Thematic analysis – Replication

Template

1. **Replicated experiment**
    a) **General difficulty**
        - **Easy task (5)**
        - **Balanced/some difficulties (4)**
    b) **Prior knowledge of the sensors/actuators (3)**
    c) **Positive thoughts on hardware deployment (5)**
2. **Refactorings in TDD**
    a) **Some refactoring (4)**
    b) **No refactoring (1)**
3. **Test cases in NO-TDD**
    a) **No tests (2)**
    b) **Tested anyways (2)**
4. **Application of TDD for ESs**
    a) **Would use TDD going forward/ imo TDD is more suitable for ESs (5)**
    b) **Would use NO-TDD going forward/ imo NO-TDD is more suitable for ESs (2)**
    c) **Not sure/depends on the complexity (2)**
5. **Prior testing experience**
    a) **No prior testing experience (2)**
    b) **Low testing experience (only unit testing) (4)**
    c) **Some familiarity with TDD (3)**
6. **Seminars and exercises**
    a) **Easy to follow/overall good (6)**
    b) **Other**
        - **Too fast for some things / some difficulties (2)**
        - **Would have liked more explanation on TDD (1)**

Student_01 (NO-TDD)

1. For me development of this task was not very hard after the previous tasks. I also feel like at this point I am familiar with some of the libraries.
2. Still wrote tests afterwards. However, in my opinion there are downsides since you're not sure you're testing everything. Writing tests before would be better but also harder.
3. Lectures and seminars were clear, but some more explanation would have been better on TDD. In my opinion TDD is better for ESs because some of the requirements may be harder to understand all at once, but I feel more comfortable with NO-TDD going forward, just because I am more used to it, regardless of ES or not. If I was more experienced with TDD I would for sure use it in the future. Also, a great experience for my future internship; I will have an advantage when going back home.

Student_02 (TDD)

1. Good feeling because it was in line the first one. Nice to see it run and test by yourself with hardware with the sensors.

2. TDD was very useful because it helps you understand what you are programming, and it will work for sure cause you have written the tests before. Only little refactoring performed for the temperature and motor story (4th) because I wanted to clean it a bit after the tests.
3. No prior testing experience at all so it was very nice to learn about this in the lectures, especially TDD; would use TDD for sure going forward, especially with more experience. Main challenge was getting into it in the beginning. It is suitable for ES development in my opinion.

Student_03 (TDD)

1. I knew most of the sensors so developing this task was not hard for me. But really cool to develop because I wanted to see it in action on real hardware.
2. Helpful to use TDD cause the test acted as documentation. You don't need to have a "master plan" for testing the whole system. You can rather focus on each phase at a time. Could help in TDD. Performed refactoring after implementing one story (3) broke something in a previous user story (2). Also, other minor refactorings to improve overall code readability.
3. Overall good experience with the lectures and seminars. It was good to see some testing approaches in practice besides theory. Going forward it depends for me. For quick prototyping or if we do not know all the requirements (like Agile) I would use TDD for sure. As I said you don't need a master plan and you can do a little bit at a time. But if the requirements are already clear I wouldn't use TDD because I feel like it wouldn't really make much sense. Also, I work with ESs but not much with the actual hardware, so I can't really say how to really improve TDD for ESs cause for the software part I think it's fine with the tools that are available.

Student_04 (NO-TDD)

1. I liked the task overall, but it was simple for me because I already used the sensors (Lab of IoT).
2. I had no issues using NO-TDD for this task, cause is the approach I used the most and I also wrote test cases. I feel like I wrote the same number of tests I would have written by using TDD.
3. I liked the experience overall with a mixture of theory and practical stuff. I knew some TDD concepts from another course (Software Dependability). As for ES going forward and using TDD or NO-TDD, I would say it depends on the complexity of the task. For ESs, using TDD in a much more complex task would be more helpful to determine the difference with NO-TDD.

Student_05 (NO-TDD)

1. Not too easy, not too hard. Interesting to see it like this (deployed on hardware). Liked how the user stories are "incremental" and you keep adding stuff on top.
2. Did not write any test cases with NO-TDD. I didn't really feel like I needed to do it, so I preferred to just modify the source code until everything worked. Also don't have much testing experience overall.
3. Lectures were a bit fast for me. Would improve my overall experience with testing before saying for sure which to use in the future. With TDD of course you have less errors so I feel like for ES TDD could be useful if you know more about what you're doing, but generally I would use NO-TDD going forward.

Student_06 (TDD)

1. Had some trouble with the implementation of the 4th task with the servo motor, but the task was manageable overall and clear to understand.

2. I feel like for this task using TDD was helpful; I performed some refactoring, mostly for the test cases, very little or maybe nothing for the production code, I don't' remember.
3. No testing experience at all and very little experience with Python. It was hard to get into the mechanics even for the exercises during the lectures. In the beginning I preferred NO-TDD because it was more natural to test the code after; however, after applying TDD in the last tasks I liked using it. I still need a lot more practice with, but I feel like TDD would be useful for ESs going forward.

Student_07 (TDD)

1. I already knew these sensors from other courses (Lab Of IoT) and knew their interface and how they were used so the task was pretty easy for me.
2. As a result, TDD for me was not really needed for this task, it was a bit overkill. Minor refactoring. On the second task however, which I found much more complex, I feel like TDD was much more beneficial.
3. I already knew Unit Testing, and TDD was introduced in another course (Software Dependability), even if mostly from a theory point of view it has been pretty natural for me to apply it in these tasks. I am not experienced with TDD for ESs specifically so I can't really say if there is a big impact with it but probably it can be very helpful with more complex Ess.

Student_08 (TDD)

1. I felt quite comfortable about developing this task. It was nice to deploy it on hardware, afterwards. Overall, a pleasant experience.
2. Using TDD helped me with this task especially in user stories 2 and 3. For the others the was no big difference in my opinion. Did not perform any refactoring as they were not necessary in my opinion.
3. Comfortable about the lectures and seminars. Learned why TDD could be helpful. For sure testing ESs is important regardless of the approach; however, TDD gives you immediate feedback which I feel is important for ES development.

Student_09 (NO-TDD)

1. At first was a big tough but I really studied and gave it all; it was fun to see the project deployed and tested on real hardware.
2. I did not write any test cases. I really tried to do it but couldn't finish it, so I just decided to focus on the implementation. In the future I would love to write more test cases.
3. Didn't have much experience with testing before the lectures and seminars. I think TDD is more fun, and it could be useful for ES development. with no-TDD it can be a bit more complex since adding the tests after can be more challenging. When I applied TDD I didn't find any particular difficulty, however it was a bit more challenging then no-TDD, but i think that if I learned more how to use TDD, it would be better.