LaTeX 2_ϵ-Vorlage von Matthias Pospiech

Leibniz Universität Hannover

Matthias Pospiech

August 2, 2011

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

<Ort einfügen>, den <Datum einfügen> <Autor einfügen>

Contents

1 Introduction

1.1 Personal motivation

This thesis describes the analysis, enhanced design and implementation of an existing microcontroller based mesh solution [Kor09]. The current solution showed.

1.2 Research overview

2 Evaluation

2.1 Existing solution

2.2 Assumptions

2.3 Requirements

3 Hardware Design

3.1 RAM

- Harvard architecture
- RAM bus
- Latch

3.2 USB Serial Device

3.3 RFM12B Radio

3.4 Keyboard

4 Software Modules

4.1 UART

4.2 SPI

4.3 Watchdog

4.4 Clock

4.5 Shell

4.6 Network Stack

4.7 RFM12 Driver

5 Software Algorithms

5.1 Protothreads

Designing a software system that executes on embedded micro-controllers implies a lot of challenges when many software modules are involved and complexity grows. The conceptually defined modules must be somehow implemented. If the micro-controller lacks an operating system then there is no possibility of using provided abstractions and APIs for module orchestration and execution. Another challenge are limited hardware resources which prevent the deployment of many existing operating system kernels. Basically there are two types of execution models which can be implemented in micro-controllers:

Sequential execution

This type sequentially executes all modules starting from the first module until the last one. Once the last module ends the execution starts again from the first module. The pseudo code for this execution model is visible in listing 5.1.

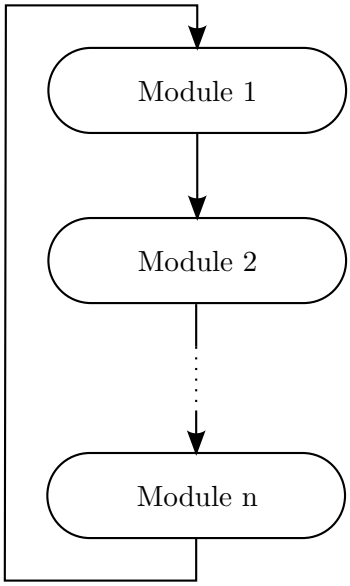

Figure 5.1: Sequential execution model

Listing 5.1: Sequential execution pseudo code

```
function main() {
    while(true) {
        module_1();
        module_2();
        ...
        module_n();
    }
}
```

Concurrent execution

"Traditional" embedded implementations are using state machines. Especially the existing thesis uses state machine based algorithms a lot, although the author does not mention this fact at all.

Alternatives:
- Heavyweight: Real Operating System. Enumerate them and compare ...
- Lightweight: Thread implementations. Problem: Each thread has its own stack which consumes a lot of memory.
- More Lightweight: Protothreads. Best compromise between classical state machines and real threads.

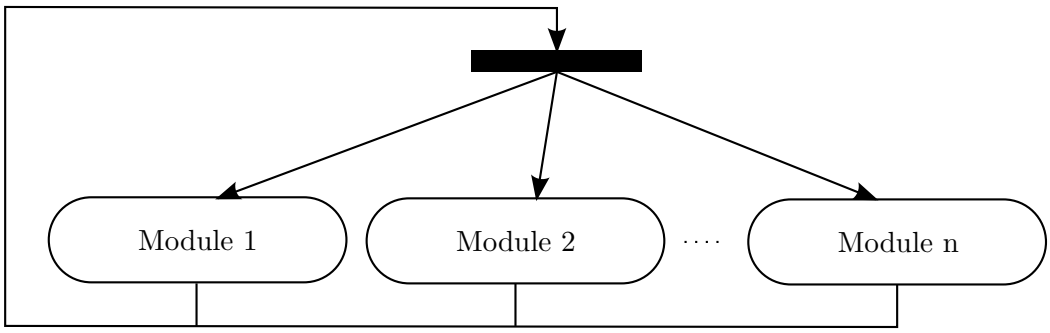

Figure 5.2: Concurrent execution model

5.2 Ring Buffers

5.3 Half-Duplex Radio Access (Petri Net)

6 Network Stack

6.1 Layer 2a: MAC Layer

6.2 Layer 2b: Logical Link Control

6.3 Layer 3: Batman Routing

6.4 Layer 7: Application

7 Research

7.1 Simulations

7.1.1 Shell

7.1.2 Routing

7.1.3 Radio Transmission

7.2 Mesh evaluation

7.3 Results

8 Conclusion

Bibliography

[Kor09] Korniowski, Marek: *Projekt odpornej na awarie sieci komputerowej z transmisją danych w pasmach nielicencjonowanych* (2009)

List of Figures

List of Tables

Danksagung