TK3: Mobile & Ubiquitous Computing

Exercise E1: IoTK Platform & Wireless

Submission Date: 03.06.2019 by 14:00 MESZ

By handing in a solution, you confirm that you are the exclusive author(s) of all the materials.

Introduction

This year, the exercises will be all about creating the IoTK, a simplified Internet-of-Things platform, from scratch. In this exercise, we focus on the hardware platform and wireless connection.

Hand-In

You have to hand-in the code before the submission date specified above. No late submissions will be allowed and results in 0 points. After the submissions, each group will present their solution to one of the tutors.

Task 1. Wireless Medium Access Control (10 points)

An important part will be the ESP32 development platform. These boards are capable of Wi-Fi and Bluetooth and are compatible with the Arduino platform. The first task focuses on first steps with this platform.

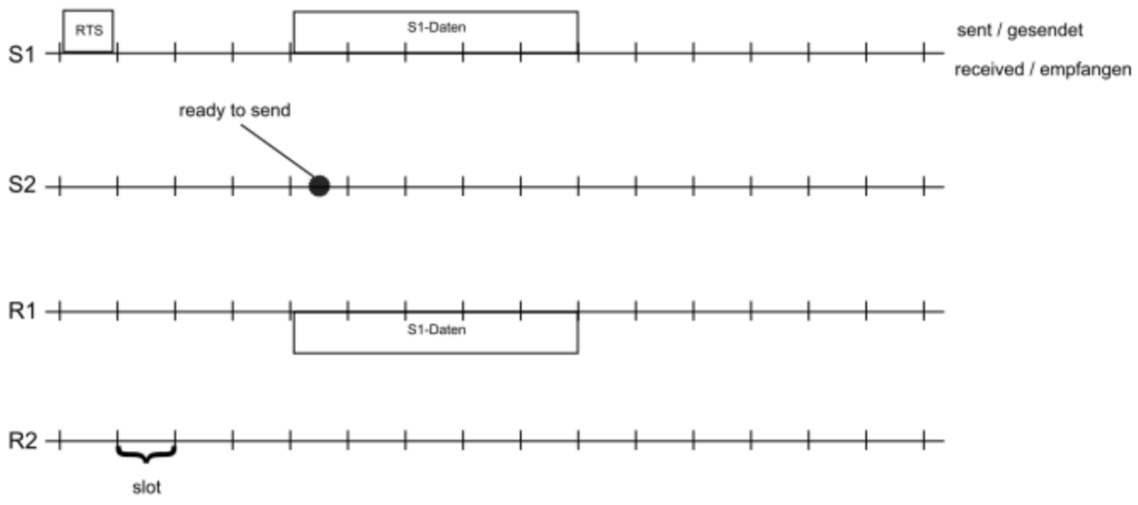
Task 1.1

Please draw a constellation of four stations (two senders (S1, S2) and two receivers (R1, R2)) where the *exposed terminal* problem can happen. Include the *transmission range* and the *detection range* for the senders into your drawing.

Task 1.2

Four stations (two senders (S1, S2) and two receivers (R1, R2)) are situated in a constellation where the *exposed terminal* problem happens. The communication is clocked, meaning transmissions start only at the beginning of a time slot. The simple *RTS/ CTS* protocol for MAC is used. IFS amounts to one time slot.

In the diagram below one transmission from S1 to R1 and the related sent RTS packet are already depicted. Please complete the drawing with sent and received RTS, CTS and ACK packets belonging to the transmission of S1. Please draw sent packets above the line and received packets below the line. Signal propagation delays can be neglected. Please pay attention to draw all packets at sender and receiver correctly.

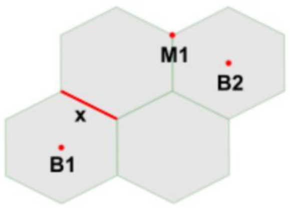


Task 1.3

Imagine a cell phone network (cell-based). Frequencies are bound to *equilateral* hexagons (see Figure below). The network contains three stations (B1, B2, M1). M1 is a mobile station. This means it will NOT always be at the location marked in the figure!

- The length of each hexagon side is *x* = 100m.
- All stations send with the same frequency f = *2.4GHz.*
- The maximum transmission power of the stationary senders *B1* and *B2* is *Pb = 100W.*
- The maximum transmission power of M1 is *Pm* = 500mW.

Note: Signal strength is damped using the *Friis transmission equation*: $A = g\,\frac{1}{f^2 d^\alpha}$, where *d* is the distance and f the frequency. Use α = 2 (line-of-sight} and $g = 5.6 * 10^{14}\ \frac{m^2}{s^2}$ for your calculations.



We consider the following scenario: *B1* and *B2* use full power to transmit to M1. M1 is located at *B2* (please note: This is different than illustrated in the figure!). *M1* wants to receive the signal from *B2.* The signal transmitted by B 1 is received as noise. There are no other sources of noise present.

Calculate the signal-to-noise ratio (SNR) for M1. Please write the complete formula and constitute all values, before giving your final answer.

Task 2. Plug 'n' Wi-Fi (10 points)

Connecting many wireless devices to an access points is often annoying. While Wi-Fi Protected Setup (WPS) provides an easy connection, its PIN method is considered insecure. Therefore, the IoTK pursues a different approach: As soon as an ESP32 is plugged into the USB port of a Raspberry Pi the Wi-Fi credentials are exchanged automatically without any further need for interaction.

Task 2.1

Write code for the ESP32 that receives Wi-Fi credential (i.e. SSID and password suitable for WPA2 personal/psk) from the serial console and stores the credentials in a permanent memory on the ESP. Moreover, the ESP should initiate a Wi-Fi connection on each boot using the stored credentials (you may use the libraries *EEPROM* or *Preferences*).

Task 2.2

Write code for the Raspberry Pi that automatically detects when an ESP32 is attached via USB. To that end, you may use an UDEV rules file with the following content (in a single line):

```
SUBSYSTEM=="tty", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", MODE="0666",
ENV{ID_MM_DEVICE_IGNORE}="1", SYMLINK+="ttyESP32", RUN+="/bin/su - pi -c /home/pi/tk3/yourScript.sh"
```

Create a file called *esp32.rules* and place it in `/etc/udev/rules.d/` folder. A restart of UDEV may be necessary (depending on the distribution via `sudo reload udev` or `service udev restart`).

If the rule is working properly, a file */dev/ttyESP32* is create to access the ESP via a serial connection and the script */home/pi/tk3/yourScript.sh* is executed as user *pi* (you may customize device location, user or script path to your needs).

Task Requirements

The following requirements must be fulfilled for this task:

- The ESP32 should signal success through permanently activating its LED1 for 10 seconds and failure through blinking LED1 for 10 seconds
- The WiFi credentials need to be permanently stored in the ESP32.
- No user interaction (besides plugging in via USB) is required to transfer the credentials
- The ESP32 connects to a WiFi on each boot if valid credentials were stored
- The Raspberry Pi automatically starts all necessary code on boot.

Tutorials

ESP

- https://github.com/espressif/arduino-esp32
- https://github.com/espressif/arduino-esp32/tree/master/libraries/Preferences
- https://github.com/espressif/arduino-esp32/tree/master/libraries/EEPROM

Raspberry Pi

- https://projects.raspberrypi.org/en/pathways/getting-started-with-raspberry-pi

UDEV & Serial Connections

- https://unix.stackexchange.com/questions/65891/how-to-execute-a-shellscript-when-i-plug-in-a-usb-device
- https://playground.arduino.cc/Interfacing/LinuxTTY