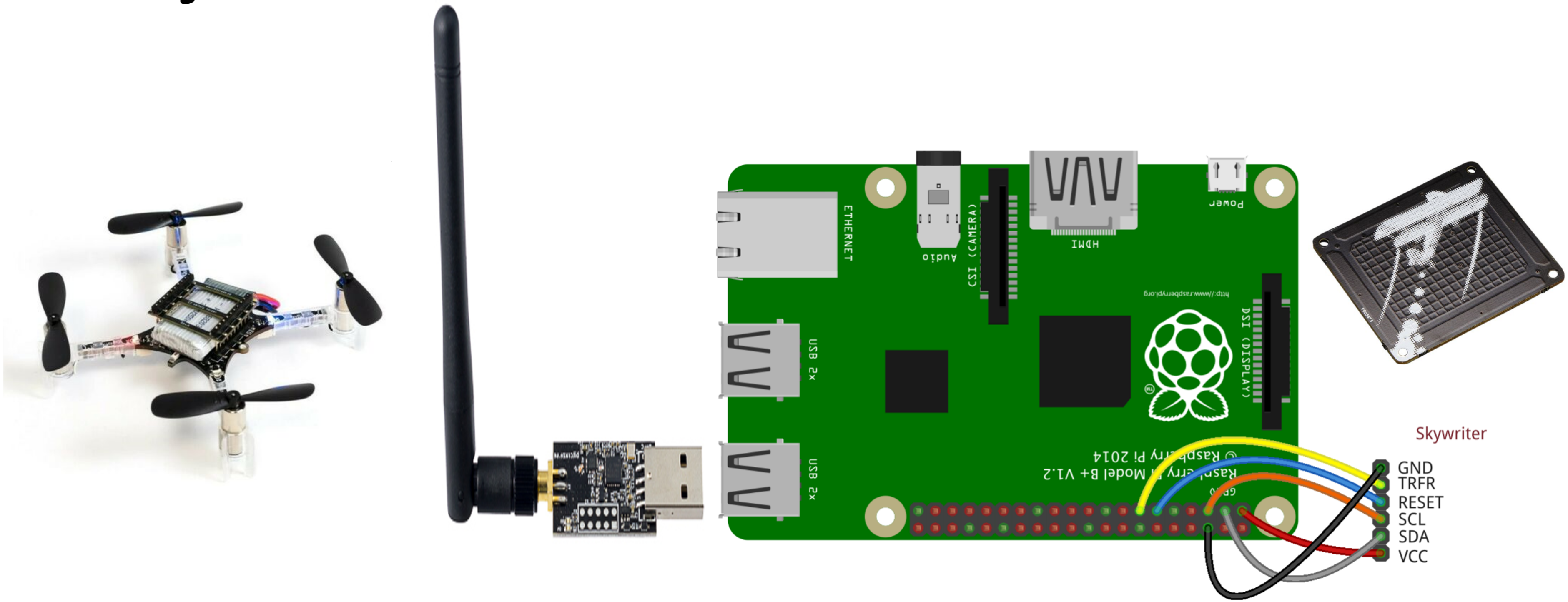
Gesture Controlled Drone

Ian Walter
Monette Khadr

Project Scope



https://www.bitcraze.io/crazyflie-2-1/

https://magpi.raspberrypi.org/articles/skywriter

System Overview

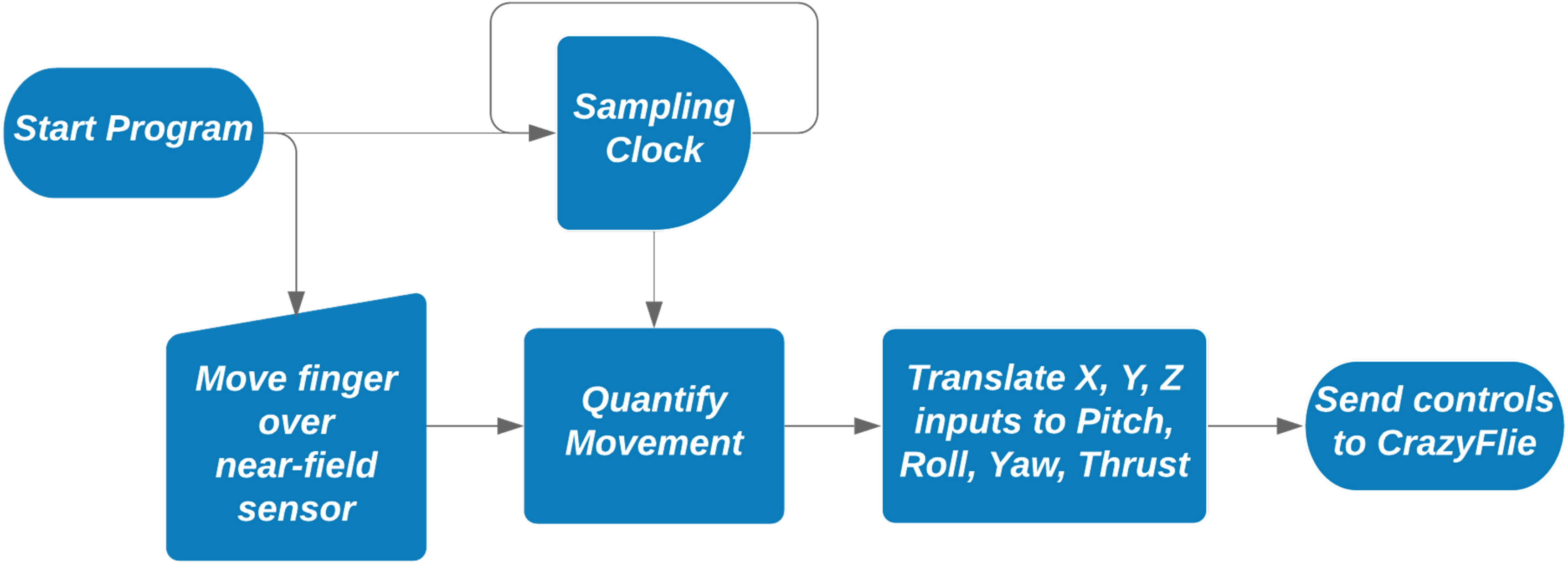


Image is original work

Skywriter HAT Sensor

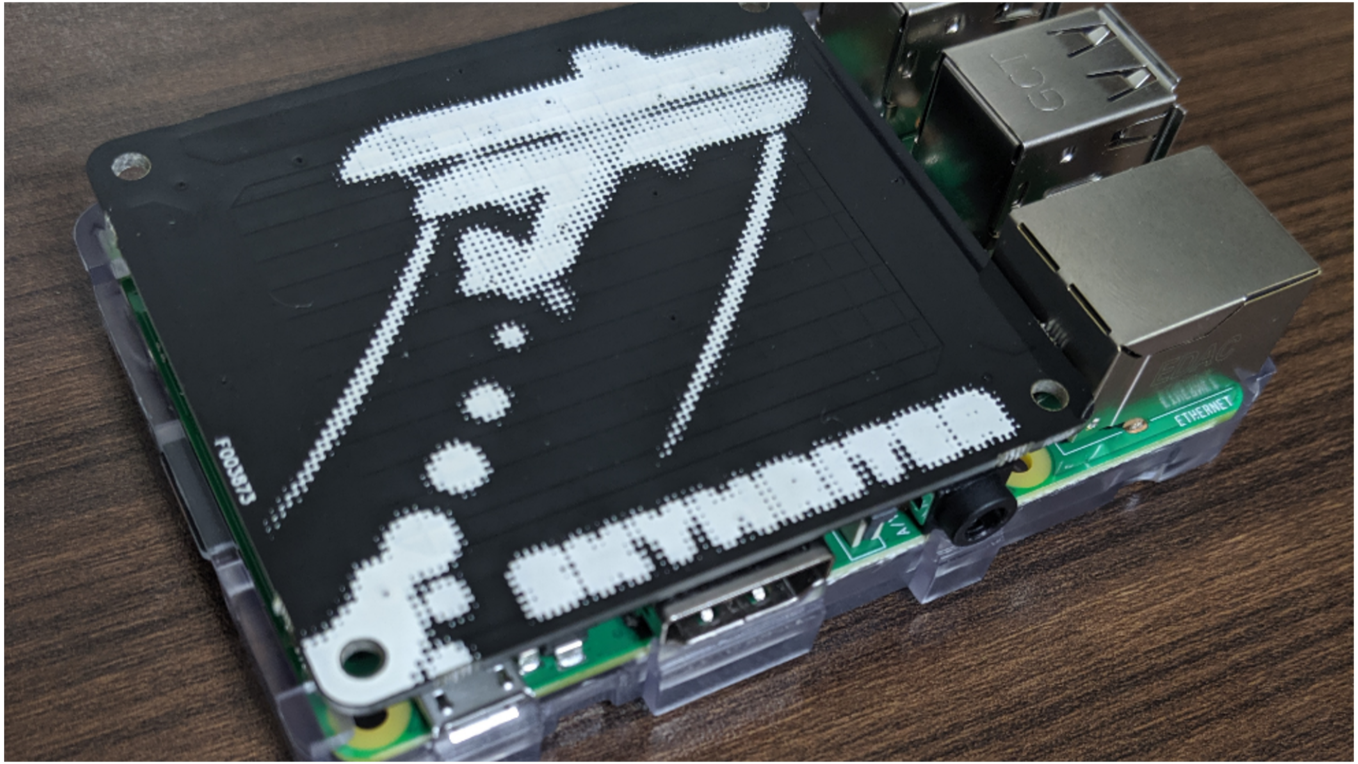
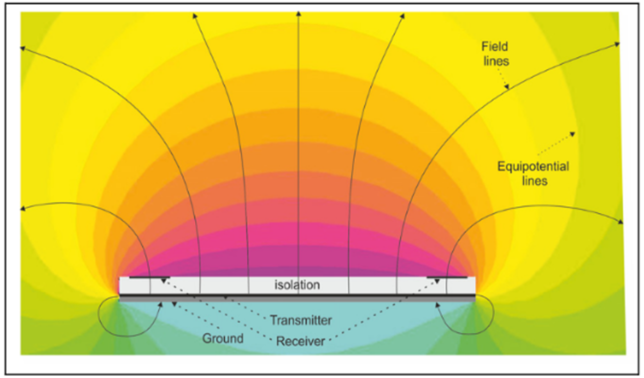


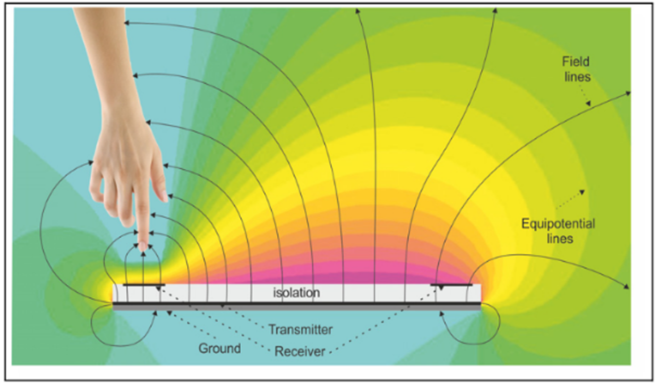
Image is original work

Skywriter HAT - How it works

Skywriter constantly emits an electric field

When conductive objects (like a finger) disturb this field, electrodes measure this disturbance

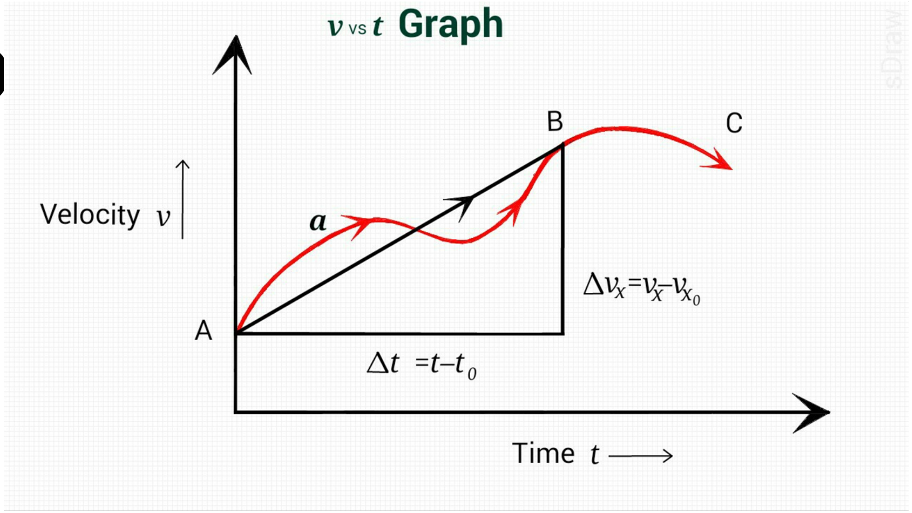




Images from Skywriter HAT Datasheet

Skywriter HAT - Mo

- Detects 200 positions per second
- 150 dots per inch
- Measured range of 0-2 cm



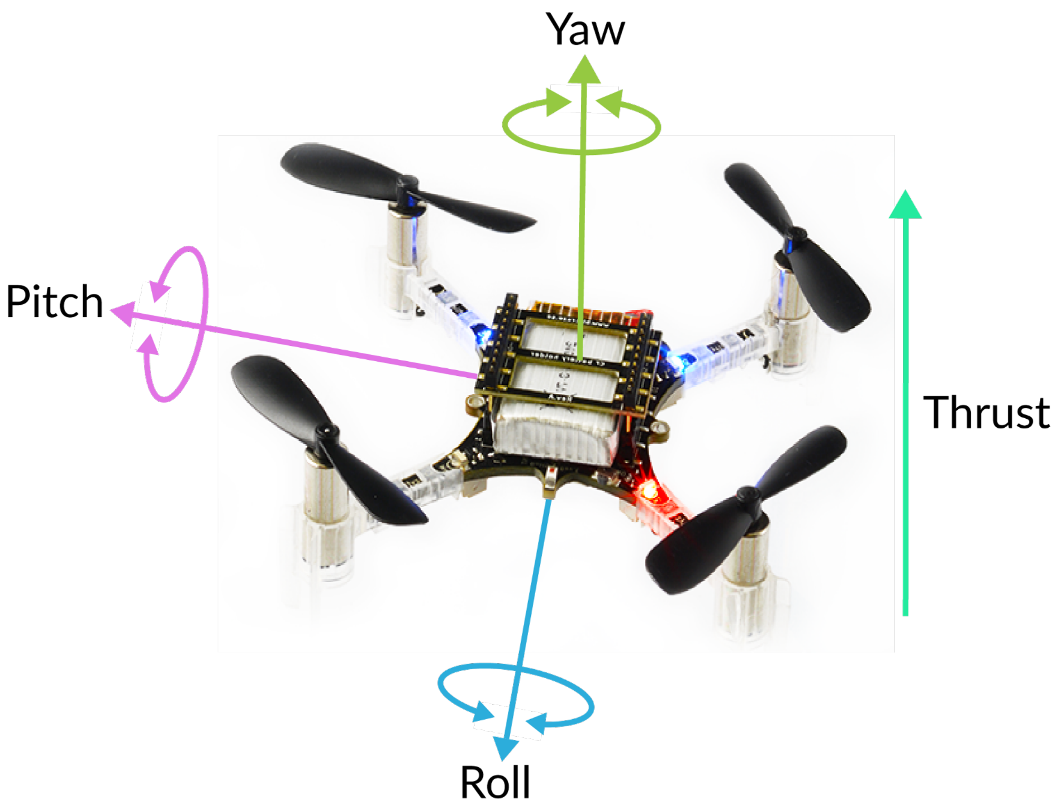
$$v_x = \frac{\Delta d_x}{t} \qquad v_y = \frac{\Delta d_y}{t} \qquad v_z = \frac{\Delta d_z}{t}$$

Drone - Crazyflie 2.0 Structure

Quadcopter, also known as quadrotor, is a helicopter with four rotors.

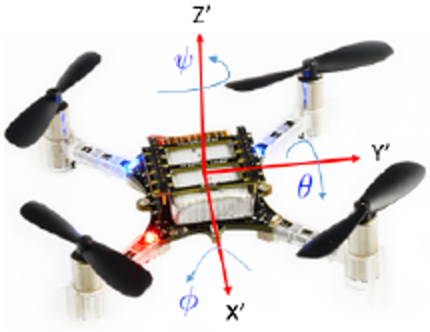
The rotors are directed upwards and they are placed in a square formation with equal distance from the center of mass of the quadcopter.

The quadcopter is controlled by adjusting the angular velocities of the rotors which are spun by electric motors.



https://www.bitcraze.io/crazyflie-2-1/

Drone - Mathematical Model



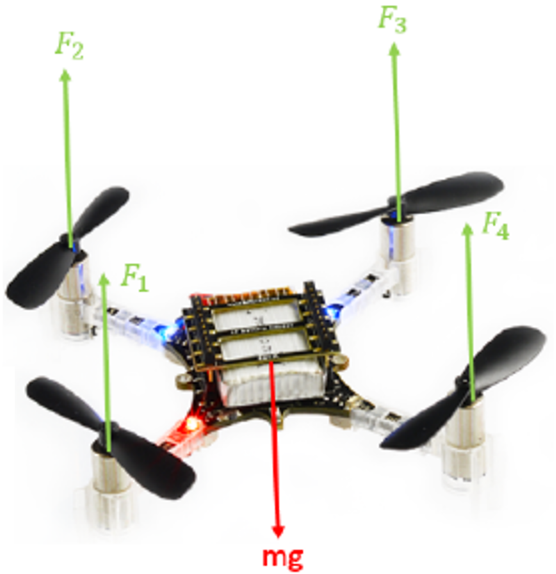
<u>Assumptions</u>

The crazyflie, our quadcopter, can be modelled as a rigid body, thus the well-known dynamic equations can be used.

It is symmetrical in its geometry, mass and propulsion system.

Its mass is fixed.

$$f = \sum_{i=1}^{4} f_i \xrightarrow{\text{Along z}} \mathbf{F}_b = \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix}$$



Drone - Mathematical Model

Newton-Euler equations:

$$\begin{bmatrix} \mathbf{F} \\ \tau \end{bmatrix} = \begin{bmatrix} m\mathbb{1}_3 & \mathbb{0}_3 \\ \mathbb{0}_3 & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \alpha \end{bmatrix} + \begin{bmatrix} \omega \times m\mathbf{v} \\ \omega \times \mathbf{I}_3 \omega \end{bmatrix}$$

total force

mass

linear acceleration

linear velocity

total torque

moment of inertia

angular velocity

angular acceleration

D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. Intl. J. Robot. Research, 31(5):664–674, Apr. 2012.

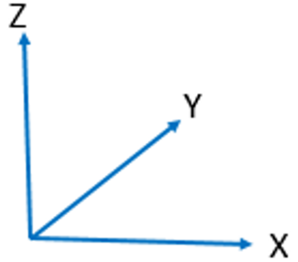
Drone - Mathematical Model

$$\boldsymbol{R} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix}$$

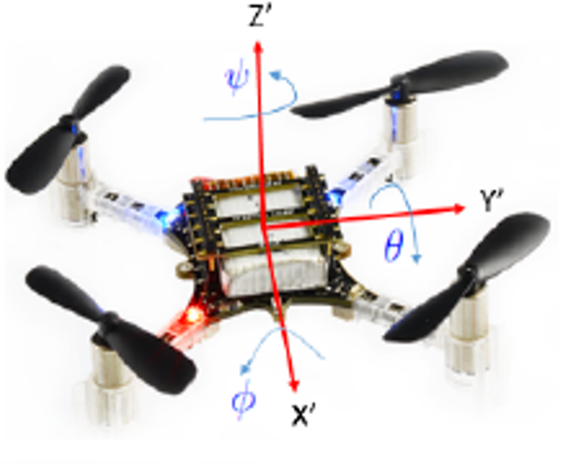
$$\mathbf{F}_e = R\mathbf{F} - m\mathbf{g}$$

Parameter	Description	Value
m_{quad}	Mass of the quadcopter alone	0.27 [Kg]
d	Arm length	39.73×10^{-3} [m]
r	Rotor radius	23.1348×10^{-3} [m]
I_{xx}	Principal Moment of Inertia around x axis	1.395×10^{-5} [Kg $\times$ m^2]
I_{yy}	Principal Moment of Inertia around y axis	1.436×10^{-5} [Kg $\times$ m^2]
I_{zz}	Principal Moment of Inertia around z axis	2.173×10^{-5} [Kg $\times$ m^2]
k_T	Non-dimensional thrust coefficient	0.2025
k_D	Non-dimensional torque coefficient	0.11

Luis, C., & Le Ny, J. (August, 2016). *Design of a Trajectory Tracking Controller for a Nanoquadcopter*. Technical report, Mobile Robotics and Autonomous Systems Laboratory, Polytechnique Montreal.



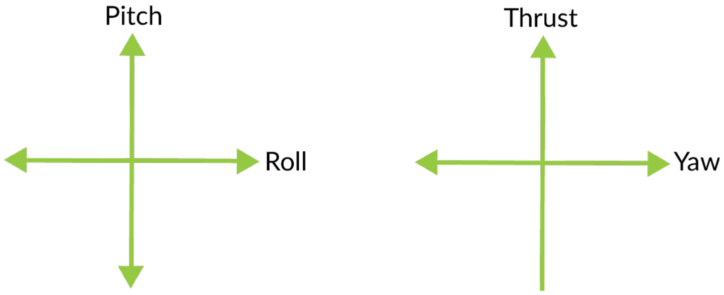
Inertial Frame



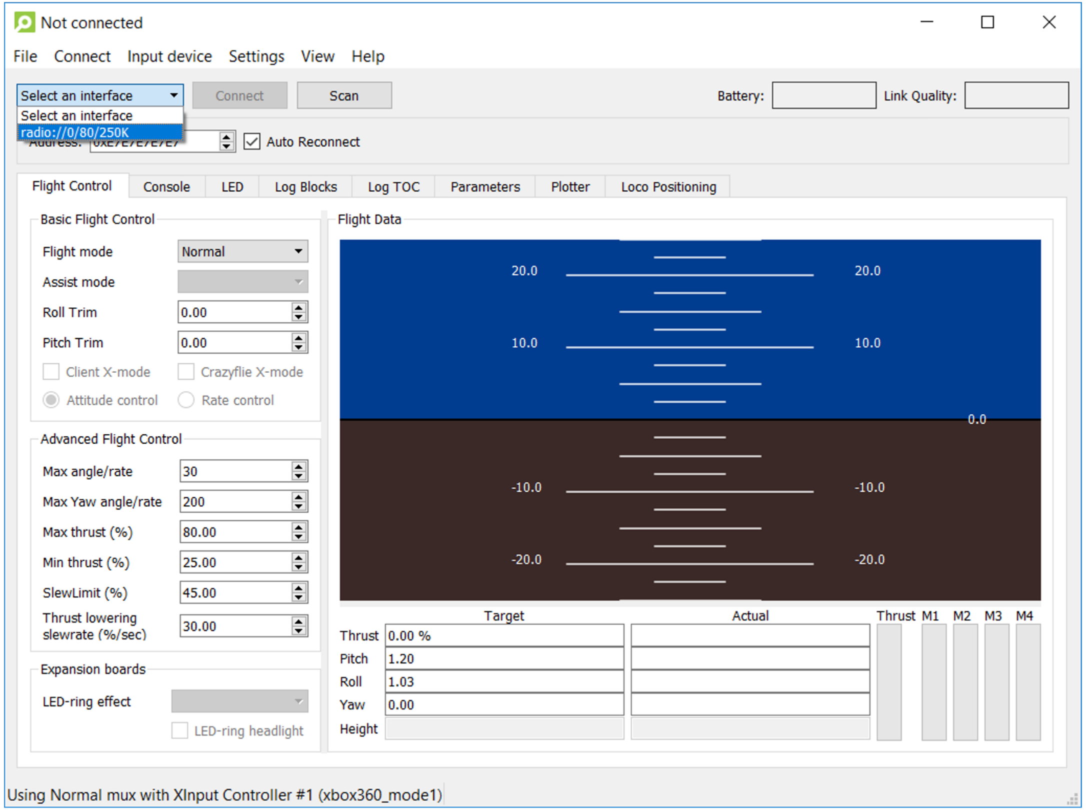
Body Frame

Resources

Mobile app



Crazyflie client



https://www.bitcraze.io/getting-started-with-the-crazyflie-2-0/

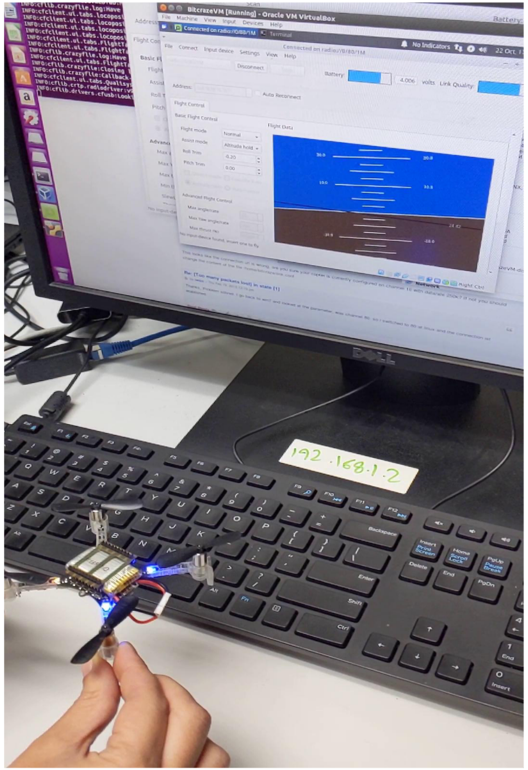
Utilized Function

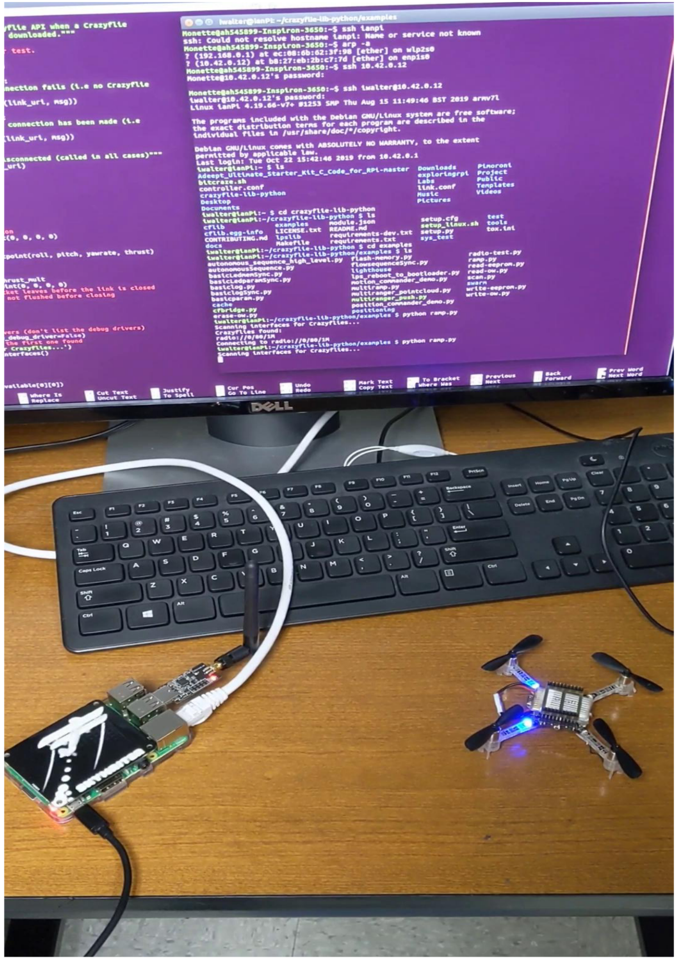
```
def send_setpoint(self, roll, pitch, yaw, thrust):
    """
    Send a new control setpoint for roll/pitch/yaw/thrust to the copter

    The arguments roll/pitch/yaw/trust is the new setpoints that should
    be sent to the copter
    """
    if thrust > 0xFFFF or thrust < 0:
        raise ValueError('Thrust must be between 0 and 0xFFFF')

    if self._x_mode:
        roll, pitch = 0.707 * (roll - pitch), 0.707 * (roll + pitch)

    pk = CRTPPacket()
    pk.port = CRTPPort.COMMANDER
    pk.data = struct.pack('<fffH', roll, -pitch, yaw, thrust)
    self._cf.send_packet(pk)
```

Alternative Function

```
def send_position_setpoint(self, x, y, z, yaw):
    """
    Control mode where the position is sent as absolute x,y,z coordinate in
    meter and the yaw is the absolute orientation.

    x and y are in m
    yaw is in degrees
    """
    pk = CRTPPacket()
    pk.port = CRTPPort.COMMANDER_GENERIC
    pk.data = struct.pack('<Bffff', TYPE_POSITION,
                          x, y, z, yaw)
    self._cf.send_packet(pk)
```

Results to date





Tasks - midpoint

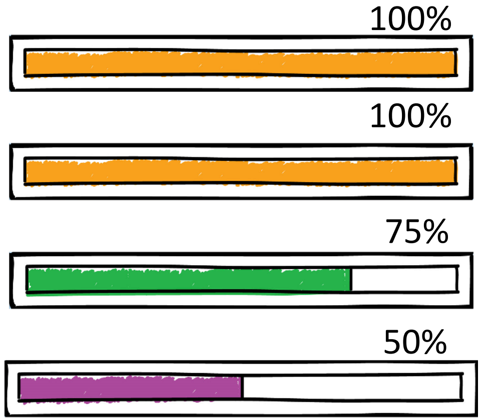
Characterize the sensor

Model the drone

Investigate the drone's API

Translate sensor readings

Fine tune drone flight



Tasks - now

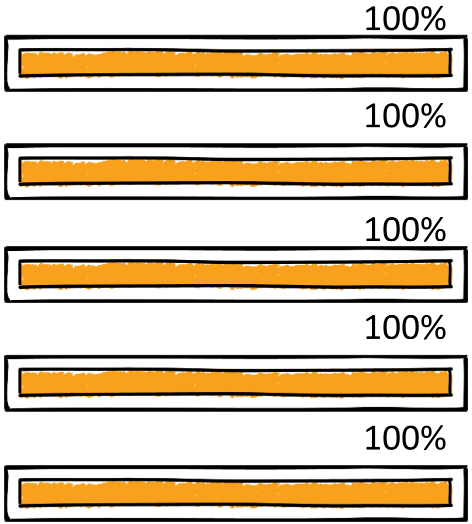
Characterize the sensor

Model the drone

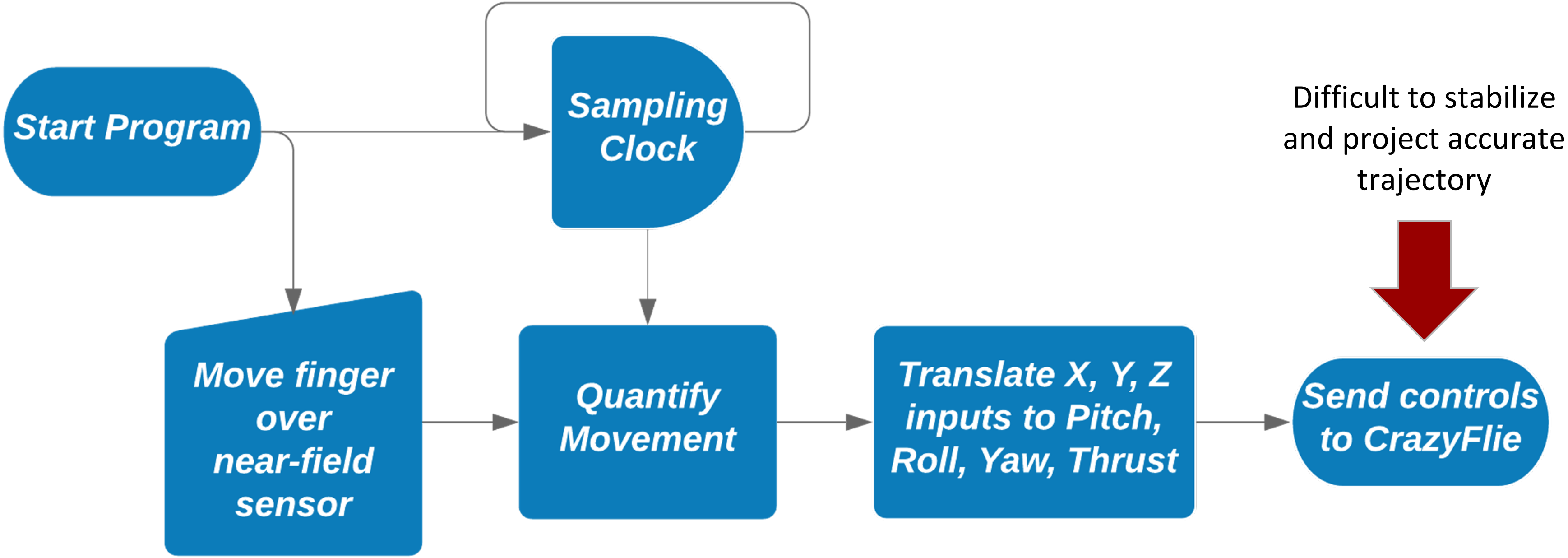
Investigate the drone’s API

Translate sensor readings

Fine tune drone flight



System Overview - Open Loop



Difficult to stabilize and project accurate trajectory

Image is original work

System Overview - Closed Loop

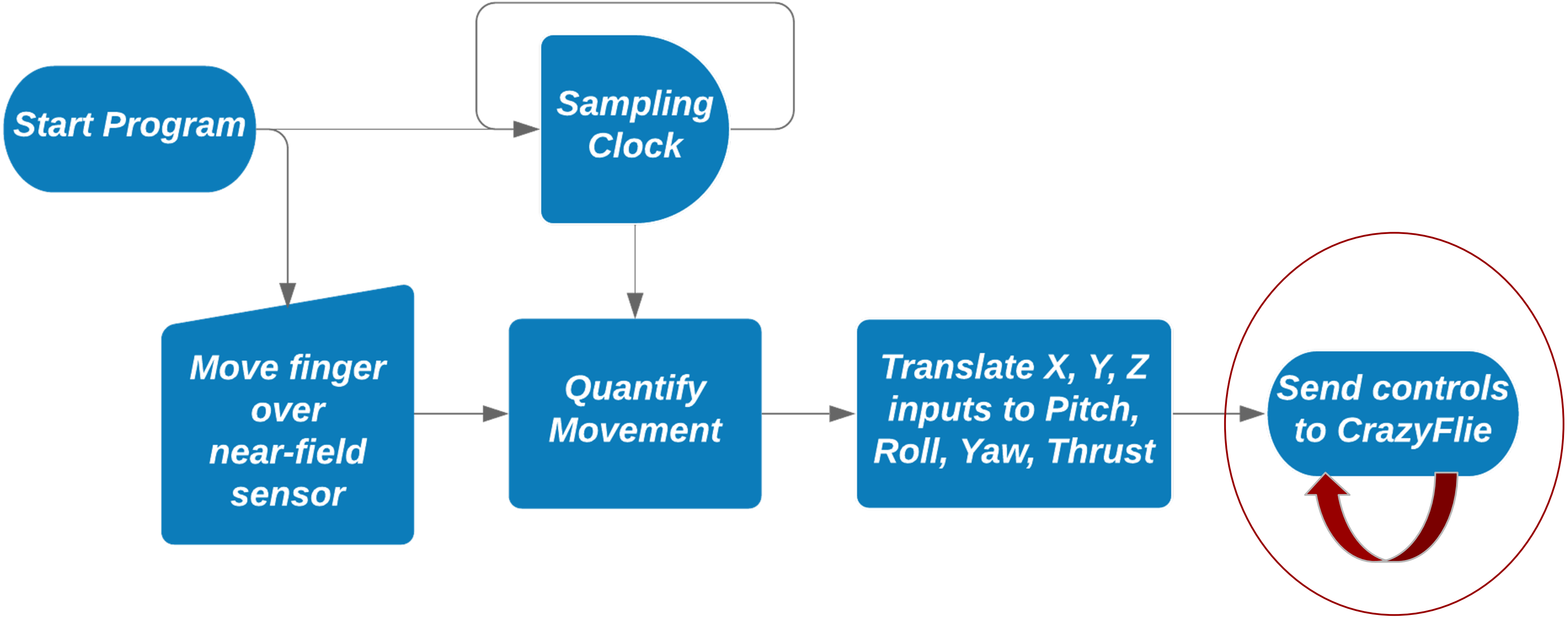


Image is original work

Calibration

1) Adjust the propellers

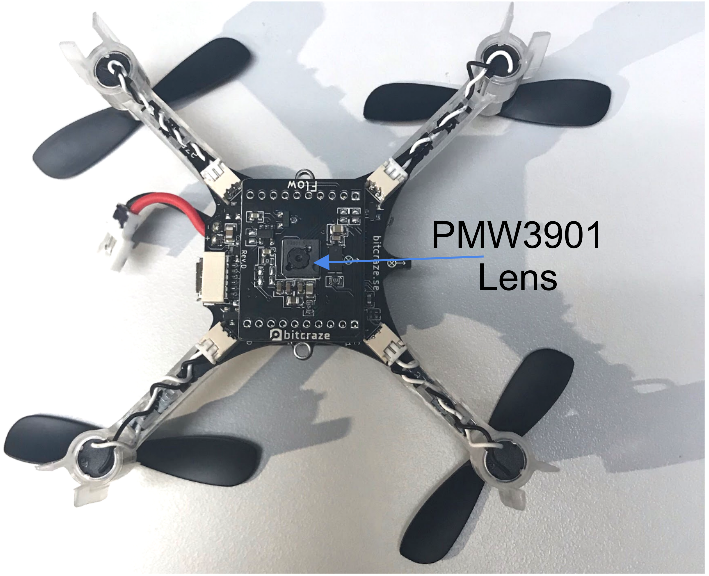
   Well balanced propellers reduce the vibrations in the the Crazyflie

1) Use pre-mounted sensor
   a) 3 axis gyro (MPU-9250)
   b) 3 axis accelerometer (MPU-9250)
   c) 3 axis magnetometer (MPU-9250)
   d) high precision pressure sensor (LPS25H)

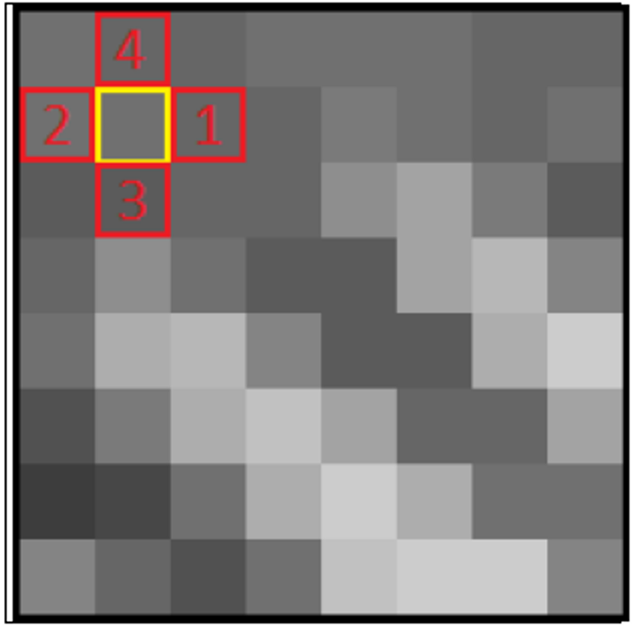
1) Install additional sensors
   a) VLS3L0x
   b) PMW3901

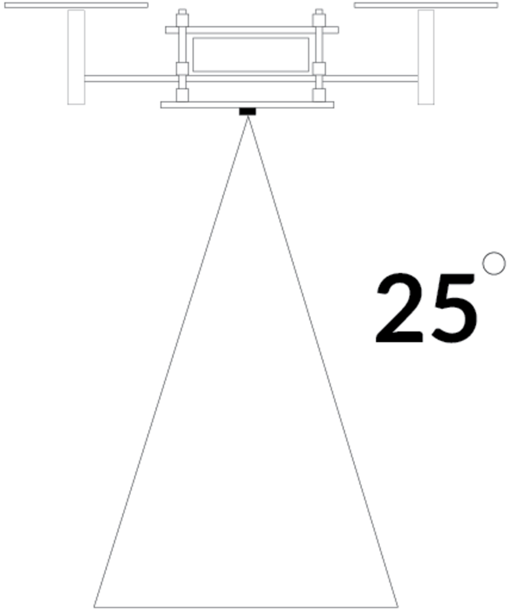
Inner State Estimation

- In order to allow stable drone flight, we added an expansion board that contains two sensors, the VL53L0x time-of-flight (ToF) sensor and an optical flow sensor PMW3901. The ToF sensor is a laser ranging sensor that measures the distance of the drone from the ground. The optical flow sensor uses a low-resolution camera to measure movements in the x and y coordinates relative to the ground.
- The PMWB901 requires SPI interface, while the VL53L0x requires I2C connectivity, the PCB board handles the connectivity constraints and allows direct communication with the drone.



Optical Flow Sensor





Gageik, Nils & Strohmeier, Michael & Montenegro, Sergio. (2013). An Autonomous UAV with an Optical Flow Sensor for Positioning and Navigation. International Journal of Advanced Robotic Systems. 10. 1. 10.5772/56813.

Why use the Kalman Filter?

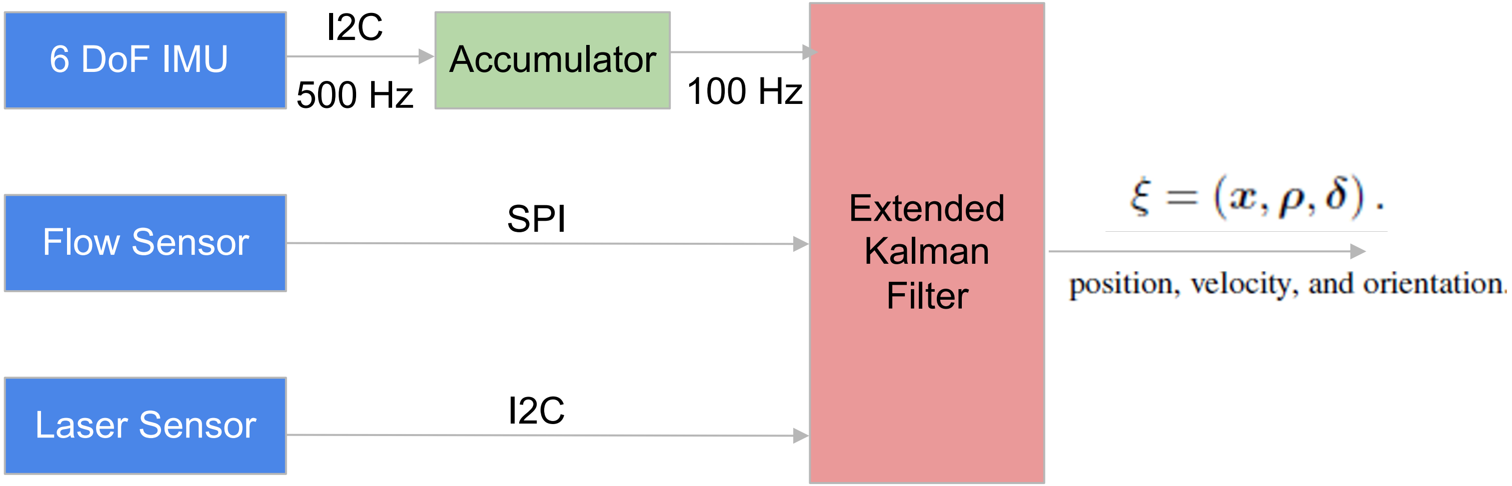
The force generated by a propeller translating with respect to the free stream will typically be significantly different from the static thrust force *f*. This deviation is given by *fa*, which is taken to be a function of the quadrocopter's relative airspeed.

$$m\ddot{x} = \mathbf{R}\ (fe_3 + f_a) + mg$$

$$z_{\mathrm{acc}} = \mathbf{R}^{-1}\left(\ddot{x} - g\right) + \eta_{\mathrm{acc}} = \frac{1}{m}\left(e_3 f + f_a\right) + \eta_{\mathrm{acc}}.$$

$$z_{\mathrm{gyro}} = \omega + \eta_{\mathrm{gyro}}$$

Fusing Sensor Readings - EKF



Kalman's Prediction

$$\begin{aligned}
\dot{\hat{x}} &= \hat{\mathrm{R}}_{\text{ref}} \left(\mathbf{I} + [\![\hat{\delta} \times]\!] \right) \hat{\rho} \\
\dot{\hat{\rho}} &= \frac{1}{m} f e_3 + \left(\frac{1}{m} \mathbf{K}_{\text{aero}} \dot{\theta}_{\Sigma} - [\![\hat{\omega} \times]\!] \right) \hat{\rho} \\
&\quad - \|g\| \left(\mathbf{I} - [\![\hat{\delta} \times]\!] \right) \hat{\mathrm{R}}_{\text{ref}}^{-1} e_3 \\
\dot{\hat{\delta}} &= \hat{\omega}
\end{aligned}$$

Kalman Filter Implementation - Crazyflie Firmware

```
 * Primary Kalman filter functions
 *
 * The filter progresses as:
 *  - Predicting the current state forward */
static void stateEstimatorPredict(float thrust, Axis3f *acc, Axis3f *gyro, float dt);
static void stateEstimatorAddProcessNoise(float dt);

/*  - Measurement updates based on sensors */
static void stateEstimatorScalarUpdate(arm_matrix_instance_f32 *Hm, float error, float stdMeasNoise);
static void stateEstimatorUpdateWithAccOnGround(Axis3f *acc);
#ifdef KALMAN_USE_BARO_UPDATE
static void stateEstimatorUpdateWithBaro(baro_t *baro);
#endif

/*  - Finalization to incorporate attitude error into body attitude */
static void stateEstimatorFinalize(sensorData_t *sensors, uint32_t tick);

/*  - Externalization to move the filter's internal state into the external state expected by other modules */
static void stateEstimatorExternalizeState(state_t *state, sensorData_t *sensors, uint32_t tick);
```

https://github.com/bitcraze/crazyflie-firmware/blob/6308ff47ff4d4691f9b7f6f991564244c76d7910/src/modules/src/estimator_kalman.c#L1034-L1098

Flight Dem

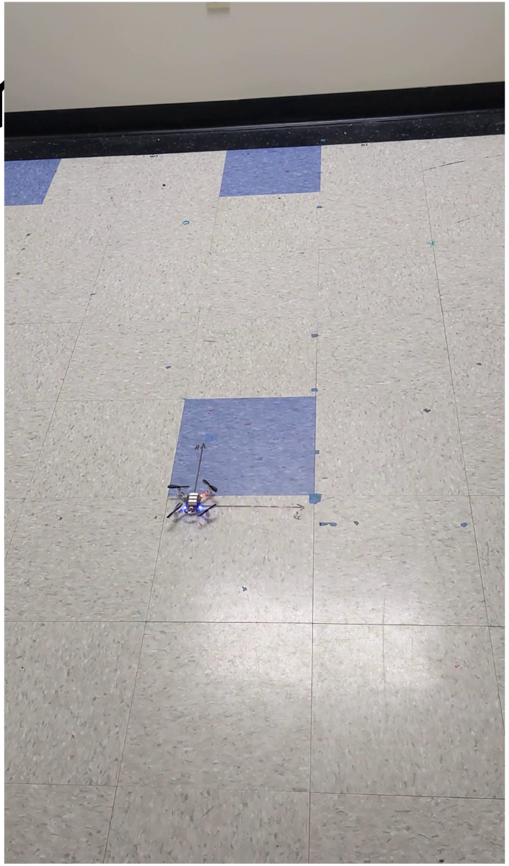
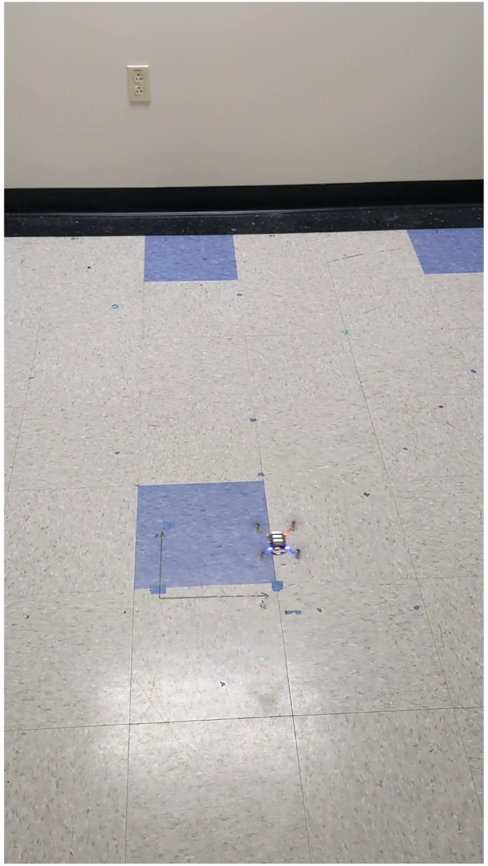


Figure 8



Precoded Sequence

Sensor-Drone integration

- Large issue with combining the dependency packages (sensor requires root, drone does not)
- Specified a fixed velocity at which the drone flies (this was a design choice)
- Code

Analysis

Issue	"Why"	Effects
Extremely noisy measurements	No re-calibration of sensor	Very inaccurate and inconsistent measurements
Static or slow movement causes large shifts in measured value	Sensor does adapt to sustained distortions in EM field	Very difficult to do slow or precise movements

These can be seen in the flight demonstrations!

Solutions + Conclusion

- Primary issues are due to the near-field sensor being relatively cheap ($20)
- Able to compensate for some issues (noisy measurements) by reducing spatial resolution through averaging
- A more suitable (cheap) sensor would have been an optical flow sensor (fairly robust for $40)
- More numerical analysis on drone positioning could be done utilizing something like optitrack system

Flight demonstrations (with sensor)