: “Who has photonvision.local?”
3. PhotonVision Responds:
   * The PhotonVision device replies with its IP address.
   * The OS receives the response and passes the IP to the browser.
4. Browser Connects:
   * The browser uses the IP address to initiate an HTTP connection to the PhotonVision dashboard.
   * You see the web interface where you can configure cameras, pipelines, and NetworkTables settings.

**✅ Requirements for This to Work**

* mDNS Support on Your OS:
  + Windows: Requires Bonjour (installed with iTunes or separately).
  + macOS and Linux: Native support.
* Multicast Enabled on Network:
  + Routers/switches must allow multicast traffic.
* Same Subnet:
  + Your computer and the PhotonVision device must be on the same local network.

**🧪 Troubleshooting Tips**

* If photonvision.local doesn’t resolve:
  + Try accessing the device via its IP address.
  + Use a network scanner (like Advanced IP Scanner or nmap) to find the device.
  + Ensure Bonjour or Avahi is running on your system.

## How mDNS and TCP Work Together

**mDNS (Port 5353) — Name Resolution**

* + mDNS runs on UDP port 5353 and is used for resolving hostnames to IP addresses on a local network.
  + When you type http://photonvision.local:5800, your computer sends a query over port 5353 asking:  
    “Who is photonvision.local?”
  + The PhotonVision device responds with its IP address.

**PhotonVision Dashboard (Port 5800) — Web Interface**

* Once your computer knows the IP address (thanks to mDNS), it opens a TCP connection to port 5800 on that device.
* This port hosts the web dashboard, where you can configure settings, view camera feeds, etc.

**Summary of the Flow:**

1. mDNS (UDP 5353) → Resolves photonvision.local to an IP.
2. Browser (TCP 5800) → Connects to that IP to load the dashboard.

# NetworkTables

Network Tables is a publish-subscribe messaging system. It is very simple and uses a key/value store system. Values are published to named “topics” either on the robot, driver station, or potentially an attached coprocessor, and the values are automatically distributed to all subscribers to the topic.

It was written specifically for FRC by WPIlib developers. It has tight integration with FRC libraries, vision pipelines, and dashboards like Shuffleboard or SmartDashboard.

Examples of other publish-subscribe systems are:

* DDS (Data Distribution Service): I used this at StillRiver Systems for patient monitoring of a Photon Radiation Therapy machine
  + Advanced robotics, using ROS2, uses DDS under the hood
* OPC-UA
  + The IoT (Internet of Things) and large-scale telemetry utilize publish-subscribe

Why use it over existing technologies?

* NetworkTables is easy to use, with no external brokers, servers, or configuration.
* Perfect for quick development cycles by high school students.
* More complex systems (like DDS, MQTT, or ROS) would add complexity without benefit in most FRC contexts.
* FRC robots operate on a closed network, not the internet.
* No need for scalable, secure, or brokered communication like you'd need in production/industry.

**Concepts:**

* Topic: A named data channel. Topics have a fixed data type (for the lifetime of the topic) and [mutable](https://docs.wpilib.org/en/stable/docs/software/frc-glossary.html#term-mutable) properties (since if it is name modifiable, there is no need to publish anything about it since it never changes).
* Publisher: defines the topic and creates and sends timestamped data values.
* Subscriber: receives timestamped data value updates to one or more topics.
* Entry: a combined publisher and subscriber. The subscriber is always active, but the publisher is not created until a publish operation is performed (e.g. a value is “set”, aka published, on the entry). This may be more convenient than maintaining a separate publisher and subscriber.
* Property: named information (metadata) about a topic stored and updated separately from the topic’s data. A topic may have any number of properties. A property’s value can be any data type that can be represented in JSON.

**Usage**

* PhotonVision tries to connect to a NetworkTables server. Normally, the RoboRIO is the server, but in your case, your VM must act as the server
* Start the NetworkTables in code:

In your Java robot code (running in simulation), you need to start the NetworkTables server manually:

* This allows PhotonVision to connect to your VM as if it were the RoboRIO.

import edu.wpi.first.networktables.NetworkTableInstance;

public class Main {

public static void main(String[] args) {

NetworkTableInstance.getDefault().startServer(); // Starts the NT server

RobotBase.startRobot(MyRobot::new);

}

}

* The ‘Settings’ page of the PhotonVision web page must use the IP address of the VM (Virtual Machine) that is running the Simulator.
  + This is not the case if using the RoboRIO, then it must use 1721

I added the following to verify that I can read from the PhotonVision camera:

import org.photonvision.PhotonCamera;

import org.photonvision.targeting.PhotonTrackedTarget;

public class Robot extends TimedRobot {

private static final String kDefaultAuto = "Default";

private static final String kCustomAuto = "My Auto";

private String m\_autoSelected;

private final SendableChooser<String> m\_chooser = new SendableChooser<>();

private final Field2d m\_field = new Field2d();

private final PhotonCamera camera = new PhotonCamera("Arducam\_OV9281\_USB\_Camera");

/\*\* This function is called once when teleop is enabled. \*/

@Override

public void teleopInit()

{

// Query the latest result from PhotonVision

var result = camera.getLatestResult();

boolean hasTargets = result.hasTargets();

System.out.println("hasTargets = " + hasTargets);

if (hasTargets)

{

// Get a list of currently tracked targets.

//List<PhotonTrackedTarget> targets = result.getTargets();

// Get the current best target.

PhotonTrackedTarget target = result.getBestTarget();

// Get information from target.

double yaw = target.getYaw();

double pitch = target.getPitch();

double area = target.getArea();

double skew = target.getSkew();

System.out.println("yaw = " + yaw);

System.out.println("pitch = " + pitch);

System.out.println("area = " + area);

System.out.println("skew = " + skew);

}

}

## PhotonCamera NetworkTables Descriptions

| **Class** | **Description** |
| --- | --- |
| PhotonCamera | Contains all of the subscriber information about the PhotonVision camera |
| PhotonPipelineResult | Returns the most recent information from Network Tables. |
| PhotonTrackedTarget | This has the yaw, pitch, area, skew, fiducial ID, etc… |

Subscriber: Arducam\_OV9281\_USB\_Camera

Topics :/Arducam\_OV9281\_USB\_Camera/pipelineIndexRequest

Property: TBD

# Lab Setup

Settings Page:

Networking:

Team Number/NetworkTables Server Address = 10.0.0.163

* Set to this for my VM at home
* Change this to a static IP address available on the 10.17.21.x subnet

Hostname = photonvision-1721-lab

* Use this when connecting to the PhotonVision dashboard

<http://photonvision-1721-lab.local:5800>

* This uses mDNS to connect to the photonvision-1721-lab host
* It then connects to the custom TCP port of 5800 for the dashboard

# Useful commands

**Shutdown the OrangePi**

sudo shutdown -h now

**Remote login**

ssh pi@10.0.0.183

Pw: raspberry