1:30 HR

Need...More...Power

Regulate your robot's speed as slope angles change.

Introduction

Gear up for an exciting new sensor built into your robot - an **Inertial Measurement Unit (IMU)**, also known as a **6-Axis Gyroscope-Accelerometer**. Trying saying that three times fast!

Think of this sensor a bit like an inner compass, giving your robot a keen sense of tilt, position, and motion. With its help, we can collect useful information such as the speed of Alvik, the angle of slopes, or which direction your robot is facing.

This data is crucial to success in our next mission. The goal is to ensure Alvik maintains speed as it moves along inclined paths, automatically regulating power to the wheels as the degree of inclination changes. The steeper the path, the more power we must give the wheels to compensate the force of gravity.

Ready? Let's hit the slopes!

Learning Objectives

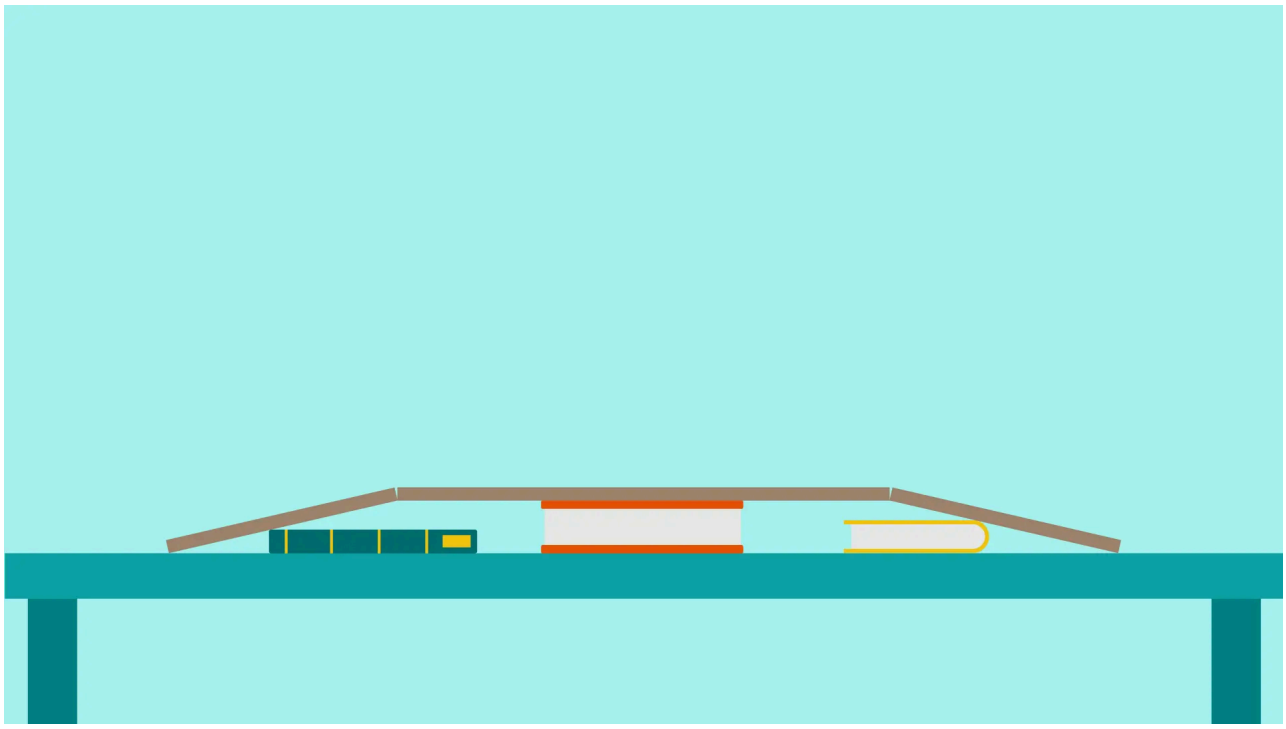
- Understand how the IMU sensor works.
- Develop a program to read and print sensor data.
- Interpret readouts to identify slope angles and orientation.
- Automatically convert angle changes to new wheel speeds.
- Observe changes in acceleration with variations in slope.

Setting Up

For this project, we will need to create one or more ramps. This can be done with any flat object, ideally around 50-100 centimeters in length.

- **Ramp Support** (books or notebooks)
- **Ramp** (wooden board, cardboard sheet, etc.)
- **Tape Measure or Ruler**
- **Notebook** (recommended)

In the example construction below, we are using sheets of plywood and books to adjust height to different angles. In this configuration, we have an incline to climb, a flat plane at the peak, and a decline to descend.

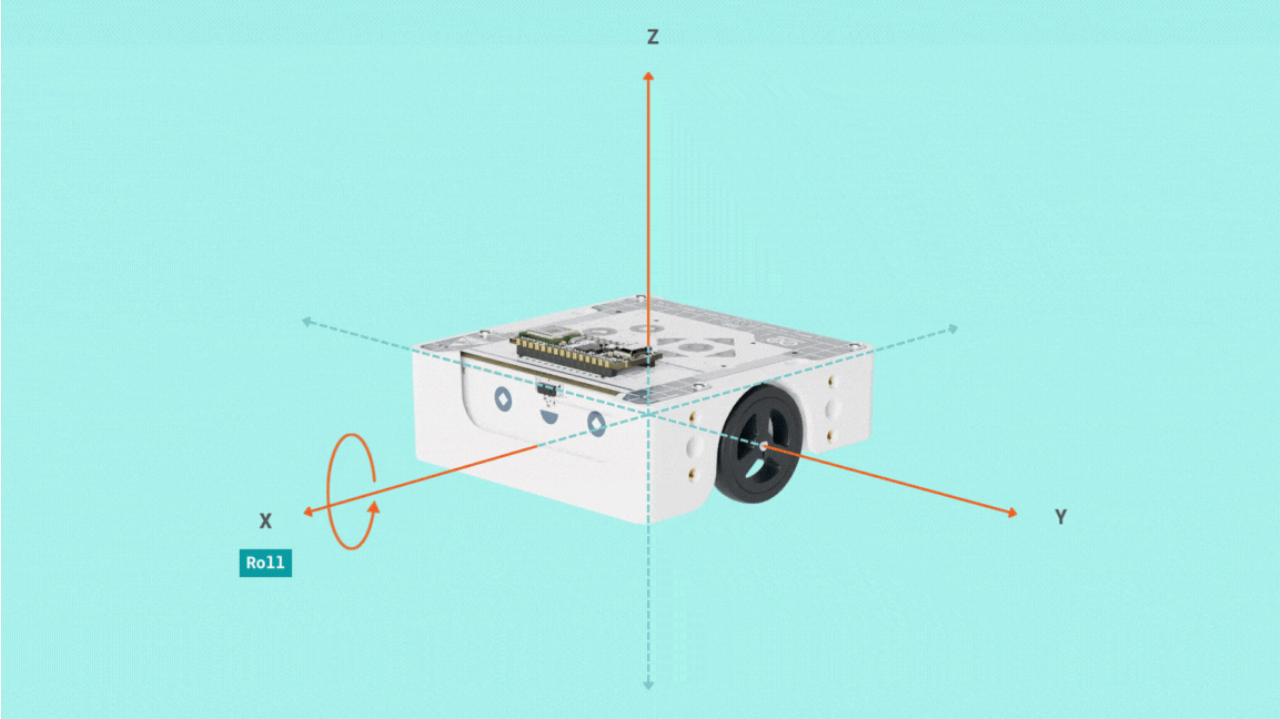


IMU

The 6-Axis Gyroscope-Accelerometer is a type of IMU sensor, which stands for Inertial Measurement Unit. An IMU is a device that helps gadgets like drones, smartphones, and video game controllers know where they are and how they're moving. It does so with two main parts: an accelerometer and a gyroscope.

The accelerometer measures acceleration, which is any change in the gadget's speed or direction. Imagine you're on a skateboard - when you push off the ground, you speed up. The accelerometer in an IMU can sense that speed increase, whether it's forward, backward, or sideways.

The gyroscope measures how much something is tilting or turning. It keeps track of movements like **pitch** (tilting up or down), **roll** (leaning side to side), and **yaw** (turning left or right). By combining these measurements, the IMU can figure out exactly how a gadget is positioned and moving at any moment. This is crucial for keeping things stable and responsive, whether you're flying a drone through the air or tilting your phone to steer in a racing game.



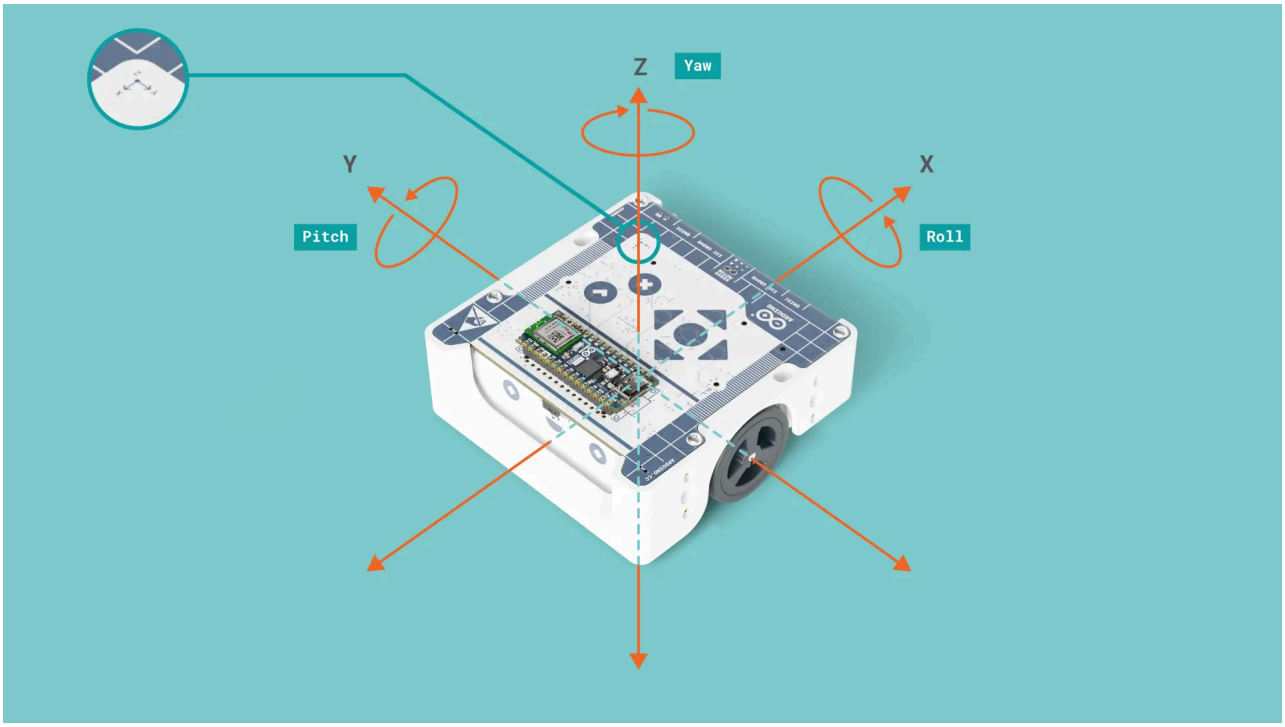
Sensor Test

Create a new file called *imu_test.py* where we will write a simple script to view the readouts of the IMU sensor.

```
from arduino import *
from arduino_alvik import ArduinoAlvik

alvik = ArduinoAlvik()

def setup():
  alvik.begin()
  delay(1000)

def loop():
  roll, pitch, yaw = alvik.get_orientation()
  # Uncomment next line for rounded readout values
  # roll, pitch, yaw = round(roll, 2), round(pitch, 2), round(yaw, 2)
  print(roll, "|", pitch, "|", yaw)
  delay(500)

def cleanup():
  alvik.stop()

start(setup, loop, cleanup)
```

The method `get_orientation()` returns IMU sensor values for roll, pitch, and yaw - each which are assigned to corresponding variables in a single line. As you

observe the printed readouts, you will notice that **float values** are quite long, so we have an optional line of code to round floats using the `round()` function. You may update your printed readouts by uncommenting this line.



Look carefully at the top of your robot where you will find a printed symbol showing the direction of the X, Y, and Z axes. As shown in the previous illustration, these axes represent the center of rotation for roll, pitch, and yaw movements. For example, tilting Alvik forward and backward represents the pitch movement, which is the same as turning around the Y axis.

Take a moment to rotate your robot along each axis while observing the roll, pitch, and yaw values printed to the terminal. Yaw is much like a compass, but rather than using North, South, East, and West for finding its orientation, we simply use a range from 0 to 360 degrees. Roll and pitch, on the other hand, are measured in a range of -180 to 180 degrees.

Once you feel comfortable understanding the relationship between movements and the values displayed in the terminal, move on to the next section to put it into practice.

Adaptive Speed

Alright, it's time to test out the ramp you've built. Start a new script in Arduino Lab and save it as ***adaptive_speed.py***. This script may look pretty long, but we will break it down piece by piece to help you understand what's going on under the hood.

```
from arduino import *
from arduino_alvik import ArduinoAlvik

alvik = ArduinoAlvik()

def setup():
  alvik.begin()
  delay(1000)

def loop():
  roll, pitch, yaw = alvik.get_orientation()
  print(pitch)
  delay(100)

  BASE_SPEED = 10
  MULTIPLIER = 2  # Change multiplier to adjust extra_rpm strength
  extra_rpm = abs(pitch) * MULTIPLIER
  adapted_speed = BASE_SPEED + extra_rpm

  # Robot will only move if slope is detected within 15 and -15 pitch
```

The main loop starts off by gathering and printing data to the terminal. Since we are focused on the angles of slopes as we drive forward and backward, we only need to know Alvik's change of pitch, which we have assigned to the variable `pitch`.

Next, we have some **constants** that never change and formulas that will later help your robot adapt its speed. Ideally, the constants (shown in ALL CAPS) should be declared before the `def setup()`, but we will keep everything together for now.

```
BASE_SPEED = 10
MULTIPLIER = 2  # Change multiplier to adjust extra_rpm strength
extra_rpm = abs(pitch) * MULTIPLIER
adapted_speed = BASE_SPEED + extra_rpm
```

The `BASE_SPEED` is the minimum RPM of Alvik's wheels while moving. In the last line of this block, we have `adapted_speed`, which represents the base speed of 10 RPM plus a little extra push depending on the steepness of the slope angle. The greater the angle, the more power we give to the motors by adding `extra_rpm` to the `BASE_SPEED`.

Since upward slope angles are negative and downward slope angles are positive, the formula of `extra_rpm` includes the `abs()` function to provide the absolute value of `pitch`. In other words, no matter the readout, `pitch` will always be a positive value when calculating `extra_rpm`.

Now, you may be wondering why we have `MULTIPLIER`, and the answer is simple - it's basically like nitro! If you want Alvik to strongly adapt it's speed, increase the `MULTIPLIER` value to boost the `extra_rpm`.

```
# Robot will only move if slope is detected within 15 and -15 pitch
if pitch < 15 and pitch > -15:
    # Incline slope
    if pitch < -1:
        alvik.set_wheels_speed(adapted_speed, adapted_speed)
    # Decline slope
    elif pitch > 1:
        alvik.set_wheels_speed(-adapted_speed, -adapted_speed)
    # No slope
    else:
        alvik.set_wheels_speed(0, 0)
    delay(500)
else:
    alvik.set_wheels_speed(0, 0)
```

You should be familiar with nested conditions from our last lesson. The first condition `if pitch < 15 and pitch > -15:` provides us with a realistic working range for experimentation. If slopes fall outside of this range, Alvik will have a difficult time climbing such steep angles, and so we simply set the wheels to `(0, 0)` as shown in the final `else:` statement.

However, if the first statement is `True`, we check three possible scenarios. First, if `pitch < -1`, we can conclude that your robot is on an inclined slope and must adapt its speed accordingly using `adapted_speed` as its RPM values. Second, if `pitch > 1`, it's clear that Alvik is on a downward slope and must adapt its speed in reverse, hence the negative `adapted_speed` for reverse RPM. At last, any value between -1 and 1 represents a relatively flat surface, so speed values are set to zero.

> **NOTE:** During testing, you may wish to disconnect the programming cable. Remember to simply transfer the script to your board and update *main.py*

with `import script_name` so that the program automatically runs at reboot.

Now that you understand the code, take a moment to test the results on the ramp you built. Place your robot on the upward or downward slope and observe its behavior. As you are testing, try lifting and lowering the ramp with your hand. You should notice the motors respond to changes in slope. Once the robot reaches a flat surface, it should stop entirely until placed on a slope once again.

Oh Yaw

To consolidate what you have learned, we have a very unique challenge that will require some critical thinking. We understand how to convert IMU readings to adapted values that regulate speed. By converting sensor values to wheel movements and strategically using conditions, there are many responsive movements we can program.

Let's focus on yaw, an IMU readout measured from 0º to 360º about the Z axis. Imagine Alvik is placed on a notebook - as you slowly spin the notebook around, we would observe a change of degrees representing the direction your robot is facing. As you rotate, wouldn't it be cool if Alvik could adaptively use a spin turn so that it always faces one direction? Is it possible... you bet!

The goal in this challenge is to program your robot to adjust its wheel speeds so that it always faces 180º. When spun around on a flat object such as a notebook in a clockwise or counterclockwise direction, the wheels must change power and direction to realign to the target orientation.

Good luck - try not to get dizzy!