libre_bus

Łukasz Drzensla

April 2024

1 Introduction

This document defines requirements that the discussed device should meet. It also includes prototype proposals (circuit diagrams, evaluation boards etc).

2 Purpose

libre_bus is a system designed for use in a public transport bus or tram. It is assumed that the users of the system can be split into three groups: administrators, drivers and passengers. The first group uses the system rarely (that is: only when there is a need to perform configuration) and it is assumed that only one administrator uses the system at the same time. Members of the drivers group also may use the system singly at the same time. Whereas many passengers may use the system at the same time. Main use cases of the system for each group are shown on the diagram below.

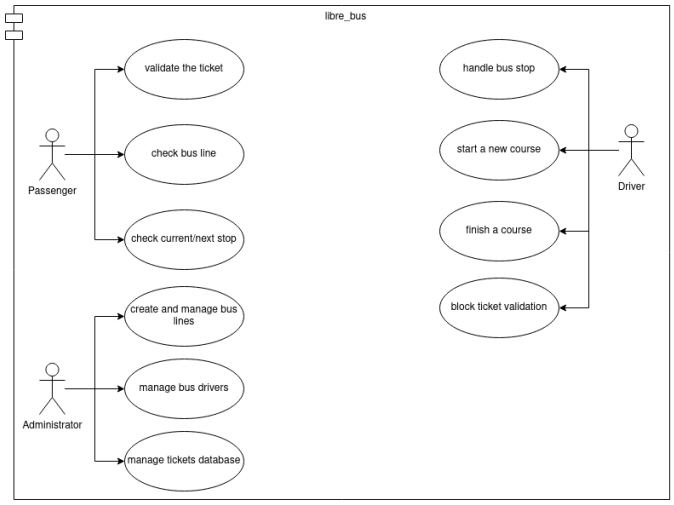

Figure 1: Main use cases diagram

Activities performed by a user from each group are presented on a diagram below. It should be noted that these tasks include activities indirectly connected with the utilisation of the system. The diagrams have been provided to better depict when and how the system is used.

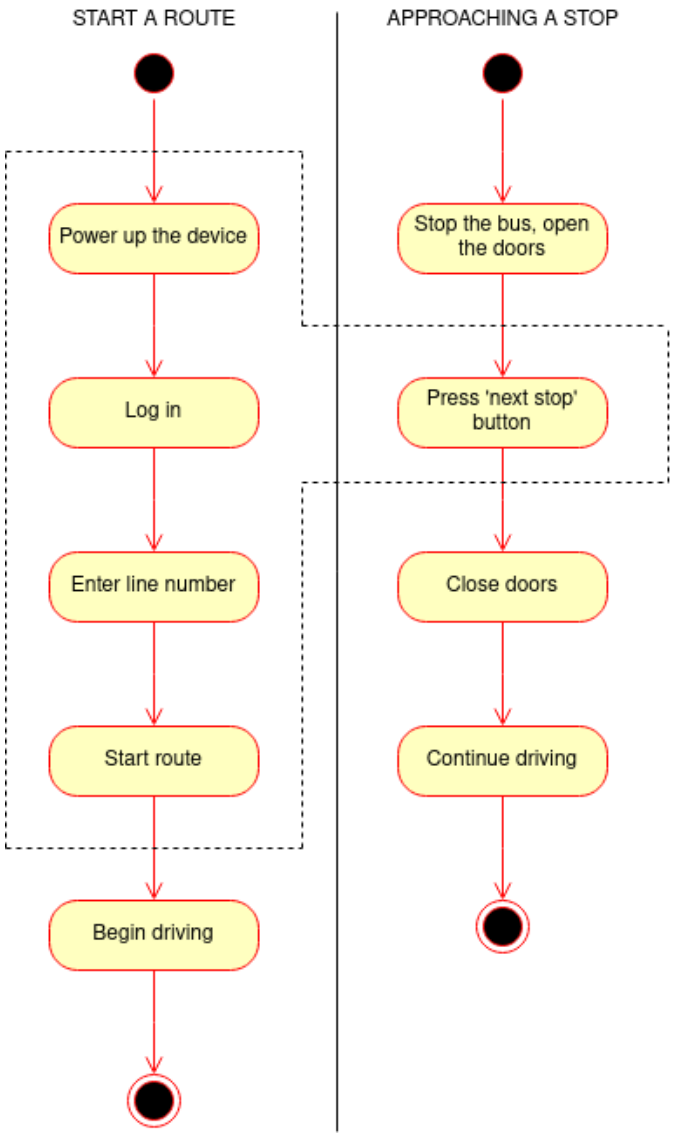

Figure 2: Driver activities flowchart. Activities that involve usage of the system are enclosed within dashed lines.

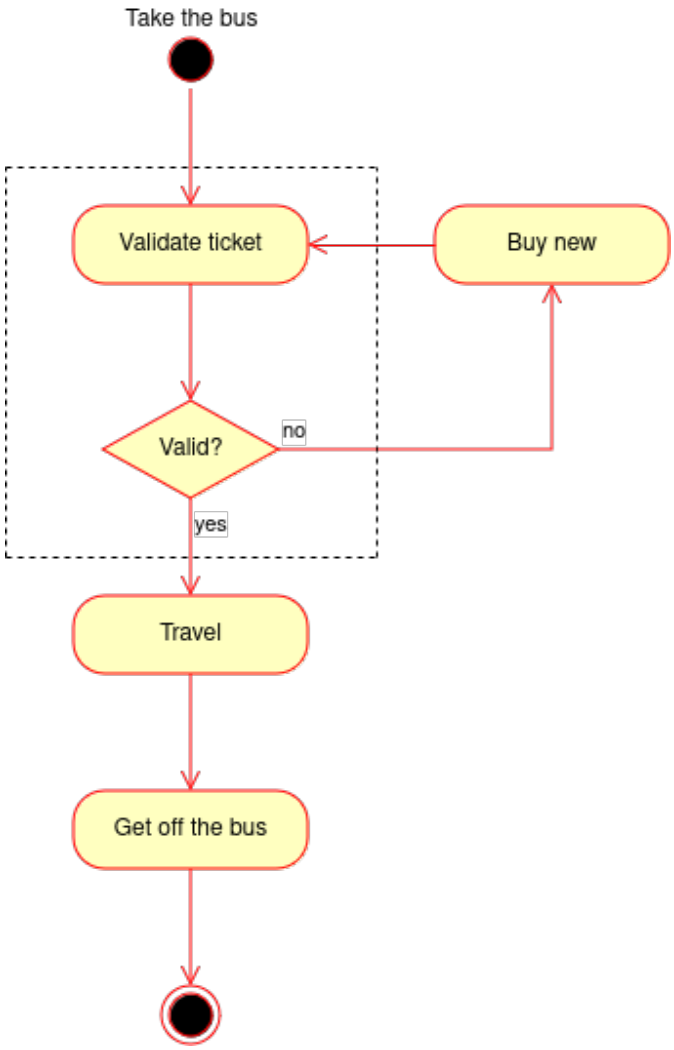

Figure 3: Passenger activities flowchart. Activities that involve usage of the system are enclosed within dashed lines.

3 Main components

The system consists of three main component: Driver Interface Device, Master Device and Passenger Devices. It should be noted that there shall be only one Driver Interface Device, one Master Device but there might be multiple Passenger Devices. The diagram below presents their relation.

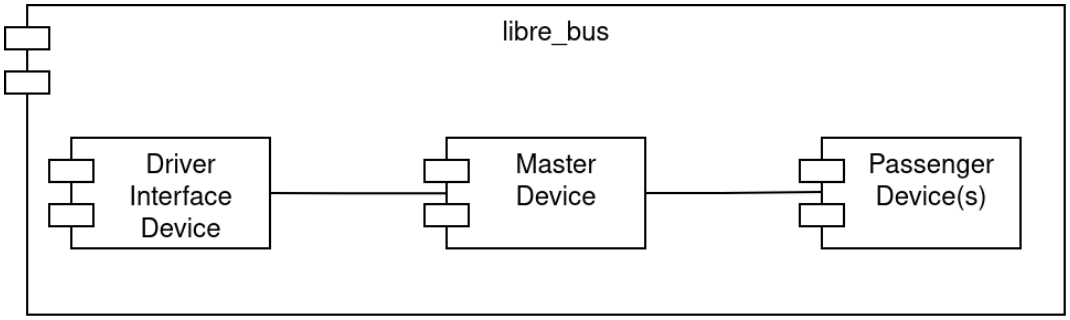

Figure 4: Main components

Requirements these devices should meet are listed below.

3.1 Passenger Device

- enable ticket validation (through RFID cards)
- display the status of ticket
- display the current/next stop and bus line
- show information whether ticket validation is locked or not (LEDs)
- there might be multiple passenger devices in the system

3.2 Driver Interface Device

- enable driver login
- enable choosing bus line
- easy bus stop switching (when bus stopped and proceeds going to the next one)
- enable locking/unlocking ticket validation
- there might be only one device in the system

3.3 Master Device

- handle requests from driver and passenger devices and reply to them
- store ticket info database
- store driver info database
- provide a tool for an administrator to use for management (may be accessed from ssh login)
- (optional) display current/next stop

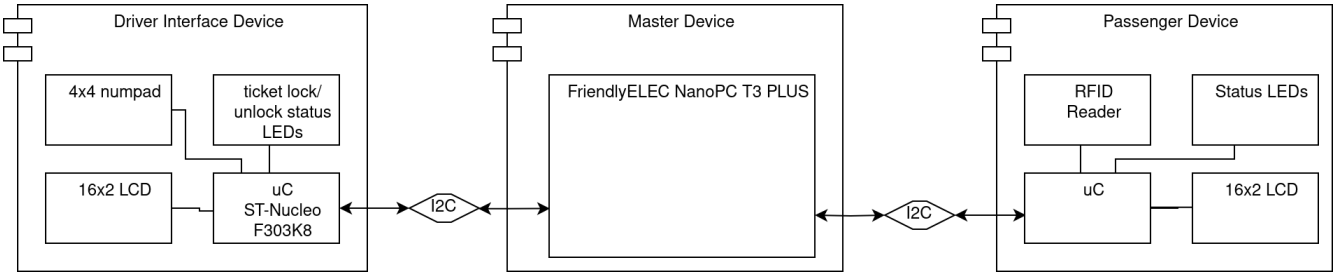

Figure 5: More elaborate diagram presenting the components. Rationale for sub-component choices is stated later in the document in chapters dedicated to the specific components.

4 Device Design

This chapter introduces concepts of design of three main components. It shall be noted that as the intended purpose of the system is operation in a vehicle. Thus the available voltage is assumed to be equal to 12 V.

4.1 Passenger Device

The main objective of passenger device is to provide interface for a passenger to validate tickets and get feedback whether their ticket has been validated or not. This can be accomplished with a RFID reader and LCD screen (eg 16x2) and optionally LEDs. This peripherals need to be driven by a microcontroller which shall be connected to the master device via the chosen bus - that is I2C. It should be noted that there might be many passenger devices.

4.1.1 Hardware

For the purpose of evaluation and development Arduino Uno has been chosen as the passenger device. It communicated with master device via I2C.

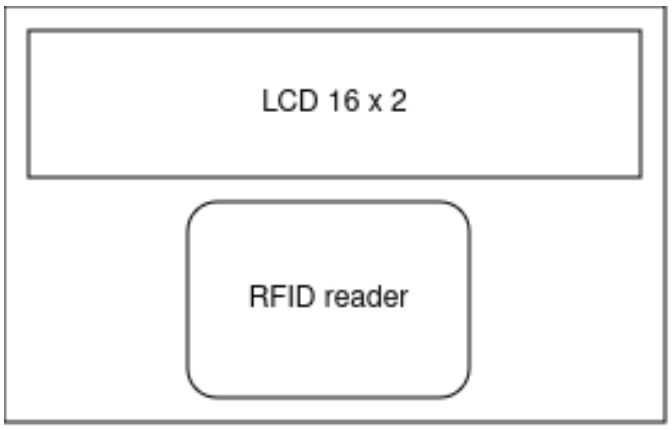

Figure 6: Passenger device mockup

4.1.2 Software

Software should handle the following scenario: a passenger validates a ticket. The device should ask the master whether the passenger has a valid ticket, check the response and prompt adequate output. The desired behaviour is presented on the flowchar below.

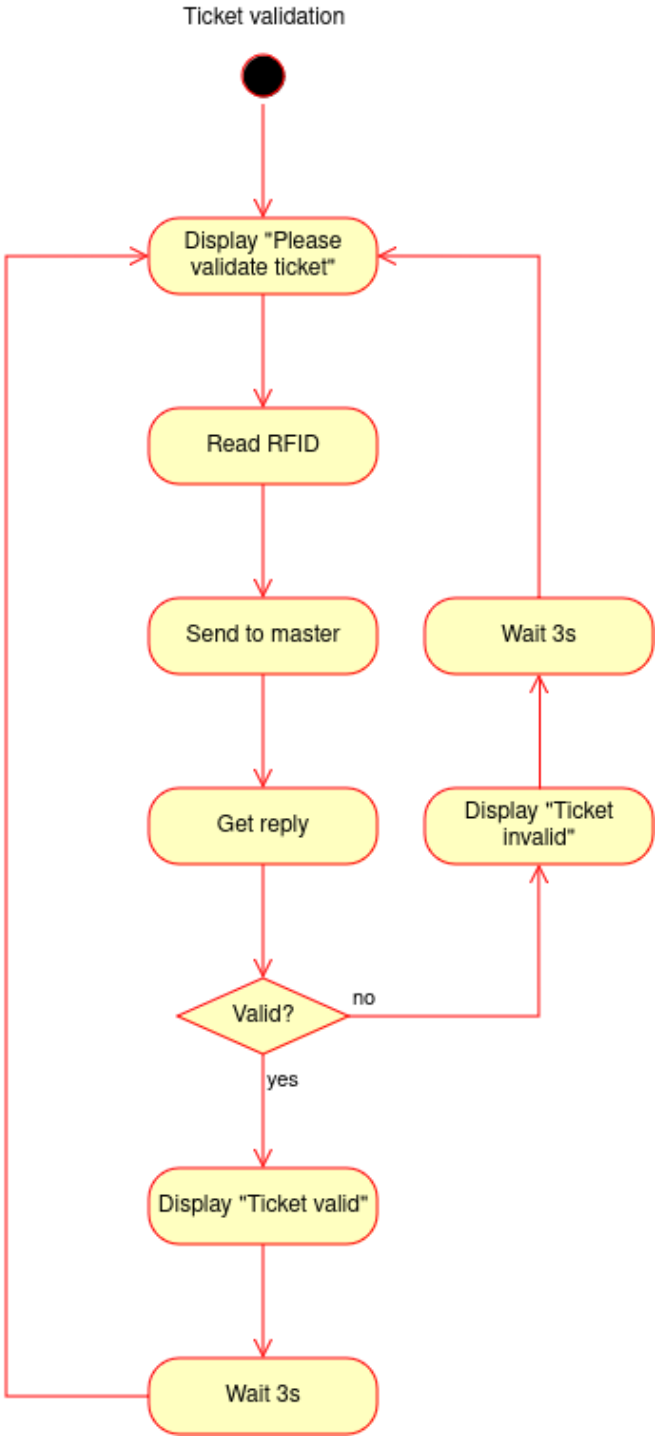

Figure 7: Visualisation of activities performed by the passenger device.

4.2 Driver Interface Device

The Driver Interface Device shall provide the driver with all the functionality to fulfill the use cases presented for the driver in figure 1 and elaborated in chapter 3.2.

4.2.1 Hardware

The device shall allow driver login. Each driver is given a unique ID and a passcode (later called collectively "login credentials") by the company which utilises the system. An ID is a 4-digit value that consists of numbers from 0 to 9. The same applies to the passcode. Thus the interface device should contain a numpad which would allow input of numbers from 0 to 9 and a button for confirmation. Aside from that the interface shall include a button for marking a bus stop as 'done' which indicates that th bus has stopped and proceeds to go to the next stop. Also it should provide a button for locking and unlocking ticket validation. This functionalities will be used very often hence the decision to dedicate buttons to them. A 4x4 numpad has been chosen to serve the purpose of the input for this device. Left buttons will be used for more generic functionality: menu, browse menu (arrows) and go back. These will allow starting a new course, finishing a course etc. As stated above the text, two LEDs shall be present: one indicating enabled ticket validation (green) and disabled ticket validation (red). Only one LED can be on at the same time. The device shall also contain a power switch.

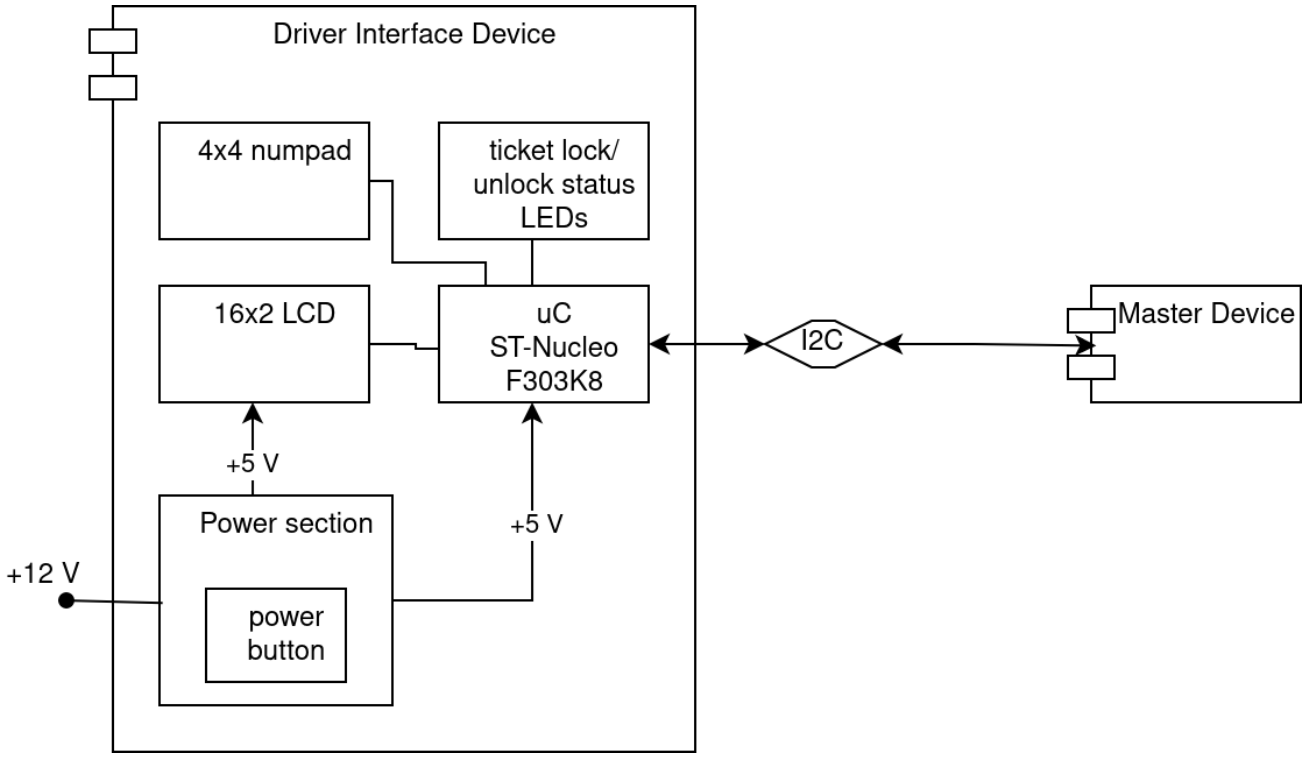

Figure 8: Diagram showing the design of the Driver Interface Device

The device is driven by a microcontroller. After careful consideration ST-Nucleo F303K8 has been chosen for development purposes. The chosen 16x2 LCD is driven by HD44780 driver. The chosen buck-boost converter for the power section is S13V10F15.

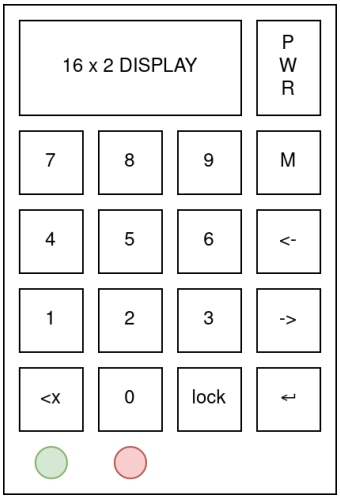

Figure 9: Driver Interface Device mockup

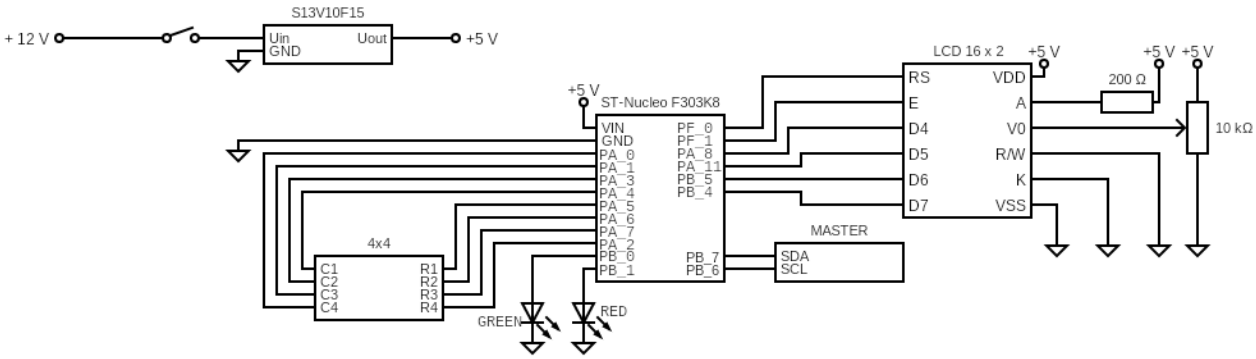

Figure 10: Driver Interface Device circuit diagram

4.2.2 Software

Software should be able to handle the following scenarios: driver login, choosing bus line, next stop handling, blocking ticket validation. Diagrams presenting desired behaviour for each scenario are presented below.

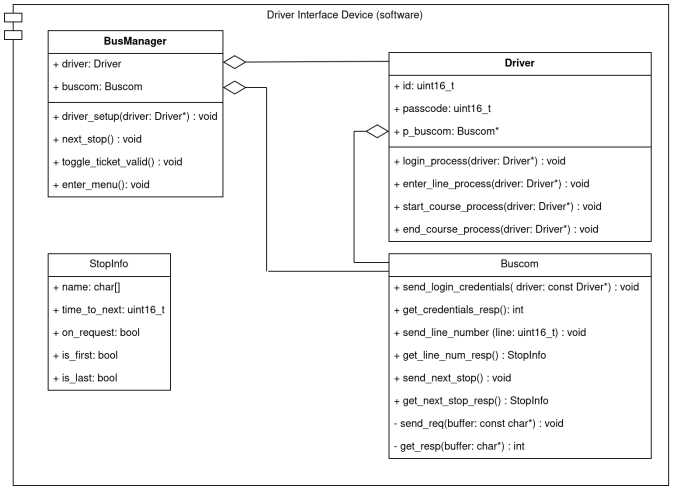

Figure 11: Classes diagram for Driver Interface Device

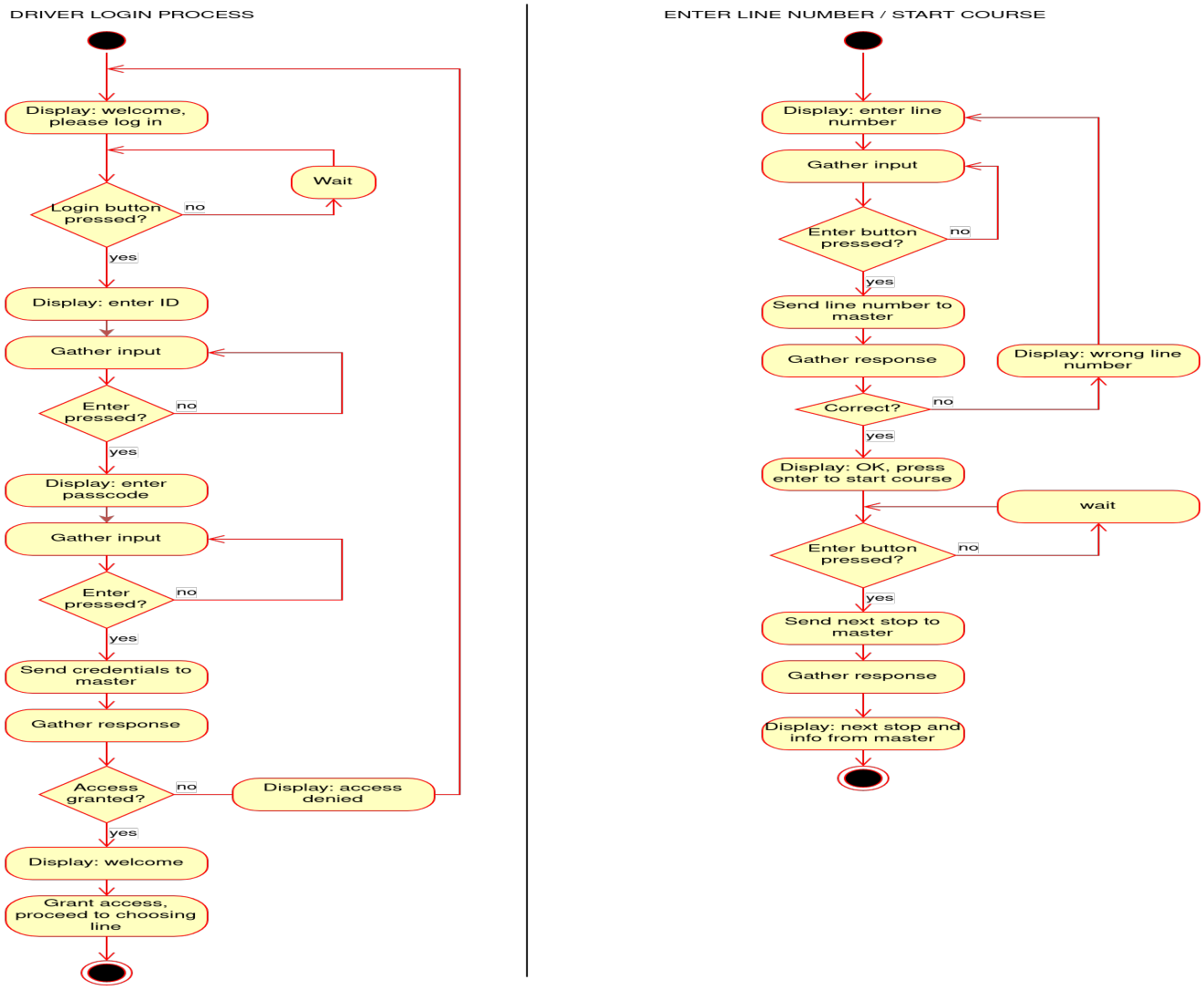

Figure 12: Flowcharts of two tasks which a driver executes at the beginning of a route.

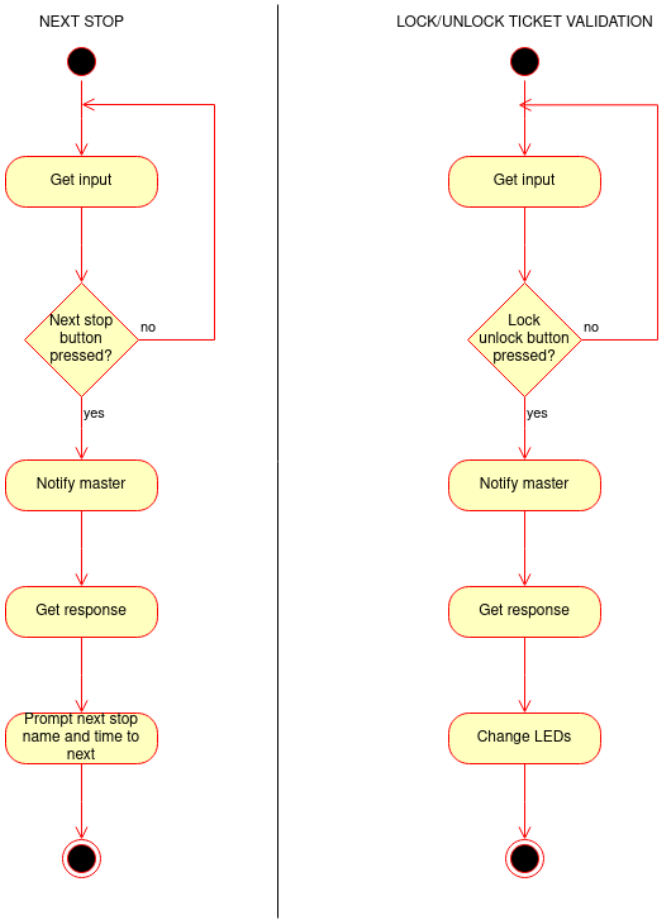

Figure 13: Flowcharts of two important driver tasks: proceeding to next stop and lock/unlock ticket validation

4.3 Master Device

The main objective of the master device is processing requests from Driver Interface Device and from Passenger Devices and replying to them. Master device keeps records of drivers and passenger tickets. It shall also allow for maintenance and configuration to be performed by an administrator using an engineering tool. Master device should run Linux and allow command line access (connection via ssh). It shall have a static additional IP address configured on the Ethernet port: 169.254.100.1 to allow ssh connection using link-local like connection.

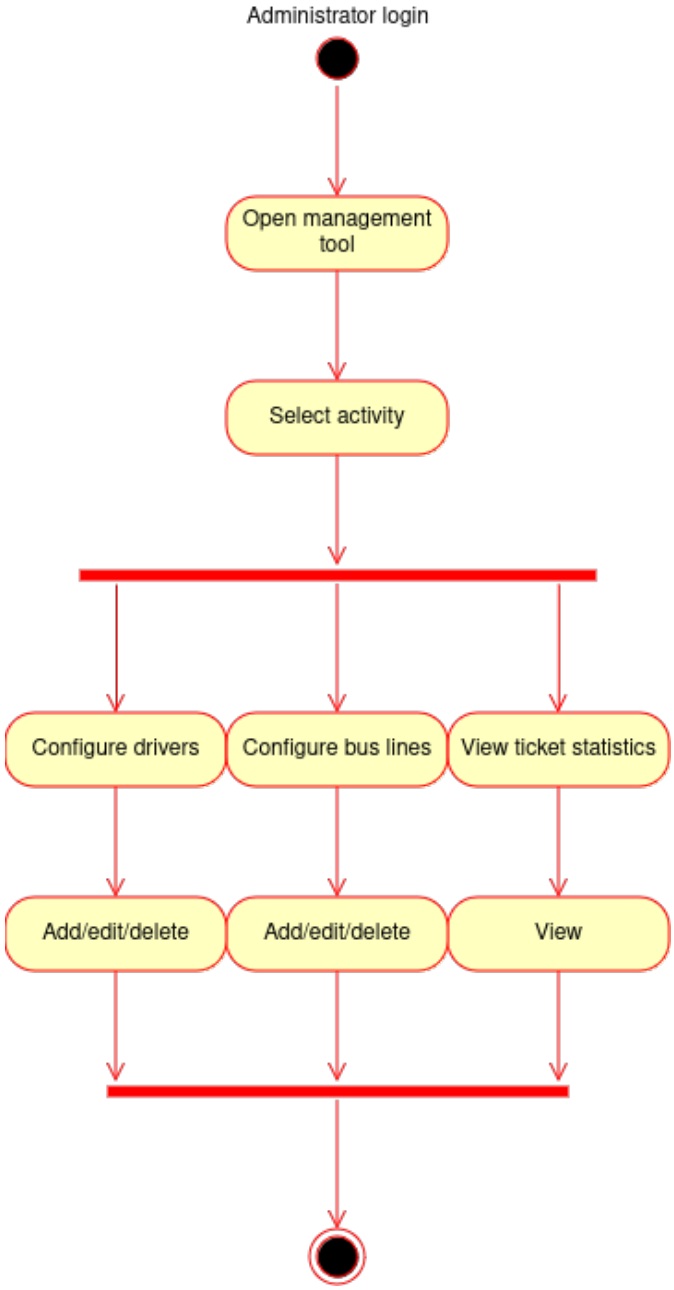

Figure 14: Activities administrator should be able to perform using the management tool on the master device

4.3.1 Hardware

For the purpose of evaluation and development FriendlyELEC NanoPC T3 PLUS has been chosen as the master device. It communicated with Driver Interface Device and Passenger Devices via I2C.

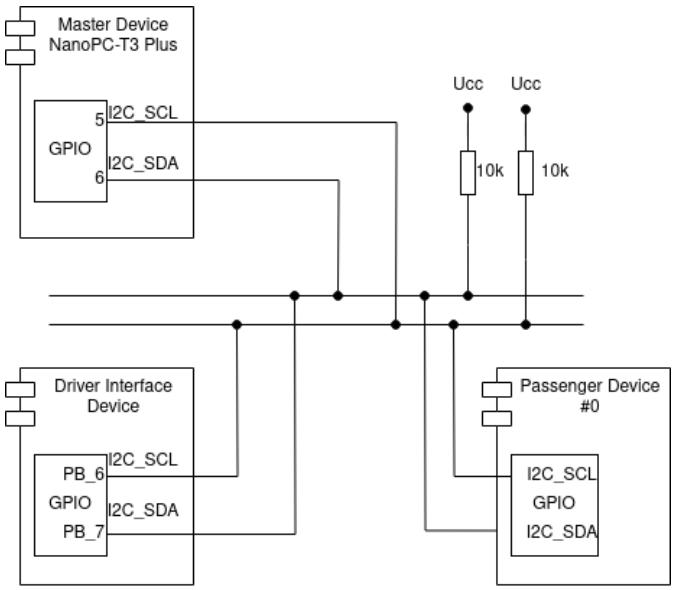

Figure 15: Diagram showing I2C wiring diagram.

4.3.2 Software

Driver info and bus lines shall be stored as XML files. Driver info should contain each driver name, surname, ID and passcode. Passcode should be stored as a hash. A bus line should have two child nodes for each route direction. A route node contains two or more nodes, one for each bus stop. Their order in file corresponds to their order in a bus route. A bus stop node should contain bus stop name, a boolean value whether a stop is 'on request' or not and time in minutes that takes the bus to reach the stop from the previous one.

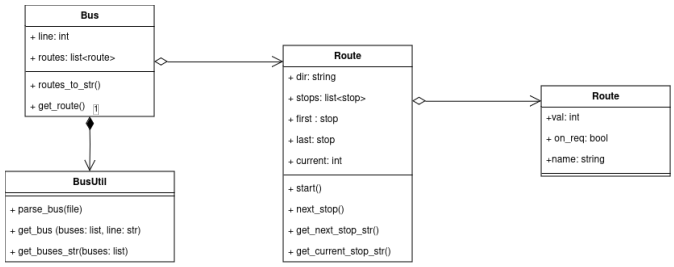

Figure 16: Classes diagram for master device intended for handling requests for driver device

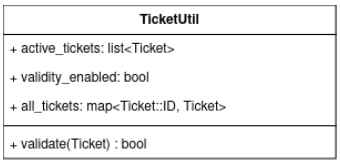

Figure 17: Class for master device intended for handling requests for passenger device

4.3.3 Protocol

For the purpose of communication between components a protocol needs to be defined or chosen. This document proposes defining a new application level protocol called "Buscom". Buscom shall be able to handle the following functions: login credentials, next stop, line number and ticket validation. Functions with their codes (FC = Function Code) and info whether they are Queries (Q) or Responses (R) are presented in the table below. Frame has a fixed size of 36 bytes.

Frame begin is an identifier which tells the protocol stack a new frame has started. It can be any number which can fit into two bytes. Proposal in this document is 0xA9. The protocol shall be defined in a separate document.

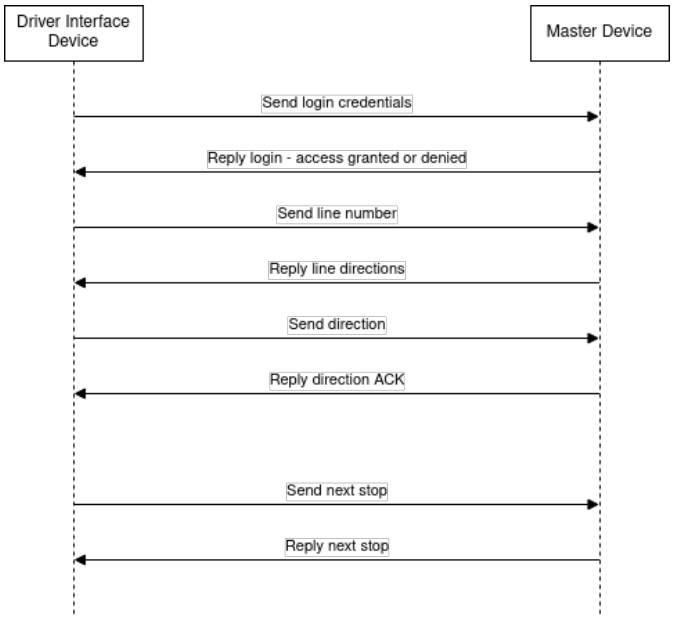

Figure 18: Frames that are exchanged between master device and Driver Interface Device during starting a bus route and during the whole route (separated section at the bottom). The last two frames are exchanged more than once in a route.

FC	Function	Q/R
00	Login credentials	Q
01	Login credentials	R
02	Next stop	Q
03	Next stop	R
04	Line number	Q
05	Line number	R
06	Ticket validation	Q
07	Ticket validation	R

Table 1: Function codes

Header [2 B]	Payload [32 B]	CRC [2 B]

Table 2: Buscom frame

Frame begin [1 B]	FC [1 B]

Table 3: Buscom header

4.4 Prototype

A prototype of the Driver Interface Device has been created. As stated in this document before, the microcontroller chosen for the evaluation purpose is ST-Nucleo F303K8. It drives an LCD 16x2 display and two LEDs on a prototyping board. A button matrix 4x4 is also connected. Software has been written to provide the crucial functionality however without communication with the master. Instead, a dummy implementation has been introduced. It should be noted that the LCD driver has been created using https://controllerstech.com/interface-lcd-16x2-with-stm32-without-i2c/ tutorial. A picture of the device is presented below. The developed prototype allow for driver login (single driver), choosing bus line (one defined), locking/unlocking ticket validation and handling bus stops ("next stop" functionality).

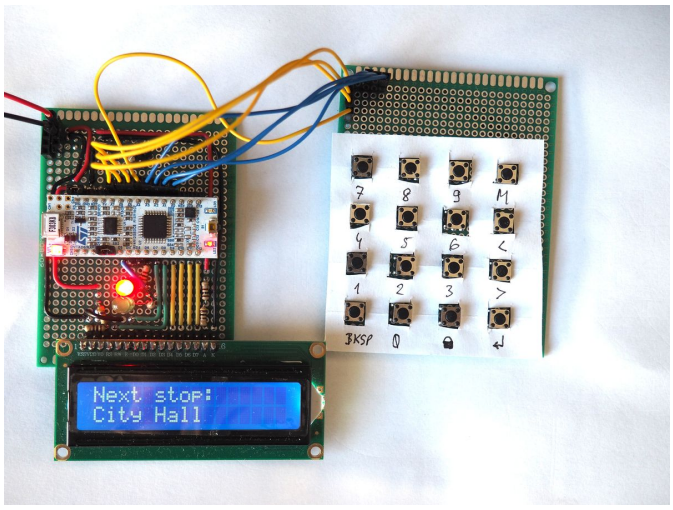

Figure 19: A photo of the prototype of the Driver Interface Device. The driver has already logged in and stared a route. Ticket validation is locked (red LED on).

The next step in prototyping shall be implementation of communication between master device and driver interface device. Firstly, Buscom protocol shall be fully defined and then implemented generically for usage on both sides. Secondly, appropriate software for handling driver device requests shall be developed for master device.