bachelor's thesis

Communication for distributed control of slotcar vehicular platoon

Anastasia Vlasova

May 2015

Supervisor: Herman Ivo Ing.

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Control Engineering

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Control Engineering

BACHELOR PROJECT ASSIGNMENT

Student: **Anastasia Vlasova**

Study programme: Cybernetics and Robotics
Specialisation: Systems and Control

Title of Bachelor Project: **Communication for distributed control of slot-car vehicular platoon**

Guidelines:

1. Implement an interface to a communication chip through an appropriate bus. The goal is to prepare a simple API, which will be then used by upper control layers.
2. Prepare a communication infrastructure for a platoon control algorithms, in which vehicles share their states with the others (CACC, LQR, ...).
3. If possible, implement identification algorithm of the car's position in the platoon.

Bibliography/Sources:

[1] C. Siva Ram Murthy, B.S. Manoj: Ad Hoc Wireless Networks : Architectures and Protocols, Prentice Hall, 2004
[2] Sarangapani J. : Wireless Ad hoc and Sensor Networks: Protocols, Performance, and Control, CRC Press, 2007
[3] Drew Gislason: Zigbee Wireless Netowrking, Newnes 2008

Bachelor Project Supervisor: Ing. Ivo Herman

Valid until the summer semester 2015/2016

[illegible]
prof. Ing. Michael Šebek, DrSc.
Head of Department

L.S.
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

[illegible]
prof. Ing. Pavel Ripka, CSc.
Dean

Prague, December 12, 2014

Acknowledgement

I would like to offer my sincerest gratitude to my supervisor, Ing. Ivo Herman, who lead me through this project, helped me with the development and writing of this thesis. I'd like to thank Ing. Dan Martinec and Bc. Martin Lád for valuable advises. I am also grateful for the support of my family and friends.

Declaration

I declare that I worked out the presented thesis independently and I quoted all used sources of information in accord with Methodical instructions about ethical principles for writing academic thesis.

Prague, 22th 2015

[illegible]

Abstract

This research focuses on the preparing of the inter-vehicles communication system for the slotcar platoon model. The communication is implemented as a feed-forward signal and improves the existing cascade feedback controller, which uses only the onboard sensors measurements. The slotcar platoon model consists of three slotcars provided by Carrera. Each vehicle is equipped with ARM Microcontroller and radio board with CC2530 wireless MCU which runs full ZigBee Pro solution. This thesis also deals with the ZigBee network processor configuration and contains the description of the ZigBee hardware driver implementation. Being equipped by wireless communication capabilities slotcars communicate with each other and the base station. It allows to control vehicles wirelessly, read out data during the experiments and identify car's position in the platoon.

Keywords

ZigBee; CC2530; CC2530ZNP-Pro; STM32F4; SPI; radio board; slotcar; cognitive vehicle platooning; CACC; position identification…

Abstract

Tento dokument ze zabývá implementací bezdrátové komunikace mezi autodráhovými autíčky. Tato komunikace je použita pro implementaci dopředného řízení, o které je rozšířen již implementovaný zpětnovazební kaskádní regulátor vzdálenosti, používající pouze veličiny měřené lokálními senzory na autíčku. Každé autíčko je vybaveno ARM procesorem a komunikačním čipem CC2530, poskytující ZigBee Pro standard. Tato práce se také zabývá samotnou konfigurací a popisem tohoto čipu. Použitím bezdrátové komunikace jsou autíčka schopna posílat data během experimentu do počítače. To umožnuje bezdrátové ovládání kolony, konfiguraci parametrů a určení pořadí autíček.

Klíčová slova

ZigBee; CC2530; CC2530ZNP-Pro; STM32F4; SPI; radio board; slotcar; CACC; určení pozice. . .

Contents

Abbreviations

Abbreviation	Phrase
AP	Application Processor
API	Application Programming Interface
APL	Application layer
CCA	Clear Channel Assessment
CRC	Cyclic Redundancy Code
CSMA-CA	Carrier Sense Multiple Access - Collision Avoidance
EM	Evaluation Model
FCS	Frame Check Sequence
FFD	Full-Function Device
GTS	Guarantee Time Slots
HW	Hardware layer
MAC	Medium Access Control layer
MCU	Microcontroller Unit
MHL	Message Handler Layer
MPL	Message Program Layer
NWK	Network layer
PAN	Personal Area Network
PHY	Physical layer
RF IC	Radio Frequency Integrated Circuit
RFD	Reduced-Function Device
SPI	Serial Peripheral Interface
ZC	ZigBee Coordinator
ZDK	ZigBee Development Kit
ZDO	ZigBee Device Object
ZED	ZigBee End Device
ZHA	ZigBee Home Automation
ZNP	ZigBee Network Processor
ZR	ZigBee Router

1 Introduction

The main goal of the project is to implement a communication infrastructure for platoon control algorithms, in which vehicles share their states with the others. To do it we need to choose a suitable communication standard and implement the solution into the slotcar. This thesis describes the implementation process, which includes the study of the vehicle hardware and software, the SPI initialization, detailed research on the ZigBee communication standard and creating the ZigBee hardware driver which provides the transmission and reception services. Different algorithms were implemented in the main program layer as well, e.g. position determination in the platoon, address information or parameters setting.

1.1 Motivation

The Super Smart Vehicle System(SSVS) was created to make multiple improvements such as the following: reduce car accidents, optimize speed and handling, increase fuel efficiency and prevent traffic jams. The SSVS was introduced by JSK (Association of Electronic Technology for Automobile Traffic and Driving) in Japan in 1990 [1]. The SSVS consists of 4 fields: information systems for a single vehicle, information systems for inter-vehicles, information systems for vehicle-to-road relations, and studies on vehicle-to-driver relations [2].

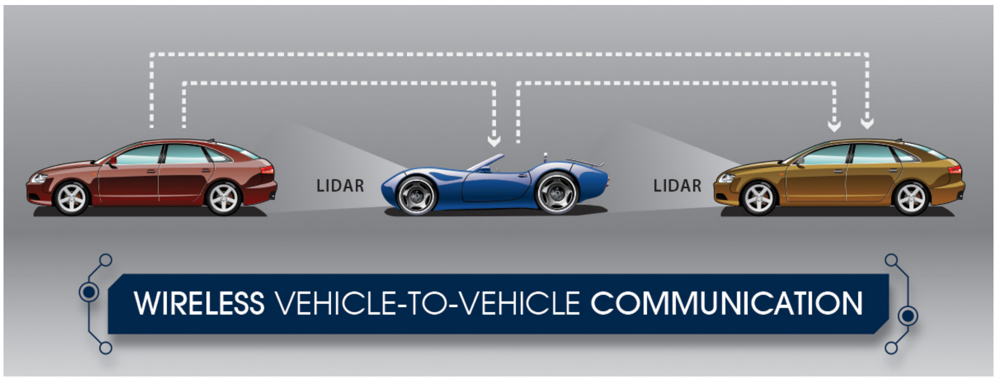


Figure 1. Autonomous Vehicle Design (Source: www.nuvation.com)

One of the main goals of this work is to extend the already existing slotcar platoon model [3] to the "cognitive vehicles platooning", where vehicles are coupled together by wireless communication. Previous works were focused on creating the controllers to follow the reference distance and speed based only on sensor measurements. Using the preceding or the leading vehicle measurements in a combined feedback and feed-forward loop improves the response to the preceding car and minimizes the control signal value. The motivation was not only to improve the existing platform, but try different algorithms on it which could hardly be realized without data exchange. For example, car's position identification in the platoon.

Through the course of this research project further knowledge about microcontrollers and communication protocols (especially ZigBee) was gained.

1.2 The Slotcar platooning project

This project was started in the Department of Control Engineering by the AA4CC group[3]. The goal of this work is to create a slotcar platoon model for testing distributed control algorithms on it.
The main part of the system is a slotcar by Carrera. It is equipped with custom-made electronics. The following are the most important components placed on the main board:

- ARM processor by ST Microelectronics STM32F405RGT6,
- debug communication processor Nordic Semiconductor nRF24L01,
- gyro with integrated accelerometer LSM330DLC,
- INA213AIDCKT current-shunt monitor,
- QRE1113 reflective object sensor is used for speed measurements.

. A VEMT3700F silicon NPN phototransistor and a VSML3710 infrared emitter are placed on the distance measurement board. There are two such boards in the slotcar: for back and front distance measurements. On the separate board is located the ZigBee CC2530 communication processor. The hardware design and implementation was done by Ing. Jaromír Dvořák.

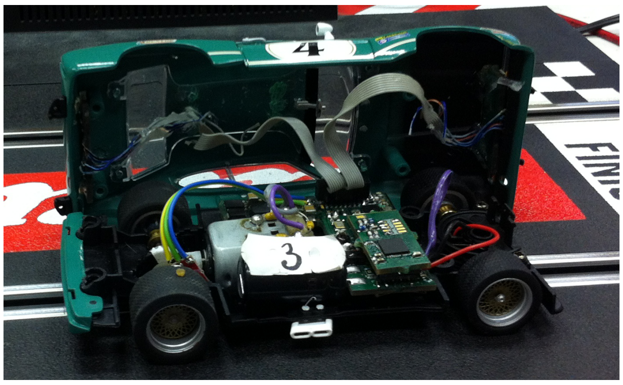


Figure 2. The slotcar with the secondary ZigBee radio board

The project is being developed in Eclipse CDT (C/C++ Development Tooling) [4]. In the project are used the ARM Cortex Microcontroller libraries. Jaromír Dvořák wrote the basic program, system functions and the Bootloader program to load the compiled code into the chosen slotcar. After the run of the project the program asks for the 'car address', which is a number from 1 to 10 written on the vehicle. For loading Nordic nRF51-Dongle is used. The slotcar bond graph modeling, identification and linearization were done by Martin Lád [5]. He also designed and implemented two speed measuring methods (by IRC sensor and back emf) and speed regulator. One of the main achievements was enabling of regulation of low speeds such as $200\,mm \cdot s^{-1}$.
The creation of the distance measuring sensor is described in the thesis of Jan Moravec [6]. There are shown results of testing distance control algorithms on the car platoon.
Results of the current project were tested with the android application, which was written by Alexander Dubeň. A more detailed description of provided experiments are placed in the final chapter.

2 CC2530 communication processor

The CC2530 is an IEEE 802.15.4 compliant true System-on-Chip by Texas Instruments. It supports the ZigBee, ZigBee PRO, and ZigBee RF4CE standards. The CC2530 offers a high-performance microcontroller core, up to 256-KB Flash, 8-KB RAM for ZigBee profiles and an extensive peripheral set including 2 USARTs and 21 general-purpose GPIO pins [7]. It also has very low power consumption, excellent receiver sensitivity and robustness to interference. In comparison with the previous generation, the CC2430, the CC2530 supports ZigBee PRO mesh network applications.

2.1 The secondary radio board implementation

The CC2530 requires very few external components. The secondary radio board contains:

- ZigBee CC2530 communication processor,
- ABM3B 32Mhz crystal,
- TLV70030DCKT linear regulator,
- debug connector,
- SPI connector,
- RF IC.

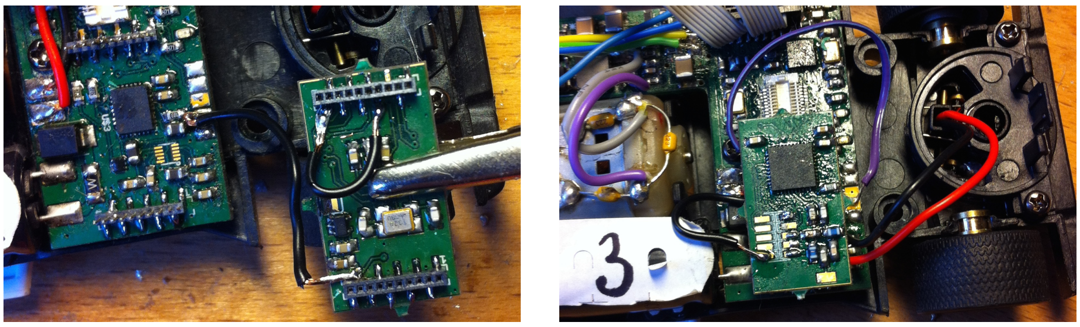


Figure 3. Connecting the secondary radio board to the main slotcar board

To check the board functionality we connected the 3.3V and GND pins to the power supply. The normal current draw for the secondary radio board is 0,3 A. Some of boards showed 0 or high (0,8 A) current draw. The cause wasn't detected (it could be an open or a short circuit). Functional boards were programmed via SmartRF05EB evaluation board with `CC2530ZNP-Pro.hex` file. The evaluation board is equipped with a debug connector that allows debugging and programming of an external RF microcontroller from Texas Instruments. The pin-out of the connector is placed in the SmartRF05 Evaluation Board User's Guide [**p.[22]**, 8]. The secondary radio board pin-out is shown in Figure 4. Only three manufactured boards were detected by the evaluation board.

Next, the secondary radio board was connected to the main board. Table 16 in Appendix A shows the CC2530 and radio connector pin configuration. The 3.3V, GND and RESET (SPI signal) pins were misplaced in the secondary radio board connector

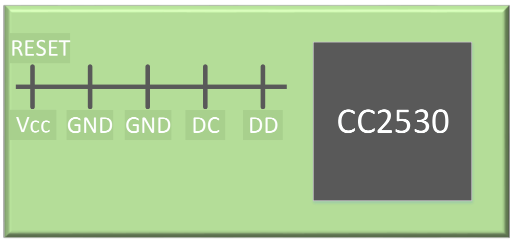


Figure 4. Secondary Radio Board Debug Connector

and were connected with the main car board with wires. The scheme of Secondary Radio Board and Secondary Radio Connector is placed in `slotcar_v1r1.pdf` [9].
The radio board was implemented into three slotcars. A new version of the slotcar is currently being developed and shall be ready at the end of the May 2015. Because of this new model becoming available soon no further investments in the current model have been done, not to mention time limitation.

2.2 Prototype model for the ZigBee driver development

Directly developing of the ZigBee driver on the slotcars has one disadvantage: it's difficult to debug. There are several reasons for this:

- The electronics are custom-made, there are no guarantees that it will work correctly. Therefore, it's hard to tell if the problem is in the code or in the hardware.
- Constantly soldering and unsoldering of debug wires can have a bad effect on the other working parts.
- During the configuration phase (before the device starts in the network) the communication between ARM and CC2530 processors can be seen only on the oscilloscope.

Hence the evaluation modules were used as a prototype model and as a study platform for ZigBee solution and configuration. It was used as a base station during the project development as well.

CC2530 ZigBee Development Kit

The CC2530ZDK is a compliant System-on-Chip solution based on the CC2530 system-on-chip (SoC), and contains all hardware, software, tools, and documentation necessary to build a ZigBee compliant product [10].

The SmartRF05EB (evaluation board) is the main board in the kit with a wide range of user interfaces. The EB is the platform for the evaluation modules (EM) and can be connected to the PC via USB to control the evaluation module. The CC2530EM (evaluation module) contains the RF IC, necessary external components and matching filters for getting the most out of the radio. The module can be plugged into the SmartRF05EB. The EM is used as a reference design for a RF layout.
The CC2531 USB Dongle is a fully operational USB device that can be plugged into a PC. The dongle comes preprogrammed to work as a Packet Sniffer. It can be used together with SmartRF Packet Sniffer application [12]. This application was used during the software development to see data exchange between slotcars.
To program the ZNP image (hex-file) with the SmartRF Flash Programmer [13] onto

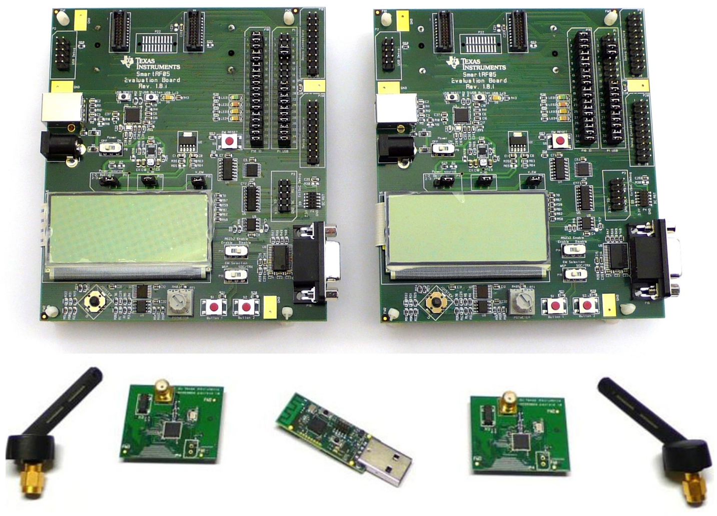


Figure 5. CC2530 ZigBee Development Kit (Source: [11])

a CC2530EM, the EM needs to be plugged directly into one SmartRF05EB board. The hex file used in this project is placed in `TexasInstruments\Z-StackHome1.2.0\Projects\zstack\ZAP\ZNP-HexFiles\CC2530ZNP-Pro.hex`. The detailed instruction, how to connect and program devices, is placed in "Z-Stack User's Guide For CC2530 ZigBee-PRO Network Processor Sample Applications" [11].

STM32F401C-DISCO

The CC2530EM interfaces through an SPI to the same ARM processor as placed on the main slotcar board. The ARM processor is placed on the STM32F401C-DISCO board. It's a very suitable solution for the project start. It's possible to connect 4 evaluation boards per SPI and 2 modules are enough to create a network and test the communication between them.

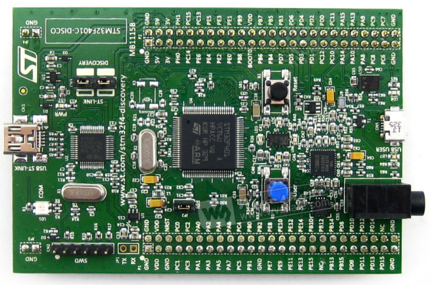


Figure 6. STM32F401C-DISCO (Source: http://www.wvshare.com/)

The development was started with a STM32F407-DISCOVERY board. Later, DISCOVERY board was replaced with a DISCO board. The main difference: the CPU clock speed must be reduced from 168 MHz to 84 MHz, more information in

Table 15. A user manual for Discovery kit for STM32F401 line [14] is available on the STMicroelectronics website. The ST-LINK/V2 programming and debugging tool is integrated on the STM32F401 DISCO board. The board connects to the computer via USB cable type A to mini-B. The ST-LINK/V2 USB driver must be installed on the computer. For upgrading or downgrading the software on a board ST-LinkUpgrade software is used. Usually it's placed in the `STMicroelectronics\STM32ST-LINKUtility\ST-LINKUtility` folder. This project uses the Keil MDK Version 5 software as a development environment. MDK contains all development tools including IDE, Compiler, Flash Programmer and Debugger. The firmware version on ST-Link must be compatible with Keil MDK version. The corresponding firmware is placed in the `Keil_v5\ARM\STLink\ST-LinkUpgrade`. The STM32 firmware used in this project has version V2.J20.

3 ZigBee standard specification

The vehicle communication standard should fulfil the following criteria:

- reliability and robustness,
- low cost,
- low power consumption,
- easy implementation,
- ubiquity and public availability,
- large network support,
- possibility to regulate data flow,
- mesh topology.

Slotcars were equipped with the nRF24L01 module by Nordic Semiconductor. It's a cheap, ultra low power 2.4 GHz ISM band wireless solution device. However, in practice it was found to be difficult to configure: instead of the expected 20 to 30 meters, the range under 0.5 meter was gained. The Nordic solution isn't a well-known standard and it led to implementation problems.

The cheap Bluetooth module with data range 723 kbit/s is a good solution for slotcar to base station communication, but doesn't support other topologies. It can have only eight nodes, which isn't sufficient for future platform development.

ZigBee is the most expensive solution and the slowest one (250 kbit/s) of the aforementioned, but it has the biggest range (to 300 m), it supports 65000 nodes and uses different topologies such as mesh, star and tree 7.

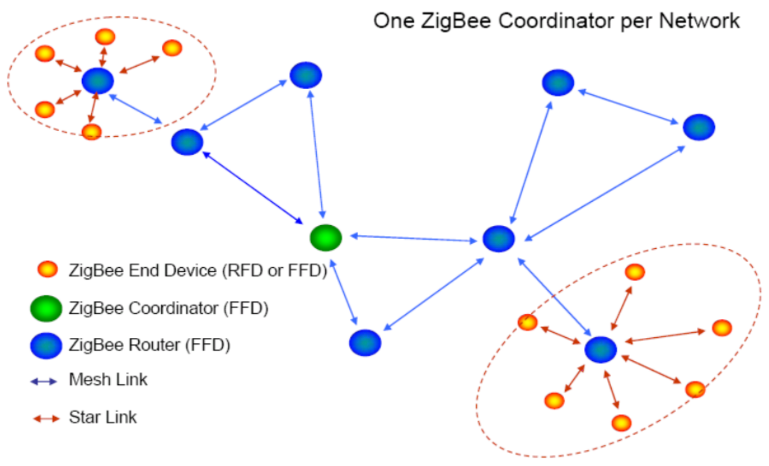

One ZigBee Coordinator per Network
ZigBee End Device (RFD or FFD)
ZigBee Coordinator (FFD)
ZigBee Router (FFD)
Mesh Link
Star Link

Figure 7. ZigBee Network Model (Source: measure.feld.cvut.cz)

ZigBee is a reliable and flexible standard. It defines a set of communication protocols for low-data-rate short-range wireless networking [15]. ZigBee was developed by the ZigBee Alliance. This technology is used in commercial building and home automation, security and healthcare. Such networks have become the widely adopted wireless sensor network standard. It offers a possibility to use it in the real time systems, although it's not its main purpose.

ZigBee is based on an IEEE 802.15.4 standard, a part of the IEEE family of standards for physical and link layers. It offers a good coexistence among 2.4GHz RF products (Bluetooth, Wireless USB, Wi-Fi) and low bit error rate. ZigBee has very good anti-interference performance, however it's important to choose the right frequency in case of Wi-Fi operating within a very short range (2 meters). More information can be found in the Atmel AT02845 application note [16]. To reduce channel load ZigBee standard allows the ACK mechanism to be disabled. So there is more 'free space' for transmission and it reduces waiting time before the node can send the data. It's very useful in case of the real time control, where timeliness may be more important than reliability. In other words, we prefer to lose some data rather than to resend them and cause delay in the data flow.
ZigBee offers basically four kinds of different services:

1. association and authentication (to allow valid nodes to join to the network),
2. extra encryption services [17],
3. routing protocol [18],
4. application services, e.g., binding service for linking devices that are related.

We will concentrate on the first one. Later in this chapter the base characteristics of the IEEE 802.15.4 standard and ZigBee network processor configuration will be described. The process of the network creation, joining and data exchange will also be demonstrated.

3.1 The ZigBee base: IEEE 802.15.4

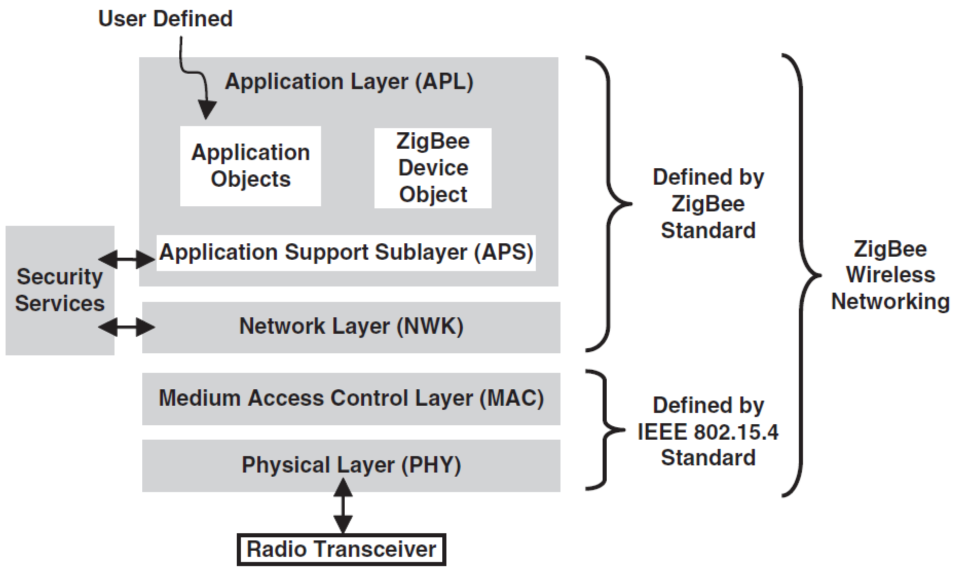

User Defined
Application Layer (APL)
Application Objects
ZigBee Device Object
Application Support Sublayer (APS)
Security Services
Network Layer (NWK)
Medium Access Control Layer (MAC)
Physical Layer (PHY)
Radio Transceiver
Defined by ZigBee Standard
Defined by IEEE 802.15.4 Standard
ZigBee Wireless Networking

Figure 8. ZigBee Wireless Networking Protocol Layers (Source: [15], p.[5])

The IEEE 802.15.4 is a standard which specifies the physical layer and media access control (MAC) for low-rate wireless personal area networks [19]. It provides compatible interconnection for wireless data communication and is designed for low cost, low data rate and low power devices.
The IEEE 802.15.4 operates in unlicensed radio frequency (RF) bands: the 868 MHz band in Europe, the 915 MHz band in North America, Australia and the 2.4 GHz band worldwide.

The PHY layer is responsible for activating and deactivating the radio transceiver, transmitting and receiving data and selecting the channel frequency. When a device wants to transmit it performs two functions: Energy Detection(ED) and Clear Channel Assessment(CCA). It scans the medium and reports if the channel is not in use by any other device.
802.15.4 uses two techniques to avoid data collision: CSMA-CA and GTS. The Carrier Sense Multiple Access - Collision Avoidance is the most common: each node listens to the medium. If another node transmits, other nodes wait for a random time. The Guarantee Time Slots (GTS) uses a centralized node which gives slots of transmission time to each node. For better coexistence DSSS (Direct Sequence Spread Spectrum) is used. It extends the transmitted signal over the larger bandwidth than is needed. The resulting signal appears as a noise signal and shows greater resistance to interference [20].
The MAC provides the interface between the PHY and the next higher layer above the MAC - NWK layer, generates beacons, synchronizes the device to the beacon, employs the CSMA-CA for channel access, manages GTS channel access, provides personal area network (PAN) association and disassociation services and support for security. 16-bit Frame Check Sequence, based on the Cyclic Redundancy Check (CRC), is used in the IEEE802.15.4 to detect if any of bits were received incorrectly.
Two types of physical devices are provided in the IEEE 802.15.4 wireless network: full-function devices(FFDs) and reduced-function devices (RFDs). FFD Devices can perform all available operations within the standard, such as coordination tasks, sending tasks and routing mechanism. RFD Devices have limited capabilities and can talk only with an FFD device.
The Network layer (NWK) and Application layer (APL) layers are defined by the ZigBee standard and will be described later.

3.2 Interaction between the ZNP and the application processor

The communication between the CC2530 ZigBee network processor and the application processor (the main slotcar processor) is provided via SPI. The slotcar processor runs the application code, which uses CC2530 ZigBee Network Processor (ZNP) Application Program Interface. The CC2530 ZigBee Soc runs full Z-Stack solution (ZigBee Pro) and provides connectivity with the IEEE 802.15.4 Radio. Figure 9 shows the interaction process between the ARM processor and the network processor.

3.2.1 Initializing SPI

The first thing that should be done is SPI initialization. The CC2530-ZNP pin configuration is described in CC2530ZNP Interface Specification [21]. The SPI Interface must be configured with following parameters:

- SPI master,
- Bit order MSB first,
- Clock speed must be greater than 500kHz and less than 4 MHz (we set SPI frequency to 1,35 MHz),
- Clock polarity 0 and clock phase 0 on CC2530.

All aforementioned parameters must be set during SPI initialization. It’s necessary to assign pins explicitly to the SPI interface, enable all the clocks and the SPI Interface.

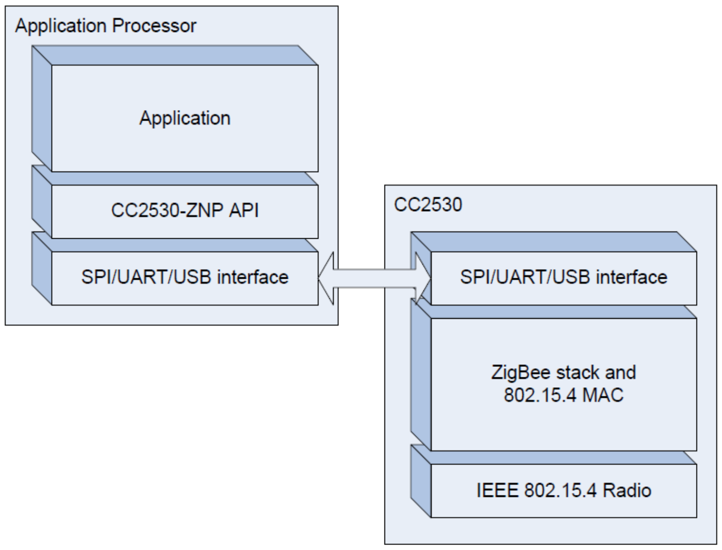


Figure 9. Interaction between the AP and the ZNP (Source: [11])

For communication are required RESET, CFG0, CFG1, MISO, MOSI, CS, SCK, SRDY and MRDY signals:

- RESET is used to hard reset the module.
- CFG0 indicates if there is a 32kHz crystal connected to the CC2530-ZNP (in this project the CFG0 pin is set low and the CC2530 uses the internal RC oscillator).
- with CFG1 the user can choose UART or SPI transport mode. To use SPI transport mode, the CFG1 pin should be set high).
- MISO, MOSI, SCK and CS are the standard SPI signals. The master role has the application processor and the slave role has the network processor. So the Chip Select signal should be low. MOSI is the master line for sending data to the slave and MISO is the slave line for sending data to the master.
- MRDY is Master ReaDY, an active low signal. It is set by the application processor when it has the data to send.
- SRDY is Slave ReaDY, a bi-modal signal. It indicates if the network processor is ready to receive or send data.

Since the program is written for the application processor, the communication process will be described from the AP side of view. The protocol scenarios are as follows:

- POLL - The network processor(CC2530-ZNP) requests to start the communication.
- SREQ - The application processor sends a command to the network processor and waits for SRSP frame (synchronous response).

To indicate the right protocol scenario we need to follow MRDY and SRDY signals. If the SRDY level is changed before the MRDY level, the network processor wants to transmit to the application processor data (POLL command). If the AP has the data for the network processor, then the MRDY level is changed first (SREQ command). After both processors show that they are ready, the application processor starts to transmit data. If it has no data to send (the data request is initialized by the network processor - POLL command), the application processor transmits zeros. The AP waits for the ZNP ready status (SRDY high) and starts to receive data. Each data unit has a Command field. It helps to determine the purpose of the following payload. It can be the response to the command, e.g. the application processor sends the request to establish the network, the ZNP responds with successful or unsuccessful status. Or it could be received data from another node. Figure 10 shows the general frame and the command field formats. The Type field is 0 for POLL, 1 for SREQ, 2 for AREQ and 3 for SRSP commands. The ID field maps to a particular interface message. The Subsystem Values and Names are placed in Table 19 in Appendix B.

Bytes: 1	2	0-250
Length	Command	Data

(a)

Bits: 7-5	Cmd0 4-0	Cmd1 7-0
Type	Subsystem	ID

(b)

Figure 10. (a) General frame format (b) Command Field (Source: [21])

STM32F4 External interrupts

To process the incoming data we need to configure the SRDY pin to be an interrupt. STM32F4 has 7 interrupt handlers for GPIO pins: handler for pins connected to line 0, ..., 4, 5-9, 10-15. For PC1(SRDY) pin EXTI1_IRQHandler is used. It's possible to configure if a falling or rising edge will be detected. In this project a falling edge detection is used. To distinguish SREQ and POLL commands the MRDY pin value is checked also. At the interrupt moment it's low for SREQ command (the data need to be sent) and high for POLL command (the data need to be received).

3.3 CC2530-ZNP power-up procedure

In this section the power-up process of the CC2530-ZNP is described. We followed the recommended approach as specified in the CC2530ZNP Interface Specification [21]. All below mentioned commands are placed in the Appendix C:

1. The application processor and CC2530 power up.
2. The application processor initializes its SPI interface and configure EXTI Line1(SRDY) and disables interrupt.
3. The application processor sets CC2530 RESET pin low, holding CC2530 in reset.
4. The application processor sets the CC2530 CFG0 low (internal RC oscillator) and CFG1 high (SPI transport mode).
5. The application processor sets CC2530 RESET pin high and CC2530 starts operation.
6. The SYS_RESET_IND (Table 22) message is sent by the CC2530 network processor to the application processor using the POLL command. This command ensures the proper reset of the radio and brings the radio into configuration mode. The ZNP response includes a reason value which indicates why the network processor was reset. It can be one of the following values:
   - power-up,
   - external (the RESET signal was driven low for a time greater than 200 ns),
   - watch-dog (a hardware fault or program error).

Because we hold CC2530 in reset by setting the RESET pin low, the ZNP will return the external reset reason.

7. The application processor sends the ZCD_NV_STARTUP_OPTION command (Table 23) to overwrite all the configuration parameters. It's the SREQ command. The parameter 0x00 prevents network state loss or erase of the existing configuration caused by an accidental device reset. If the network parameters need to be changed, the value of the parameter must be 0x02.
   The response form is in Table 24. All response packets contain the status value. The status value 0x00 indicates that the command was successfully executed. The 0x01 value indicates ZFailure. Other values of status parameters are described in the 4.7 chapter in CC2530ZNP Interface Specification[21].
8. Application processor sets CC2530 RESET_N pin low, holding CC2530 in reset. Modification of ZCD_NV_STARTUP_OPTION parameter requires a CC2530-ZNP device reset before it can take effect, hence the second reset is required.
9. The application processor sets CC2530 RESET_N pin high and CC2530 starts operation.
10. The CC2530 response with SYS_RESET_IND.

During the development phase it was discovered that it's necessary to have a delay after changing the RESET_N state (approximately 20 ms). Without a delay the network processor behaves erratically, e.g. not responding to SREQ command or it sends a failure response. The project uses the delay function from STM32 examples. It's not properly calibrated, so the delay time can differ for other MCU types.

3.4 CC2530-ZNP startup procedure

After the power-up procedure, some mandatory commands should be executed to configure the ZigBee device:

1. **Logical type**. The application processor sends ZCD_NV_LOGICAL_TYPE SREQ command (Table 27) to configure the device role to the CC2530-ZNP device. There are three types of logical devices in a ZigBee network: coordinator (ZC), router (ZR) and end device (ZED):
   - A ZC is the main device, it scans the RF environment, chooses a channel and a network identifier (PAN ID), forms the network and might bridge to other networks. There can't be two coordinators in the same network.
   - The ZR device passes data from other devices and allows child nodes to connect to it. The ZC and ZR devices have the radio always on.
   - The ZED talks to its parent node, it can't relay data from other devices. These devices have a sleep mode in order to conserve power.

   The value parameter can be set 0x00 for coordinator, 0x01 for router and 0x03 for end device.
2. **Channel**. The application processor sends ZCD_NV_CHANLIST SREQ command (Table 25) to configure a physical channel of the RF spectrum. There are 16 available channels. Although as was previously mentioned the IEEE 802.15.4 adopts many mechanisms for coexisting with other 2.4GHz radio products, we can adopt additional measures. Figure 11 shows that channels 15, 20, 25 and 26 are not overlapped with Wi-Fi channels so it's safe to use them. For slotcar communication we will need only one channel, so we can choose the most suitable one, viz. 15. The Coordinator can choose multiple channels from the aforementioned by multiple

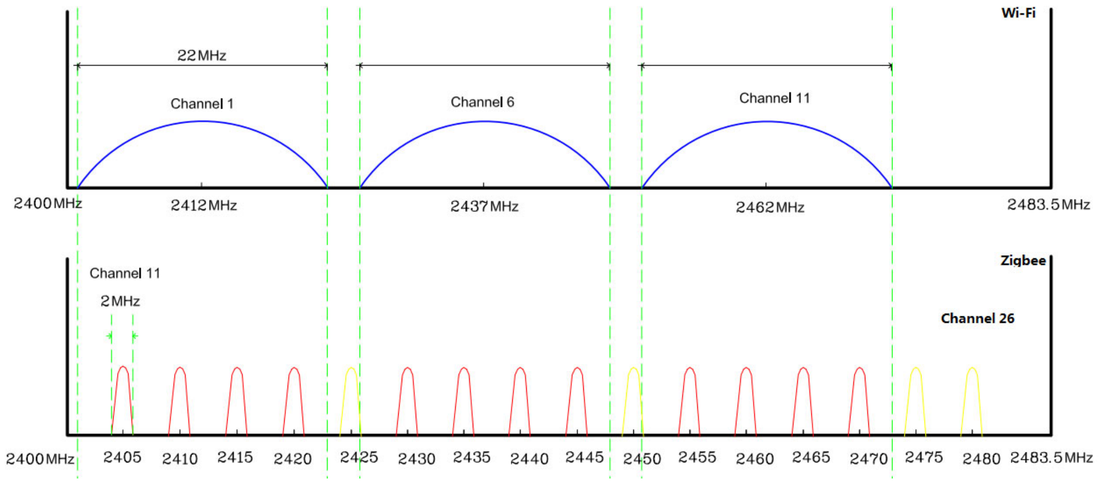


Figure 11. Wi-Fi and ZigBee Overlapping Channels in 2.4GHz ISM Band. [16], p.[7]

calls of the channel configuration command, perform energy scan and establish the network on it.

3. **PAN ID**. The application processor sends ZCD_NV_PANID SREQ command (Table 29) to configure the address of a network within a channel. Personal Area Network Identifier (PAN ID) was implemented for the logical separation of ZigBee nodes located in the same physical channel or vicinity. ZigBee PAN IDs are 16-bit numbers that range from 0x0000 to 0x3FFF. The 0xFFFF PAN ID specifies that the application requests the ZigBee stack to obtain the identifier dynamically by detecting other networks operating in the same frequency channel and choosing a PAN ID that does not conflict with theirs. However we register only one channel (15) and non-default PAN ID (0x0003) for the Coordinator and the same for Routers to prevent the joining to the wrong ZigBee network operating on the same channel or with the same PAN ID.
4. **Simple descriptor**. The highest protocol layer in the ZigBee wireless network, the APL layer, hosts the application objects [**p.[22]**, 15]. Application objects are developed to control and manage protocol layers in a ZigBee device. There can be 240 application objects in a single device (Figure 12). Each application object has a unique endpoint address (endpoint 1 to endpoint 240). The ZigBee Device Object (ZDO) endpoint address is zero. Its function is to be the network manager.

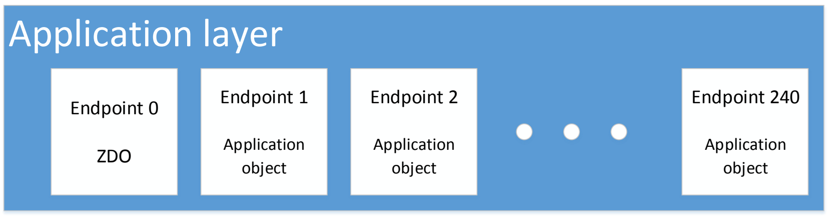


Figure 12. The Application Objects on the APL layer

The endpoints are the main interface between the user application and the stack. The user application can manage the network by making requests and handling callbacks to this object. Hence the software should be the same for all slotcars

and we don't need to separate received information for different applications, it's very important to configure the same endpoint to prevent the confusion in message addressing.
Figure 13 shows the Application object structure. The obligatory part of the Application object build up is to configure the simple descriptor parameters. The set of stack parameters that need to be configured to specific values, along with the above device type values, is called a stack profile [**p.**[**3**], 22]. It defines the network type and shape and the features that are available to applications, e.g. the types of security. All devices in a network must conform to the same stack profile (i.e., all devices must have the stack profile parameters configured to the same values). The ZigBee Alliance defined ZigBee and ZigBee PRO stack profiles. ZigBee Pro is used in this project.
The 'profile' shown in Figure 13 is an Application profile or a stack 'subprofile'. It's a set of agreements on application-specific message formats and processing actions, a fully compliant ZigBee solution. The ZigBee standard offers the option to use public application profiles in developing an application. For example, Z-Stack Home is TI's ZigBee Home Automation (ZHA) compliant protocol stack for the CC2530 and CC2538 System-on-Chip. It already has prepared functions for creating an application for smarter homes. The range of public application profiles is 0x0000 to 0x7fff. Slotcar uses a manufacturer-specific profile, 16-bit number with a range of 0xBF00 to 0xFFFF, viz. 0xBF00.

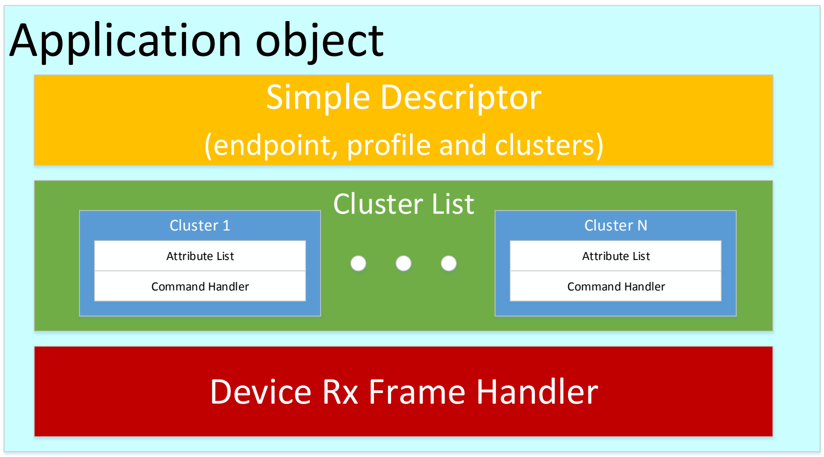


Figure 13. The Application Object structure (Source: [23])

Clusters registration is a part of the Application object configuration as well. A cluster is a list of attributes and a command handler. When the data come to the Application object, Device Rx Frame Handler function checks the incoming message cluster ID and calls the Command Handler corresponding to the received cluster ID. The command handler actions can be defined by the user or be given by chosen specification and depends on received attributes. There are ZigBee Cluster Libraries, e.g., lightning, general, measurement & sensing. Each Application Profile supports some of them.
We use the manufacturer-specific profile (own profile), so we can't use already prepared ZigBee Cluster Libraries. At this point it should be noted that slotcars communicate not only using ZigBee standard, but also Nordic, which will be replaced with Wi-Fi in the next model. Hence the network processor should be

responsible only for transmitting (addressing), receiving and proceeding data to the application processor. It means that we don't need cluster services.
All afore-mentioned parameters must be configured by calling the AF_REGISTER SREQ command (Table 31). We set the EndPoint parameter to 0x08, AppProfId to 0xBF00 and the number of Input and Output cluster Ids (AppInClusterList and AppOutClusterList) to 0x00.

5. **Device starts in the network.** Finally, the ZDO_STARTUP_FROM_APP SREQ command (Table 33) is sent by the application processor and starts the device in the network. StartDelay parameter specifies the time delay before the device starts in milliseconds. We didn't set any delay, so the slotcars start to look for the network immediately after the configuration. The response status value can be 0x00 for restored network state, 0x01 for new network state or 0x02 for leave and not started.

Network formation

After the user requested the device to start the CC2530 network processor will send to the application processor the ZDO_STATE_CHANGE_IND responses (Table 35). The ARM should send POLL commands, read responses and check the ZDO status. The status shows in which state a network configuration is. Figure 14 shows how the Coordinator forms the network.

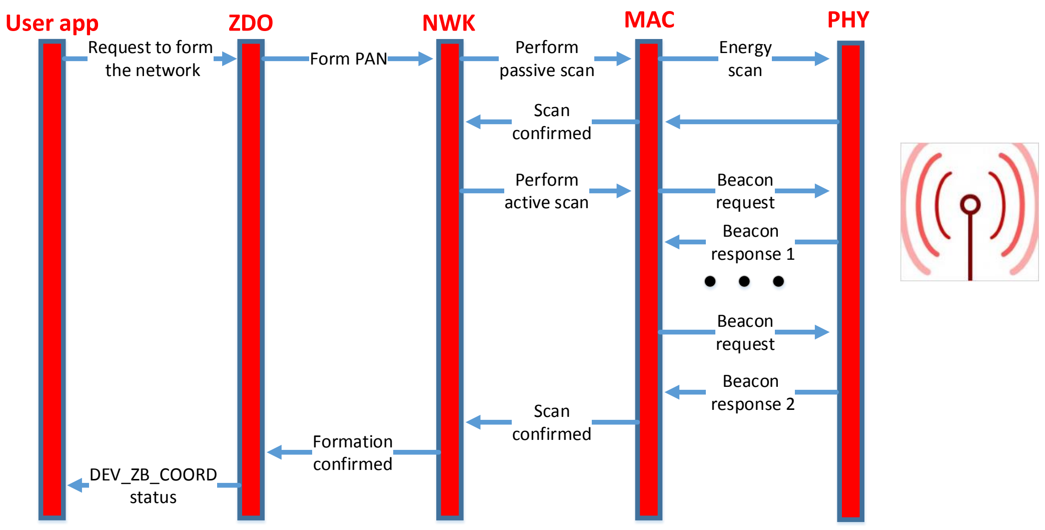


Figure 14. NWK formation (Source: [24])

The NWK layer is responsible for managing the network formation and routing. First, the ZDO calls the network formation function. The coordinator provides the energy scan and will choose from the channels with the lowest amount of traffic and the fewest networks. In our case the coordinator will start operate on the channel chosen by the user. Then the ZDO broadcasts a beacon request (a packet which contains information about the device PAN ID). All nearby ZigBee devices will send a beacon back. The PAN ID exchange is called the active scan. There can be only one coordinator in the network and only the coordinator can form a network, determine a unique PAN ID and a channel. When the network formation is finished, the ZDO returns the

DEV_ZB_COORD status. It means that the device has successfully started as a Coordinator. See the ZDO status table 20.

Network Discovery/Join

Figure 15 shows the joining process of the Router to the Coordinator.

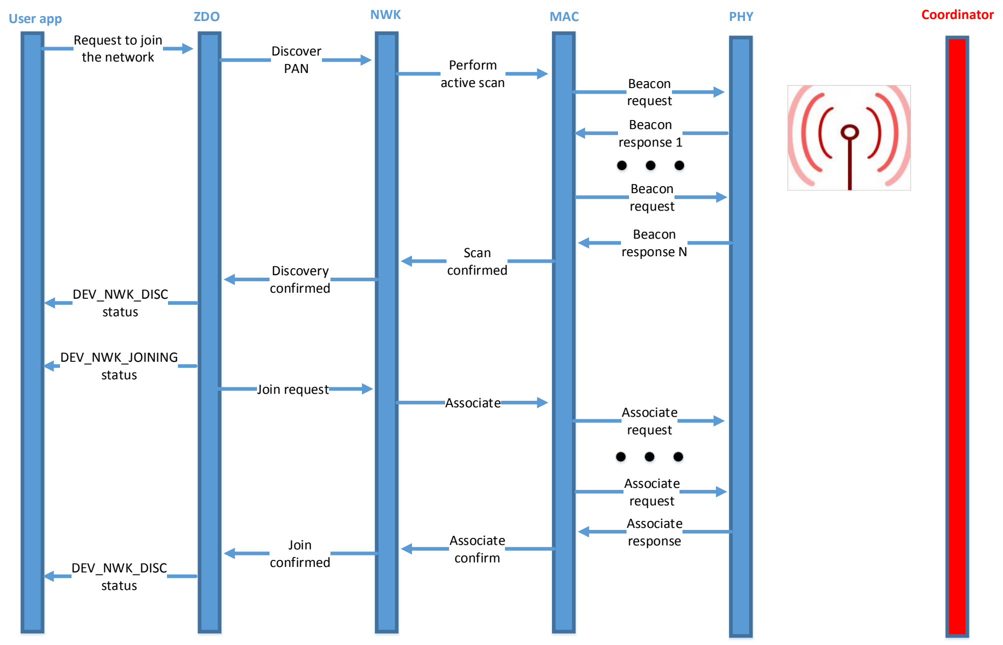


Figure 15. NWK Discovery/Join (Source: [24])

ZDO requests Network(PAN) discovery, the discovery function will call the MAC's active scan service and it will perform the Beacon request. When other devices see the beacon request, they will respond with an 802.15.4 beacon frame. It contains MAC information about the responding device and a beacon payload for generic data. Within that payload, the responding device will include ZigBee network information such as the protocol ID and version, amount of routers and end devices allowed to join, the device profile that is being used, and other somewhat useful information. ZDO returns DEV_NWK_DISC status (discovering PAN's to join).
From active scan information the router should choose the Coordinator (slotcars will join to the coordinator operating on the registered channel and with the same PAN ID). Two coordinators on the same channel and with the same PAN ID can't exist.
After ZDO sends DEV_NWK_JOINING (joining a PAN) to the user application and calls a network join request, it calls the MAC's association service and initiates an association request to the Coordinator. The response will pass up to the network layer via the MAC's association response. If the join request was successful, the device will update it's NWK and MAC information tables to include the new network address, PAN ID, and also update the neighbour table to specify its parent, the network join confirmation is proceeded to the ZDO. The user receives the DEV_ROUTER status (device joined, authenticated and is a router). After joining, the device will broadcast a device announcement which includes a 16-bit network address, a 64-bit IEEE address and the network status.

The Router joining process from the Coordinator side in shown in Figure 16. When

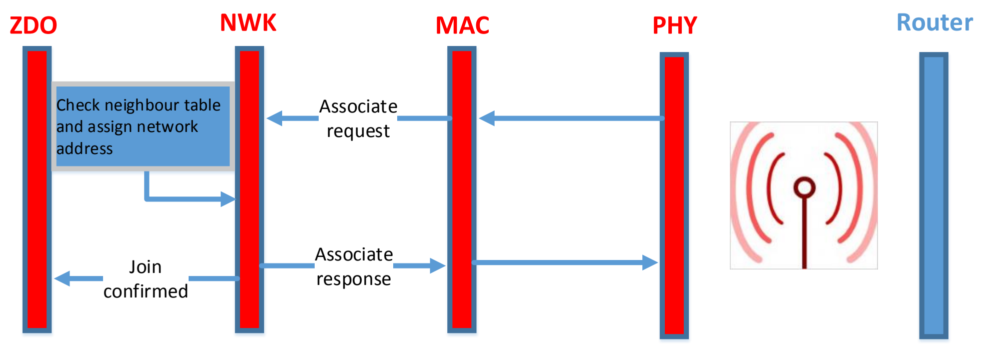


Figure 16. The parent side of the join process (Source: [24])

a MAC association request arrives to the potential parent, it sends an indication to the network layer that a device is trying to join. The potential parent will then look through its neighbour table to see if the 64-bit IEEE address of the joining device already exists. If not, the parent allows devices to join it and adds the device to the neighbour table, generating a new 16-bit network address for it. Then this information is sent out as a MAC associate response. If it exists, then the parent sends the same network address, which was generated, when requesting device was asked to join the first time.

Data Transfer

After the power-up and startup procedures, devices can start to transmit and receive data. When the interrupt on SRDY line is received, we check the MRDY level to assure that it's a POLL command. The user application sends zero payload and receives from the network processor AF_INCOMING_MSG response (Table 38). This response contains received data. The data field can contain 0 to 99 bytes of data without any security.
The Application processor uses AF_DATA_REQUEST SREQ command (Table 36) to build and send a message through the AF layer. Besides the address and the destination endpoint the user can set options such as bypass routing, request APS acknowledgement for this packet, etc. The options field is organized as a bitmask. A value of 0x80 means that bit 7 is set and routing will be bypassed for the packet, a value 0x10 sets bit 4 to request APS acknowledgement for the packet. Other options can be found in CC2530-ZNP Interface Specification [21]. The Radius value specifies the number of hops allowed delivering the message and can be configured as well. We will use this ZigBee feature to reduce message retransmission when we will need to increase the information transmission speed. In case of important information exchange the radius value will be increased. For example, slotcar address information or initial parameters sent by the user to the vehicles.

In the ZigBee Pro stack profile network with mesh topology, only the CSMA-CA technique is used to avoid data collision. The device (router or coordinator) transmits once the channel is clear. The data exchange is shown in Figure 17. The frame format of the Beacon as standardized in the IEEE 802.15.4 specification includes GTS field, but in the context of ZigBee this field is 0. It means that the Coordinator doesn't permit GTS requests.

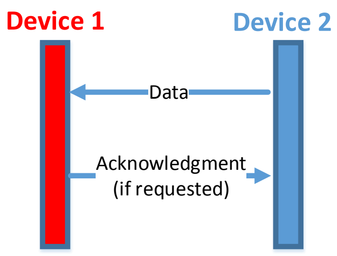


Figure 17. Non-Beacon Data Transfer

The data path in NWK layer

Two diagrams in this section explain how data is proceeded in the Network Layer. Data to be transmitted flows down from an upper layer. Received data come up from the lower layer or do a U-turn if they should be retransmitted. All slotcars and the base station are in range of each other, that means that they all have each other's address in the neighbour table. The data flow is restricted to the highlighted area.

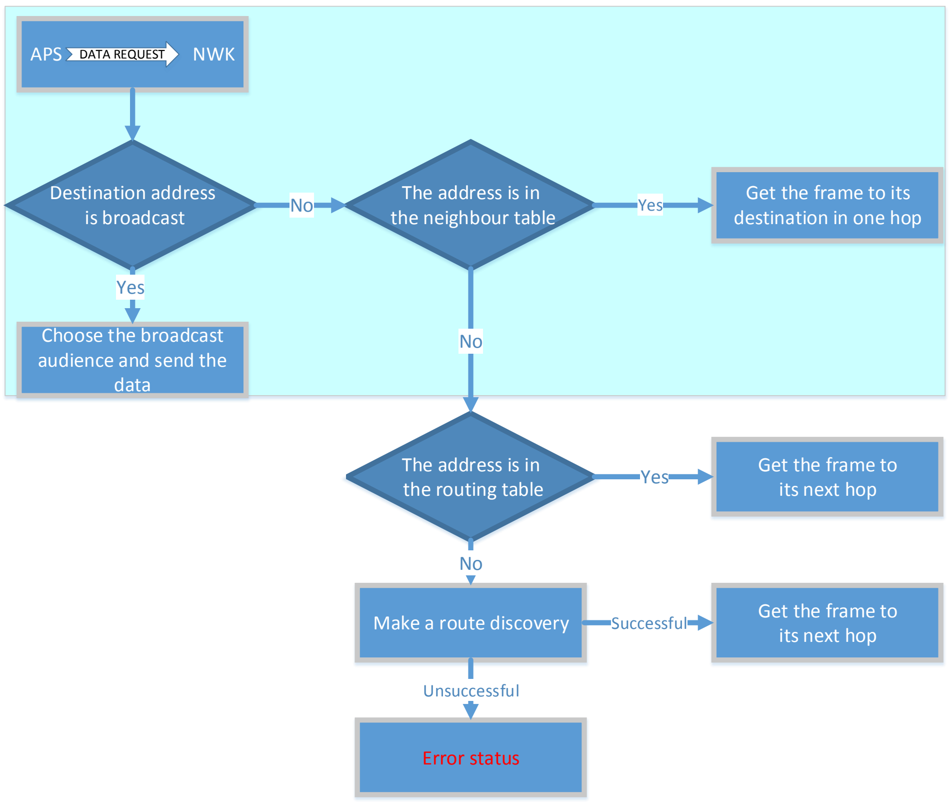

APS
DATA REQUEST
NWK
Destination address is broadcast
No
The address is in the neighbour table
Yes
Get the frame to its destination in one hop
Yes
Choose the broadcast audience and send the data
No
The address is in the routing table
Yes
Get the frame to its next hop
No
Make a route discovery
Successful
Get the frame to its next hop
Unsuccessful
Error status

Figure 18. Data Transmission on NWK Layer

When a frame is received, it's stored and the indication is sent to the MAC layer. The MAC layer takes the frame up, decodes the MAC header and, if it's a MAC data frame passes it up to the NWK layer using the Data Indication service.

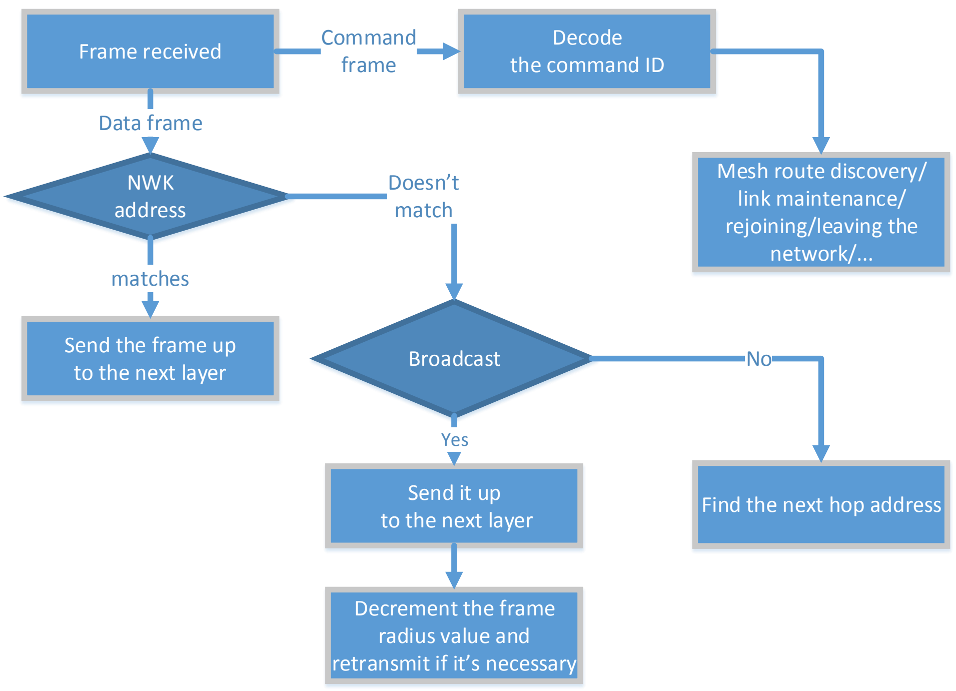

Frame received
Command frame
Decode the command ID
Mesh route discovery/ link maintenance/ rejoining/leaving the network/...
Data frame
NWK address
Doesn't match
matches
Send the frame up to the next layer
Broadcast
No
Yes
Send it up to the next layer
Find the next hop address
Decrement the frame radius value and retransmit if it's necessary

Figure 19. Data Reception on NWK Layer

3.4.1 ZigBee addressing

Each device in a network has a unique address. The IEEE 802.15.4 uses two methods of addressing: 16-bit short addressing and 64-bit extended addressing. It's possible to use either 16-bit or 64-bit addressing. The short address allows communication within a single network. Using the short address reduces the length of the packet and saves memory, which is required for storing an address. ZigBee 2007 Pro and newer versions use "Stochastic Addressing", that means that addresses are randomly chosen and assigned to joining devices by the coordinator. The coordinator address is always 0x0000 and the broadcast address is 0xFFFF. Other special addresses could be found in Table 17 in the Appendix B. To find out the short address device we need the ZB_GET_DEVICE_INFO command (Table 39). It is a SREQ operation. This command retrieves a device information. The Param variable is the identifier of the device information. If a 0x02 parameter is used, CC2530 returns its short address. See the full list of parameters in Table 21. The value always has 8 bytes in length even though the actual value may be smaller in size. The remaining bytes can take any value. ZB_GET_DEVICE_INFO command is called after the Router returned DEV_ROUTER status.

3.5 Summary

The whole process of power-up and startup procedure of Coordinator and Router and the entire communication between devices is shown in Figure 20. The short CC2530-ZNP configuration instruction is placed in the Appendix C and includes all necessary packets.

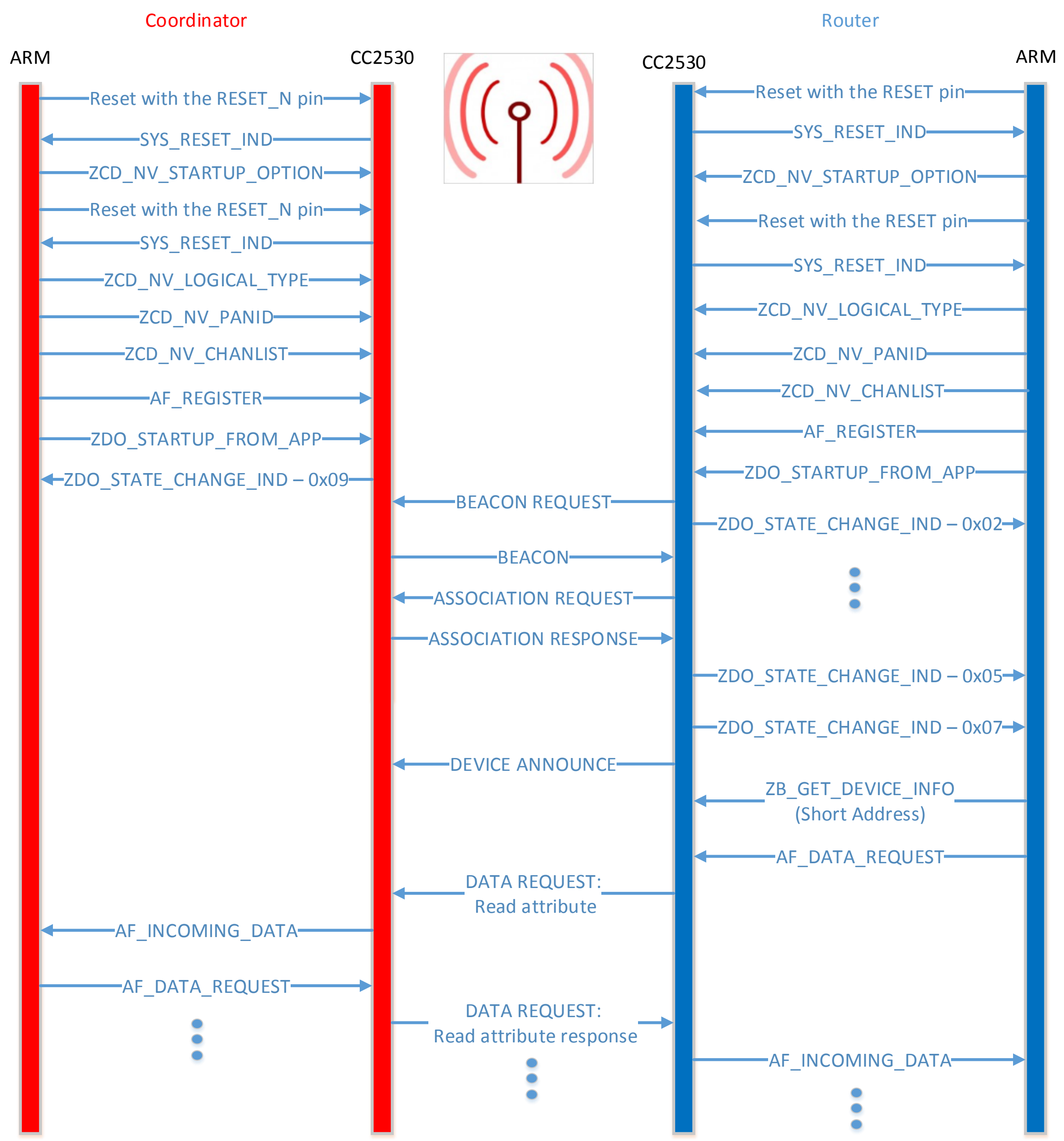


Figure 20. Power-up, startup and communication process between Coordinator and Router

4 Communication infrastructure

The unified communication stack was created by Jaromír Dvořák and Martin Lád. It is independent on the particular hardware implementation (ZigBee, Nordic, . . .). The unified communication stack allows implementation of new communication drivers. It also allows the creation of new services in the main program layer. Figure 21 shows the data passing structure.

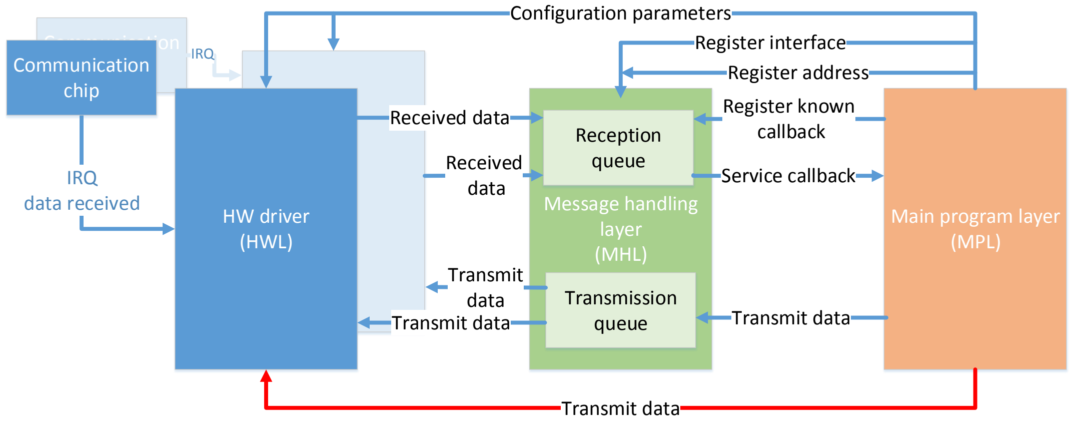


Figure 21. Data passing structure

Each layer provides services for the next layer. The main program layer (MPL) finishes the processing of the data, does the system setup and provides callbacks for different message types. The MPL layer can give the transmit data command direct to the HW layer. The Message handling layer (MHL) stores data in queues and calls service functions corresponding to the received message type. The MHL calls the HW layer transmit function when there are data in transmission queue. The HW layer receives and transmits data and forwards received data to the MHL layer. One of the goals of this project is to implement the HW ZigBee driver. The first section 4.1 describes the main ZigBee HW layer functions.

4.1 The ZNP driver implementation

The HW ZNP driver implements the following functions:

- **zigbee_conf**(**zigbee_settings** settings) - this function includes the SPI initialization, interrupt configuration and CC2530-ZNP power-up and start-up procedures. The `zigbee\_conf` is called in the main function. As an input it requires the `zigbee\_settings` structure with following variables:
  - device logical type,
  - ZigBee logical address,
  - channel (15),
  - PAN ID (0x0003),

– pointer to the function that should be called when the HW layer has data for MHL layer.

Slotcars are programmed as Routers to have the possibility to directly communicate with each other and the base station. The CC2530 evolution model connected to the SmartRF05EB board has a Coordinator role.
Many address types were already mentioned. Here is a quick summary:

– **Logical address or slotcar number**. The current platform consists of 10 slotcars. As previously mentioned, every slotcar has its number with a range of 1 to 10. Three of cars have the CC2530 communication chip implemented: 5, 6 and 10.

– **ZigBee short address**. It's the 16-bit short address assigned to every Router (slotcar) by the Coordinator (base station). This address type was described in the chapter 3.4.1.

– **ZigBee logical address**. It's the number in range of 101 (0x65) to 110 (0x6E). 99 (0x63) is for the Coordinator and 100 (0x64) is for broadcasting. To determine the slotcar ZigBee logical address to the car number 100 is added (car number + 100).

In the previous chapter 3.3 were described more configurable parameters than in the `zigbee_settings` structure. Every ZNP initialization and configuration step has its own function. These functions are placed in the `zigbee\_conf`. The parameters such as the application endpoint or the application profile could be changed there.

- **zigbee_transmit_data**(**Packet** packet) - the purpose of this function is to build and send the data through AF layer. The Packet structure is shown in Figure 22.

Bytes: 1	1	1	1	0 - 94
message type	payload length	destination address	source address	payload

Figure 22. Packet structure

The destination and source addresses help to distinguish the destination or the source HW layer. The ZigBee HWL will be called for data transmission if the destination address value will be equal or more than 99. The same for the received data and the source address. The message type determines what function must be called. In the main function it is possible to register known callbacks. The `zigbee\_transmit\_data` transmits data regardless of the `msg\_type` value.

- **ZIGBEE_ISR**() - this is the `EXTI15\_10\_IRQHandler` interrupt service routine handler for pins connected to lines 10 to 15. It manages the data reception and passes payload (the data field) to the `zigbee\_received\_data` function.

- **zigbee_received_data**(**int*** received_payload) - this function checks the first `received\_payload` byte, which contains the message type. If it's registered, the function fills the Packet structure and forwards it to the MHL layer by calling the `zigbee\_callback`.

4.2 Main program layer services

After all necessary configurations, the program will enter the main event loop. Each program cycle the following functions are performed:

- data transmission from the transmission queue,
- data processing from the reception queue,
- measurements updating,
- automaton step execution (Figure 23).

Only the first two automaton steps are executed automatically. Then the slotcar enters the ‘waiting state’ and transition to other states can be done by sending a command with the defined message type.

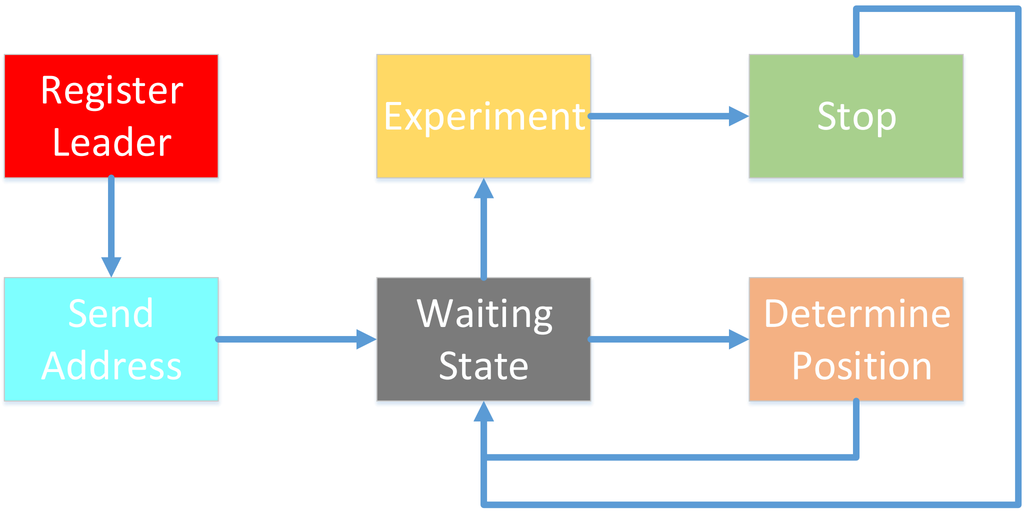


Figure 23. Main cycle state automaton

Register leader

To control the car platoon wirelessly it’s necessary to know the leader vehicle. For the next 10000 cycles (each cycle lasts approximately 1 ms) the slotcar waits for the front distance value. The distance sensor may take several seconds to settle to its final output value. Then it compares this value with the maximum distance (30 cm) and based on the result determines if it’s in front or not. Despite of the result the program goes to the next state.

Send address

At this step the slotcar announces itself by broadcasting its ZigBee short address and ZigBee logical address. The leader slotcar broadcasts its position number also. It’s necessary to bind the car number with its short address. Hence the address table is implemented. When the short address is assigned, the device adds it to the address table on the (ZigBee logical address - 100) position and then broadcasts it with the message type 0x3C. When the packet with this message type is received, the device adds the received logical and short addresses to the address table and rebroadcasts its own address with the message type 0x3D. When the packet with 0x3D is received, the device adds received addresses to the address table without rebroadcasting its own. Figure 24 shows the described algorithm.

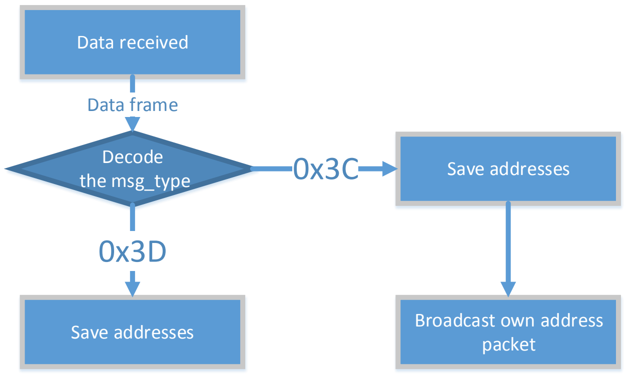


Figure 24. Addresses exchange

Table 1. Address packet

msg_type	length	dest_addr	source_addr	Payload: 0
0x3C/0x3D	0x05	0x64	MY_ADDR + 100	1 (ACK)

1	2	3	4
short_addr LSB	short_ addr MSB	ZigBee log_addr	if leader

To ensure that the address packet will be received by all active slotcars and the base station, acknowledgement is required. The payload[0] indicates if the ACK is on (1) or off (0).

Slot-car parameters configuration

After the slotcar has announced itself in the network and has entered the waiting state, the base station sends the 'get parameter' packet (Table 3) to determine slotcar default parameters:

number	parameter's name	range	data type
1	regulation type	non, speed, distance	int
2	reference PWM duty cycle	<0,1>	float
3	reference speed [mm/s]	<0,1000>	int
4	reference front distance [mm]	<50,300>	int
5	measured speed [mm/s]	<0,1000>	int
6	speed reg proportional term	<0, inf>	double
7	speed reg integral term	<0, inf>	double
8	distance reg proportional term	<0, inf>	double
9	distance reg integral term	<0, inf>	double
10	distance weight	<0, inf>	float

Table 2. Slotcar's parameters

- reg = NON. No regulator will be used, the speed of car can be set by PWM value.
- reg = SPEED. Speed regulator will be used, the speed of car can be set by reference speed value.
- reg = DISTANCE. Distance regulator will be used, the distances can be set by reference distance value.

The measured speed value is read out for control. Before the experiment start it should be 0.

Table 3. Get parameters packet

msg_type	length	dest_addr	source_addr
0x47	0x00	ZigBee log_addr	0x63

Table 4. Get parameters packet response

msg_type	length	dest_addr	source_addr	Payload: 0
0x48	0x39	0x63	MY_ADDR + 100	1 (ACK)

1 - 4	5 - 8	9 - 12	13 - 16	17 - 20	21 - 28
reg	pwm	ref_speed	ref_distance	meas_speed	reg_speed_kp

29 - 36	37 - 44	45 - 52	53 - 56
reg_speed_ki	distance_kp	distance_ki	distance_weight

The default parameters can be changed by sending the ‘set parameter’ packet (Table 5). Each parameter has a corresponding number (Table 2). The parameter value follows its number. The user chooses the number of parameters that he wants to set and the destination slotcar. The parameters readout or change can be done during the ‘waiting’ and the ‘experiment’ states.

Table 5. Set parameters packet

msg_type	length	dest_addr	source_addr	Payload: 0
0x46	0x00 - 0x44	ZigBee log_addr	0x63	1 (ACK)

1	2	3 - ...	... (1 byte)	... (n bytes)	...
count	param_number	value	param_number	value	...

Position determination

Except for the leader slotcar, the rest of the platoon doesn’t know its position. By sending the ‘determine position’ command (Figure 6) the user will obtain packets with position information from each car (Table 7). When the leader receives this command, it broadcasts the ‘position’ packet (Table 7) and checks if there is another car behind. If not, it will broadcast the ’determine position finished’ packet (Table 8). Regardless the result the car will change its position by setting the pwm duty cycle to 0,5 value for next 8000 program cycles (it will travel approximately 600 mm). Other vehicles after receiving 'determine position' command save the initial front distance value and start to compare it with the updated value each program cycle. The distance change that could be reliably detected by the car is 50 mm. The vehicle, which detects the front distance change increases the position number from the last received 'position' packet and repeats the same procedures as the first slotcar except one step. It sets pwm duty cycle not for 8000 program cycles but until the front distance is less than the reference distance chosen by the user or the default one. We need to have a reference distance between slotcars to prepare the platoon for the experiment. The aforedescribed algorithm is shown in Figure 25. The 'determine position' step isn't obligatory. It can be skipped if the information about car position isn't required for the experiment.

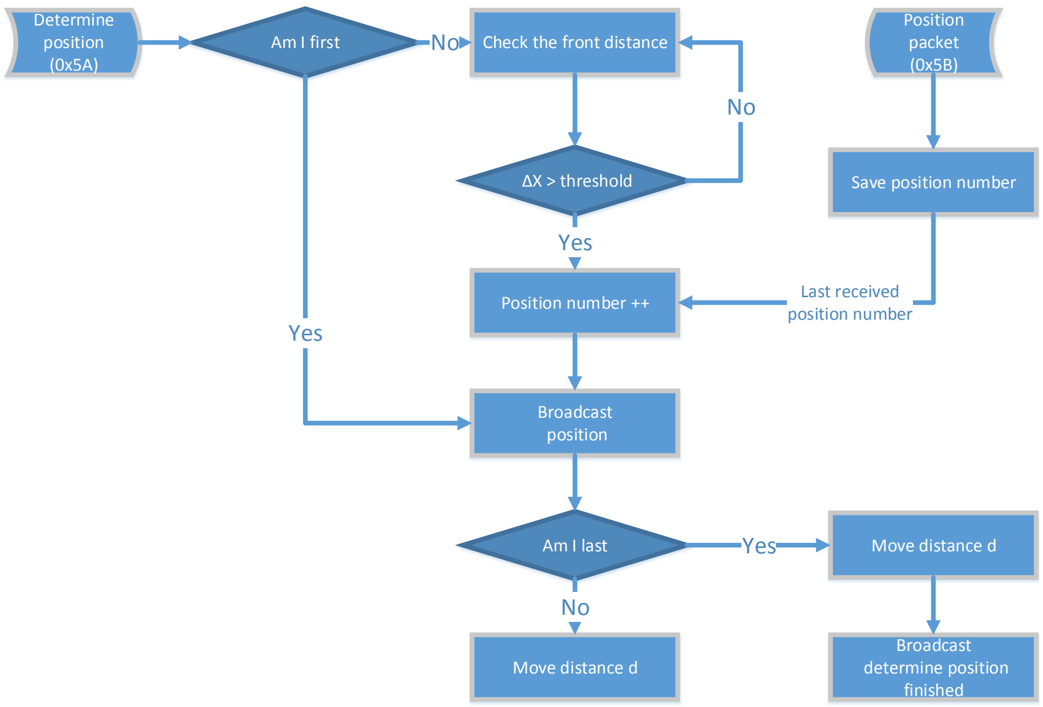


Figure 25. Process of position determination

Table 6. Determine position packet

msg_type	length	dest_addr	source_addr
0x5A	0x00	0x64	0x63

Table 7. Position information

msg_type	length	dest_addr	source_addr	Payload: 0
0x5B	0x39	0x64	MY_ADDR + 100	1 (ACK)

1
position__number

Table 8. Determine position finished

msg__type	length	dest__addr	source__addr	Payload: 0
0x5B	0x39	0x64	MY__ADDR + 100	1 (ACK)

1
position__number

Experiment

The user begins the experiment by sending the 'start experiment' packet (Table 9).

Table 9. Start experiment

msg__type	length	dest__addr	source__addr	Payload: 0
0x3E	0x00	0x64	0x63	time [ms]

When the leader car receives the 'start' packet, it activates the timer interrupt and starts to broadcast 'states' packet at specifically timed intervals. This packet includes position number, measurements and the address of the node, which must broadcast as next. The slotcars measure and broadcasts the following values:

- measured speed,
- front distance,
- back distance,
- speed regulator output,
- distance regulator output.

Table 10. States packet

msg__type	length	dest__addr	source__addr	Payload: 0
0x3F	0x1E	0x64	MY__ADDR + 100	0 (no ACK)

1	2	3 - 6	7 - 14
next__ node__addr	position__number	speed	reg__speed__Y

15 - 22	23 - 26	27 - 30
reg__distance__Y	distance__front	distance__back

The 'next node' address choice depends on how the transmission is executed: in the position order (Figure 26) or in the car number order (Figure 27). When using the position order the following vehicle will broadcast next. The position determination must be done beforehand.

If using the car number order the slotcar will choose the car number below its own in the address table and with nonzero ZigBee short address. The leader starts from

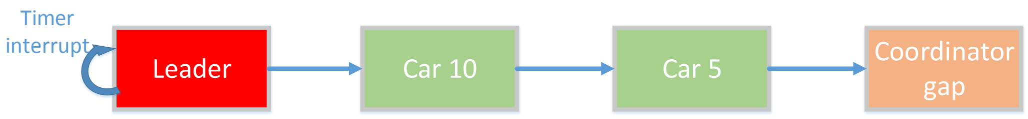


Figure 26. Data exchange in position order

the beginning of the table and can't choose its own address. If there are no active addresses left, the 'next car' field will have a Coordinator address value. The model with interrupt is reliable. In case of the data becoming lost, the data exchange will be restored by the leader.

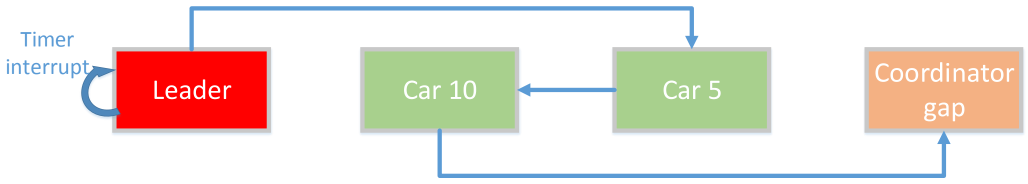


Figure 27. Data exchange in car number order

We use the TDMA (Time division multiple access) method to order the data flow. The number of the vehicles remains constant during the experiment, so the information can be transmitted in suitably fixed intervals. The 'start experiment' packet also includes the time in ms, which determines the transmission interval for each car. The Coordinator has its own transmission slot where it can send, e.g. the new reference speed for the leader. The time slot allocation is a difficult task in the system where beacon requests are disabled and we can't afford to send additional packets for timers synchronization (it will slow down measurements exchange). Hence the timer is enabled only in the leader vehicle. The timer interrupt TIM2 has 84MHz clock. Two values are required for configuration: prescaler and period. These parameters are 16 bit (max value 65535). The timer prescaler specifies the prescaler value used to divide the TIM clock. The timer period specifies the period value for reload. In other words the $timer\_clock/prescaler$ is timer speed and $period$ is a distance. We need to count the time by dividing these two values:

$$\frac{(prescaler + 1) \cdot (period + 1)}{84 \cdot 10^6}.$$

In the actual computation the program adds 1 to each value in case the user will set them on 0. We set the prescaler to 84000. The period value is $transmission\_interval \cdot (car\_count + 1)$, where the car_count is the count of nonzero ZigBee short addresses in the address table and 1 is added for the base station. So the transmission cycle time in ms will be:

$$(car\_count + 1) \cdot transmission\_interval.$$

The transmission cycle time value is 65535. With the time slot 20 ms, the platoon can consist of more than 3000 vehicles.
The acknowledgements aren't requested for the packet, it reduces the channel load. As

mentioned before, we would rather lose some data than resend them. During tests it was found out, that the optimal size of the time slot is 20 ms. At this transmission speed packets go in the right order. Smaller time slot size will cause the following situation where one car will transmit several packets one by one. Other vehicles will wait for a clear channel, store measurements information in the ZNP transmission queue and then transmit everything at once including the 'old' measurements. It's important to emphasize that packets will be stored in the ZNP, not in the Message handle layer (MHL) transmission queue. The request to transmit data goes directly from the Main program layer (MPL) to the Hardware layer (HWL). It saves processor time.
Although the whole transmission cycle time is strictly determined, the gap between each node transmission varies between 12 and 30 ms. The archived result is that four devices (three slotcars and the base station) can share information with each other 12 times per second.
To stop the experiment the user should send the 'stop' command (Table 11). The regulator type is switched to NON and the pwm duty cycle is set to 0. Before starting the new experiment, the user should again set necessary parameters.

Table 11. Stop

msg_type	**length**	**dest_addr**	**source_addr**
0x50	**0x00**	**0x64**	**0x63**

Two CC2531 USB Dongle devices are used for data exchange control. On one of them has the Packet Sniffer application installed. The other one is programmed with CC2531ZNP-Pro firmware and plugged into the tablet with the android application written by Alexander Dubeň. The first application is very useful to control the joining and association process, address exchange and position determination process. Unfortunately, Texas Instruments doesn't provide the tool to export data from Packet Sniffer to another program, e.g. Matlab. So the tablet took over the Coordinator role from the CC2530 evaluation module. It can not only send required commands but it provides additional capabilities, such as graph plotting and the data export into Matlab. The time assigned by the android application to the received packets is precise enough to make some conclusions about measurements exchange influence on the regulator work. Provided experiments are described in the next chapter.

5 Experiments

After the program enters the ‘Experiment’ state, besides the measurements exchange, it starts to execute the regulation function each program cycle. The output of this function is the pwm duty cycle, which controls the motor. There are two types of regulation in the car: speed and distance. The leader slotcar has the speed regulator. For other cars the user can set the regulation type. This chapter describes the regulation model and demonstrates how the data exchange influences the control signal value of the distance regulator (its output).

5.1 The regulation model

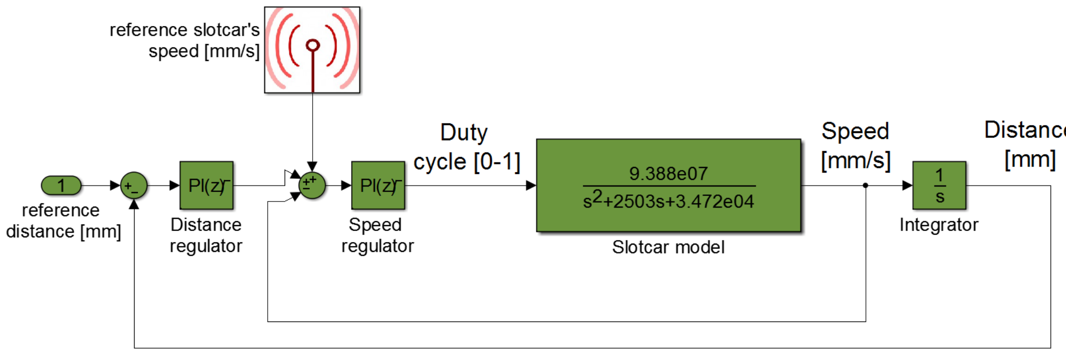


Figure 28. The regulation model with feed-forward loop realized by wireless data exchange

The car model identification and linearization is placed in the bachelor thesis [5]. The model is represented by transfer function (Figure 28). The input of the system is pwm duty cycle in range of -1 to 1 and the output is measured speed. The difference between a measured process variable and a desired setpoint is the PI speed controller input. The speed controller implementation was done by Martin Lád [5]. The regulator’s values are:

- Proportional term is 0,002,
- Integral term is 0,01.

The distance controller includes the speed loop. The output of the outer distance controller determines the set point (reference speed) for the inner speed controller. The output of the inner controller is used to adjust the control variable. This is called cascade control [25]. It is used to improve dynamic performance. Due to the integrator which converts speed into distance the outer loop is slower than the inner one. The PI distance controller was designed by Jan Moravec [6]. The regulator’s values are:

- Proportional term is 10,
- Integral term is 2.

To the existing feedback control loop we added the feed-forward loop. To the speed regulator input is added the reference slotcar’s speed sent by the leader or preceding car. The final regulation model is shown in Figure 28.

5.2 Experiments' results

The experiments are provided on the platoon, which consists of 3 slotcars. The leader slotcar always has a speed controller and its reference speed is set to 600 mm/s. The second and third vehicles have the distance controller, unless otherwise stated. The desired distance between cars is 150 mm. Figures below demonstrate how the experiments were performed and their results (red lines signify the wireless communication):

1. To test the inter-vehicle communication we enabled only speed regulation and set the default reference speed for the second and third slotcars to 0. The speed controller takes the predecessor's speed as an input value. All vehicles successfully started and were travelling at the desired speed during the whole test.

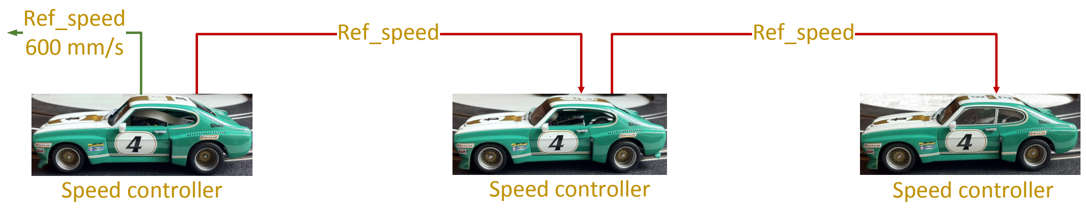


Figure 29. The model for the inter-vehicle communication testing

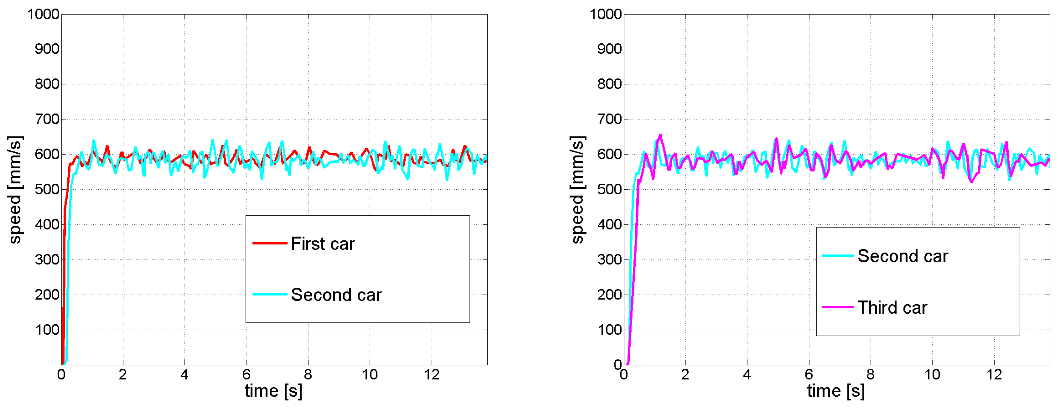


Figure 30. Measured speed

2. Next, we tested the cascade model with 0 feed-forward input. In Figure 32 we can see how the speed controller handles to follow the reference speed determined by the distance controller. The distance regulator settling time also can be determined from this test.

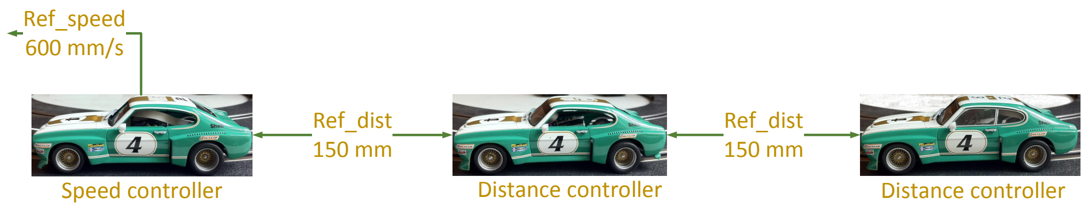
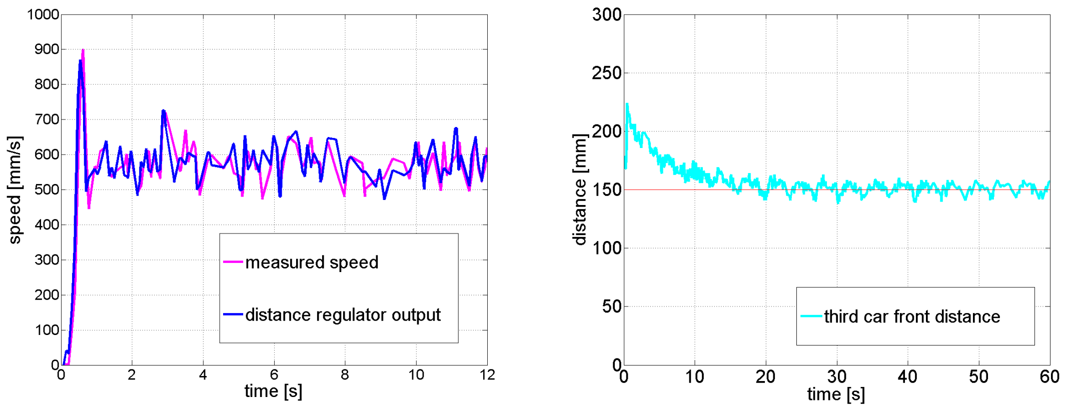


Figure 31. The vehicle platoon model with 0 feed-forward input



a) Third car measurements **b)** Third car front distance

Figure 32. The '0 feed-forward input' experiment results

3. Afterwards, we tested the regulation model shown in Figure 28. To the distance controller output is added the leader's speed. The results are shown in Figure 34. This experiment verified the assumption that the control signal value generated by the distance regulator is minimized when the leading vehicle measurements in a combined feedback and feed-forward loops are used. Also the front distance reached the reference value 4 times faster than in '0 feed-forward input' experiment.

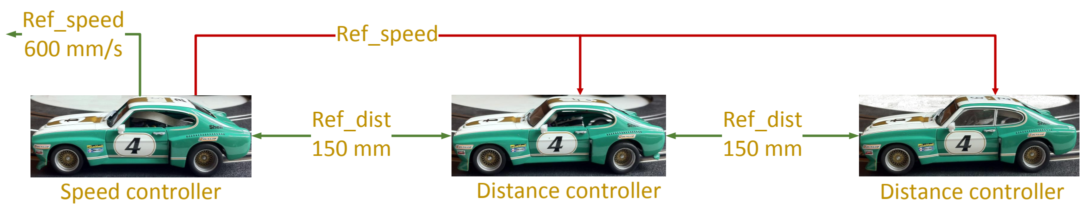
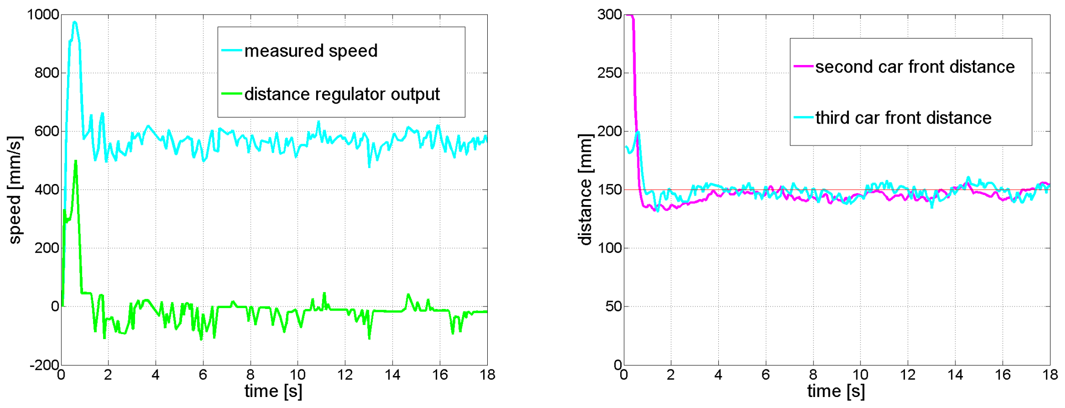


Figure 33. The vehicle platoon model with the feed-forward input determined by the leader's speed



a) Third car measurements **b)** Second and third cars front distances

Figure 34. The 'leader-following' experiment results

4. This experiment is very similar to the previous one. However the reference speed

is sent by the preceding vehicle. We can't make a conclusion about the influence of the information flow topology, because there are only three cars in the platoon.

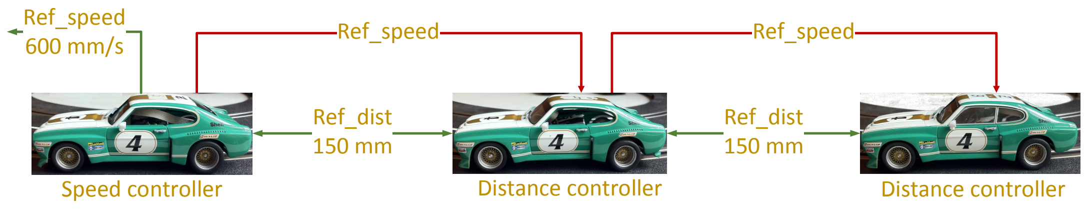
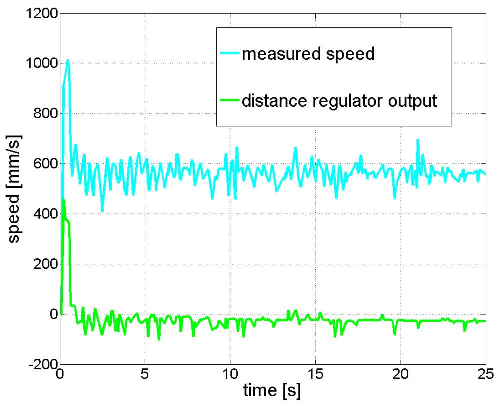
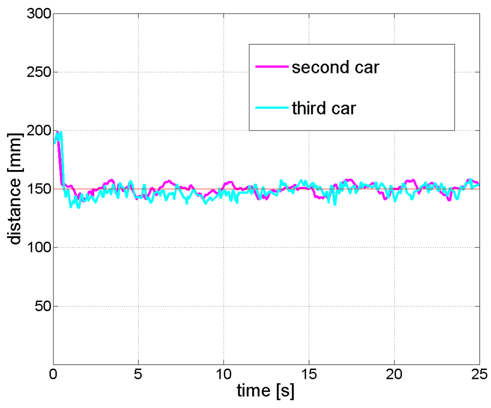


Figure 35. The vehicle platoon model with the feed-forward input determined by the predecessor's speed.



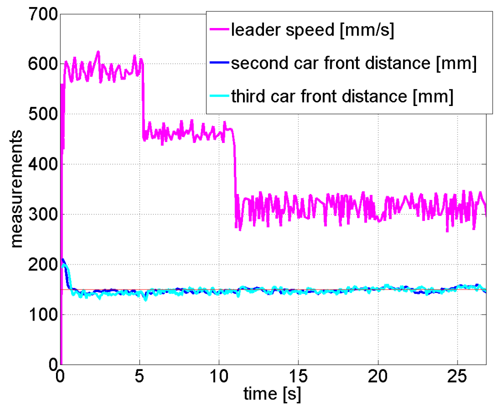
a) Third car measurements



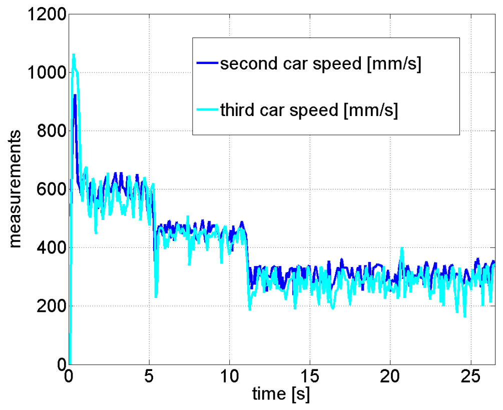
b) Second and third cars front distances

Figure 36. The 'predecessor-following' experiment results

5. Finally, we demonstrated the capability to change the leading vehicle reference speed during the experiment. The platoon configurations are the same as in the previous step.



a) Leader and second car measured speeds



b) Second and third cars measured speeds

Figure 37. Reference speed change during the experiment

5.2.1 Distance error

Figure 38 shows the main project result. It demonstrates that depending on the feed-forward loop presence the distance controller settling time differs.

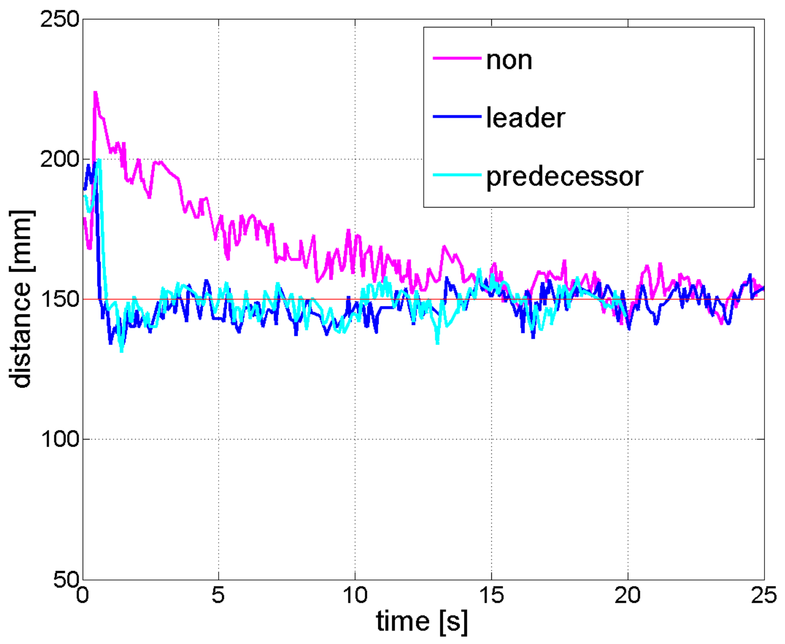


Figure 38. Third car front distance measurements

We calculated the standard deviation:

$$s = \sqrt{\frac{\sum_i (x_i - \overline{x})^2}{N-1}},$$

and the standard error for the front distance measurements:

$$\sigma_{\overline{x}} = \frac{s}{\sqrt{N}}.$$

reference vehicle	**mean value**	**standard error**	**standard deviation**	**approximate settling time**
non	151,601	0,287	5,111	16 s
leader	148,566	0,434	5,421	4 s
predecessor	148,738	0,306	4,999	4 s

Table 12. Distance error

Obtained results are placed in Table 12. We expected that the feed-forward loop will not only shorten the settling time, but provide smaller deviation from the desired value. However this second assumption wasn't confirmed.

Technical problems

The longest experiment which we managed to do lasted for 45 seconds before the car restarted. An undesired slotcar restart happens because the car isn't getting a constant supply of electricity. So the slotcar should be removed from the track and the communication order is disturbed. If the leader suddenly restarts, the whole experiment must be stopped. Power for the slotcar is carried by metal strips on the track and is pitched up by contact brushes which are placed alongside the guide flag. During the ride they can be displaced and cause the car to disconnect from the track. Some slotcar's parts may have expired as well. It's also important to remember, that we configured the communication chip inside slotcar as Full Functional Device (Router) and during constant data exchange its power consumption can't be neglected. To prevent it in the next slotcar model a supercapacitor is used for energy storage. It allows to hold the car functional for 5 s without energy supply.

5.2.2 Future work

The results gained in this project motivate to continue working on it. The new car model should remove energy supply problems and provide new sensors. It will allow better data analysis. Beyond the verifications of aforementioned experiments, we would like to test the communication model shown in Figure 39. The speed regulation cycle is faster than the distance one. Hence it would be even more interesting to see how communication influences the output of the speed regulator and if the communication speed is sufficient.

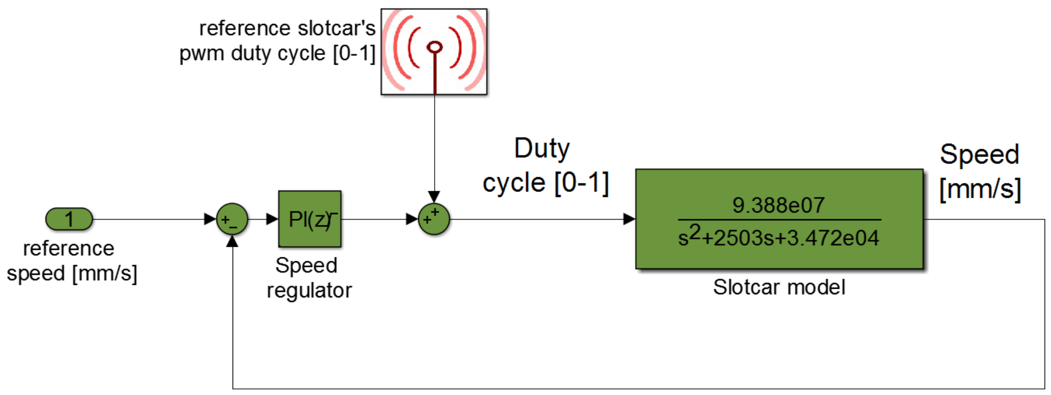


Figure 39. The speed regulation model with feed-forward loop realized by wireless data exchange

The communication analysis could have been realised in this project if there was a time to encode the timeStamp field in the received packets. Unfortunately there is no information about it in the CC2530-ZNP documentation [21].
The data exchange model could be modified as well, e.g. we can activate timer on all slotcars and add the synchronization information into the 'states' packets. Another idea for the communication model is based on the current one. However, the coordinator calls the leader to broadcast and the timer is activated only in case when the data exchange stops. The goal could be to compare communication models and find the most efficient one. Finally, we would like to provide experiments with a larger number of cars and different track forms.

6 Conclusion

In this bachelor work the concept of the inter-vehicle communication is introduced and tested on the real slotcar platoon. This thesis consists of two parts: creating the communication driver based on the ZigBee standard and preparing the communication system which allows vehicles to share their measurements.
The first phase was done on the prototype model. The ZigBee standard research was undertaken and it facilitates choosing optimal configurations, such as nonoverlapping with Wi-Fi channel, device type enabling the direct communication with each node in the network, etc.
The second phase of the project started with testing this driver on slotcars. Its functionality was confirmed. Also we prepared the state automaton in the main program cycle which determines the slotcar behaviour. It provides address exchange, position determination, parameters configuration and read out.
To the existing regulation model was added the feed-forward loop realized by wireless data exchange. Vehicles receive predecessor's or leader's speed through a wireless communication channel. The provided experiments show the smaller distance controller's corrections and 4 times faster settling of the reference distance. The results of this work proves the benefits of inter-vehicles communication. The platoon consists only of three cars equipped with the CC2530 SoC, hence we can't say how the topology of information flow influences the controllers' work. The new slotcar model currently being development promises better hardware functionality. The information gained through this research can be used for the future platform development.

Appendix A

Hardware specifications

Table 13. Connection of Kit CC2530 and Disco Kit per SPI1

	Pin CC2530 Debug	Pin CC2530 P1 + P10	STM32pin
MOSI	P18-18	P10-11	PA7
MISO	P18-12	P10-9	PA6
CLK	P18-16	P10-13	PA5
CS	P18-14	P10-29	PC5
MRDY	P18-11	P1-7	PC0
SRDY	P18-13	P1-3	PC1
CFG0	P20-18	P10-7	PC2
CFG1	P20-19	P10-1	PC3
RESET	P20-14	P10-35	PC4

Table 14. Connection of Kit CC2530 and Disco Kit per SPI2

	Pin CC2530 Debug	Pin CC2530 P1 + P10	STM32pin
MOSI	P18-18	P10-11	PB15
MISO	P18-12	P10-9	PB14
CLK	P18-16	P10-13	PB13
CS	P18-14	P10-29	PB12
MRDY	P18-11	P1-7	PE9
SRDY	P18-13	P1-3	PE10
CFG0	P20-18	P10-7	PE11
CFG1	P20-19	P10-1	PE12
RESET	P20-14	P10-35	PE13

Table 15. the CPU clock speed parameters (system_stm32f4xx.c)

	STM32F4-Discovery 168 MHz	STM32F401C-DISCO 84 MHz
#define PLL_M	**8**	**336**
#define PLL_Q	**7**	**4**

Table 16. Connection of CC2530 and Secondary Radio Board Connector

CC2530-ZNP signal	**CC2530 Pin**	**Secondary Radio Board Pin**
MOSI	**PB5**	**P$6**
MISO	**PB4**	**P$4**
CLK	**PB3**	**P$12**
CS	**PB2**	**P$10**
MRDY	**PC14**	**P$8**
SRDY	**PC13**	**P$9**
CFG1	**PC12**	**P$3**
RESET	**PC15**	**P$7**

Appendix B

ZigBee Interface parameters

Address	Audience
0xFFFF	All Devices
0xFFFD	All Devices with Receiver on Permanently
0xFFFC	Routers and Coordinators
0xFFFB	Low Power Routers

Table 17. Broadcast addresses

Channel	Value
NONE	0x00000000
ALL_CHANNELS	0x07FFF800
CHANNEL 11	0x00000800
CHANNEL 12	0x00001000
CHANNEL 13	0x00002000
CHANNEL 14	0x00004000
CHANNEL 15	0x00008000
CHANNEL 16	0x00010000
CHANNEL 17	0x00020000
CHANNEL 18	0x00040000
CHANNEL 19	0x00080000
CHANNEL 20	0x00100000
CHANNEL 21	0x00200000
CHANNEL 22	0x00400000
CHANNEL 23	0x00800000
CHANNEL 24	0x01000000
CHANNEL 25	0x02000000
CHANNEL 26	0x04000000

Table 18. Channel values

Subsystem Value	Subsystem Name
0	RPC Error interface
1	SYS interface
2	Reserved
3	Reserved
4	AF interface
5	ZDO interface
6	Simple API interface
7	UTIL interface
8-32	Reserved

Table 19. Subsystem values

Value	Name	Description
0x00	DEV_HOLD	All Devices
0x01	DEV_INIT	Initialized - not connected to anything
0x02	DEV_NWK_DISC	Discovering PAN's to join
0x03	DEV_NWK_JOINING	Joining a PAN
0x04	DEV_NWK_REJOIN	ReJoining a PAN, only for end devices
0x05	DEV_END_DEVICE_UNAUTH	Joined but not yet authenticated by trust center
0x06	DEV_END_DEVICE	Started as device after authentication
0x07	DEV_ROUTER	Device joined, authenticated and is a router
0x08	DEV_COORD_STARTING	Starting as Zigbee Coordinator
0x09	DEV_ZB_COORD	Started as Zigbee Coordinator
0x10	DEV_NWK_ORPHAN	Device has lost information about its parent

Table 20. State values for ZDO_STATE_CHANGE_IND command

Parameter	Size	Description
0x00	1 byte	Device state - See 20
0x01	8 bytes	Device IEEE address
0x02	2 bytes	Device short address
0x03	2 bytes	Short address of the parent device
0x04	8 bytes	IEEE address of the parent device
0x05	1 byte	Channel on which the ZigBee network is operating
0x06	2 bytes	PAN ID of the ZigBee network
0x07	8 bytes	Extended PAN Id of the ZigBee network

Table 21. Parameter values for ZB_GET_DEVICE_INFO command

Appendix C

Commands for CC2530-ZNP configuration

C.1 Power-up

C.1.1 System state after reset (POLL)

Table 22. RESPONSE: SYS_RESET_IND

Length = 0x06	**Cmd0 = 0x41**	**Cmd1 = 0x80**	**Reason**	**TransRev = 0x02**

ProductId = 0x01	**MajorRel**	**MinorRel**	**HwRev**

C.1.2 The device startup options configuration (SREQ)

Table 23. COMMAND: ZCD_NV_STARTUP_OPTION

Length = 0x01	**Cmd0 = 0x26**	**Cmd1 = 0x05**	**ConfigId = 0x03**

Len = 0x01	**Value = 0x02**

Table 24. RESPONSE: ZCD_NV_STARTUP_OPTION

Length = 0x01	**Cmd0 = 0x66**	**Cmd1 = 0x05**	**Status**

C.2 Start-up

C.2.1 Channel (SREQ)

Table 25. COMMAND: ZCD_NV_CHANLIST

Length = 0x03	**Cmd0 = 0x26**	**Cmd1 = 0x05**	**ConfigId = 0x84**

Len = 0x04	**Value: 0x00**	**0x00**	**0x08**	**0x00**

Table 26. RESPONSE: ZCD_NV_CHANLIST

Length = 0x01	**Cmd0 = 0x66**	**Cmd1 = 0x05**	**Status**

C.2.2 Device logical role (SREQ)

The value parameter can be set 0x00 for COORDINATOR or 0x01 for ROUTER.

Table 27. COMMAND: ZCD_NV_LOGICAL_TYPE

Length = 0x03	**Cmd0 = 0x26**	**Cmd1 = 0x05**	**ConfigId = 0x87**

Len = 0x01	**Value**

Table 28. RESPONSE: ZCD_NV_LOGICAL_TYPE

Length = 0x01	**Cmd0 = 0x66**	**Cmd1 = 0x05**	**Status**

C.2.3 PAN ID (SREQ)

Table 29. COMMAND: ZCD_NV_PANID

Length = 0x03	**Cmd0 = 0x26**	**Cmd1 = 0x05**	**ConfigId = 0x83**

Len = 0x02	**Value: 0x03**	**0x00**

Table 30. RESPONSE: ZCD_NV_PANID

Length = 0x01	**Cmd0 = 0x66**	**Cmd1 = 0x05**	**Status**

C.2.4 Simple Descriptor (SREQ). Application profile and endpoint

Table 31. COMMAND: AF_REGISTER

Length = 0x17	**Cmd0 = 0x24**	**Cmd1 = 0x00**	**EndPoint = 0x08**

AppProfId = 0x0D	**AppProfId = 0xBF**	**AppDeviceId = 0x00**

AppDeviceId = 0x00	**AppDevVer = 0x01**	**LatencyReq = 0x00**

AppNumInClusters = 0x00	**AppNumOutClusters = 0x00**

Table 32. RESPONSE: AF_REGISTER

Length = 0x01	**Cmd0 = 0x64**	**Cmd1 = 0x00**	**Status**

C.2.5 Start device in the network (SREQ)

Table 33. COMMAND: ZDO_STARTUP_FROM_APP

Length = 0x01	**Cmd0 = 0x25**	**Cmd1 = 0x40**	**StartDelay: 0x00**	**0x00**

Table 34. RESPONSE: ZDO_STARTUP_FROM_APP

Length = 0x01	**Cmd0 = 0x65**	**Cmd1 = 0x40**	**Status**

C.2.6 Device status (POLL)

Table 35. RESPONSE: ZDO_STATE_CHANGE_IND

Length = 0x01	**Cmd0 = 0x45**	**Cmd1 = 0xC0**	**State**

The application processor should receive the ZDO_STATE_CHANGE_IND commands(POLL) until the State gets the DEV_ROUTER (0x07) value in case of Router or DEV_ZB_COORD (0x09) value in case of Coordinator. The list of ZDO states is placed in the table 20.

C.3 Transmit data (SREQ)

Table 36. COMMAND: AF_DATA_REQUEST

Length = 0x0A-0x6D	**Cmd0 = 0x24**	**Cmd1 = 0x01**	**DstShortAddr = LSB**

DstShortAddr = MSB	**DestEndpoint = 0x09**	**SrcEndpoint = 0x09**

ClusterID: 0x34	**0x12**	**TransID = 0x01**	**Options = 0x90**	**Radius = 0x04**

Len	Data

For no ACK data request the options field has the 0x80 value. The data field can contains 0 to 99 bytes of data without any security.

Table 37. RESPONSE: AF_DATA_REQUEST

Length = 0x01	Cmd0 = 0x64	Cmd1 = 0x01	Status

C.4 Receive Data (POLL)

Table 38. RESPONSE: AF_INCOMING_MSG

Length = 0x11-0x74	Cmd0 = 0x44	Cmd1 = 0x81	GroupId	GroupId

ClustedId = 0x34	ClusterId = 0x12	SrcAddr	SrcAddr	SrcEndpoint = 0x09

DestEndpoint = 0x09	WasBroadcast	LinkQuality	SecurityUse	TimeStamp

TransSeqNumber	Len	Data

The data field can contains 0 to 99 bytes of data without any security.

C.5 Get device information (SREQ)

The 0x02 parameter is used, CC2530 returns it's short address. See the full list of parameters 21.

Table 39. COMMAND: ZB_GET_DEVICE_INFO

Length = 0x01	Cmd0 = 0x26	Cmd1 = 0x06	Param = 0x02

Table 40. RESPONSE: ZB_GET_DEVICE_INFO

Length = 0x09	Cmd0 = 0x66	Cmd1 = 0x06	Param

Value[7] = LSB	...	Value[0] = MSB

The value always has 8 bytes in length even though the actual value may be smaller in size. The remaining bytes can take any value.

Appendix D

The attached CD content

- Electronic version of the thesis in PDF,
- project video,
- project source code (`Platoon_implementation/fw`),
- description of the project part responsible for ZigBee driver and state automaton implementation (`Platoon_implementation/doxygen/html/index.html`).

Bibliography

[1] Li Li and Fei-Yue Wang. *Advanced Motion Control and Sensing for Intelligent Vehicles.* 2007.

[2] S. Tsugawa. “Super Smart Vehicle System -Its Concept and Preliminary Works”. In: *Vehicle Navigation and Information Systems Conference Proceedings.* 1991.

[3] AA4CC group. *Distributed control of spatially distributed systems.* URL: `http://aa4cc.dce.fel.cvut.cz/content/distributed-control-spatially-distributed-systems` (visited on 30/04/2015).

[4] Eclipse. *Eclipse CDT.* URL: `http://www.eclipse.org/cdt/` (visited on 21/01/2015).

[5] Martin Lad. “Design and implementation of a control system for a slot car”. Bachelor thesis. CTU Prague, 2014.

[6] Jan Moravec. “Distribuované řízení kolon vozidel na autodráze”. Bachelor thesis. CTU Prague, 2014.

[7] Texas Instruments. *CC2530.* URL: `http://www.ti.com/product/cc2530` (visited on 07/05/2015).

[8] *SmartRF05 Evaluation Board.* Texas Instruments. 2010.

[9] Jaromir Dvorak. *Scheme Slotcar v1r1.* 2014.

[10] Texas Instruments. *CC2530 ZigBee Development Kit.* URL: `http://www.ti.com/tool/cc2530zdk` (visited on 14/01/2015).

[11] *Z-Stack User’s Guide For CC2530 ZigBee-PRO Network Processor Sample Applications.* Texas Instruments. 2010.

[12] Texas Instruments. *SmartRF Protocol Packet Sniffer.* URL: `http://www.ti.com/tool/packet-sniffer` (visited on 14/01/2015).

[13] Texas Instruments. *SmartRF Flash Programmer.* URL: `http://www.ti.com/tool/flash-programmer` (visited on 14/01/2015).

[14] *UM1669 User manual Discovery kit for STM32F401 line.* STMicroelectronics. 2013.

[15] Shahin Farahani. *ZigBee Wireless Networks and Transceivers.* 2011.

[16] Atmel Corporation. *Atmel AT02845: Coexistence between ZigBee and Other 2.4GHz Products.* URL: `http://www.atmel.com/Images/Atmel-42190-Coexistence-between-ZigBee-and-Other-24GHz-Products_AP-Note_AT02845.pdf` (visited on 06/05/2015).

[17] *Triple Security in ZigBee: Link, Network and Application layer Encryptions.* URL: `http://www.libelium.com/security-in-zigbee-networks/` (visited on 30/04/2015).

[18] *Triple Security in ZigBee: Link, Network and Application layer Encryptions.* URL: `https://hal.archives-ouvertes.fr/inria-00187849/document` (visited on 30/04/2015).

[19] Wikipedia. *IEEE 802.15.4.* URL: `http://en.wikipedia.org/wiki/IEEE_802.15.4` (visited on 30/12/2014).

[20] Wikipedia. *Direct-sequence spread spectrum.* URL: `http://en.wikipedia.org/wiki/Direct-sequence_spread_spectrum` (visited on 08/05/2015).

[21] *CC2530-ZNP Interface Specification.* Texas Instruments. 2010-2013.

[22] Inc. Texas Instruments. *Z-Stack Developer's Guide*. 2006-2012.

[23] Freaklabs. *Zigbee Network Layer Tutorial - Part 4: Network Management 1*. URL: http://www.freaklabs.org/index.php/Blog/Zigbee/A-Brief-Tutorial-on-the-ZCL-with-Examples-from-FreakZ.html (visited on 01/05/2015).

[24] Freaklabs. *Zigbee Network Layer Tutorial - Part 4: Network Management 1*. URL: http://www.freaklabs.org/index.php/Blog/Zigbee/Zigbee-Network-Layer-Tutorial-Part-4-Network-Management-1.html (visited on 03/01/2015).

[25] Wikipedia. *Cascade control*. URL: http://en.wikipedia.org/wiki/PID_controller#Cascade_control (visited on 15/05/2015).

[26] Jennic. *ZigBee Software Architecture*. URL: http://www.jennic.com/elearning/zigbee/files/content_frame.htm (visited on 06/05/2015).

[27] AA4CC. *Slotcar Platoon System Description*. URL: https://support.dce.felk.cvut.cz/mediawiki/index.php/Slotcar_Platoon_System_Description (visited on 01/21/2015).

[28] Low Power Wireless and ZigBee Networking Workshop. *ZigBee Stack*. URL: http://processors.wiki.ti.com/images/8/8a/08_-_ZigBee_Stack.pdf (visited on 03/01/2015).

[29] Freaklabs. *Zigbee Network Layer Tutorial - Part 3: Broadcasts and Neighbors*. URL: http://www.freaklabs.org/index.php/Blog/Zigbee/Zigbee-Network-Layer-Tutorial-Part-3-Broadcasts-and-Neighbors.html (visited on 03/01/2015).

[30] Freaklabs. *Zigbee Network Layer Tutorial - Part 2: The Rx Data Path*. URL: http://www.freaklabs.org/index.php/Blog/Zigbee/Zigbee-Network-Layer-Tutorial-Part-2-The-Rx-Data-Path.html (visited on 03/01/2015).

[31] Freaklabs. *Zigbee Network Layer Tutorial - Part 1: The Tx Data Path*. URL: http://www.freaklabs.org/index.php/Blog/Zigbee/Zigbee-Network-Layer-Tutorial-Part-1-The-Tx-Data-Path.html (visited on 03/01/2015).