2017 FLYSET FTC Workshop

Hosted by

technicbots

Software Topics Session

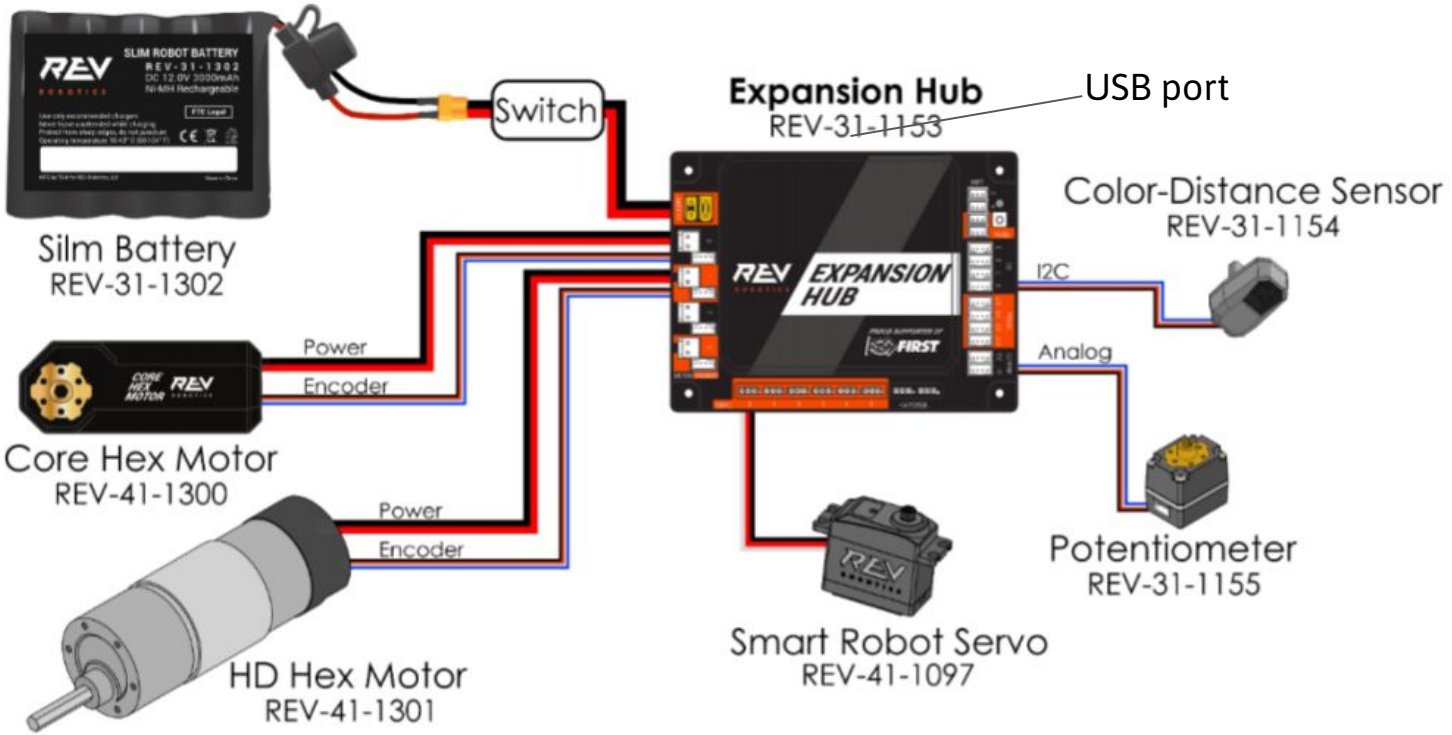
Brandon Wang

technicbots

Agenda

- Rev Expansion Hub programming
- Vision: When and Why?
- Vuforia: Setup, finding the targets, and navigation
- Further Steps
- Questions

REV Expansion Hub



Advantages of Expansion Hub

- Single module instead of 4-5
  - Much cheaper compared to buying multiple Modern Robotics modules
- Stronger connectors = less disconnects
- Built in IMU sensor
- Can connect two together to get double the motor/servo/sensor connections
- Integrated PID controller

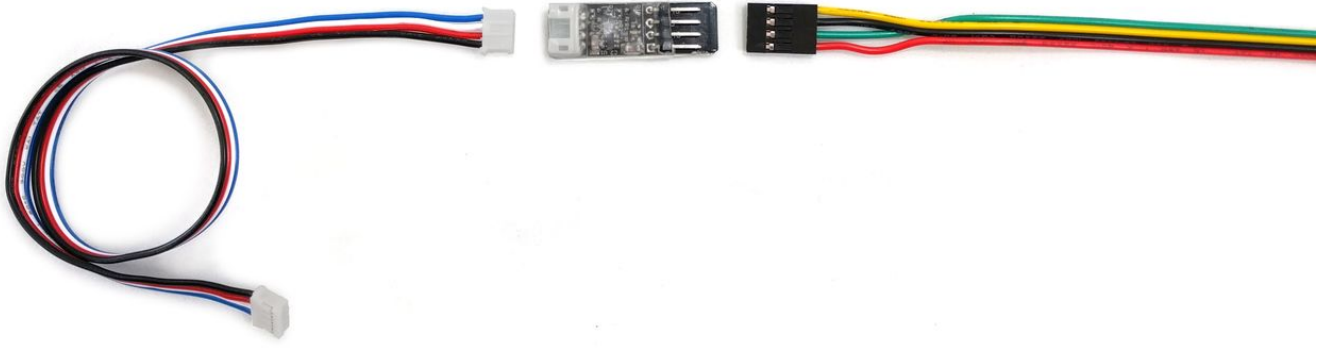
REV Expansion Hub Sensors

Color Sensor

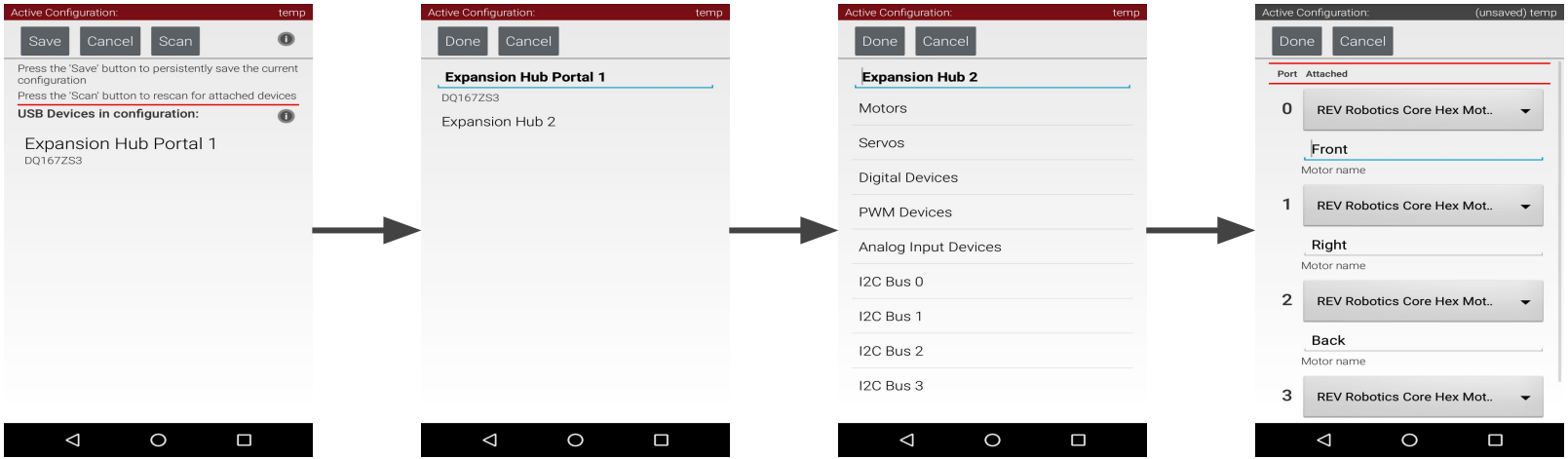
Potentiometer

Touch Sensor

REV Expansion Hub Converter



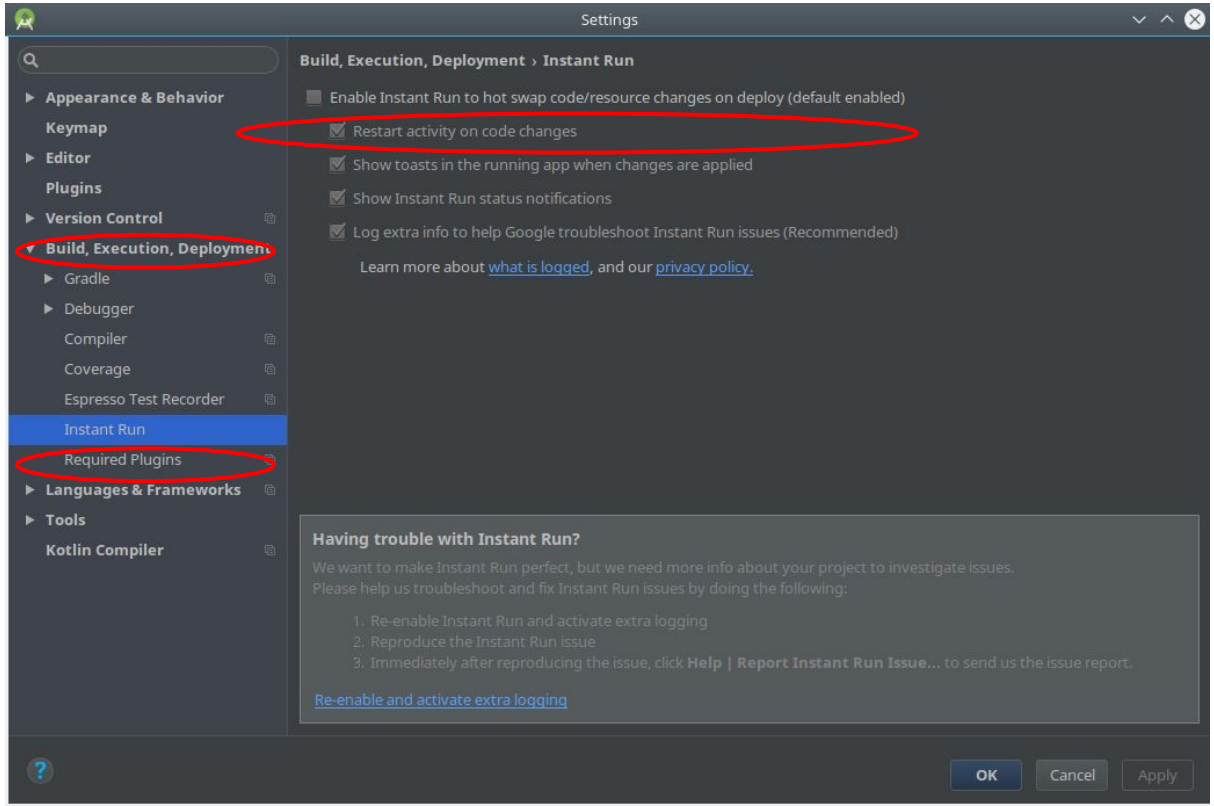
Programming with the REV Expansion Hub



- The rest is the same as with the Modern Robotics controllers

```
leftMotor = hardwareMap.dcMotor.get("leftMotor");

rightMotor = hardwareMap.dcMotor.get("rightMotor");
```

Disabling Instant Run

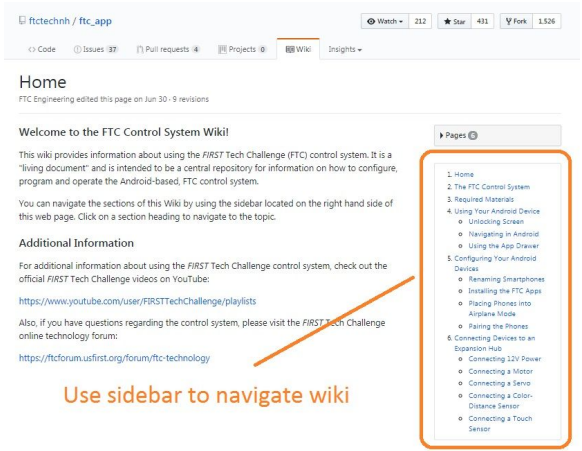


Causes errors if enabled (can't configure the robot, code not updating properly)

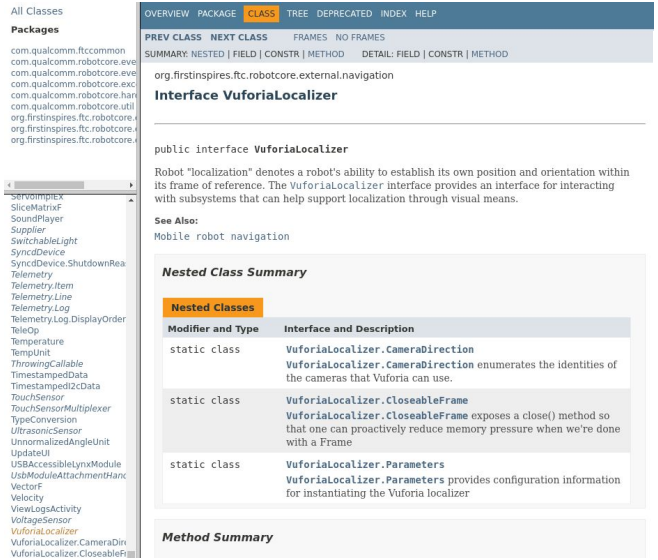
Need to do once per computer

Programming Resources

Github Wiki



Javadoc



Blocks Tutorial

FTC Blocks Programming Training Manual

REV Robotics Expansion Hub Edition

Document Version 2.1

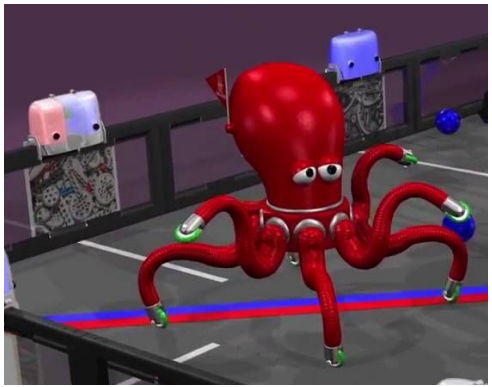
FIRST Tech Challenge

FIRST TECH CHALLENGE

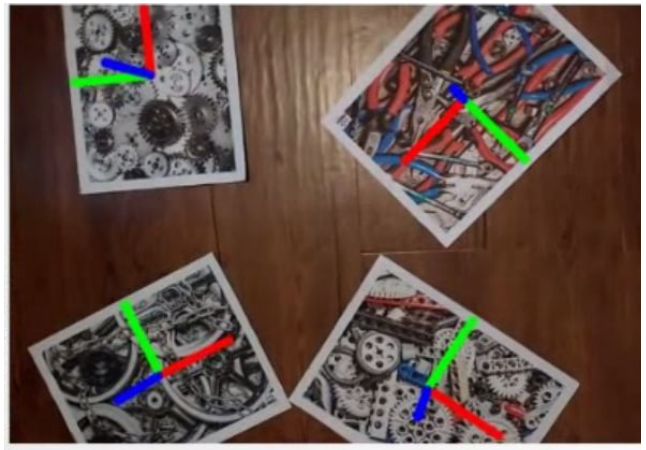
Vision

Why Vision?

- Almost certainly in next year's game
  → Usually helps to find a scoring goal
- More difficult challenges
  → Faster Positioning
  → More accurate aiming
  → Starting to become effective with more processing power
- Win Awards
- Real world applications

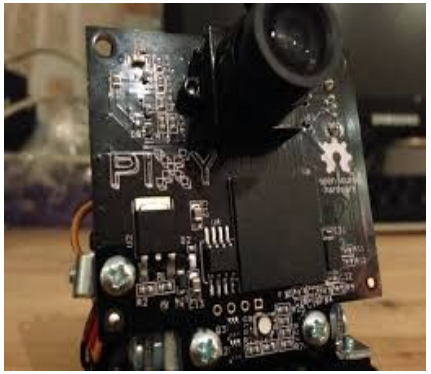


When to use Vision?



- Stable goals
  - → Vuforia Targets
  - → Things that don’t move
- When using simpler sensors/manual navigation is inefficient
- When alignment speed is important

Vision Options



CMU Pixy Camera

+ Does processing for you
- Limited detecting quality
- Little SDK support

ZTE Speed

- Too slow for good performance

Motorola Moto G4 Play
(or Google Nexus 5 / Samsung Galaxy S5)

+ All effective and fast options
+ Can use either front or back camera

vuforia™

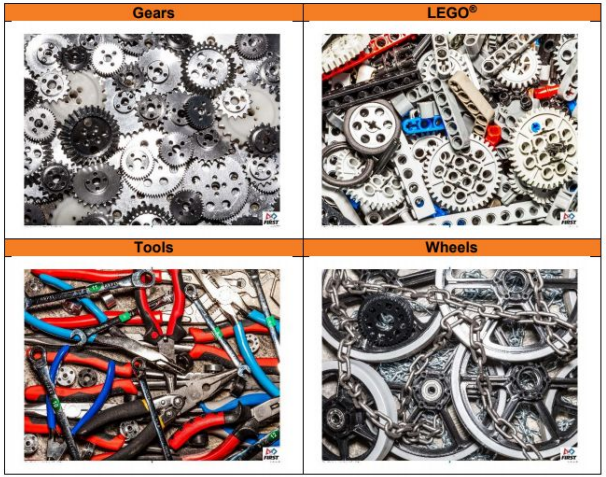
+ Comes with SDK
+ Only way to track premade targets
+ Easy to use

Other software options exist (OpenCV) that give more flexibility, but they are harder to use.

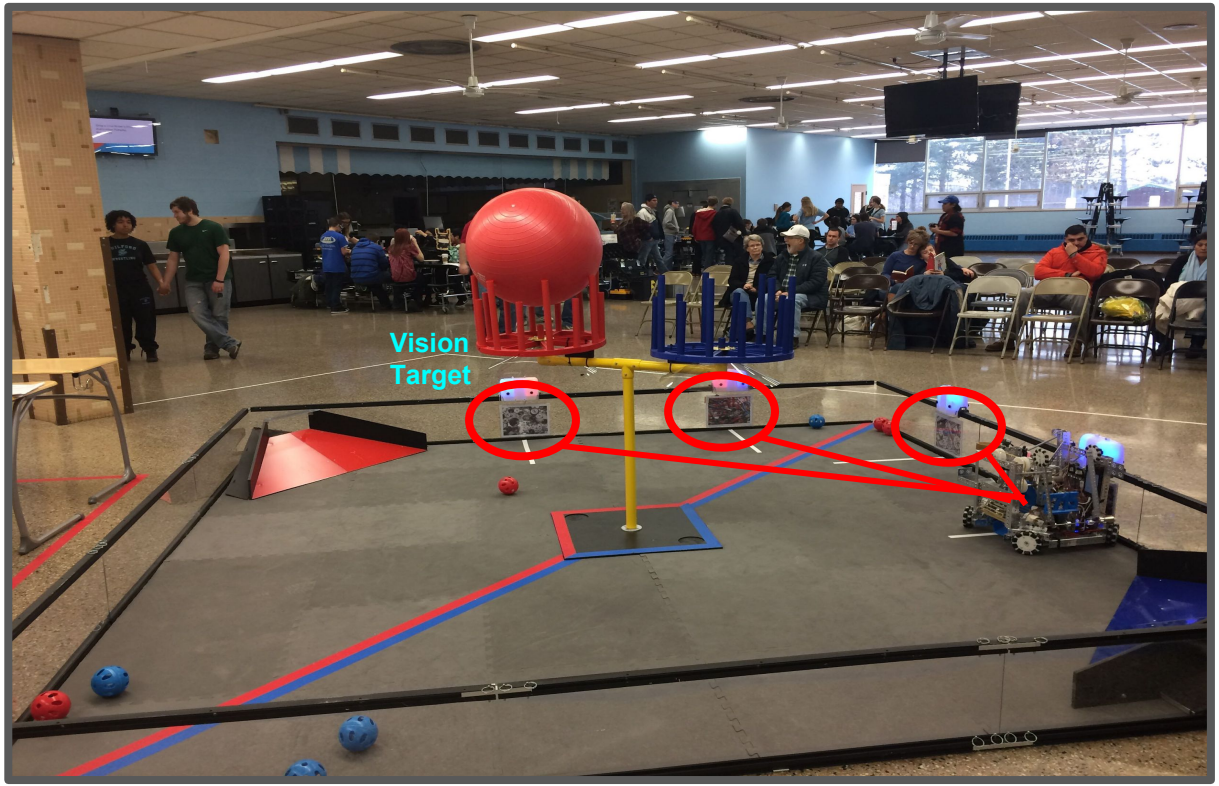
Vuforia: How it Works

Vuforia creates a “localizer” that tracks “trackables”.

- Localizer- controls camera and calculates robot position
- Trackables- targets premade by FIRST
  - Printed on 8x11 paper last year
- Outputs the location and orientation of the trackables
  - → Automatically computes orientation relative to the targets
- Can produce a guess of robot position if it sees multiple targets



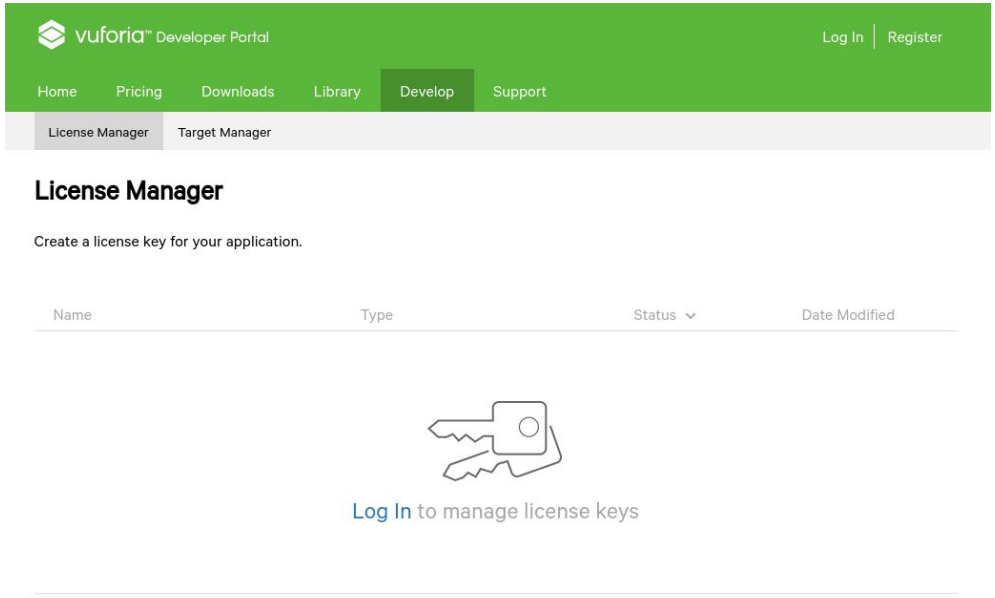

8565-D-RC
Active Configuration:
temp
Network: active, connected
Robot Status: running
Op Mode: Vuforia Tracking Demo


Vision
Target

1. Getting A License Key

https://developer.vuforia.com/license-manager

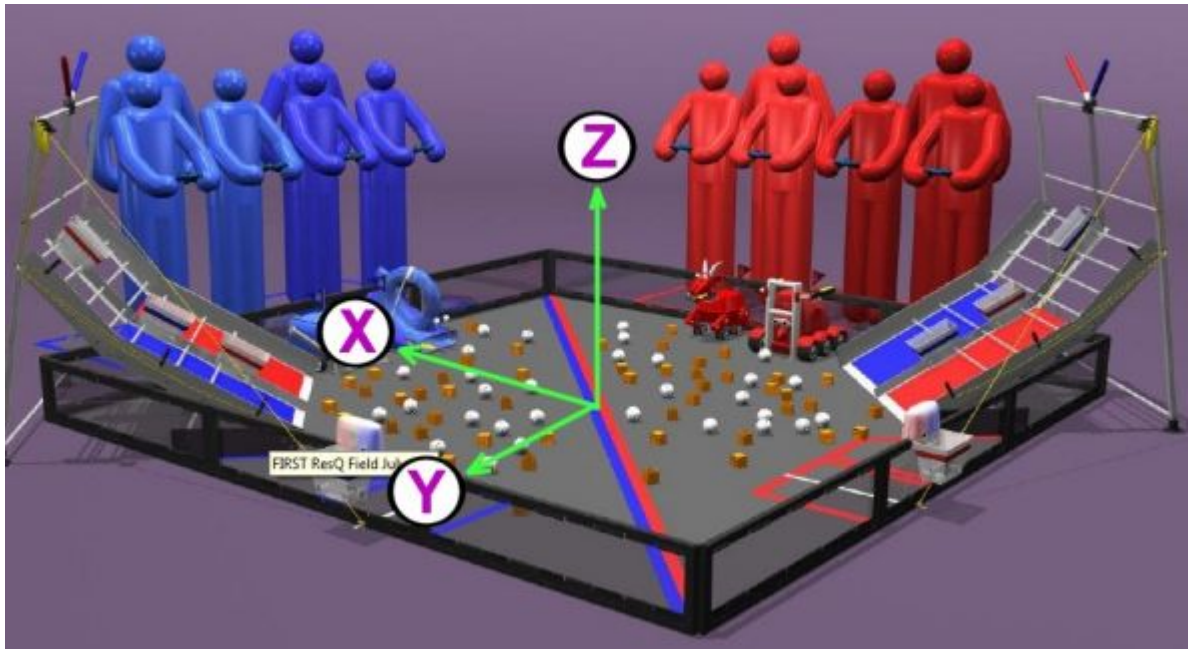
"ASYmU1X/////AAAAGeRbXZz3301OjdKqrFOt4OVPb5SKSng95X7hatnoDN..."



2. Initialization

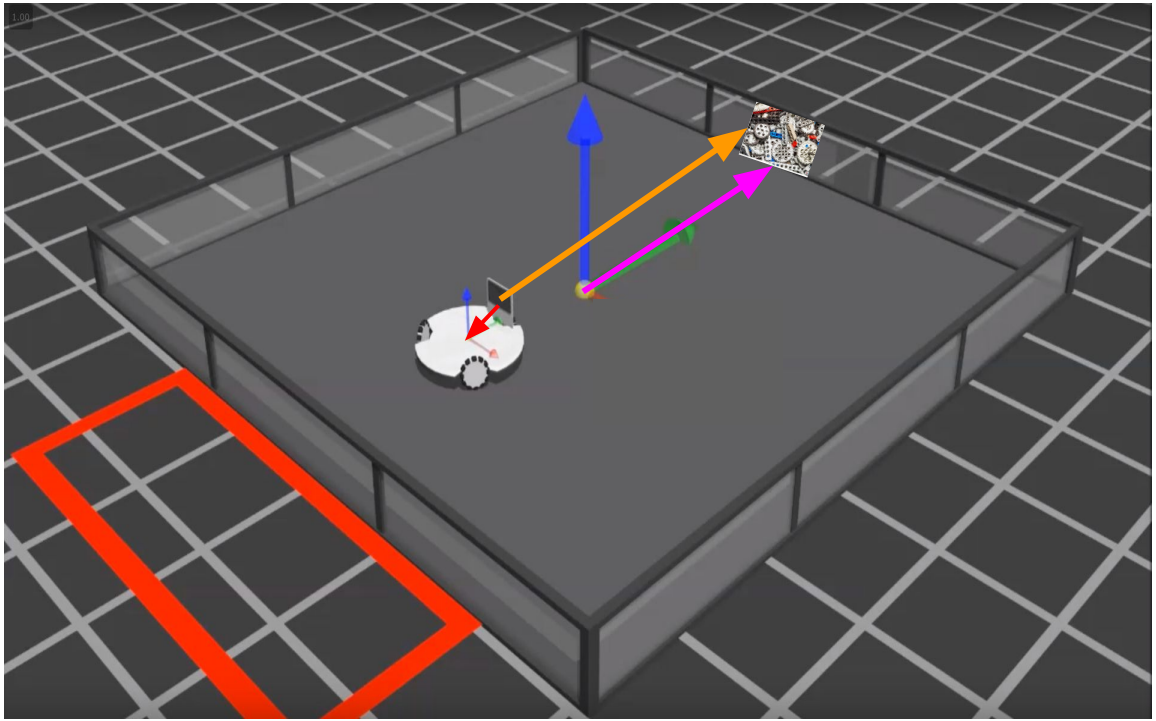
```
public VuforiaLocalizer vuforia; // The localizer
private VuforiaTrackables targets;       // List of active targets


VuforiaLocalizer.Parameters parameters = new VuforiaLocalizer.Parameters (R.id.cameraMonitorViewId);
parameters.vuforiaLicenseKey = [INSERT KEY HERE];

parameters.cameraDirection = VuforiaLocalizer.CameraDirection.FRONT;
vuforia = ClassFactory.createVuforiaLocalizer(parameters);
```

3.1 A coordinate system



The positive Y axis always extends out from the red driver station.

3.2 Orienting the Robot



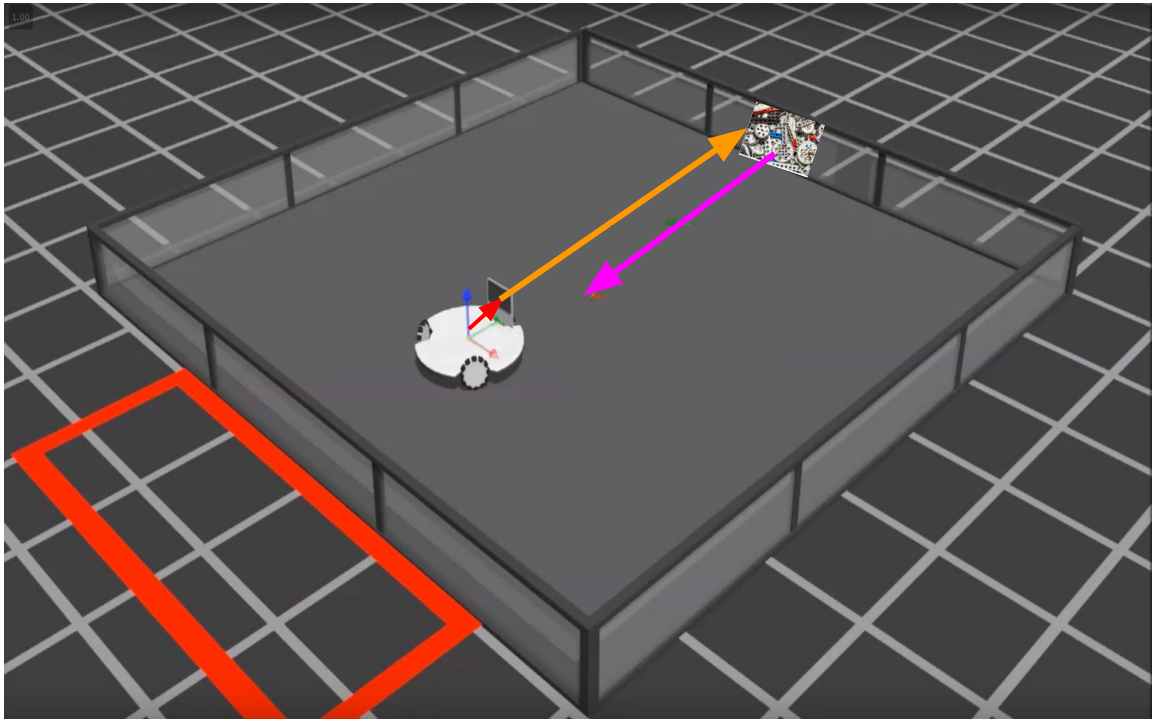
Goal: Where is the robot on the field?

Vuforia computes the camera's position relative to the targets.

You need to input where the phone is on the robot, and where the target is on the field.

Image credit Phil Malone

3.2 Orienting the Robot



Putting it together:

robot position → phone position

+

phone position → target position on field (from Vuforia)

+

target position → field position

=

Robot position → field position (AKA where you are)

Image credit Phil Malone

3.3 Defining the target position

```
OpenGLMatrix targetOrientation = OpenGLMatrix
    .translation(0, 150, 0) // Moves it 0 mm in the X axis, 150 mm in the Y axis, and 0 mm in the Z axis.

    .multiplied(Orientation.getRotationMatrix(
    AxesReference.EXTRINSIC,    AxesOrder.XYZ, AngleUnit.DEGREES, 90, 0, 0));

    // Rotates it 90 degrees clockwise around the X axis, 0 degrees around the Y axis, and 0 degrees around
    the Z axis.
```

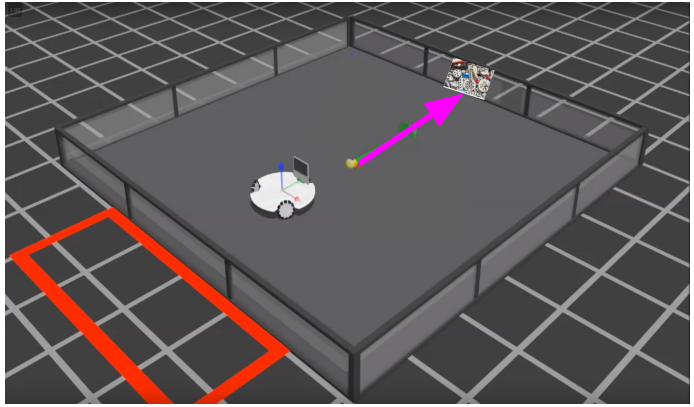
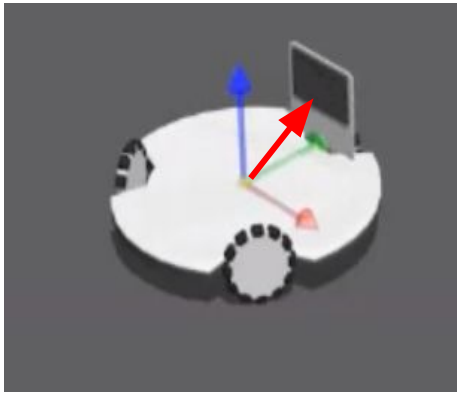


Image credit Phil Malone

3.4 Defining the phone position



```
OpenGLMatrix phoneLocationOnRobot =
OpenGLMatrix
        .translation(110, 0, 50)
        .multiplied(Orientation.getRotationMatrix(
                AxesReference.EXTRINSIC,
                AxesOrder.YZX,
                AngleUnit.DEGREES, 90, 0, 0));
```

Image credit Phil Malone

3.5 Configuring targets

- For each target, give it a location.

```
trackable.setLocation(targetOrientation);
```

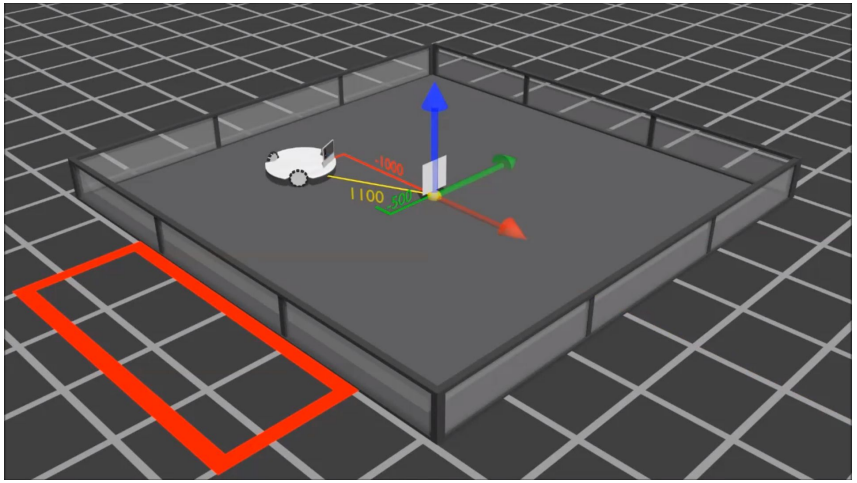
- Also tell it where the phone is on the robot

```
((VuforiaTrackableDefaultListener) trackable.getListener())
    .setPhoneInformation(phoneLocationOnRobot,
    parameters.cameraDirection);
```

4. Finding the target

```
location = listener.getUpdatedRobotLocation(); // Update the location of the robot

    if (location != null) {

     VectorF trans = location.getTranslation();       // Get a translation and rotation vector for the robot

     Orientation rot = Orientation.getOrientation(location, AxesReference.EXTRINSIC, AxesOrder.XYZ, AngleUnit.DEGREES);
      robotX = trans.get(0);       // Get the X and Y coordinates of the robot

      robotY = trans.get(1);

      robotBearing = rot.thirdAngle; // Get the rotation of the robot

}
```

5. Navigation



This assumes the target is located at the origin to simplify calculations.

1. Rotate so the robot is pointing at the target.

   Target angle = arctan(x offset / y offset)

2. Drive forwards until the target is reached.

   Distance to target = √[(x offset)^2 + (y offset) ^2]

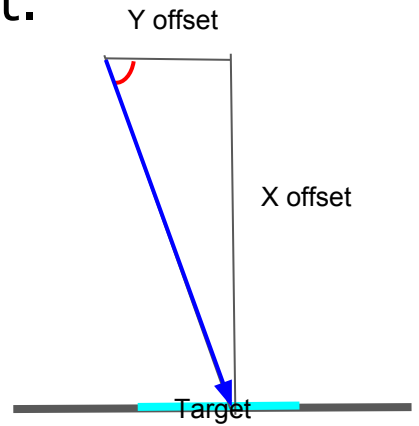


Image credit Phil Malone

Source code credit: Team 2818 G-Force

Code used in this session can be found at

https://github.com/gearsincorg/FTCVuforiaDemo

Companion video at

https://www.youtube.com/watch?v=AxKrJEtfuaI

Questions?