Setup

1. Download/copy project folder to local storage

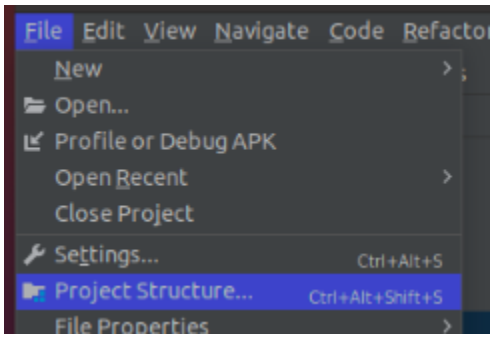
2. Download Gstreamer Android binaries from
https://gstreamer.freedesktop.org/download/

Android Universal 1.20.3 tarball

Edit /gradle.properties in order to set gstAndroidRoot to point to the unpacked GStreamer Android binaries.

```
gstAndroidRoot=/home/weis/AndroidStudioProjects/gstreamer_binaries
```

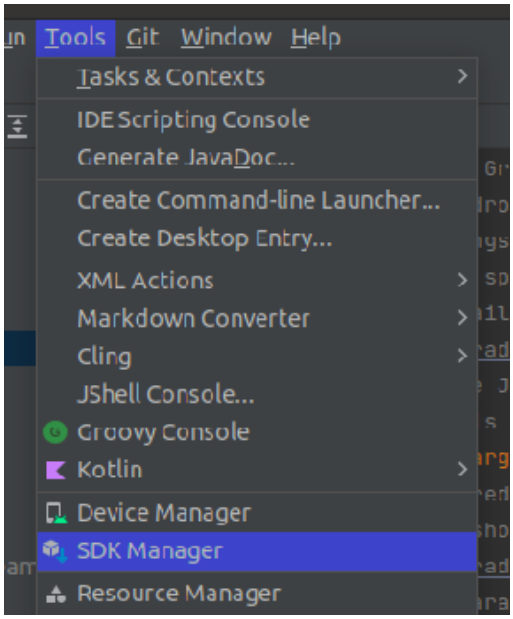
3. Launch Android-studio, opening the folder as a project, then under files, project structure.

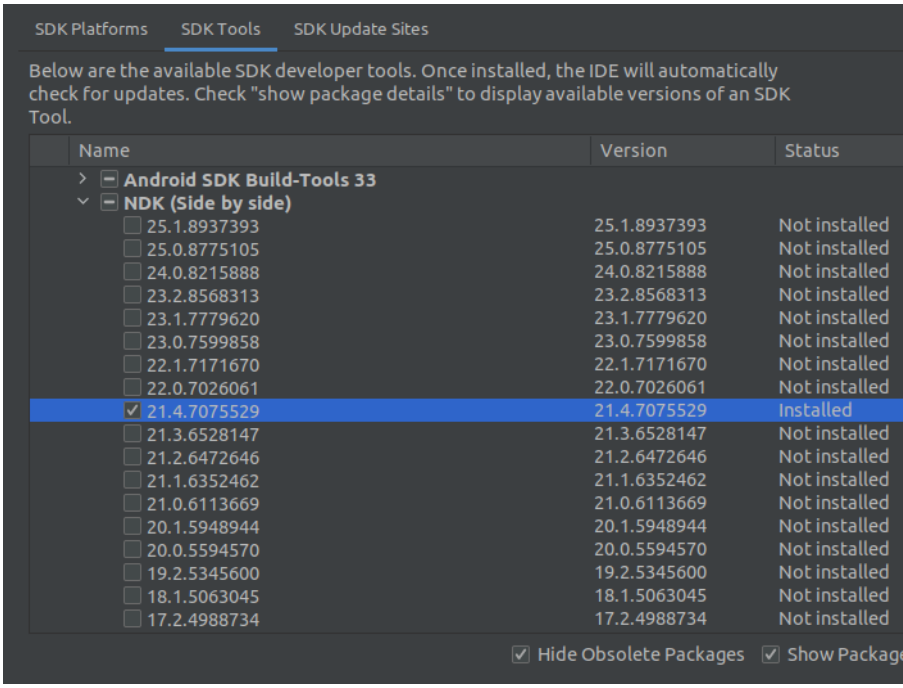


Alternatively open project structure with “Ctrl+Alt+Shift+S”

Make sure the gradle plugin version and gradle version are 7.1.3 and 7.3.3 respectively.

4. Under tools, SDK manager



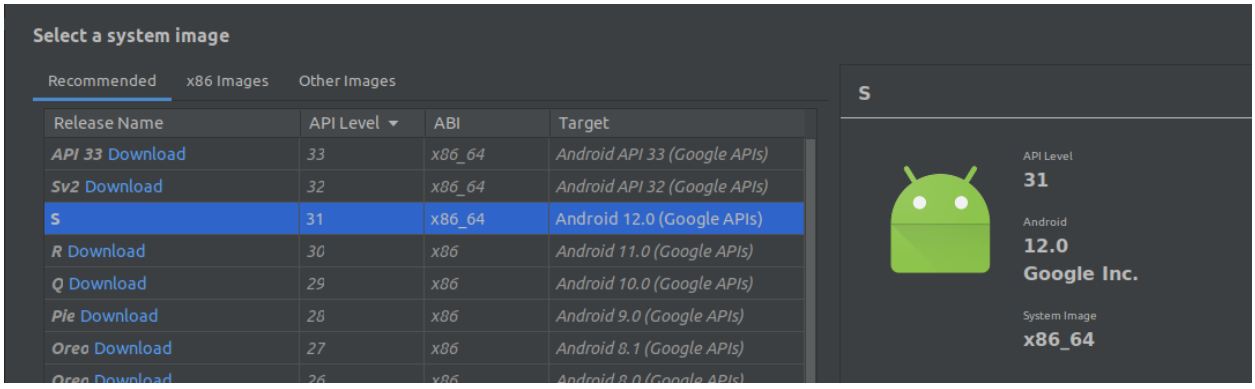


tick “Show Package Details” to select specific versions to install.

Install NDK 21.4.7075529

Application is now ready to run.

Application is tested on virtual device Pixel 3 XL running on Android 12.0



Documentation

Gstreamer adapted from
https://gstreamer.freedesktop.org/documentation/tutorials/android/video.html

Gstreamer pipeline used during testing

Nanopi video source:

gst-launch-1.0 -v v4l2src device=/dev/video0 ! 'video/x-raw, width=(int)640, height=(int)480' ! jpegenc ! rtpjpegpay ! udpsink host=$clientip port=5200

Video Player client (android app):

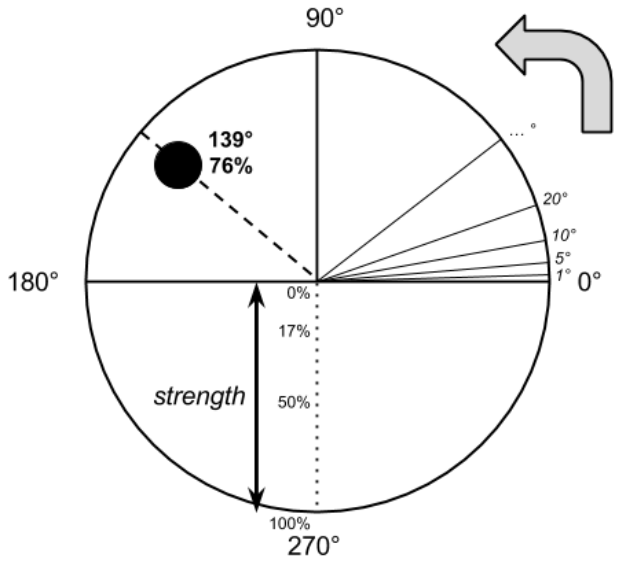
udpsrc port=5200 ! application/x-rtp, encoding-name=JPEG,payload=26 ! rtpjpegdepay ! jpegdec ! autovideosink

Gstreamer video is displayed in a textureview where I can get a snapshot of the video by using getBitmap() . This is used as a workaround to grab a frame as I was unable to figure out how to grab a frame using the native gstreamer library. The captured frame will later be used to feed into the object detection model.

One downside of this method is the mismatch of pixels in the video and the captured image. Video from the root might be in 480p but the captured image is in 1080p, this may result in accuracies when fed into the object detection model.

Screen recording is currently not working. Turning on the recording will only take a single snapshot of the video, while it is possible to take a snapshot on every frame and stitch them together into a video, I have not found a way to do it yet.

Joystick module used https://github.com/controlwear/virtual-joystick-android



The angle follows the rules of a simple counter-clock protractor. The strength is percentage of how far the button is from the center to the border. A direct forward input outputs angle 90, strength 100.

Using math we convert angle+strength into absolute X Y cartesian values to have an easier time calculating the signal to send to the robot.

Currently the robot moves using a separate signal to each motor. Values range from 2 to 12. 2 being full power clockwise and 12 being full power anticlockwise. 7 will stop the motor.

Forward movement will require the Left motor = 12 , right motor = 2

Right turn , Left motor = 12 , right motor = 12

We can get each motor signals by using

Left motor = X + Y Right motor = Y - X

And normalizing their range to 2-12 and 12-2 respectively.

Sent strings can be adjusted according to the robots input signals.

Tensorflow implementation adapted from https://github.com/tensorflow/examples/tree/master/lite/examples/object_detection/android_play_services

Both ObjectDetectorHelper.kt and OverlayView.kt are classes provided by tensorflow lite. OverlayView provides a surface to draw the bounding boxes from the results of the detection model. ObjectDetectorHelper takes in config options and the model and sets it up for use. Model selection and options can be hardcoded in or coded to be selectable during runtime. (eg. different object detection models for specific objects) Custom models just need to be placed in the assets folder.