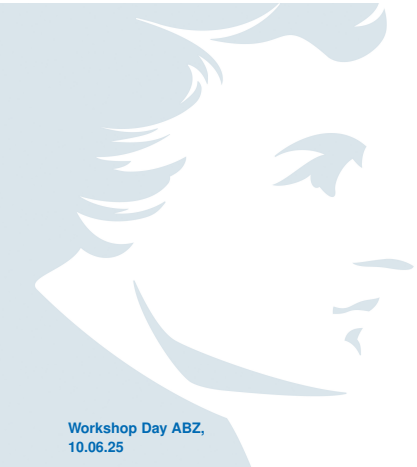
hhu.



Tutorial: Real World-Execution of Drones with ProB

Fabian Vu

Workshop Day ABZ,
10.06.25

Background: Crazyflie Drones

- **Programmatic control of multiple drones via Python-Library** `cflib`[1]
- Decks with sensors for: measuring distances in multiple directions, positioning, camera images

[1] https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/

Background: ProB, SimB, VisB

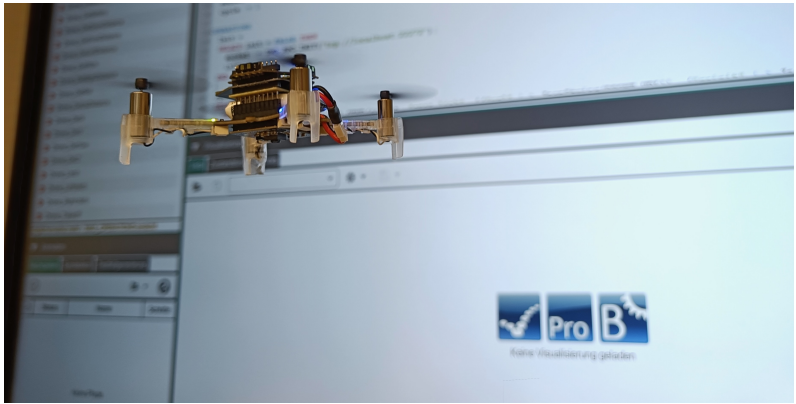
- ProB - Animator, constraint solver, and model checker for formal methods, including B
- SimB - Simulator built on top of ProB
- VisB - Domain-specific visualization tool for formal models

Background: B Method

- Formal method - for specifying and verifying software systems
- Example:

```
MACHINE Drone
VARIABLES x, y
INVARIANT x ∈ 0..100 ∧ y ∈ 0..100
INITIALISATION x := 0 || y := 0
OPERATIONS
  move_right = SELECT x < 100 THEN x := x + 1 END;
  move_left = SELECT x > 0 THEN x := x − 1 END
  ...
END
```

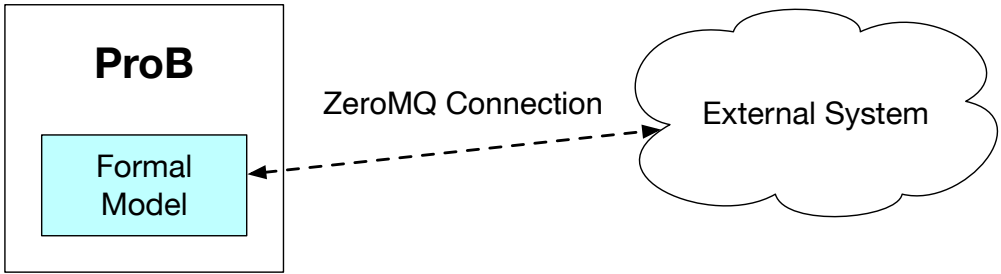
Controlling Drones with ProB: DEMO



Controlling External System with ProB

Challenge: How can we control an external system with ProB?

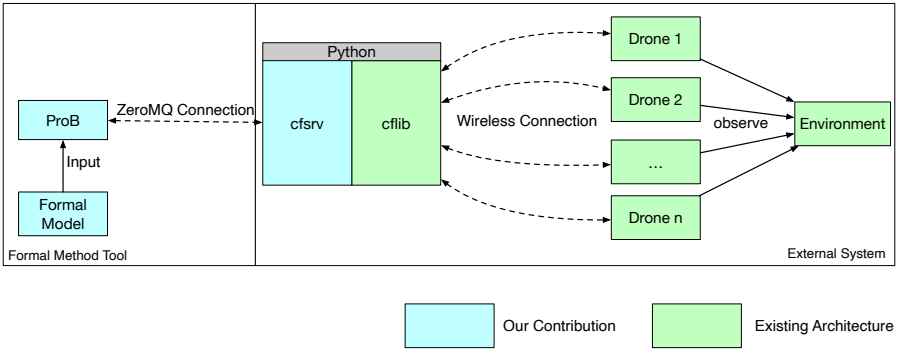
- Step 1: We implemented an external library (`LibraryZMQ_RPC`) in ProB to communicate with external system
- External system - requires server to communicate with ProB



Controlling Drones with ProB

Challenge: How can we control the drones with ProB?

- Step 2: For drones, we implemented a server (`cfsrv`) that communicates with ProB's `LibraryZMQ_RPC` (Step 1) and with the drones via `cflib`



Controlling Drones with ProB

Challenge: How can we control the drones with ProB?

- Step 3: Implement a Drone communicator in Classical B, where operations make use of external functions from `LibraryZMQ_RPC` (Step 1) to invoke Python functions that control the drones (Step 2)

Part from B Model:

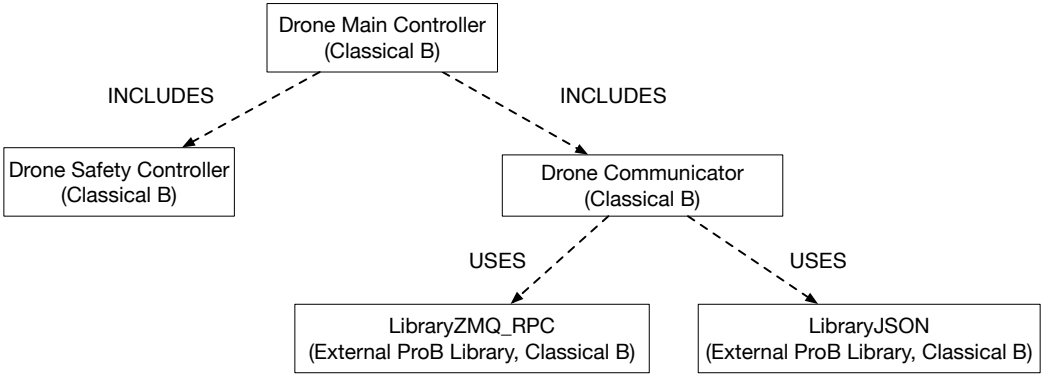
```
MACHINE DroneCommunicator USES LibraryJSON,
        LibraryZMQ_RPC
CONSTANTS DRONE_URL, ...
VARIABLES socket, ...
OPERATIONS
Drone_Takeoff =
SELECT init = TRUE THEN
  VAR res IN
    res := RpcSuccess~(
      ZMQ_RPC_SEND(socket,
        "takeoff", {("url" |-> JsonString(DRONE_URL))}))
  END
END;
...
```

Python Function:

```
from zmq_rpc import JsonRpcServer, JsonRpcRequest
import cflib
import cflib.crazyflie
...

def takeoff(self, url, height=1.0):
  ...
  mc = self._commander[url]
  if isinstance(mc, MotionCommander):
    mc.take_off(height=height, velocity=0.5)
  elif ...:
    ....
  else:
    raise AssertionError("unknown commander")
```

Modeling Safety Controller for Drones



Main and Safety Controller for Drones

Safety Controller:

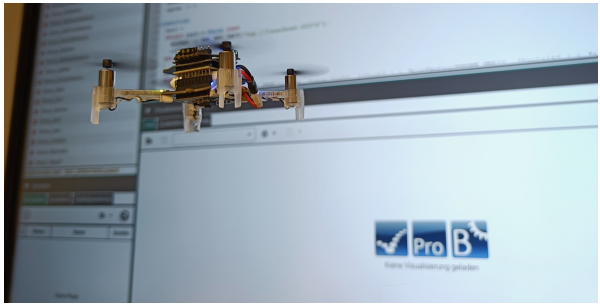
```
FORWARD(dist) =
SELECT
  updated = TRUE ∧
  dist < sensor_data(FORWARD_SENSOR) − SAFETY_DIST ∧
  ...
THEN
  y := y + dist ||
  updated := FALSE
END
```

Main Controller:

```
MAIN_FORWARD(dist) =
BEGIN
  SafetyController.FORWARD(dist);
  Communicator.Drone_Forward(dist)
END
```

DEMOs: Real-World Animation of Drones

1. Interaction with Drones while reading sensors automatically
2. Autonomous Flying of Drones



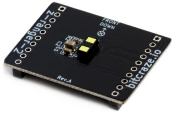
Conclusion

- Crazyflie drones - usable for teaching formal methods with real-world system
- Real-World Execution of Drones (Manual and Autonomous)
- Future research directions:
  - Applying formal methods to UAVs
  - Applying formal methods to robotic systems, AI systems, multi-agent systems
- People involved: Maike Angelike, Julius Armbrüster, David Geleßus, Philipp Körner, Lukas Lang, Michael Leuschel, Miles Vella, Fabian Vu

Appendix

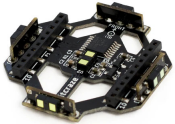
Crazyflie Drones - Decks and Sensors

Flow deck
measures distance to ground

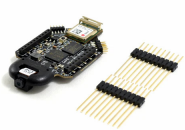


Z ranger deck
also measures ground distance

Positioning deck
tracks location via positioning nodes



Multi-ranger deck
6 sensors, all directions



AI deck
camera for image recognition

and more ...

Modeled Aspects

- Modeling of environment
  - Static and dynamic obstacles
  - Other drones
- Movement of drones + reading sensor data
- Collision avoidance + safety distances
- Inaccuracy in movements and sensor data
- Relative position vs. absolute position
- Single agent vs. multi agent

Tasks for Drones

- Mapping of environment
- Coverage of an area
- Chasing other drones
- Wall following
- ...

Validation and Verification

- Verification: Proving, Model Checking, Abstraction
- Validation: Trace Replay
- Mockup of interaction to real drones - for replaying traces from real-world execution

Challenges

- State Space Explosion
- Inaccuracy of sensors, movements, and positions
- Big-step control of drones
- Reliance on firmware of drones