Using **JeVois** in an **FRC** environment

By FIRST Team 2073

JeVois is a self contained “Smart Camera”. It has an image sensor, Linux based OS, USB and TTL Serial connections. This single unit would be used to replace a USB, or other, Web Camera and an off board image processor like a Raspberry Pi. It can also be used to replace Vision tracking code from being run on the RoboRio, or Driver’s Station.
Contained inside the JeVois’ tiny footprint of just 1.5" x 1.25" x 0.75", is a Quad Core CPU and a Dual Core GPU. While this document will not be covering in detail how to take advantage of all the capabilities of the JeVois, it will provide you with enough information to start using it for your next FRC Robot.

In addition to the JeVois, you will want to have a couple additional items. Links will be provided at the end of this document on where to find find these items.

1) 8GB micro SD card.
2) Arduino Pro Micro
3) Windows computer (Preferably not Win 10), Mac PC, or Linux PC.
4) DC to DC USB out 5v regulator.

In order to keep this document as specific as possible to FRC, it will refer the reader to many of the JeVois resources that already exist, and they are EXTENSIVE!

Keep in mind, this is a “Smart Device” and not just a replacement for your off board processor and camera(s), though it is that, and more. All communications to and from the JeVois pass through the JeVois engine. You will need to learn a few tricks to get this little device to perform in a way you are familiar with and this document will endeavour to help point out those tricks.

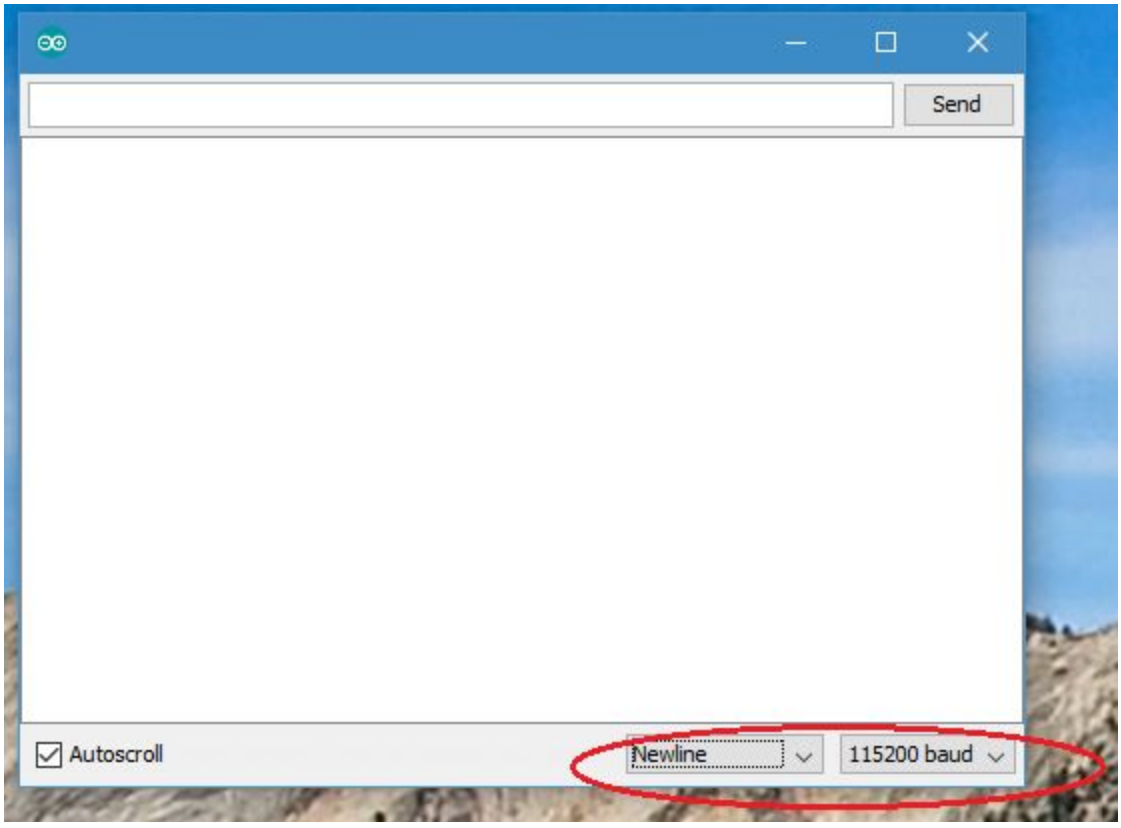
For reference, in the development of these processes, a Windows 8 laptop was used. ~~The experience with Windows 10 was a less than favorable~~. As of 12-8-17, the latest JeVois image corrects the errors with Windows 10. Additionally, this document is written for using **Python** based **OpenCV** scripts to perform vision tracking.

Notepad++ was chosen as the script editor for OpenCV scripts as well as other text files used for JeVois.

The first step, once you acquire your camera, is to make sure you can communicate with it. For this process, you will need to flash the JeVois image to a micro SD card. The easiest tool to do this is Etcher. Use the latest image for the JeVois. (**PLEASE NOTE**: on 12/8/2017, a new image was released that addresses Windows 10 compatibility. If the “Latest Image” link does not get you v. 1.6, or higher, please use this link for image version v1.6)

Once you have the card flashed, follow the directions for your PC’s OS at http://jevois.org/doc/User.html. Start with the Quick Start Guide. It is recommended that you perform all the steps through the “Command-line interface users guide”. The rest of the steps are nice to know, but can be followed up on at a later time.

After following through these steps, you may want to set up an Arduino Pro Micro as a “Serial Passthrough”. This is one of the “Examples” provided with the Arduino Integrated Development Environment. It can be found under /Examples/Communication/SerialPassthrough”. Just make sure to set both Serial0 (USB) and Serial1(hardware TTL) to 115200. The Serial Port User’s Guide shows how to wire it to an Arduino UNO, but the same connections work for the Pro Micro. One caveat, wire the red wire to “Vcc” on the Pro Micro. Once you have the Arduino connected, you can pass all command line commands through either the Arduino Serial Monitor (make sure to set “Newline”).



The other option, which is also recommended is connect to the JeVois over USB with the Terminal program such as Putty. With serial access over both ports available, you have multiple

options for getting data in and out of JeVois as well as images. One example is to have streaming video to pass through the RoboRio to the driver's station as well as sending targeting data over TTL serial to the RoboRio.

Common commands you will be using on a command line:
setpar serlog USB This tells the JeVois engine to route error messages out the USB port. This is VERY helpful when your script throws an error. Options for this command are: [None|All|Hard|USB]

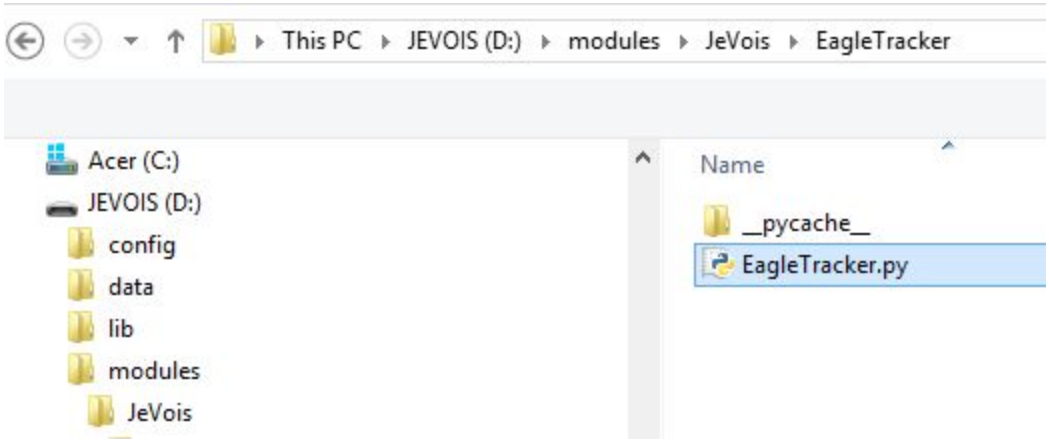
setpar serout Hard This tells the JeVois engine to route send.Serial statements out the hardware serial port. This is useful in routing targeting data out to the RoboRio. Options for this command are: [None|All|Hard|USB]

usbsd This instructs JeVois to share it's SD card just as a thumb drive would. This is how you edit your scripts on the JeVois.

streamon / streamoff These two commands are used to manually start the streaming of images and/or data without the need of video capture software.

When creating your own OpenCV script to run on the JeVois, there are a few tips that will make success easier to achieve.

1) Create a folder under the JEVOIS (D:)/ modules/Jevois folder on the SD drive. Name it the name of the script you want to run. Ie. “EagleTracker”
2) In your folder. Create the Python script with the same name as the folder. Ie. “EagleTracker.py”



3) In your script, create a class with the same name as the folder and the script.
4) Lastly, the class will contain at least one “def” named “process”. **This is where your script will reside**.

```
27  class EagleTracker:
28      # ######################################
29      ## Constructor
30      def __init__(self):
31          # Instantiate a JeVois Timer to measu
32          self.timer = jevois.Timer("Catbox", 1
33
34      # ######################################
35      ## Process function with USB output
36      def process(self, inframe, outframe):
37          # Get the next camera image (may bloc
```

Note in the image above “inframe” and “outframe”.
“inframe” is the name of the raw image grabbed by the JeVois engine for you. “outframe “ is the name of the image sent out of the script to the JeVois engine be sent out the USB connection.

5) The last major thing to note is, you MUST NOT create a While loop to repeatedly execute your script. The Jevois engine will handle the repeated calls to “process” for you.

We found that it was easiest to copy one of the Demo Python Scripts found in the JeVois image and use it as a basis for our first script. Rename it, edit it, and use its structure to get you started. The best success we had was when we started with the “PythonTest.py” script. It has

examples for sending and receiving serial data (more on this below) as well as sending a processed image out.

```
        # We are done with the output, ready to send it to host over USB:
        outframe.send()

        # Send a string over serial (e.g., to an Arduino). Remember to tell the JeVois Engine to display those messages,
        # as they are turned off by default. For example: 'setpar serout All' in the JeVois console:
        jevois.sendSerial("DONE frame {}".format(self.frame));
        self.frame += 1

    # ###################################################################################################
    ## Parse a serial command forwarded to us by the JeVois Engine, return a string
    def parseSerial(self, str):
        jevois.LINFO("parseserial received command [{}]".format(str))
        if str == "hello":
            return self.hello()
        return "ERR: Unsupported command"
```

Additionally, found in this Demo script is another interesting and helpful method. If you type “help” at the command line, Jevois will return a long set of available commands and configuration parameters, their options, and their current state. In addition, there is a section labeled “Module Specific Help”. You can populate this section with helpful information about your “module” by filling in the “supportedCommands” method with your text.

```
    # ###################################################################################################
    ## Return a string that describes the custom commands we support, for the JeVois help message
    def supportedCommands(self):
        # use \n seperator if your module supports several commands
        return "hello - print hello using python"
```

You now have the basic tools you will need to actually start tracking targets.
Most of you, if you have read this far, will be locating targets by using color thresholding and shape detection. This can be done by manually entering upper and lower Hue, Saturation and Values. But you will tire of this trial and error method very rapidly! So, how can you address this? Our approach was to create a second script specifically for gathering the calibration/tuning values. These values are stored in a file on the JeVois. When the tracking scripts launches, it grabs these calibration values from the file. This “calibration” script allows you to send commands in through the serial link(s) to adjust the tuning values in real time. It is very similar to the tracking script in it’s operation, but it adds the ability to receive commands and modify the values. This method, while being considerably faster, still gets tedious very quickly. So, how can we make tuning your tracking values easier and faster? Our approach was inspired by the GUI tool that can be found in the JeVois User Tutorials, “Tuning the color-based object tracker using a python graphical interface”. We modified this script to send commands that match what our tuning script expects. This script is written in Python and we can run it on a Windows PC. It most certainly should be able to be run under MAC or Linux as well with little to no modifications. Our goal is to release our code for this project near the end of Dec., 2017.

If you have worked through some of the tutorials, you will be familiar with how you get JeVois to run your scripts. You will need to add it to the “videomappings.cfg” file.

```
407  MJPG 320 240 15.0 YUYV 320 240 60.0 JeVois EagleTracker *
408  MJPG 320 240 30.0 YUYV 320 240 60.0 JeVois EagleTrackCal
```

We create two different scripts, one for tuning/calibrating the tracking code and the tracking code itself. You can see these two separate video mappings above.
The syntax for entries in the “videomappings.cfg” file are as follows:
MJPG 320 240 30.0 is the streaming video over USB format. YUYV 320 240 60.0 is how the image format from the camera will be acquired. MJPG/YUYV is the pixel formatting, 320 240 is the resolution, 30.0/60.0 is the FPS. JeVois is the “Vendor”, or sub-folder in reality, where the folder with your code resides. And finally EagleTrackerCal is the actual script name without extension.
Read through the top half of the videomappings.cfg file to get a more detailed description of what options are available for camera acquisition and streaming output.

Initialization configuration options

One thing you will want to take advantage of is the ability to pre-configure the JeVois at startup.
Here are two simple ways to do this. One way will be common for all scripts run on JeVois, the other sets configuration options for the particular module being run.
To set configuration for all scripts, edit the JEVOIS/config/initscript.cfg file.
You can enter commands in here as you would the command line. For example, one setting you might want is setting the serial out to go out the hardware TTL port. ie. *setpar serout Hard* . You can do many other things here as well, like locking exposure and white balance settings. In addition, you can launch your code without having to open a display window and select the resolution, frame rate and format, more on this later.
The second option will be script specific. You do this by placing a text file named *script.cfg* in the same folder as your script. By setting parameters with this file, it will only set those parameters for the script in that folder. You may want to use this option if you intend to change which script is running, on the fly.

Serial data out, USB and/or TTL

It is not the intent of this guide to tell you what data or how you should send or use the data you generate. The intent is to show you some options and let you decide what your approach will be. This section will cover what, why, then how we on Team 2073 do with our targeting data. Hopefully it will get you where you want to be quickly.

First let's start with "what":

We start with a binary value that represents whether we are tracking or not. Then we send the "x" and "y" coordinates of the center pixel of an identified target. Lastly, we can add the time the exposure was acquired. This last value is used in advanced target location prediction.

Now for the "why"

We choose these data points because they are the most useful.
The "x" value it the simplest piece of data for the RoboRio to use as it can tell the robot, or one of its subsystems, which way to turn. We find that many times, it is even better to calculate how many degrees off center the camera is from the target. At times, we may also substitute a distance calculated to the target in place of the "y", or sometimes we just add the distance to the data sent in addition to the "y".

And now for the "how"

We use JSON packets to carry the data. JSON is an industry standard and most programming language already have libraries to support it. The first step in this is to build the data set, then encode it into a JSON. It looks like this in our code.

```
86          #Build "pixels" array to contain info desired to be sent to RoboRio
87          pixels = {"Trk" : 1, "XCntr" : x, "YCntr" : y}
88          json_pixels = json.dumps(pixels)
```

To send the JSON to the RoboRio is a simple as

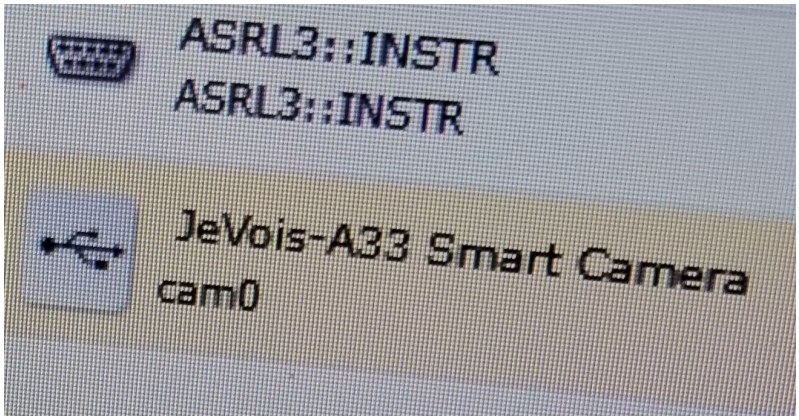
```
110         jevois.sendSerial(json_pixels)
```

If you had entered at the command line, or via one of the pre configuration options, *"setpar serout Hard",* the data would be sent out the hardware TTL port. You can do the same with *"setpar serout USB"* if you prefer. The one caveat is that if you are using the USB out to stream video, sending data out the same port may not have the results you desire. One consideration you will want to think about, the TTL will, by default, operate at 115.2Kbs while the USB port can operate at 480Mbs.

On the RoboRio side, accessing this data is as simple as using the WPILib SerialPort API. This is for the RS232 port. (Thanks Ron R. FRC Team 2607) Similar information is available for the USB ports here.

A common request we have received is "**How do you receive the targeting data on the roboRIO?**" So, in an attempt to help address this request, here is a quick look at some Java

code we put together to at least point you in the right direction. With this you should have a good starting point.
The first piece of information you will want to gather is where the roboRIO see's that your camera has attached. We did this by looking at the Web Dashboard on the roboRIO once we connected the camera's USB cable to it. Near the bottom of the screen you should see something that looks like this.



The important part to identify is the "0" after "cam". This indicated the first USB device. This means that in your code, you will need to use "kUSB". If the number had been 1 or 2, you would use "kUSB1" or "kUSB2" respectively.

Here is a snippet of code we put together that will allow you to verify you are receiving your data stream. All it does is print put the string, but at least you should see it come in.

import edu.wpi.first.wpilibj.SerialPort;

```
public class Robot extends SampleRobot {
//config a serialport and specify baudrate and port on the Rio
	SerialPort cam = new SerialPort(921600, SerialPort.Port.kUSB);

	@Override
	public void operatorControl() {
		while(true){
			try{
				System.out.println(cam.readstring());
			}catch(Exception e){
				System.out.println("Error");
			}
			Timer.delay(.005);
		}
	}
```

Starting your script on power up

It is more than likely you will want your JeVois to start running your script and sending data upon powering up. This is a fairly easy process to achieve.
The JEVOIS/config/initscript.cfg file on the SD card is read during the boot process. Not only can it be used to pre configure many parameters of the JeVois system, it can also be used to start running your script. You will want to enter a few commands for this to work.

```
10  setmapping2 YUYV 320 240 30.0 JeVois EagleTrkNoStream
11  setpar serout USB
12  streamon
```

The first line configures the JeVois camera for acquisition and launches your tracking code. The second line is used to tell JeVois which port to send the data out. The third line starts the actual stream coming from the camera sensor into JeVois and any data you are sending out. An additional command you may want to add would be ***setpar serlog None***. This would prevent log and error messages from being sent to the RoboRio.
One caveat, when you launch your code on power up, you will need a way to stop the process if you need to access the code for any reason. To do this, you will need to issue a “streamoff” command at the command line. If you are sending serial data out the USB port, it will be impossible to issue the command through USB. You will need to issue the command through the TTL serial port. This is where the Arduino Pro Micro, or similar board, comes in really handy!

Powering JeVois

The JeVois draws approximately 3.5 watts. That means at 5v, it draws about 700mA. The RoboRio USB port can provide up to 900mA. Although this theoretically is enough, it is really asking the RoboRio to approach a power limit for the USB ports. Using a “Y” USB cable will work, but then all the ports are taken. You may want consider the following option. Optimal target tracking usually is achieved by reducing the exposure to the point that the only thing visible in the image is the target.That will render the image basically useless for anything else. So, in that case, don’t attach the USB cable from the JeVois to the RoboRio, just attach it to a USB power source like the ones mentioned in the end of this document and send the targeting data to the RoboRio via the TTL serial link.

Supporting devices and where to get them

8GB micro SD card.
This can be along with the JeVois. This same package is also available from Amazon. The card is also available from a plethora of other options. Amazon is easy.

Arduino Pro Micro is available from Jevois, Amazon and SparkFun, as well as other places.

JeVois camera is available from Jevois and Amazon. Keep in mind. There are three packages available from each. Basic camera, camera with cables and SD card, developers kit. Just choose the version you prefer. It is highly recommended to get the “Beginner kit as it comes with the SD card, Serial cable and Y-USB cable.

An optional piece of hardware to consider is +5V USB DC-DC supply that can be powered off the PDB. This will provide all the power needed to run the JeVois and not put the load on the RoboRio USB port(s). Something like these from Amazon will suffice.