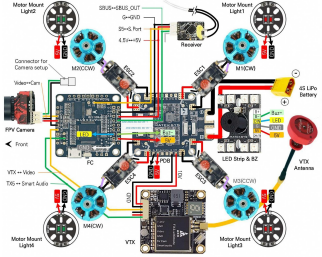


Why real-time operating systems?

Kizito NKURIKIYEYEZU, Ph.D.

Case study of a Real-time system

FIG 1. Crazyflie—a programmable nano-quadcopter[1]

[1] https://www.bitcraze.io/products/crazyflie-2-1/

Crazyflie—hardware

FIG 2. Crazyflie—a programmable nano-quadcopter

Crazyflie—hardware

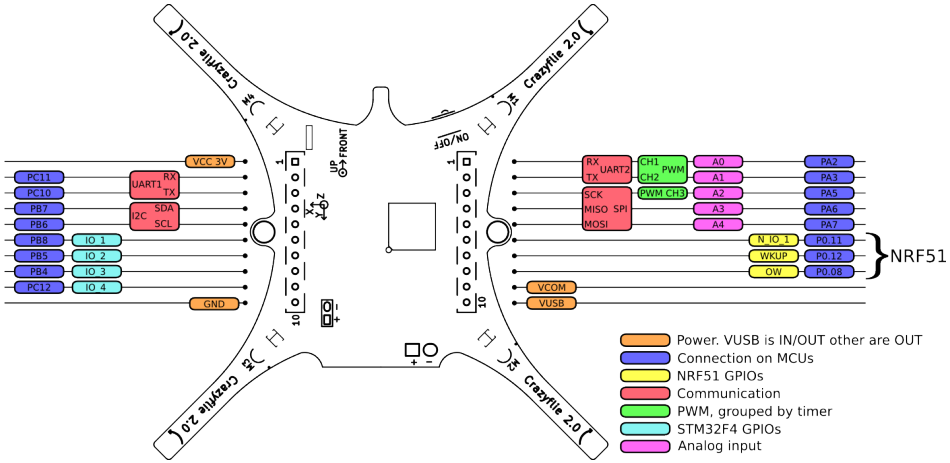


FIG 3. Crazyflie—a programmable nano-quadcopter

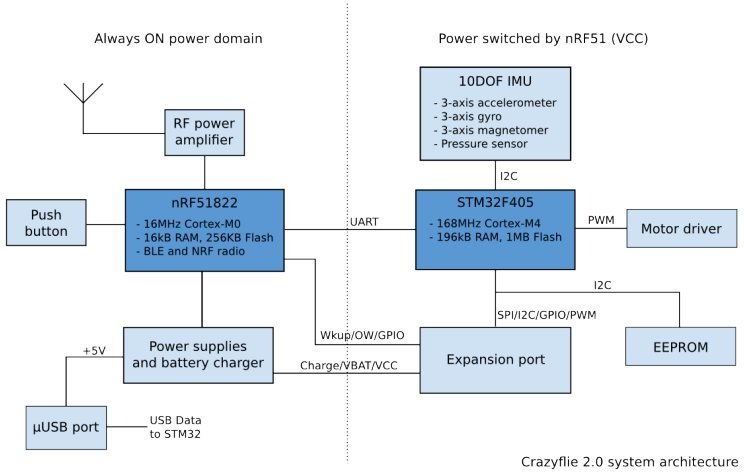


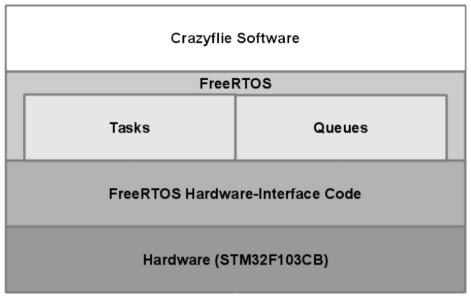
FIG 4. Crazyflie —High-level System Architecture[2]

[2] https://wiki.bitcraze.io/projects:crazyflie2:architecture:index

- nRF51822—low power CPU
  - enabling power to the rest of the system
  - battery management and voltage measurement
  - wireless radio (boot and operate)
  - detect and check expansion boards
- STM32F405—performance CPU
  - brain of the whole drone
  - responsible for flight control
  - Algorithms for DSP, PID etc
- UART—universal asynchronous receiver / transmitter
  - communication protocol
  - exchange of data packets to and from interfaces (wireless, USB)
- EEPROM—electrically erasable programmable read-only memory
  - used for firmware (part of data and software that usually is not changed, configuration data)
  - can not be easily overwritten in comparison to Flash

High-Level Software View

- Use FreeRTOS which we will use in the labs of this course[a]
- Real-time tasks for motor control (gathering sensor values and pilot commands, sensor fusion, automatic control, driving motors using PWM (pulse width modulation, . . .)
- non-real-time tasks (maintenance and test, handling external events, pilot commands, . . .).



The end