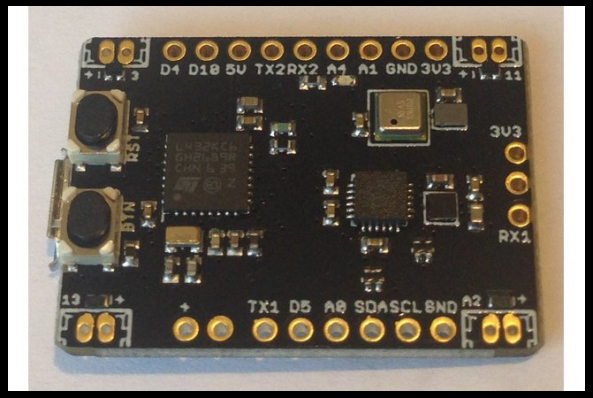
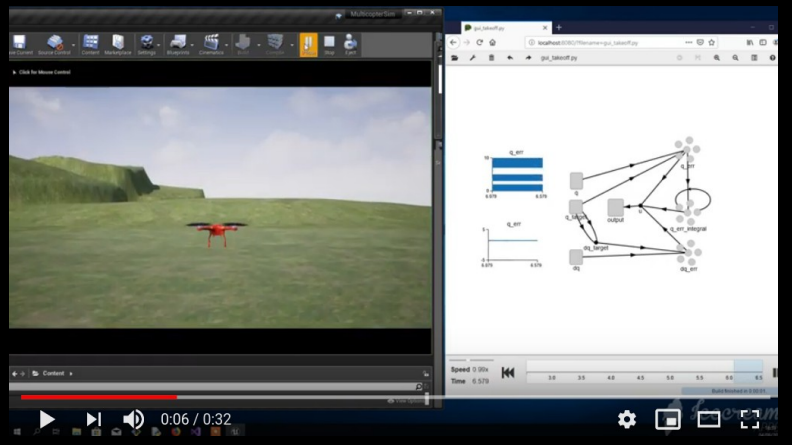
Robustness Through Simplicity: A Minimalist Gateway to Neurorobotic Flight

Simon D. Levy
Washington & Lee University

Robust Artificial Intelligence for Neurorobotics
28 August 2019

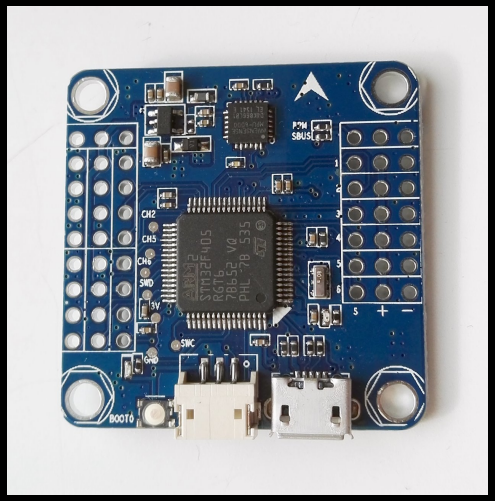




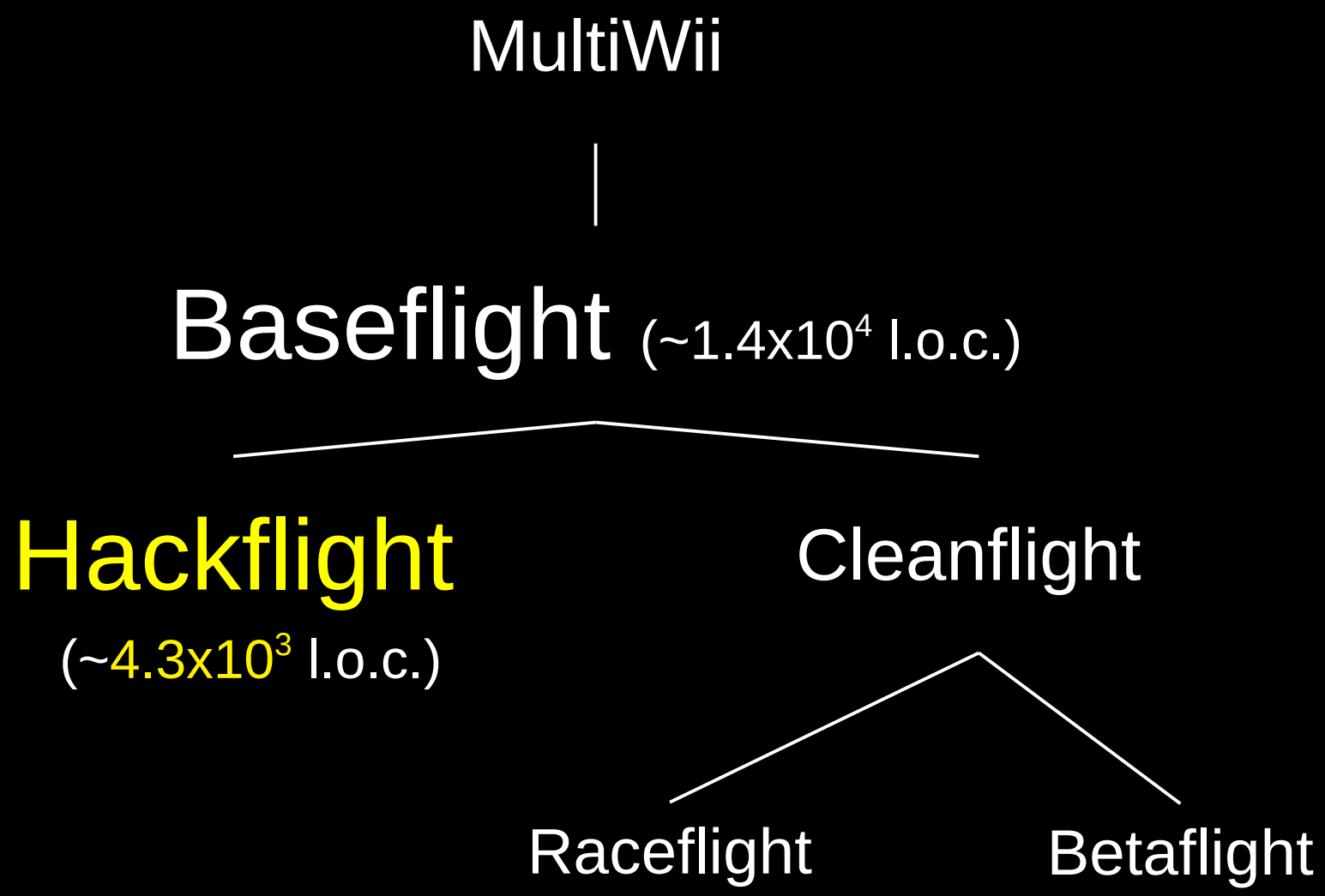
Part I: Hardware

Part I: ~~Hardware~~ Firmware

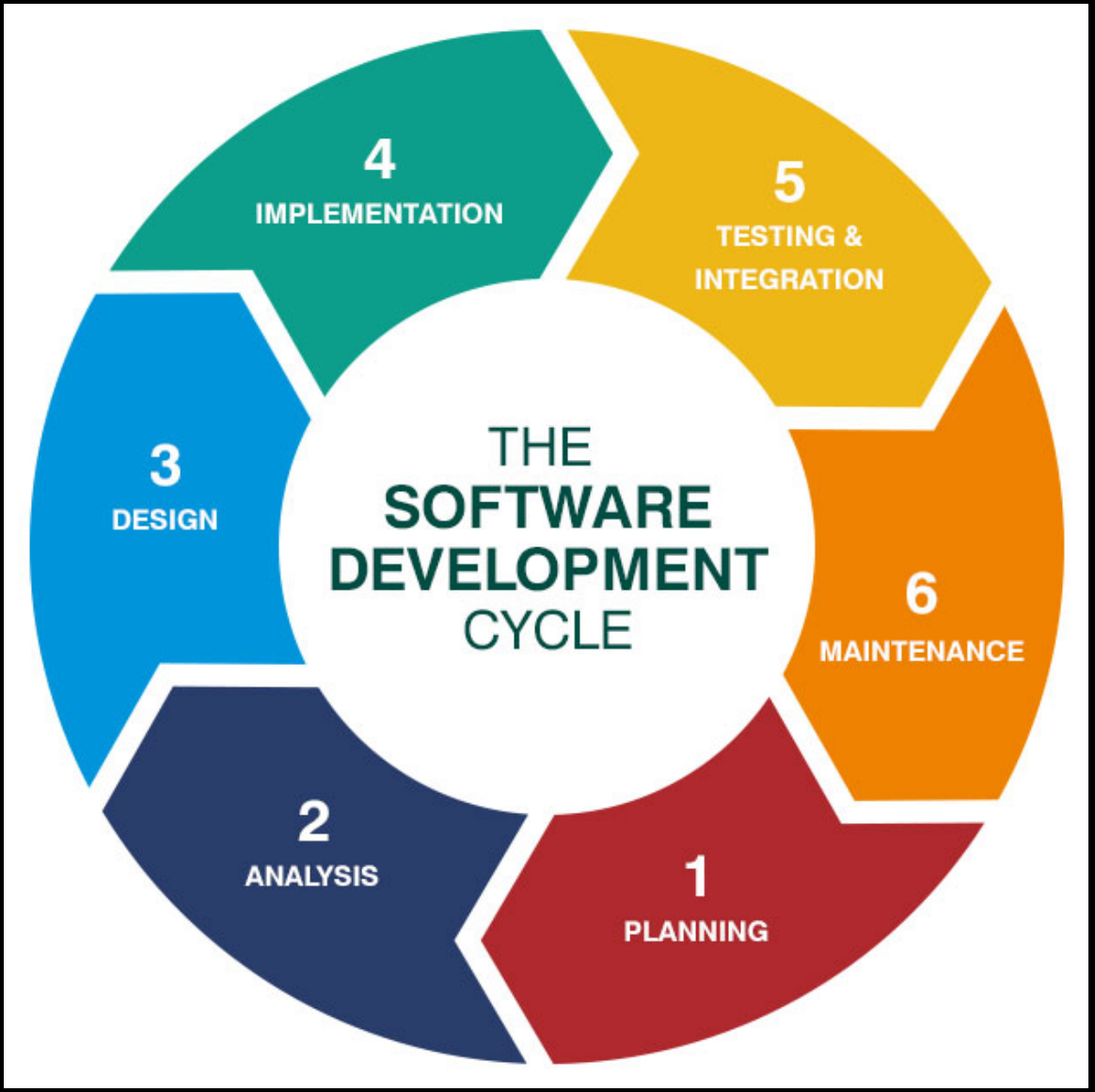
- ArduPilot: ~2.3×10^6 lines of code
- Cleanflight: ~8.5×10^5 lines of code



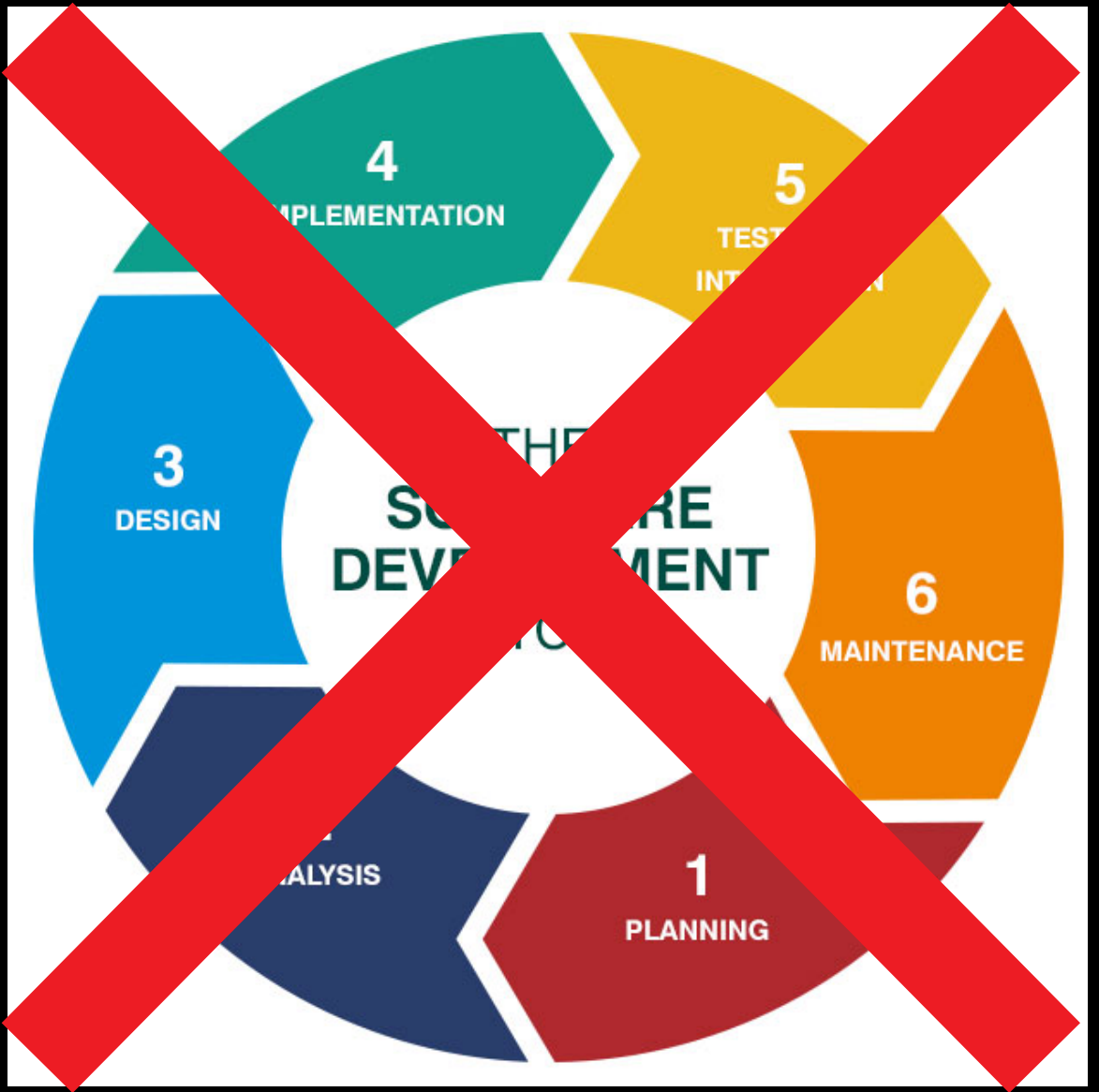
Firmware Genealogy

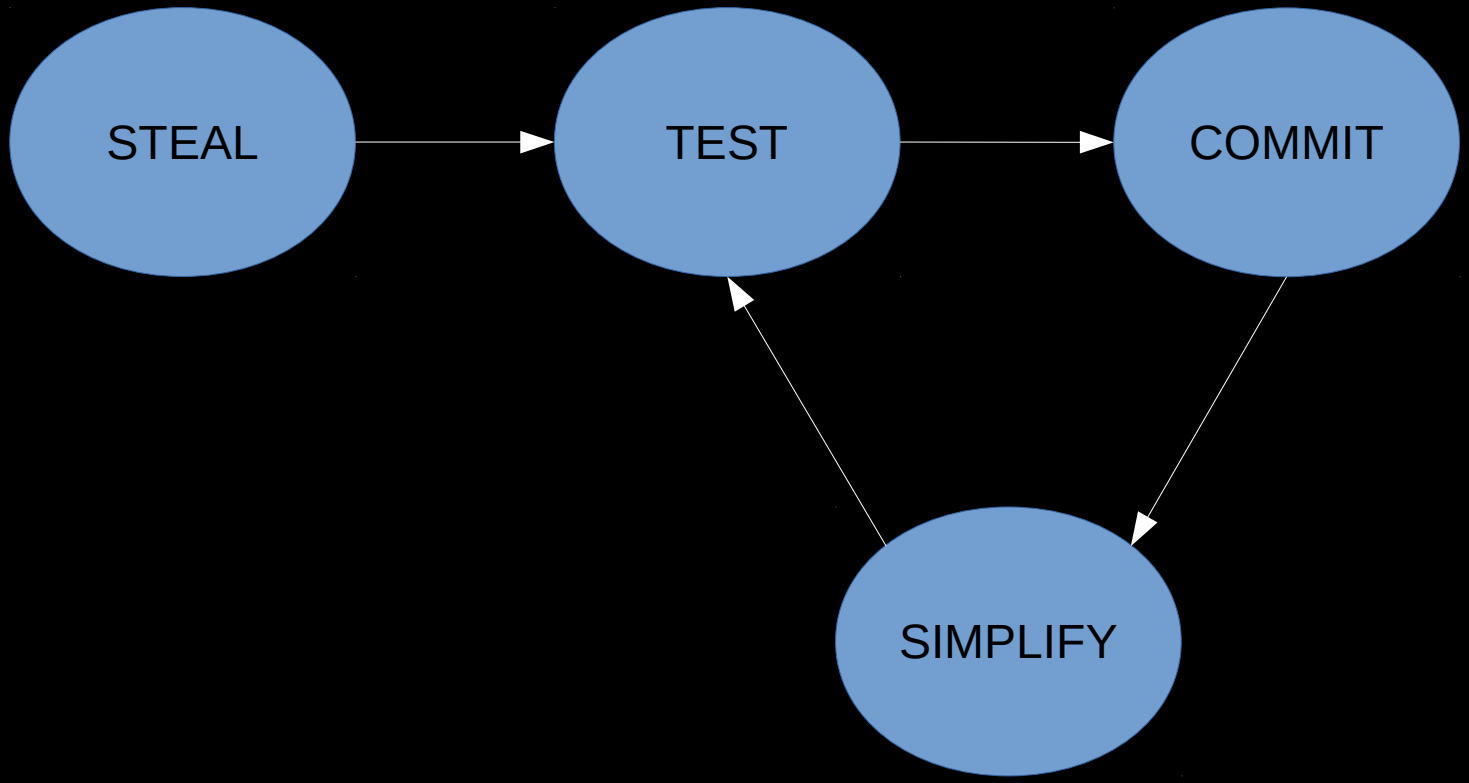


Principle #1:
It Always Has to Fly



https://online.husson.edu/software-development-cycle/


4
5
3
DESIGN
6
MAINTENANCE
1
PLANNING


STEAL
TEST
COMMIT
SIMPLIFY

Principle #2:
Bottom-Up / OOP Design

```
hf::Hackflight h;

hf::DSMX_Receiver rc = hf::DSMX_Receiver(CHANNEL_MAP);

hf::MixerQuadX mixer;

hf::Rate ratePid = hf::Rate(
        0.225f,     // Gyro pitch/roll P
        0.001875f,  // Gyro pitch/roll I
        0.375f,     // Gyro pitch/roll D
        1.0625f,    // Gyro yaw P
        0.005625f); // Gyro yaw I

hf::Level level = hf::Level(0.20f);

void setup(void)
{
    // Add Level PID for aux switch position 1
    h.addPidController(&level, 1);

    // Initialize Hackflight firmware
    h.init(new hf::Ladybug(), &rc, &mixer, &ratePid);
}

void loop(void)
{
    h.update();
}
```

https://github.com/simondlevy/Hackflight/blob/master/examples/Ladybug/LadybugDSMX/LadybugDSMX.ino

```
bool mixerIsTricopter(void)
{
#ifdef USE_SERVOS
    return (currentMixerMode == MIXER_TRI || currentMixerMode == MIXER_CUSTOM_TRI);
#else
    return false;
#endif
}

bool mixerIsOutputSaturated(int axis, float errorRate)
{
#ifdef USE_SERVOS
    if (axis == FD_YAW && mixerIsTricopter()) {
        return mixerTricopterIsServoSaturated(errorRate);
    }
#else
    UNUSED(axis);
    UNUSED(errorRate);
#endif

    return motorMixRange >= 1.0f;
}

// All PWM motor scaling is done to standard PWM range of 1000-2000 for easier tick conversion with legacy code / configurator
// DSHOT scaling is done to the actual dshot range
void initEscEndpoints(void)
{
    // Can't use 'isMotorProtocolDshot()' here since motors haven't been initialised yet
    switch (motorConfig()->dev.motorPwmProtocol) {
#ifdef USE_DSHOT
    case PWM_TYPE_PROSHOT1000:
    case PWM_TYPE_DSHOT1200:
    case PWM_TYPE_DSHOT600:
    case PWM_TYPE_DSHOT300:
    case PWM_TYPE_DSHOT150:
        disarmMotorOutput = DSHOT_DISARM_COMMAND;
        if (feature(FEATURE_3D)) {
            motorOutputLow = DSHOT_MIN_THROTTLE + ((DSHOT_3D_DEADBAND_LOW - DSHOT_MIN_THROTTLE) / 100.0f) * CONVERT_PARAMETER_TO
        } else {
            motorOutputLow = DSHOT_MIN_THROTTLE + ((DSHOT_MAX_THROTTLE - DSHOT_MIN_THROTTLE) / 100.0f) * CONVERT_PARAMETER_TO_PE
        }
        motorOutputHigh = DSHOT_MAX_THROTTLE;
        deadbandMotor3dHigh = DSHOT_3D_DEADBAND_HIGH + ((DSHOT_MAX_THROTTLE - DSHOT_3D_DEADBAND_HIGH) / 100.0f) * CONVERT_PARAME
        deadbandMotor3dLow = DSHOT_3D_DEADBAND_LOW;

        break;
#endif
```

https://github.com/cleanflight/cleanflight/blob/master/src/main/flight/mixer.c

Principle #3:
Separate the Drivers from the Algorithms

```
//---------------------------------- Core functionality ----------------------------------------------
virtual bool  getQuaternion(float quat[4]) = 0;
virtual bool  getGyrometer(float gyroRates[3]) = 0;
virtual void  writeMotor(uint8_t index, float value) = 0;
virtual float getTime(void) = 0;

//------------------------- Support for additional surface-mount sensors -------------------------------
virtual bool  getAccelerometer(float accelGs[3]) { (void)accelGs;  return false; }
virtual bool  getMagnetometer(float uTs[3]) { (void)uTs;  return false; }
virtual bool  getBarometer(float & pressure) { (void)pressure;  return false; }

//------------------------------- Serial communications via MSP ----------------------------------------
virtual uint8_t serialAvailableBytes(void) { return 0; }
virtual uint8_t serialReadByte(void)  { return 1; }
virtual void    serialWriteByte(uint8_t c) { (void)c; }

//------------------------------- Reboot for non-Arduino boards ----------------------------------------
virtual void reboot(void) { }

//---------------------------------------- Safety --------------------------------------------------------
virtual void showArmedStatus(bool armed) { (void)armed; }
virtual void flashLed(bool shouldflash) { (void)shouldflash; }
virtual bool isBatteryLow(void) { return false; }
```

https://github.com/simondlevy/Hackflight/blob/master/src/board.hpp

```c
static void scalarUpdate(kalmanCoreData_t* this, arm_matrix_instance_f32 *Hm, float error, float stdMeasNoise)
{
  // The Kalman gain as a column vector
  static float K[KC_STATE_DIM];
  static arm_matrix_instance_f32 Km = {KC_STATE_DIM, 1, (float *)K};

  // Temporary matrices for the covariance updates
  static float tmpNN1d[KC_STATE_DIM * KC_STATE_DIM];
  static arm_matrix_instance_f32 tmpNN1m = {KC_STATE_DIM, KC_STATE_DIM, tmpNN1d};

  static float tmpNN2d[KC_STATE_DIM * KC_STATE_DIM];
  static arm_matrix_instance_f32 tmpNN2m = {KC_STATE_DIM, KC_STATE_DIM, tmpNN2d};

  static float tmpNN3d[KC_STATE_DIM * KC_STATE_DIM];
  static arm_matrix_instance_f32 tmpNN3m = {KC_STATE_DIM, KC_STATE_DIM, tmpNN3d};

  static float HTd[KC_STATE_DIM * 1];
  static arm_matrix_instance_f32 HTm = {KC_STATE_DIM, 1, HTd};

  static float PHTd[KC_STATE_DIM * 1];
  static arm_matrix_instance_f32 PHTm = {KC_STATE_DIM, 1, PHTd};
```

https://github.com/bitcraze/crazyflie-firmware/blob/master/src/modules/src/kalman_core.c

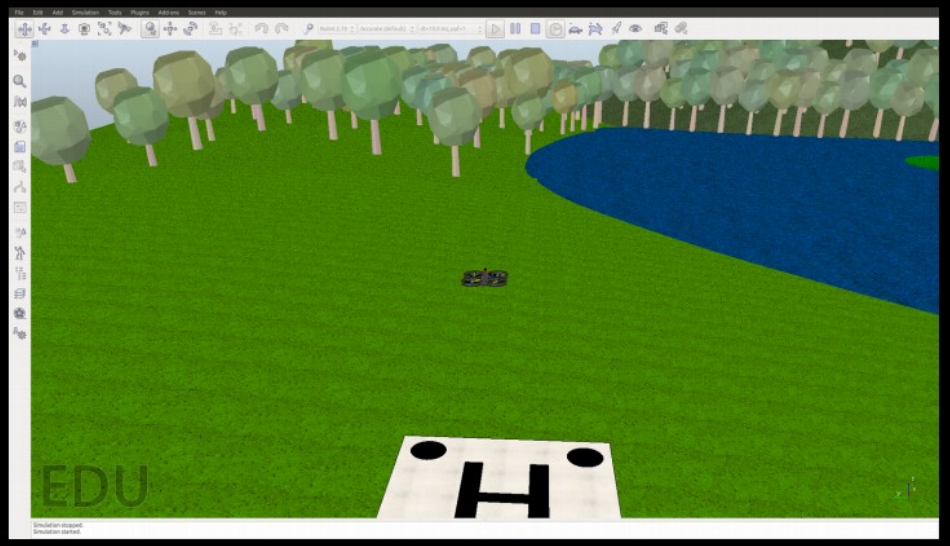
Part II: Simulation

```
//------------------------------------ Core functionality ----------------------------------------------
virtual bool  getQuaternion(float quat[4]) = 0;
virtual bool  getGyrometer(float gyroRates[3]) = 0;
virtual void  writeMotor(uint8_t index, float value) = 0;
virtual float getTime(void) = 0;
```

STM32 drivers

Firmware

V-REP Plugin



2016

Hackflight firmware on Windows #2

Open **sytelus** opened this issue on Mar 1, 2017 · 6 comments

sytelus commented on Mar 1, 2017 Collaborator

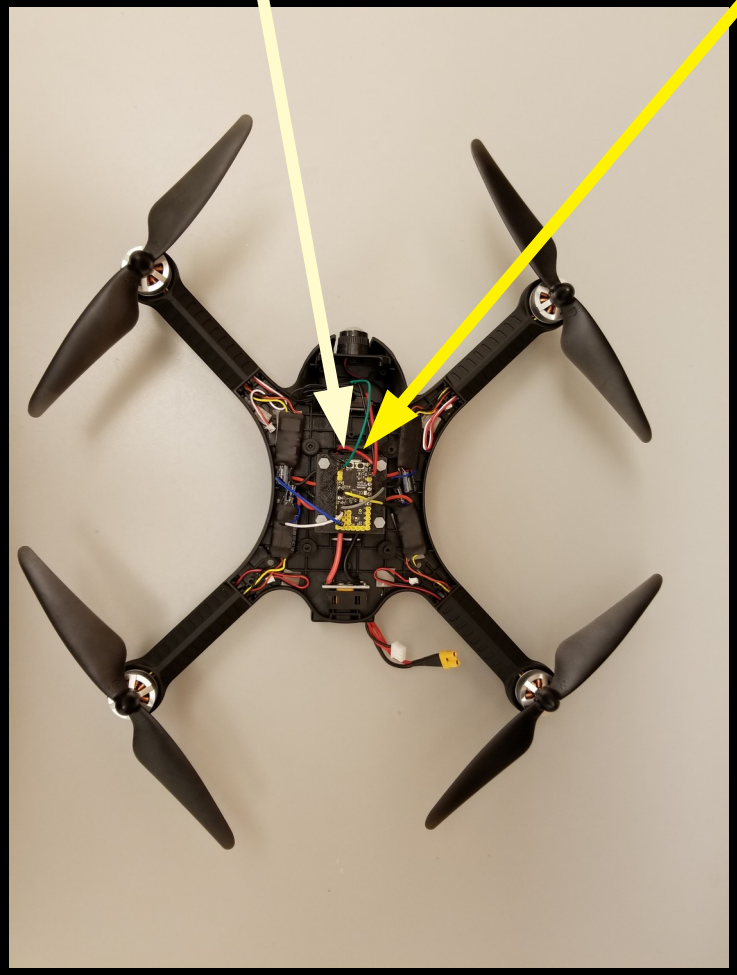
I'm looking for firmware code that I can put in AirSim. Basically, this firmware would a simple cross platform library that takes in sensor inputs from the simulator and outputs rotor controls.

Is this possible to do with hackflight firmware?

Arduino drivers

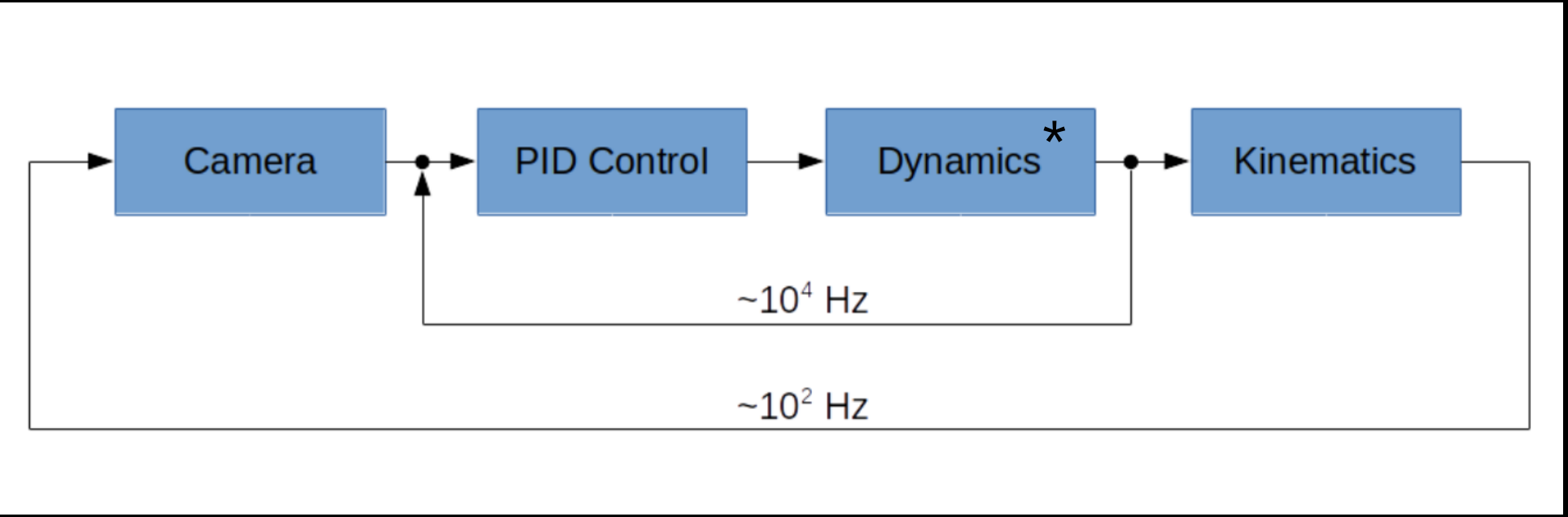
Firmware

UnrealEngine4



2018

MulticopterSim



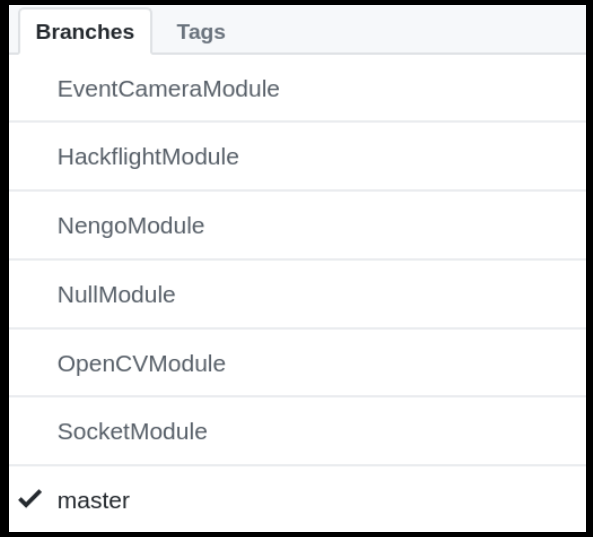
~1.5x10^3 l.o.c.

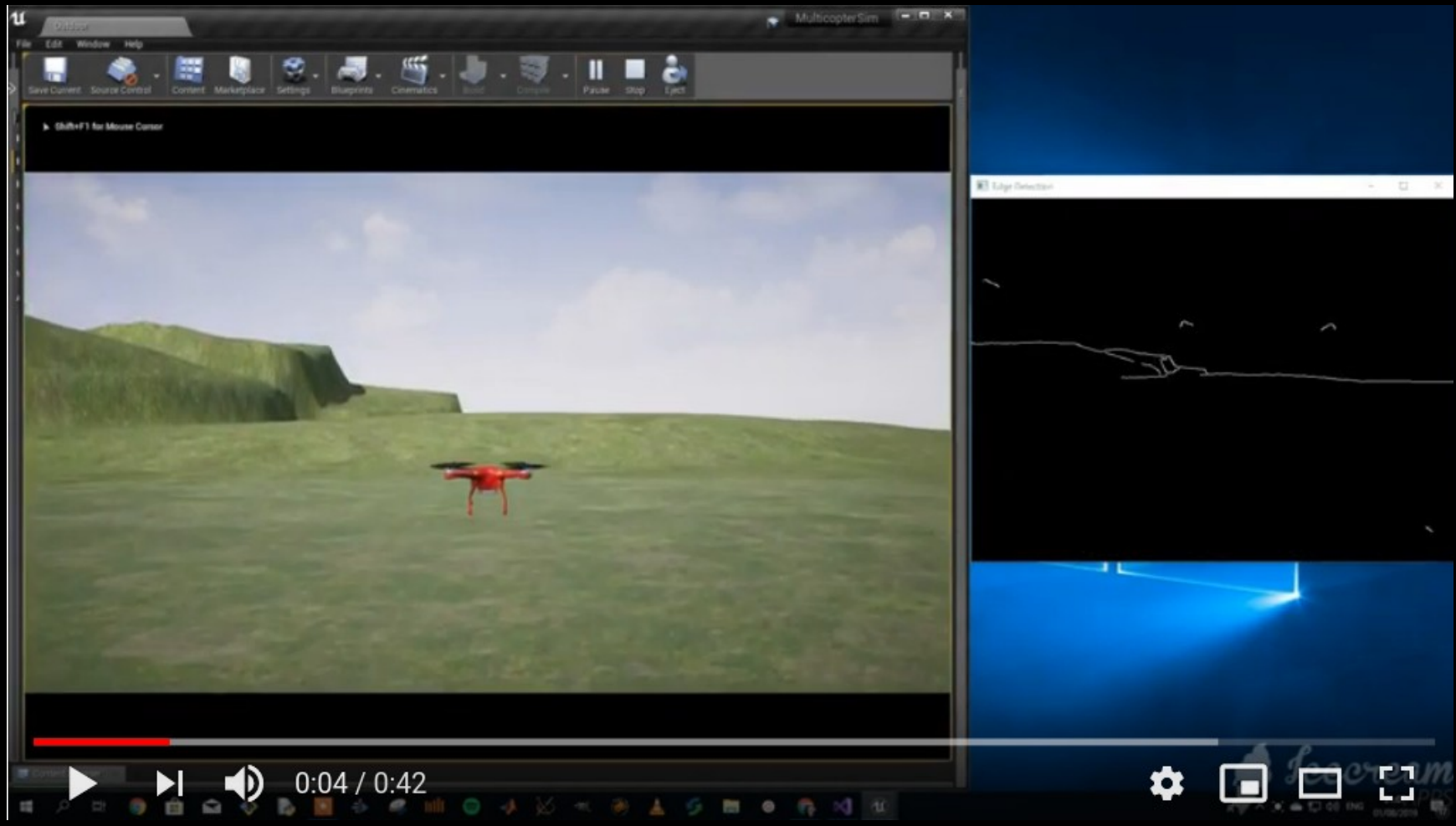
* Design and Control of an Indoor Micro Quadrotor

Samir Bouabdallah
Autonomous Systems Laboratory
Swiss Federal Institute of Technology
Lausanne, Switzerland
Email: samir.bouabdallah@epfl.ch

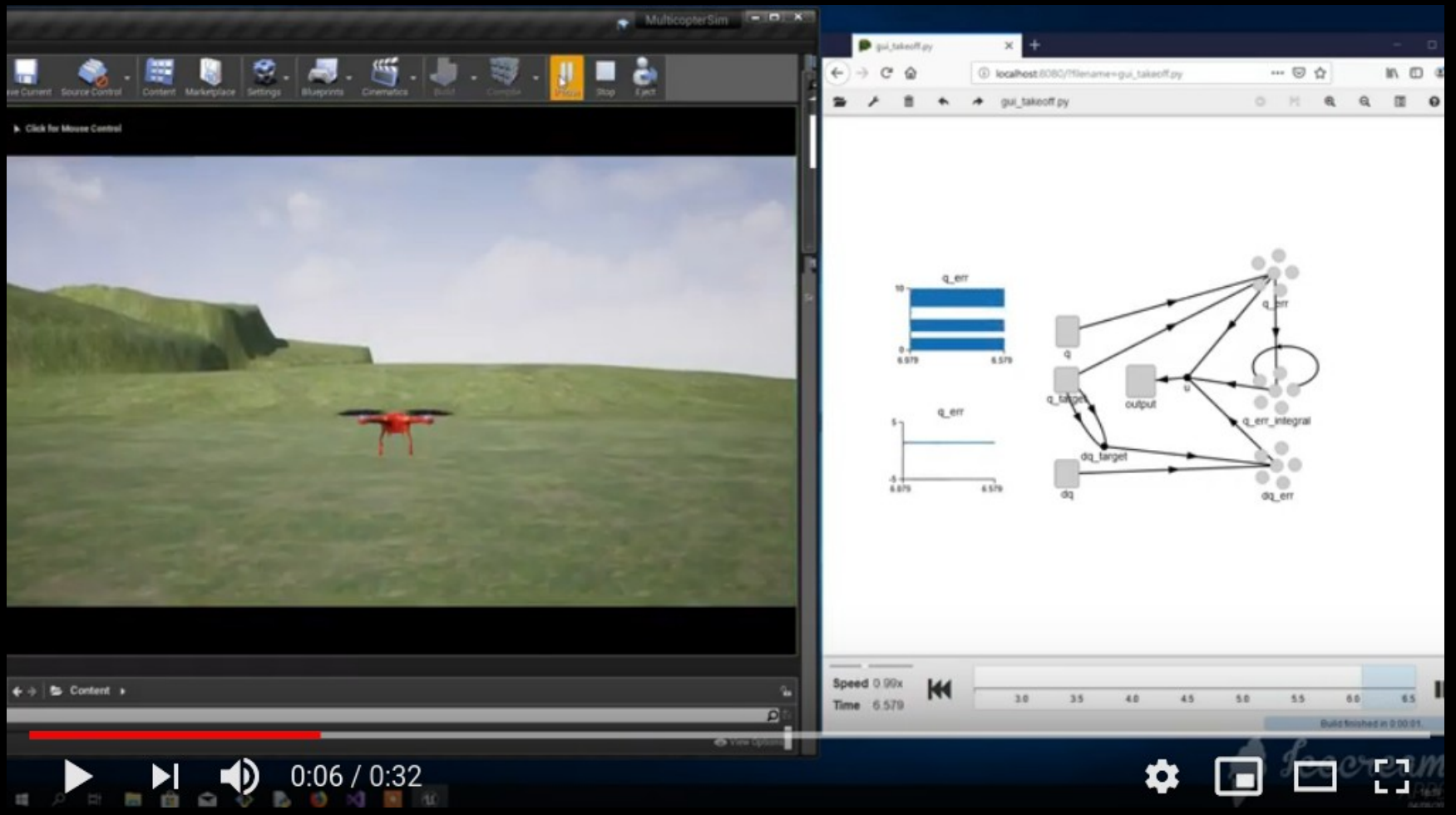
Pierpaolo Murrieri
Interdepartmental Center "E. Piaggio"
University of Pisa
Pisa, Italy
Email: p.murrieri@ing.unipi.it

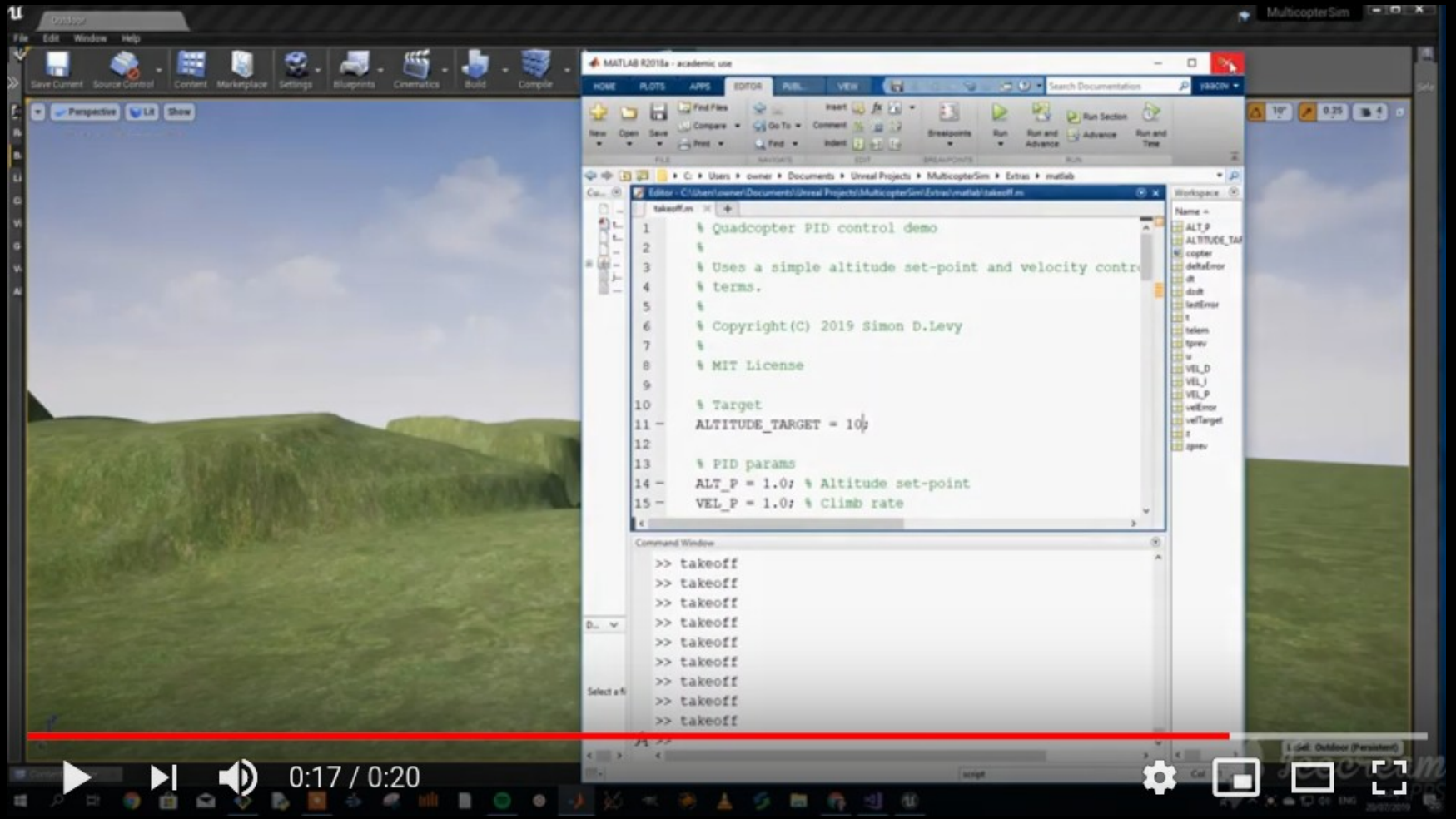
Roland Siegwart
Autonomous Systems Laboratory
Swiss Federal Institute of Technology
Lausanne, Switzerland
Email: roland.siegwart@epfl.ch





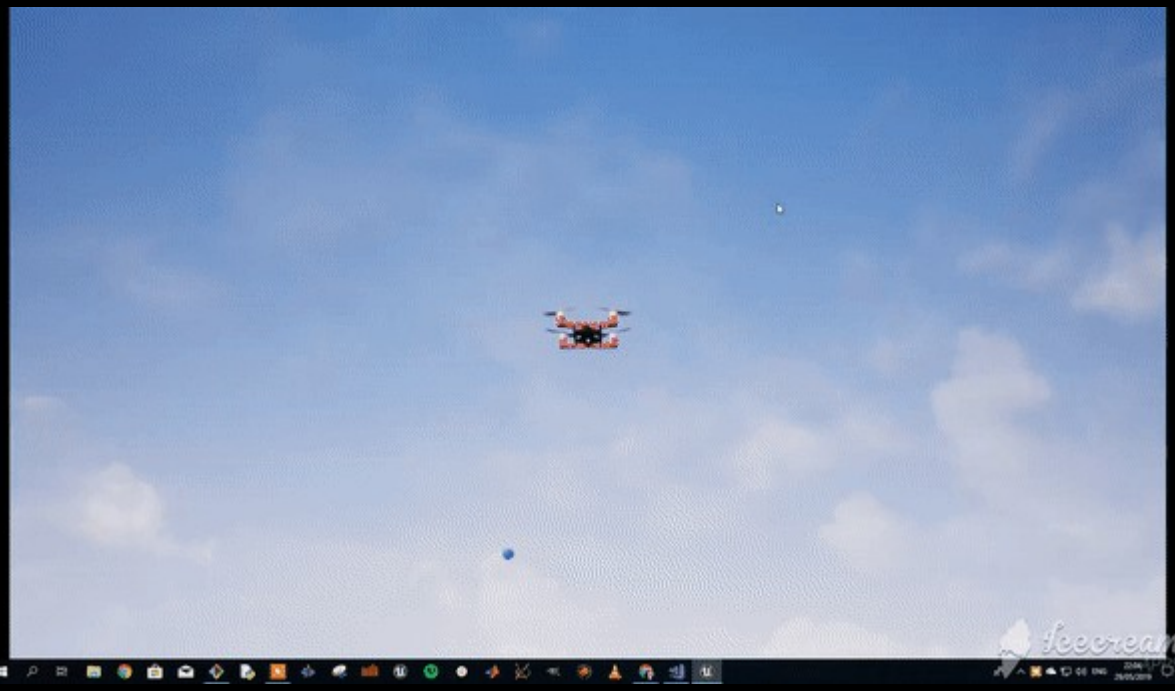
2019

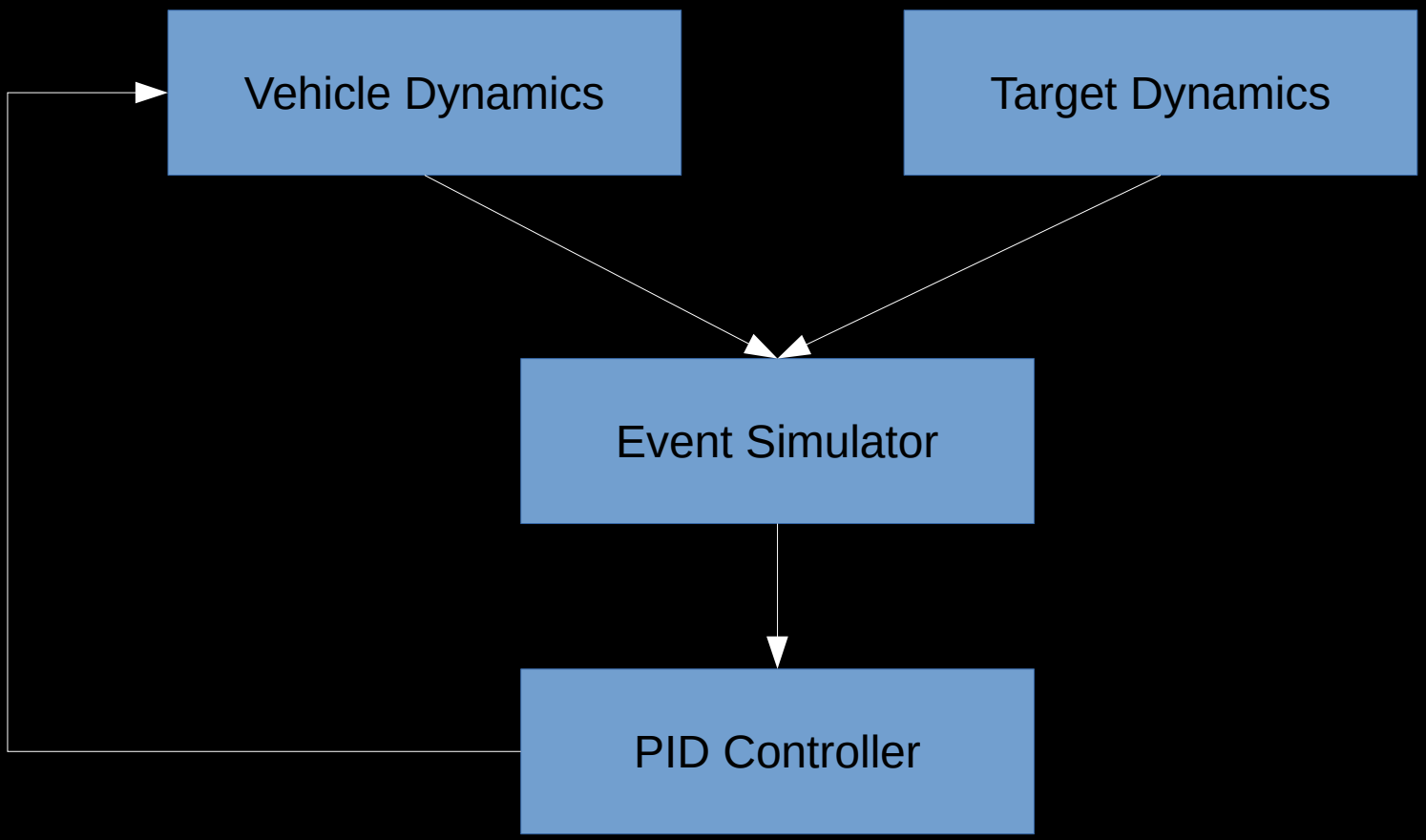

MulticopterSim
Click for Mouse Control
Content
gui_takeoff.py
localhost:8080/?filename=gui_takeoff.py
q_err
q_err
q
q_target
output
q_err_integral
dq_target
dq
dq_err
Speed 0.99x
Time 6.579
0:06 / 0:32

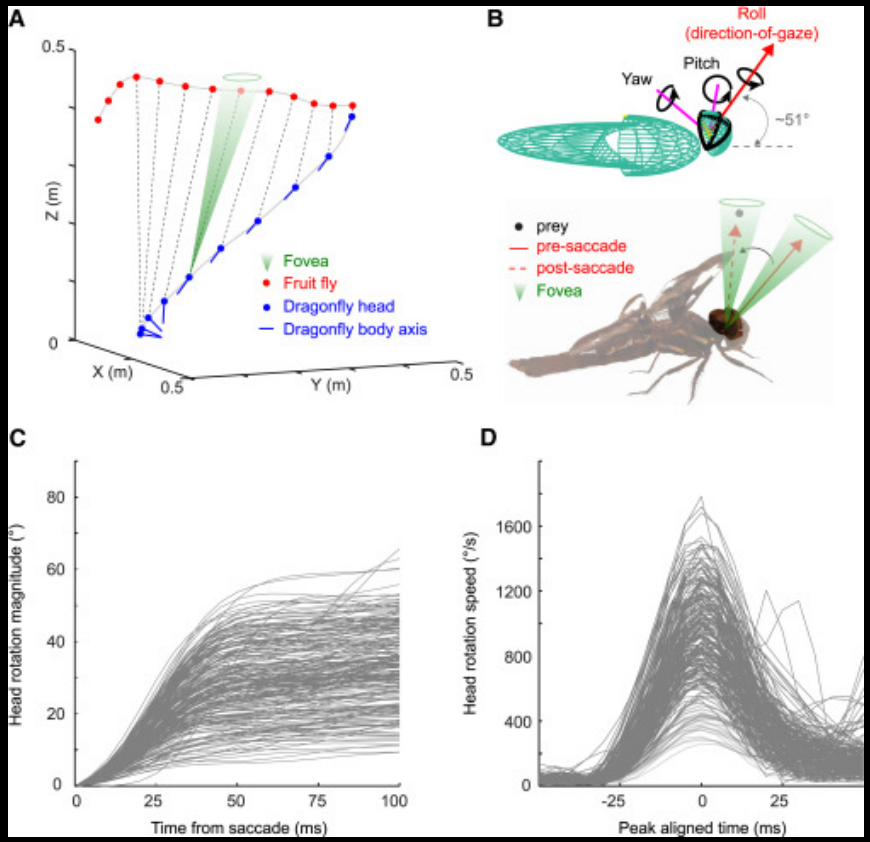

MATLAB R2018a - academic use
takeoff.m
% Quadcopter PID control demo
% Uses a simple altitude set-point and velocity contr
% terms.
% Copyright(C) 2019 Simon D.Levy
% MIT License
% Target
ALTITUDE_TARGET = 10;
% PID params
ALT_P = 1.0; % Altitude set-point
VEL_P = 1.0; % Climb rate
Command Window
>> takeoff
>> takeoff
>> takeoff
>> takeoff
>> takeoff
>> takeoff
>> takeoff
>> takeoff
>> takeoff
0:17 / 0:20

Part III: Current Work

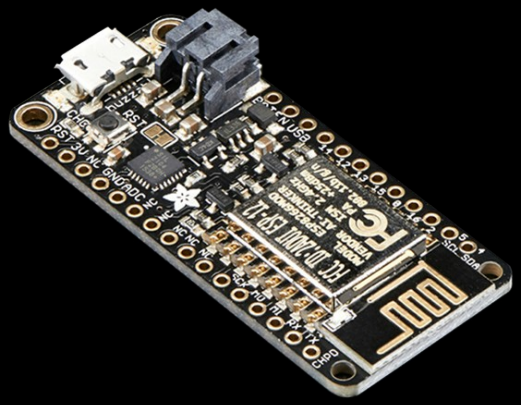
Davis 346 Dynamic Vision Sensor




Vehicle Dynamics
Target Dynamics
Event Simulator
PID Controller



Lin & Leonardo (2017)

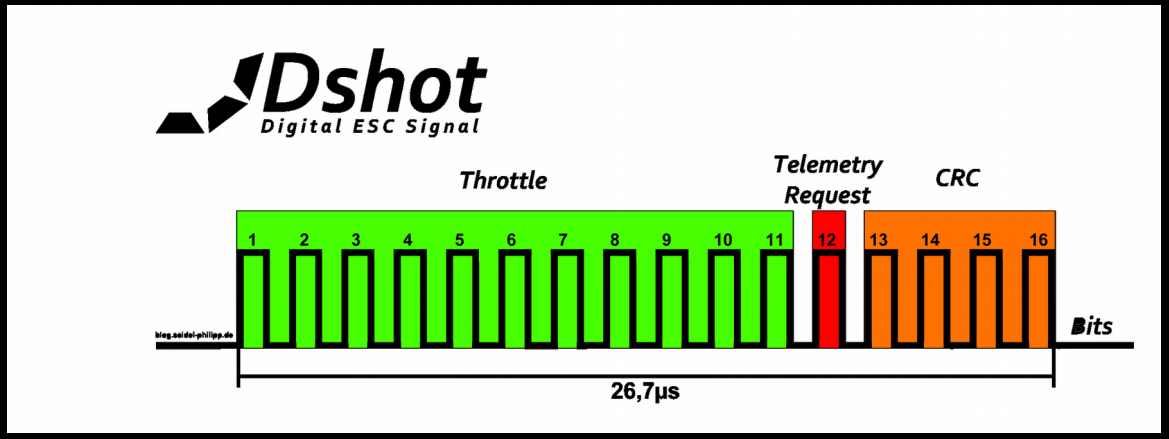


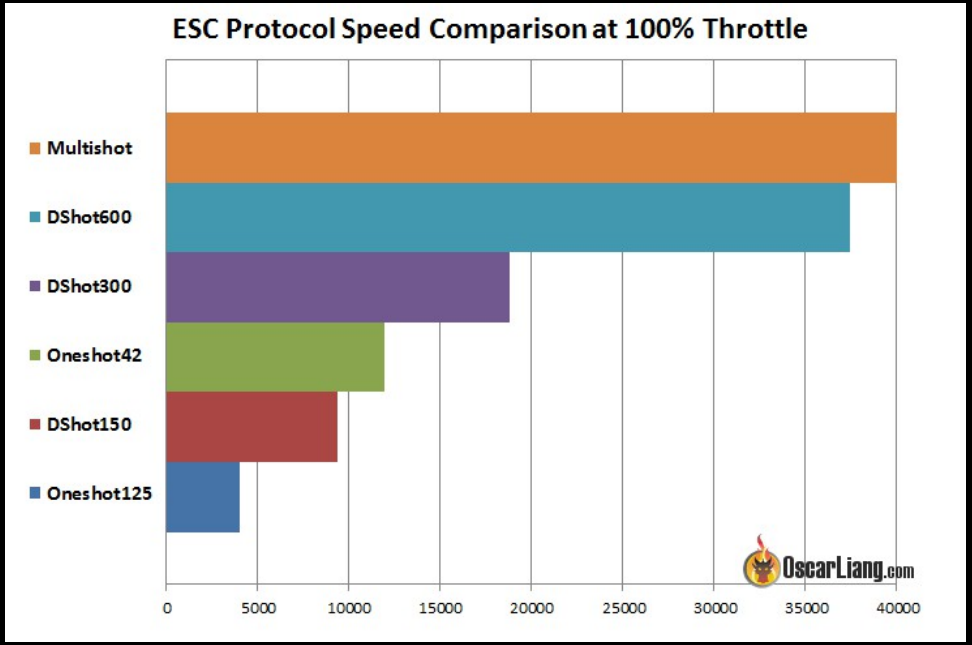
ESP32 Feather Board

2x240 MHz
Bluetooth, Wifi

Teensy 4.0

600 MHz


Dshot
Digital ESC Signal
Throttle
Telemetry Request
CRC
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Bits
26,7µs


ESC Protocol Speed Comparison at 100% Throttle
Multishot
DShot600
DShot300
Oneshot42
DShot150
Oneshot125
0
5000
10000
15000
20000
25000
30000
35000
40000
OscarLiang.com


fishpepper.de

Acknowledgment