What do drones think about?

Shallow dive into embedded Rust and sensors.

$whoami

Dawid Królak

- General purpose software engineer
- Rust enthusiast
- Amateur volleyball player
- Aerorust communit y core member
- Passionate about European space exploration and engineering programs.

https://twitter.com/@Taavicjusz

https://github.com/taavit

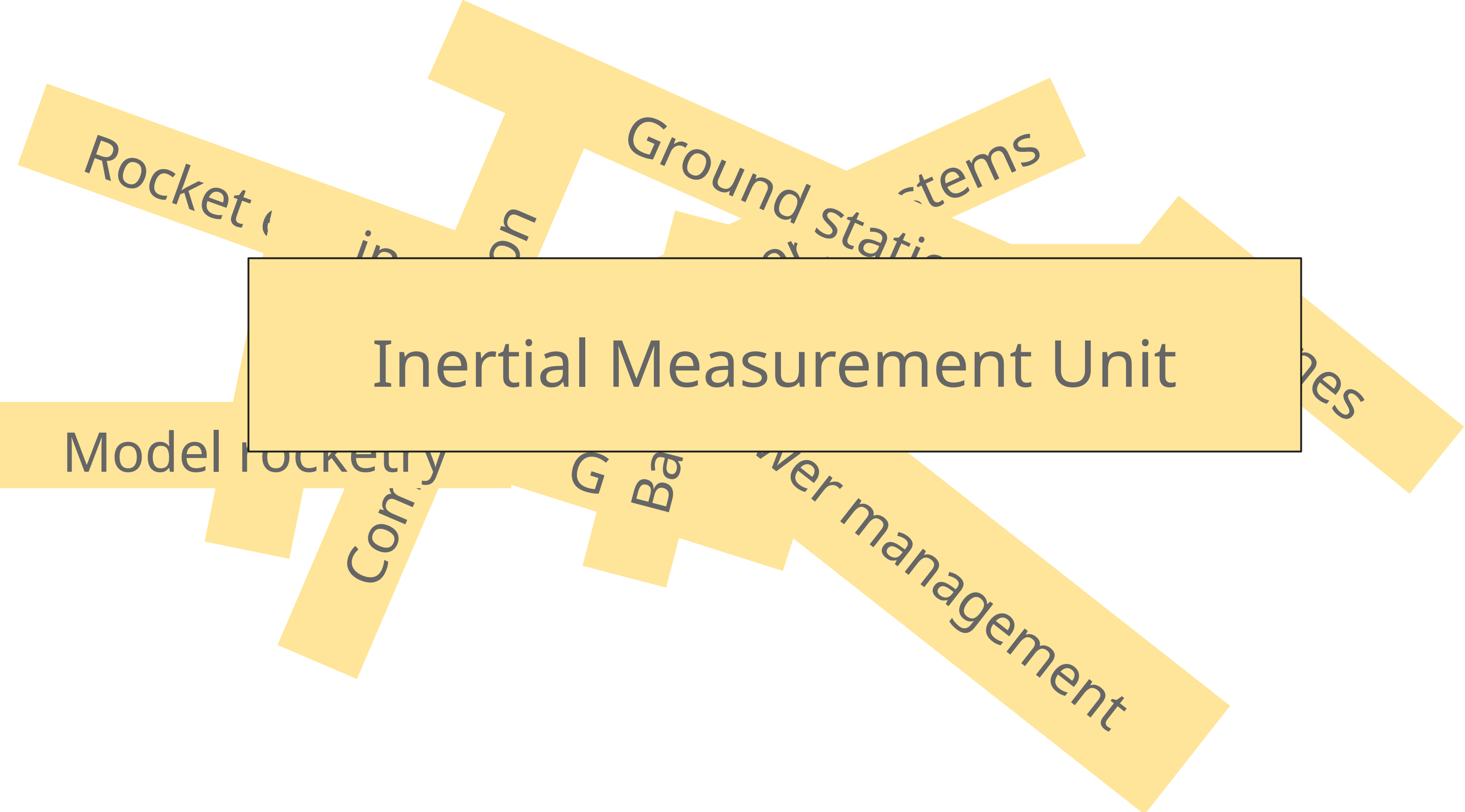
Aerorust

Our community aims to grow the Rust programming language ecosystem for aerospace applications because it "empowers everyone to build reliable and efficient software".

- Nanosat workshops
- NMEA crate
- Free-flight-stabilization
- Casual meetups

https://aerorust.org/

What to choose…



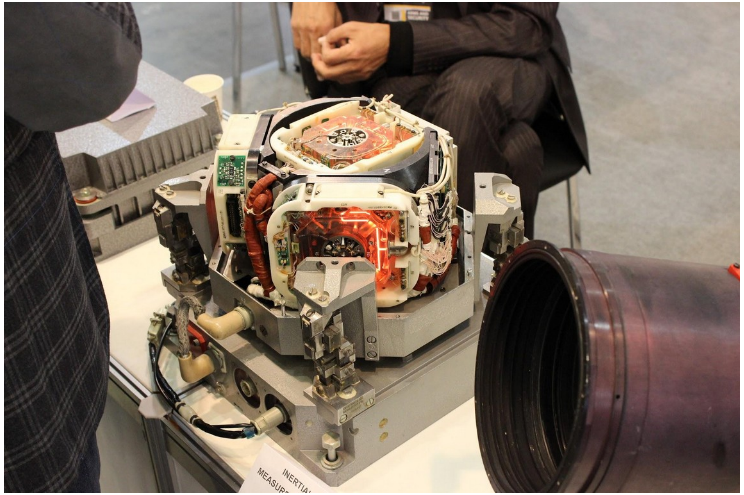
Basics

What is IMU?

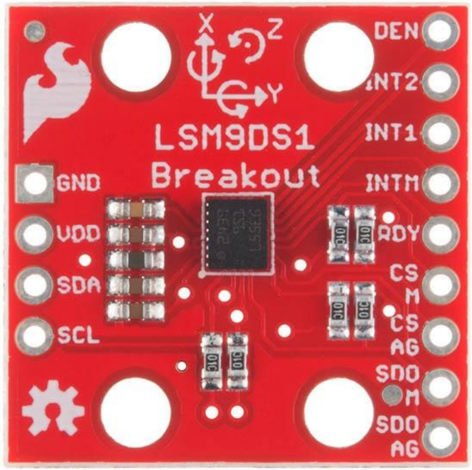
Device that measures and report forces acting on a given object that combined can provide orientation in a given geometric space (reference frame).

It’s about attitude aka orientation in space, not position. (For position we have Inertial Navigation System)

IMU in real life



VictorAnyakin, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons



What IMU reports

Quaternions

- Special number system (Complex numbers on steroids)
- Easy to calculate (matrices) by computer
- For some people new concept needed to learn

$$a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$$

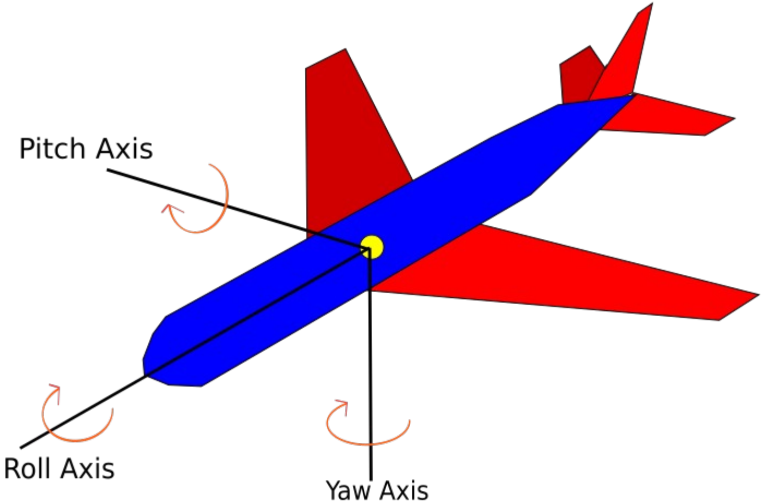
Euler angles

- Easy to understand
- Easy to read
- Not the best for machine calculation
- Good enough for now

Euler angles

To recreate orientation we have to determine:

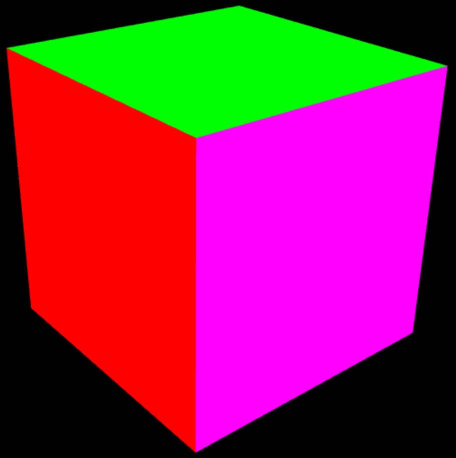
- yaw
- pitch
- roll



Use the same order of angles!

Project!

Display cube with real time orientation



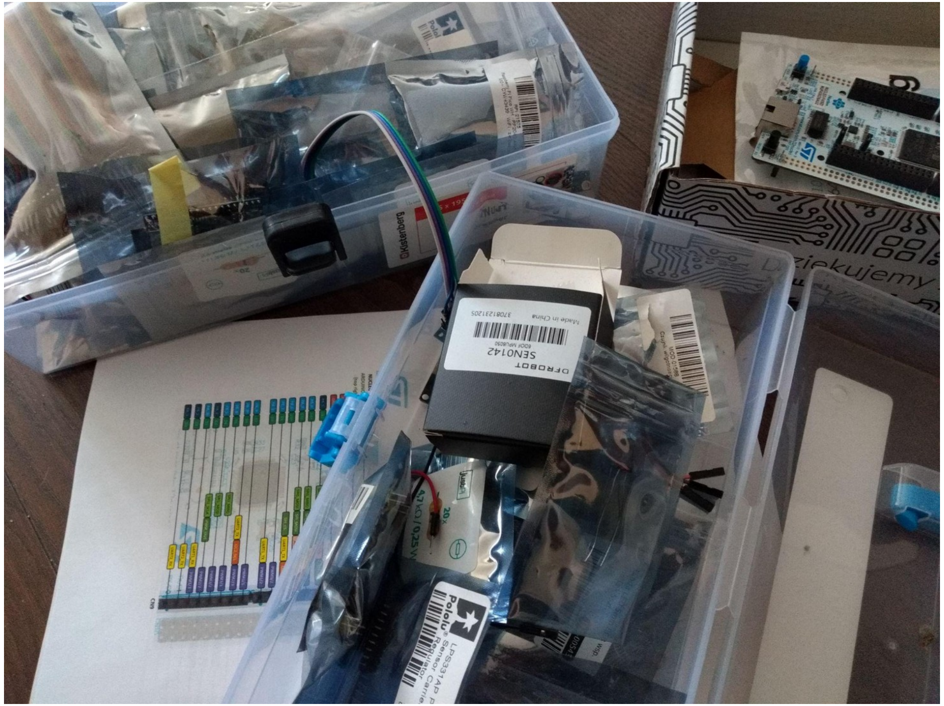
Step by step:

- Find MCU, sensors, breadboards
- Connect everything
- Write firmware
- ~~Write frontend~~
- Plug and play!

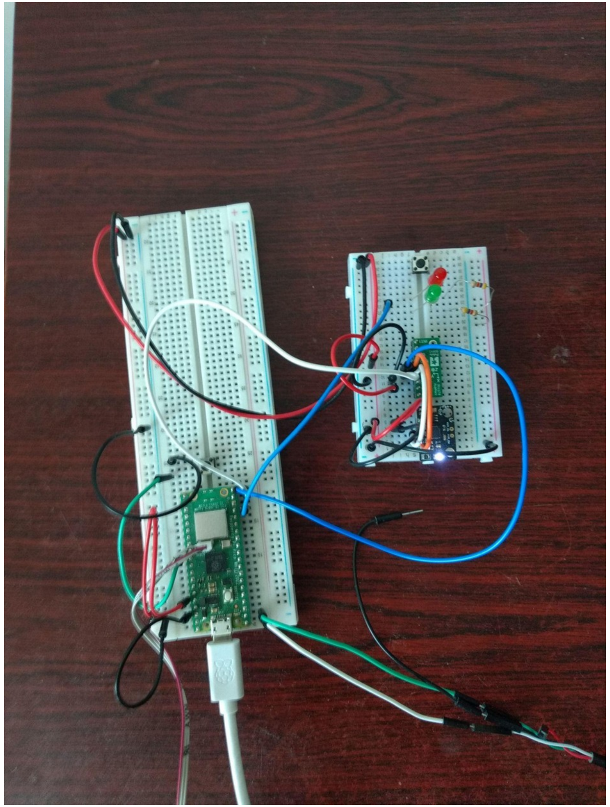
Setup hardware and software

Finding stuff

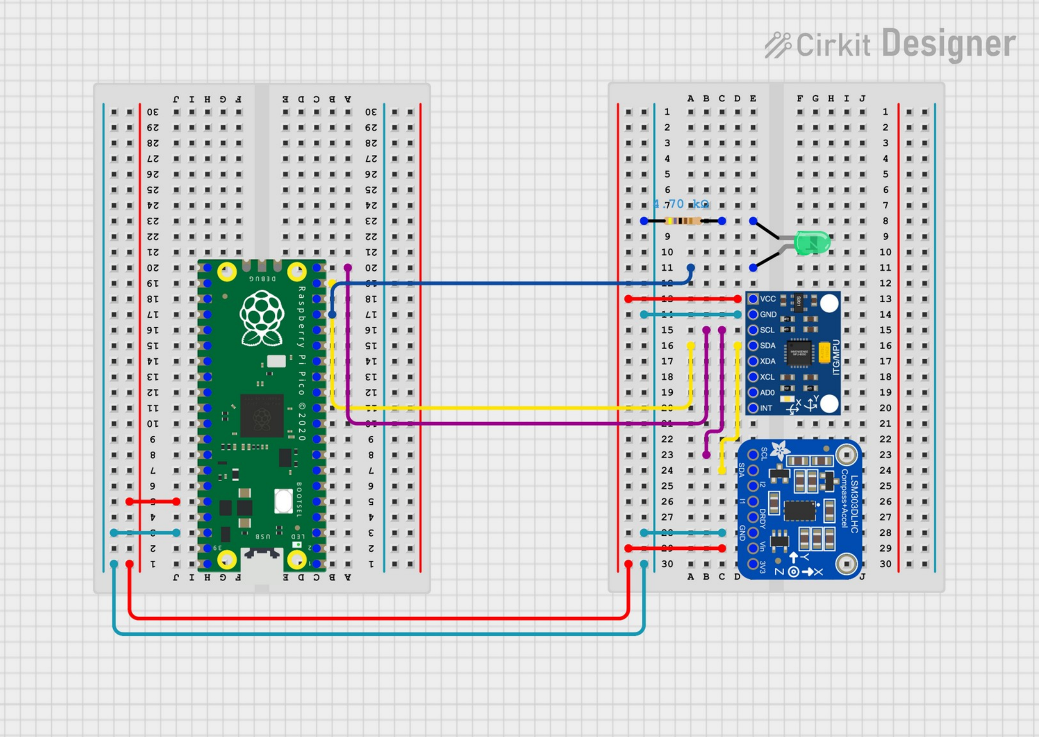
- 2x Breadboard
- Raspberry Pico
- LSM303D (acc+mag)
- MPU6050 (gyro+acc)
- 1 x LED
- 1 x 4.7k resistor



Connecting



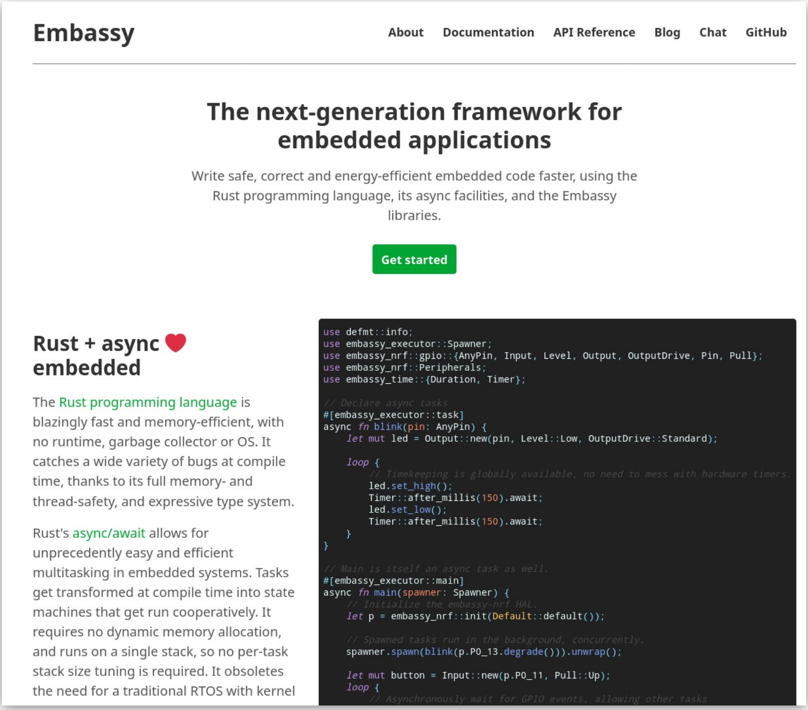
Connecting


Cirkit Designer
470 kΩ
VCC
GND
SCL
SDA
XDA
XCL
AD0
INT
ITG/MPU
LSM303DLHC
Compass+Accel
Raspberry Pi Pico ©2020

Firmware

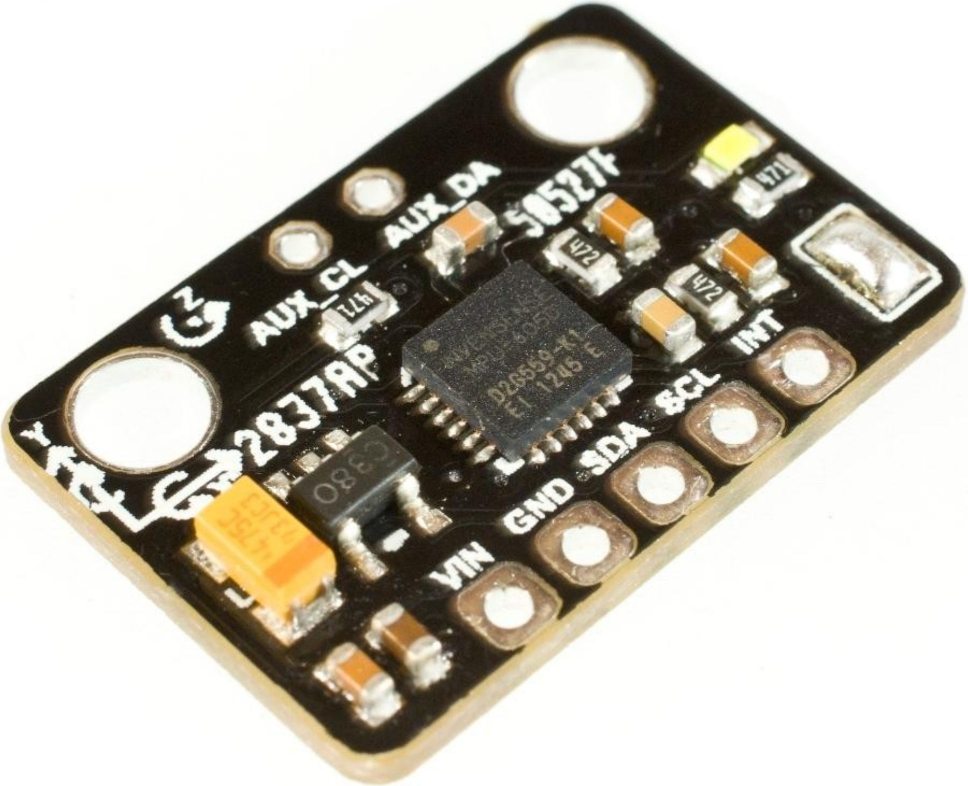
Embassy

- Multiple tasks
- Well documented
- De-facto standard for embedded rust framework



Code setup (embassy)

```rust
let p = embassy_rp::init(Default::default());
let i2c = embassy_rp::i2c::I2c::new_blocking(p.I2C1, scl, sda, Config::default());

let config = uart::Config::default();

let mut uart = uart::Uart::new(
    p.UART0, p.PIN_0, p.PIN_1, Irqs, p.DMA_CH0, p.DMA_CH1, config,
);
```



MPU6050

MPU6050 - Gyro + Accelerometer

- Ready to go synchronous driver for sensor
- Contains DMP (Digital Motion Processor)
  - Provides ready to go attitude data
  - Needs upload firmware to the sensor
  - Check licences!

Gyroscope data

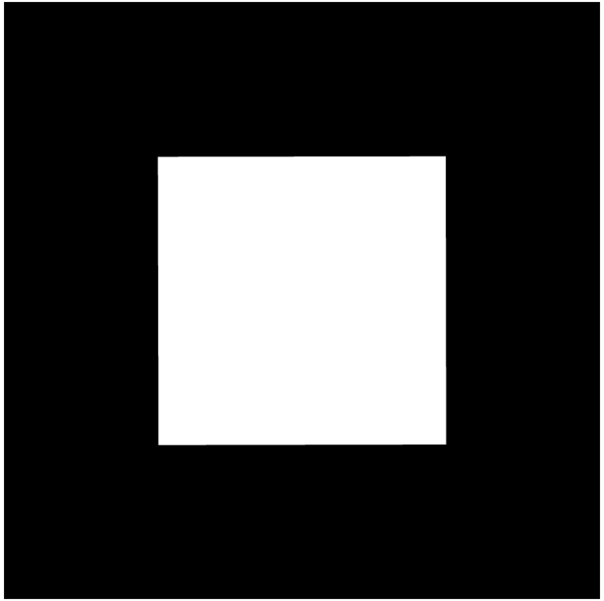
- Read angular velocity (deg/s)
- Read time between measurements
- Integrate
- Voila!

Reading data from sensor

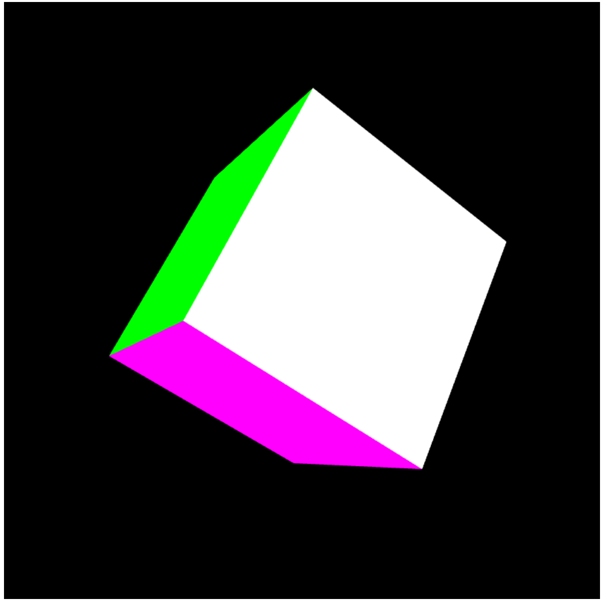
```rust
let mut now = Instant::now();
let mut pitch = 0.0;
let mut roll = 0.0;
let mut yaw = 0.0;

loop {
    let mut buffer = String::<64>::new();
    let gyro = sensor.gyro().unwrap().scaled(GyroFullScale::Deg2000);
    let cur_time = Instant::now();
    let timestep = ((cur_time - now).as_micros() as f32) / 1_000_000.0;
    now = cur_time;
    let step_x = (gyro.x() * timestep).to_radians();
    let step_y = (gyro.y() * timestep).to_radians();
    let step_z = (gyro.z() * timestep).to_radians();

    pitch += step_x;
    roll += step_y;
    yaw += step_z;

    write(&mut buffer, format_args!("/*{:.5},{:.5},{:.5}*/\r\n", f32::from(yaw), f32::from(pitch), f32::from(roll))).unwrap();
    uart.write(buffer.as_bytes()).await.unwrap();
}
```

Results

How it's started



How it's going (after few minutes)



Gyro is (not) enough

- Detect current angular velocity, not attitude (need integrate and sum)
- Noise (as every sensor)
- Drift
- Not perfect

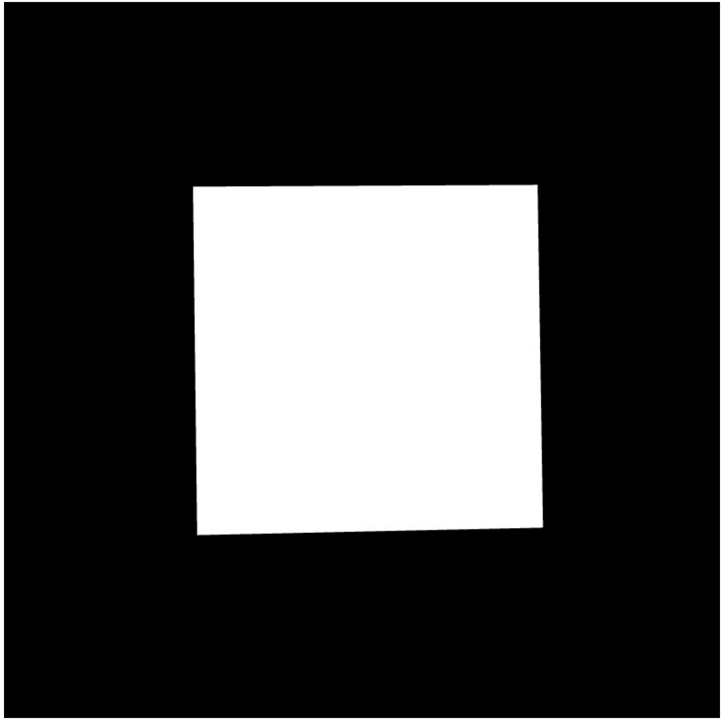
Timestep	x	y	z
0.009456	-0.111343330	0.10919662	-0.0060349824
0.009466	-0.111383624	0.10924699	-0.0060349824
0.009452	-0.111393680	0.10921682	-0.0060349824
0.009487	-0.111413874	0.10922691	-0.0060450790
0.009451	-0.111413874	0.10923697	-0.0060249628
0.009463	-0.111413874	0.10923697	-0.0060148920
0.009551	-0.111444370	0.10927763	-0.0060047274
0.009459	-0.111484630	0.10928769	-0.0059443284
0.009445	-0.111484630	0.10931785	-0.0059141736
0.009388	-0.111504614	0.10934782	-0.0059041830
0.009388	-0.111514606	0.10934782	-0.0059041830
0.009450	-0.111524664	0.10935788	-0.0058639552
0.009454	-0.111524664	0.10938806	-0.0058538940
0.009449	-0.111554830	0.10937800	-0.0058840616
0.009484	-0.111564930	0.10938810	-0.0058840616
0.009462	-0.111585066	0.10938810	-0.0058739916
0.009467	-0.111605210	0.10939817	-0.0058739916
0.009451	-0.111625330	0.10938811	-0.0058639340

Let's add accelerometer

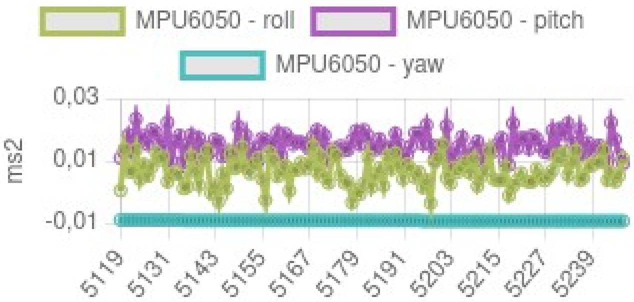
```
let accel_yaw = 0.0;
let accel_pitch = -libm::atan2f(accel.y(), accel.z());
let accel_roll = -libm::atan2f(
    -accel.x(),
    libm::sqrtf(accel.y() * accel.y() + accel.z() * accel.z()),
);
```

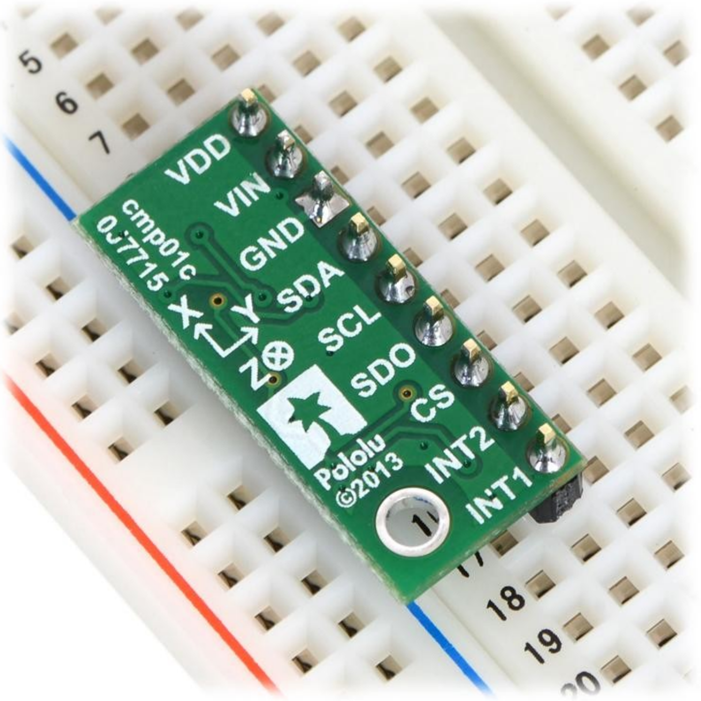
Cannot calculate yaw
This works thanks to gravity in Z axis

Results



Problem: **Noise**!

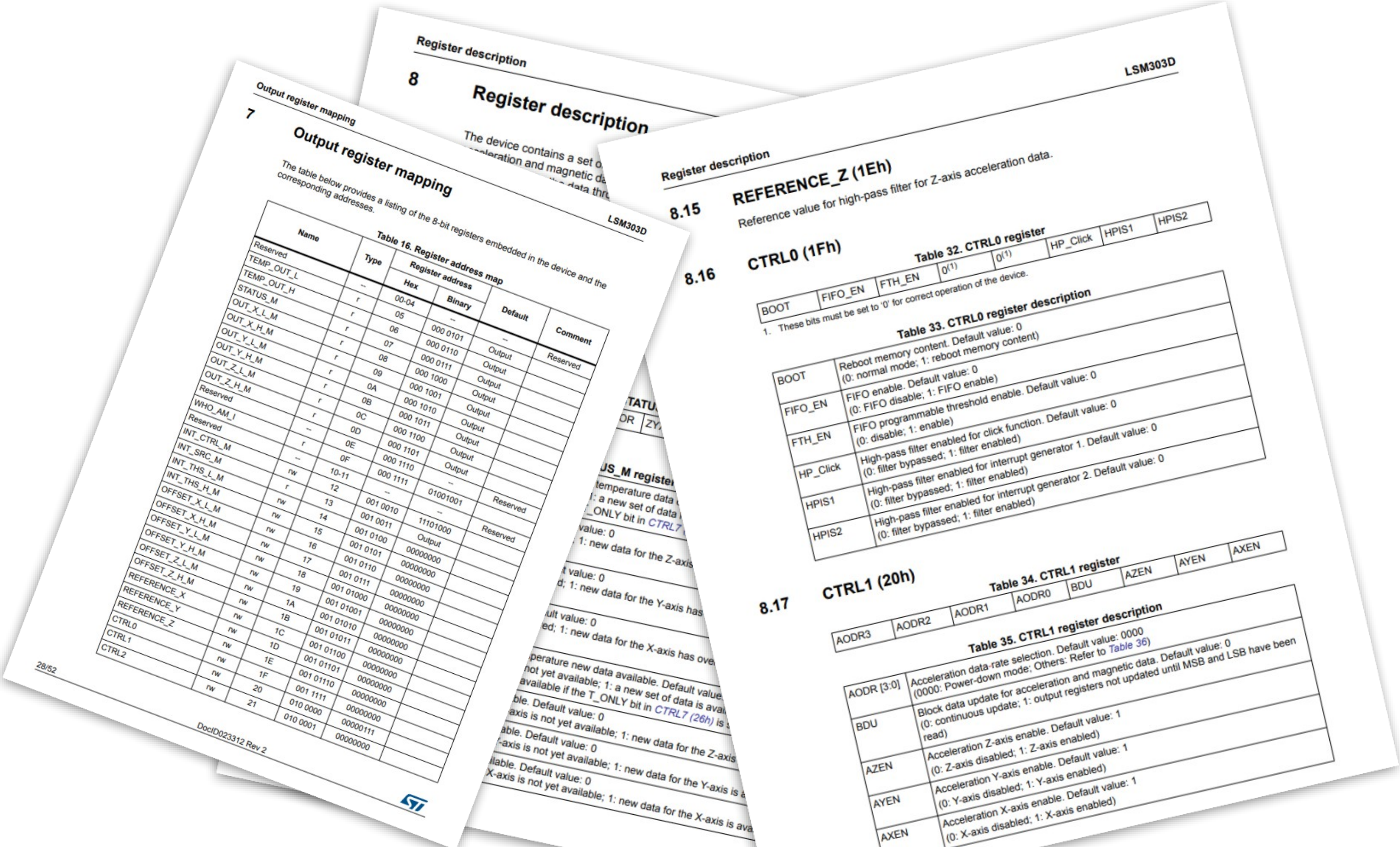




https://www.pololu.com/product/2127

LSM303D - Accelerometer + Magnetometer

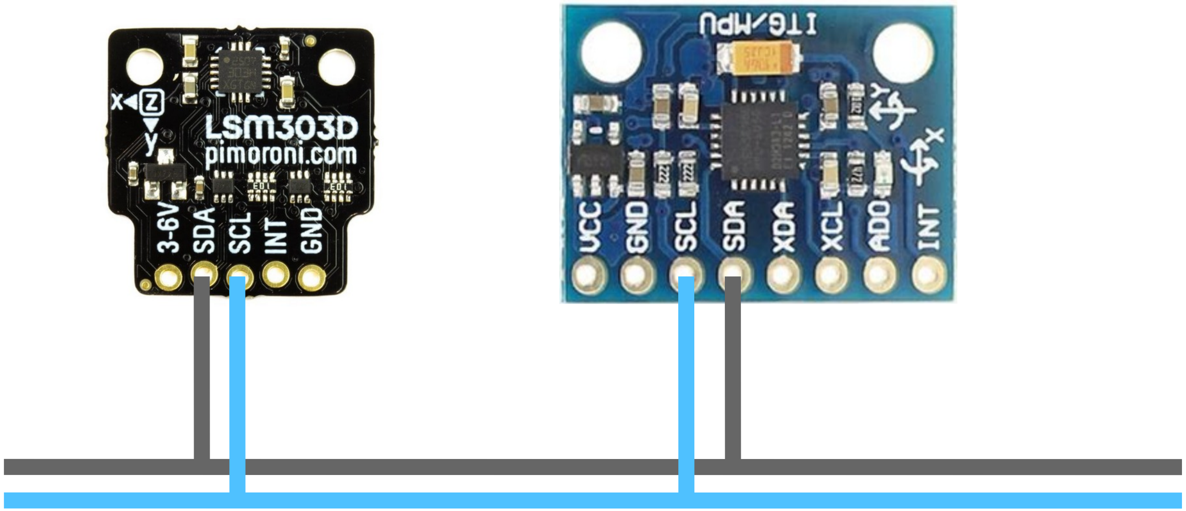
LSM303D - Driver



LSM303 - Driver

```rust
use embedded_hal_async::i2c::I2c;

pub struct LSM303D<I2C: I2c> {
    i2c: I2C,
    acc_multiplier: f32,
    mag_divider: f32,
    address: u8,
}

impl<I2C: I2c> LSM303D<I2C> {
    pub fn new(i2c: I2C) -> Self {
        Self {
            i2c,
            mag_divider: 1.0,
            acc_multiplier: 1.0,
            address: ADDRESS,
        }
    }

    pub async fn configure_int1(&mut self, configuration: Int1Configuration) -> Result<(), ()> {
        self.i2c
            .write(self.address, &[Register::Ctrl3 as u8, configuration.into()])
            .await
            .map_err(|_| ())
    }
```

I2C Bus



Who owns I²C?

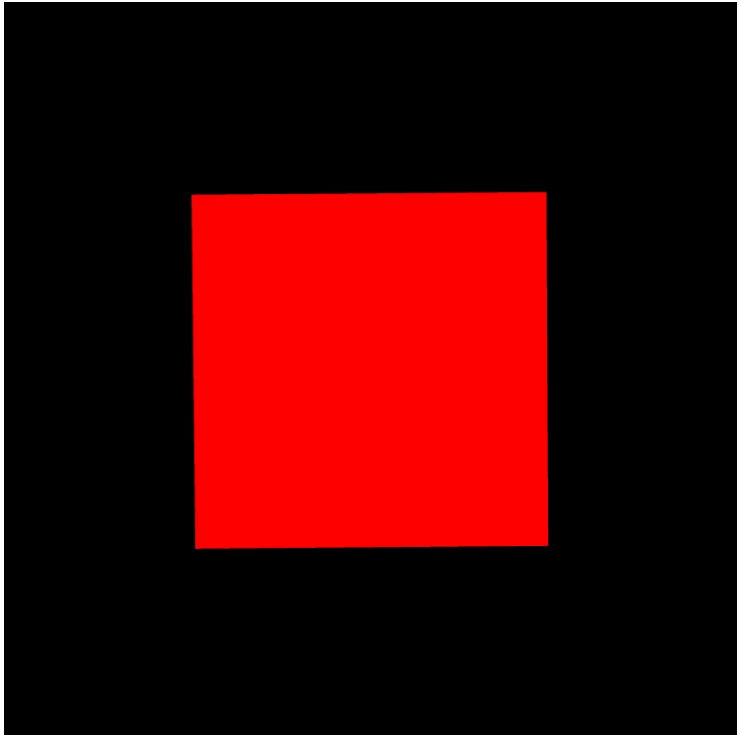
Sharing sync I²C bus

```
static I2C_BUS: StaticCell<
   Mutex<CriticalSectionRawMutex, RefCell<I2c<I2C1, embassy_rp::i2c::Blocking>>>,
> = StaticCell::new();

#[embassy_executor::main]
async fn main(_spawner: Spawner) {
///… Code
   let i2c = embassy_rp::i2c::I2c::new_blocking(p.I2C1, scl, sda, i2c_config);
   let i2c_bus = Mutex::new(RefCell::new(i2c));
   let i2c_bus = I2C_BUS.init(i2c_bus);
   let i2c_dev1 = I2cDevice::new(i2c_bus);
   let i2c_dev2 = I2cDevice::new(i2c_bus);
///… Code
}
```

Adding magnetometer

```rust
let mut lsm303 = LSM303D::new(i2c_dev2);
let lsm_config = Configuration::default().configure_magnetometer(
    MagnetometerDataRate::Hz50,
    MagneticSensorMode::ContinuousConversion,
    MagnetometerFullScale::Mag2,
    MagnetometerResolution::High,
);
lsm303.configure(lsm_config).unwrap();
loop {
    ///… Code
    let mag_result = lsm303.read_measurements().unwrap().magnetometer;
    let mag_yaw = libm::atan2f(mag_result.x, mag_result.y);
    ///… Code
}
```

Results

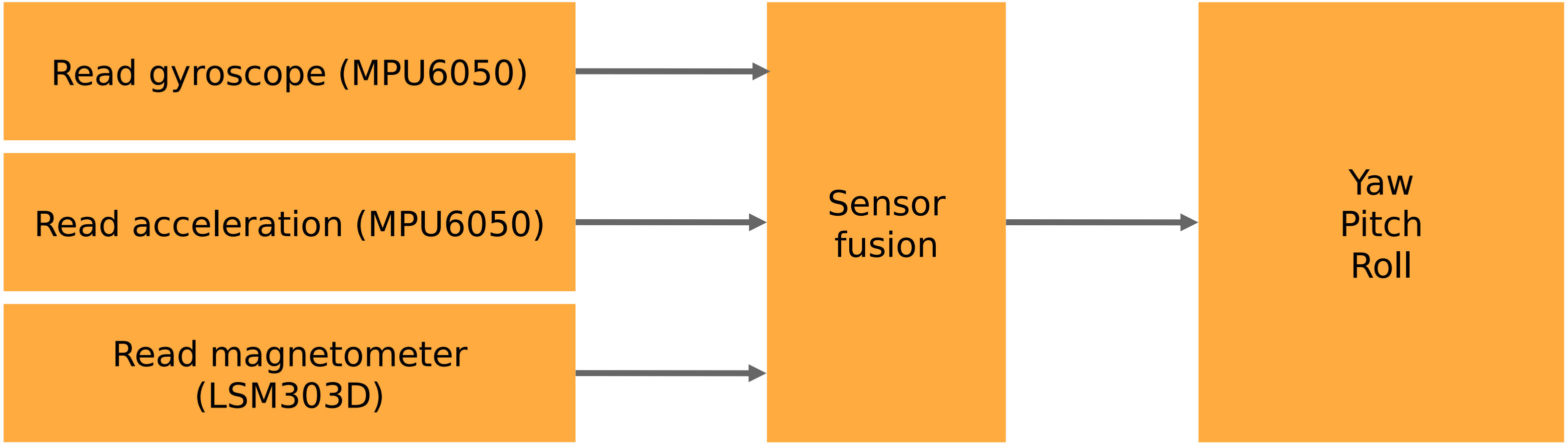


Problems:
Electric devices can affect measurement!

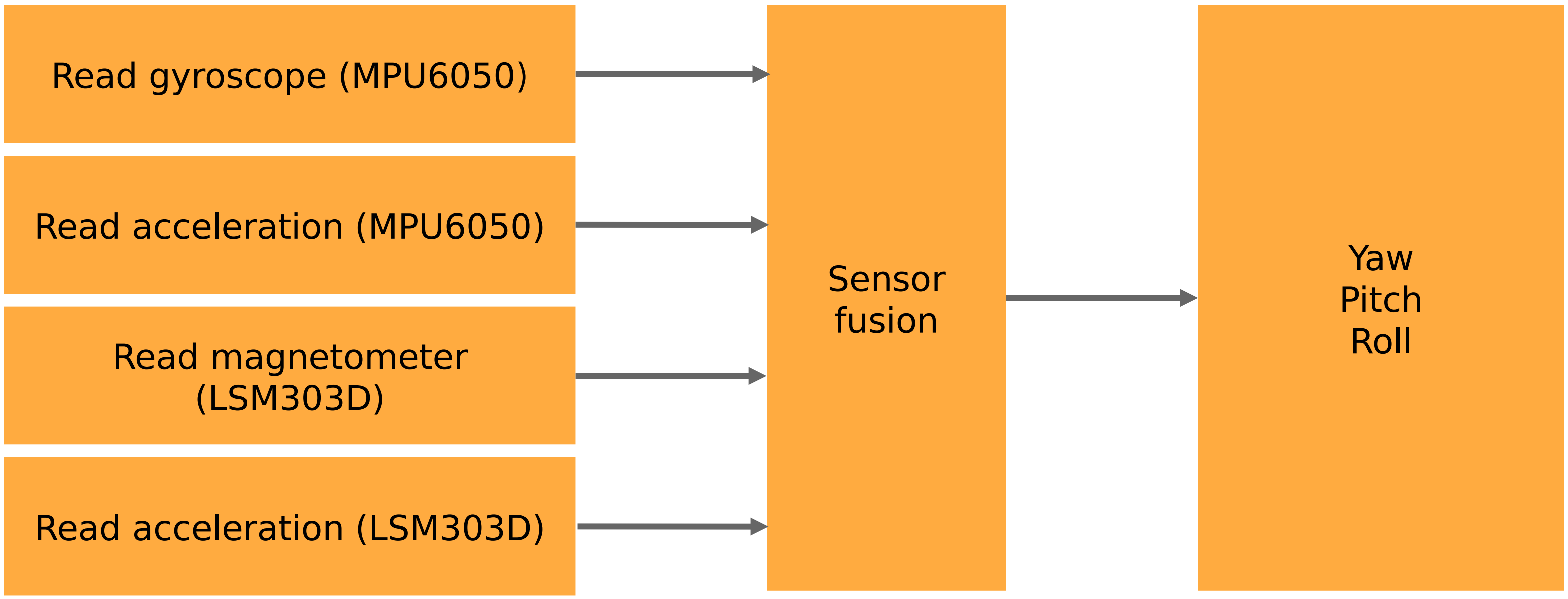
Mostly used for yaw

Sensor fusion

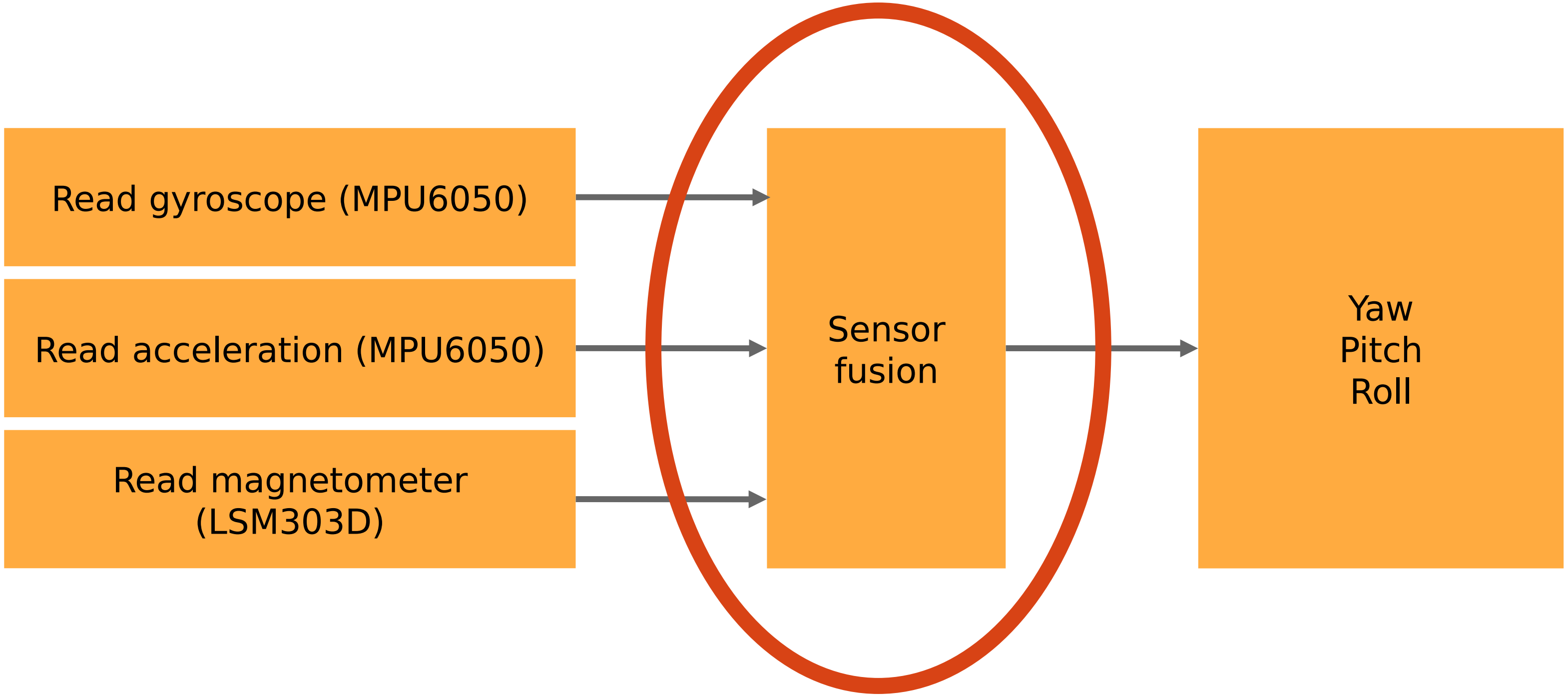
Sensor fusion



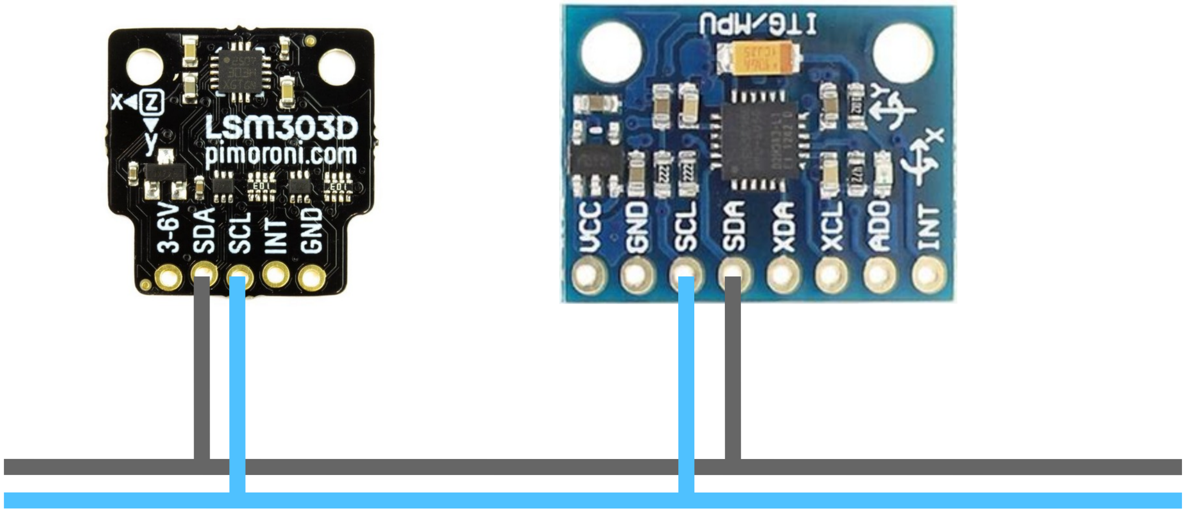
Sensor fusion



Sensor fusion

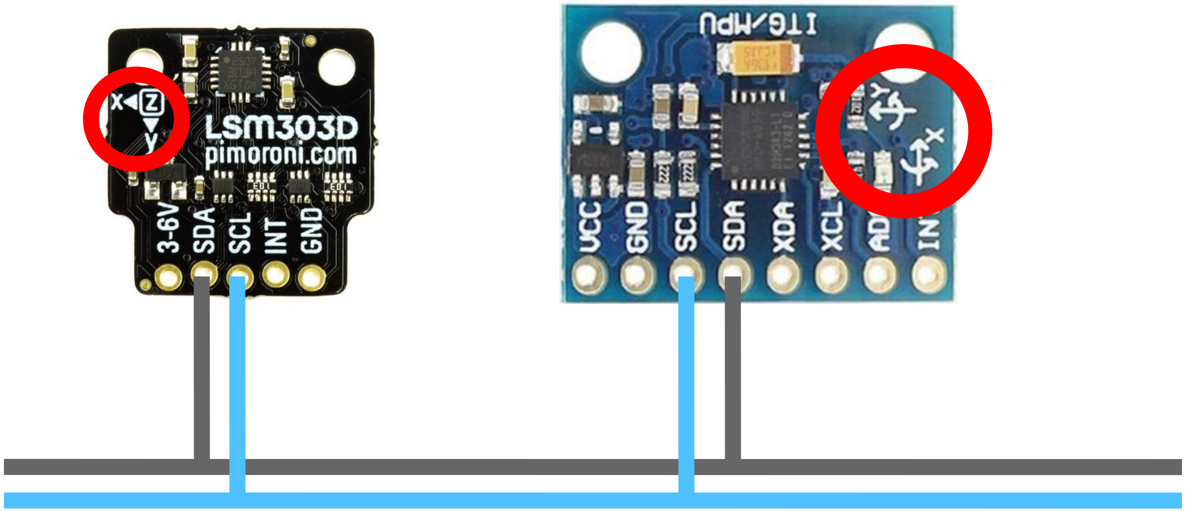


I2C Bus



Who owns I²C?

Check axis!



Double check this, trust me ;)

Data Fusion

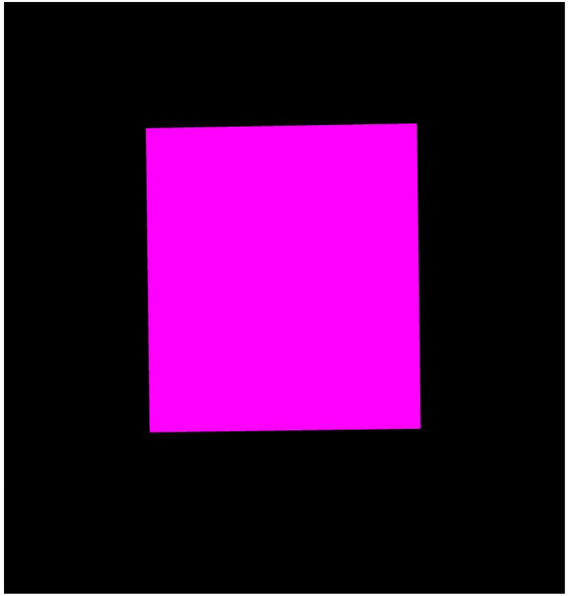
Complementary filter

1. Get calculation from gyro
2. Get calculation from accelerometer + magnetometer
3. Combine!

Code example

```
let timestep = ((cur_time - now).as_micros() as f32) / 1_000_000.0;
now = cur_time;

let time_constant = 0.75;
let a = time_constant / (time_constant + timestep);

yaw = a * (yaw + gyro.z().to_radians() * timestep) + (1.0 - a) * mag_yaw;
pitch = a * (pitch + gyro.x().to_radians() * timestep) + (1.0 - a) * accel_pitch;
roll = a * (roll + gyro.y().to_radians() * timestep) + (1.0 - a) * accel_roll;
```

Something isn’t right...

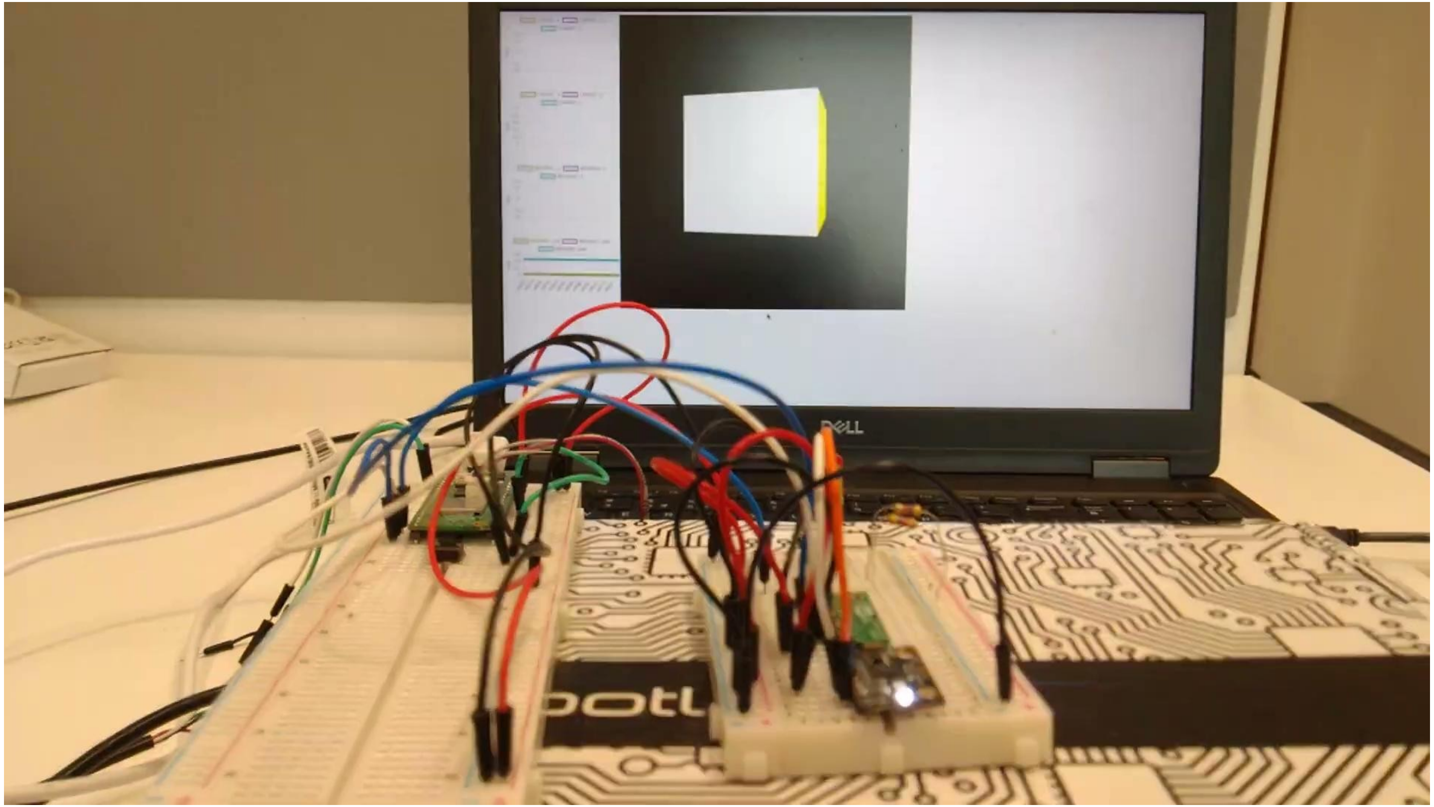


Tilt compensation

```
fn magnetic_yaw(mag_result: MagnetometerMeasurements, accel_pitch: f32, accel_roll: f32) -> f32 {
    let cx = mag_result.x * libm::cosf(accel_pitch)
        + mag_result.y * libm::sinf(accel_roll) * libm::sinf(accel_pitch)
        + mag_result.z * libm::cosf(accel_roll) * libm::sinf(accel_pitch);
    let cy = mag_result.y * libm::cosf(accel_roll) + mag_result.z * libm::sinf(accel_roll);
    libm::atan2f(-cy, cx)
}
```

Demo!

Demo



Is it end of a ride?

Or just a beginning?

Let’s explore more!

How far we are from real world application?

Very far…

What to learn next?

- Handle negatives angles
- Complementary filter, while it’s working OKish, I might want to check other filters (Kalman e.g.)
- Math - Could use some matrices instead of derived equations. Maybe go with nalgebra crate?
- Send binary data? Send stream in chunks (lower resolution in presenting data)?
- Replace RP2040 with something with FPU?
- Improve connections between components (design own hat for pico?)

Questions

Ok(())