FINDINGS DURING THE DEVELOPMENT OF THE BALANCED ROBOT PROJECT

- Tooling/technology set-up
  - Each technology (e.g. RFID control, PWM servo control, serial communication, camera/computer vision, Bluetooth...) has various offerings. But once you need a solution for a specific OS (Windows/Linux/embedded) and language (C++/Python IDE), choice turns limited and cross-compatibility an issue
  - Each processor (Arduino/Raspberry/AWS/PC) requires a separate set-up of the development tooling and selection/integration of the technologies/packages. This is complex, time consuming and unpredictable at best.
  - Pycharm is a great python IDE, allowing to work seamlessly on AWS or Raspberry Pi. Well done.
    However, some problems in the development environment (e.g. “global” statement does not work reliably in python console in Pycharm) need to be found and require work-arounds

- Basic feature implementation
  - Finding solutions to typical and widespread problems is still hard – the internet is full of suggestions or partial answers, but consolidation is lacking (since decades...)
  - Multiprocessing/-tasking/parallel processing: anything that can go wrong will go wrong (and also hide well)
  - Communication between processors is a constant source of trouble: Keep interface design simple and straightforward. Implementation must be robust first, including startup/rundown. Avoid changes of the protocol if possible

- Integration
  - Once technology is basically working, integration is the true challenge. Examples:
    - Turning off a PWM signal at the wrong time causes a shortened final pulse, causing a wrong servo setting
    - Robot positioning was unprecise for ~2cm. The root cause: Tires are not round and have soft rubber
    - Using different light (sunny/cloudy, LED vs. classic light bulbs) changes the detected RGB color of objects
  - Magic numbers (esp. thresholds, or balance control parameters): must analyze dependencies/scaling and tolerances. If success is not robust, the problem is not in the magic number, its in the algo using it.

- Process
  - Effort split: 50% tooling/tech set-up, 10% feature implementation, 40% integration/testing/debugging/tuning
  - I find the following 5 phases helpful for each feature: 1. state intent, 2. plan approach, 3. prototype the approach, 4. implement (until no known open points), 5. test/debug/tune. Time invested in 4 will save much time in 5.
  - Rich data logging (including saving images) and good error checking really helps pinpointing the origin of problems
  - Predictions, when things should be working, are too optimistic (even when taking this into account already)

R. Bruckmeier, https://github.com/robbruck/RFID-robot