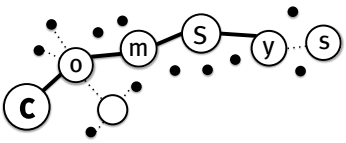
FACULTY OF
COMPUTER SCIENCE



Communication and Networked Systems

Bachelorarbeit

Reverse-engineering a De'Longhi Coffee Maker to precisely bill Coffee Consumption

Fabian Off

Supervisor: Prof. Dr. rer. nat. Mesut Güneş
Assisting Supervisor: M.Sc. Frank Engelhardt

Institute for Intelligent Cooperating Systems, Otto-von-Guericke-University Magdeburg

28. February 2018

Abstract

Abstract

For billing coffee, the faculty of Communication and Networked Systems (ComSys) is looking for a way to monitor the usage of the faculties' coffee maker. This thesis describes the reverse-engineering of a De'Longhi Coffee Maker to allow for precise billing of coffee consumption.

Currently, consumption is measured by manually keeping track of who's ordered a coffee on a sheet of paper next to the coffee maker. This method has inherent problems: as each coffee is billed the same amount, it is not based on the actual amount of coffee but rather the count of beverages. A small cup of very light coffee costs the same as a large cup of a very strong brew - even though it consumes less actual beans. Also, this method is prone to error: people may forget to make a mark when ordering and thus overall amounts may be skewed. Finally, a list kept up-to-date manually can't be analyzed over time and does not allow for reporting of coffee consumption or even prediction of when to order new coffee beans.

This thesis solves the problems associated with manually keeping track of consumption by developing an automated method of monitoring the coffee maker. This is implemented in the form of a device that monitors and controls the coffee maker. Further, a billing system is developed to order beverages as well as keep track of the actual consumption per user.

Precise measurements are conducted to confirm the assumptions about the machine's operation and to calibrate the billing to the real consumption.

As a result of the thesis, an understanding of the machine's internal communications bus and the protocol used on that bus was developed. This understanding is published in the form of a protocol definition as well as a verification algorithm to validate the received data.

Contents

List of Figures

List of Tables

Acronyms

CLI Command Line Interface. 19

ComSys Communication and Networked Systems. iii, 3–5, 8, 10–12, 21, 43, 44

DBC De'Longhi Bus Connector. 11–13, 19, 22, 24–26, 28, 29, 31–34, 37, 40, 41, 43–46, 51, 52

DMA Direct Memory Access. 10, 23–26, 34

IoT Internet of Things. 45

LDAP Lightweight Directory Access Protocol. 10, 12

MCU Microcontroller Unit. 8–10, 22, 23, 26, 32–34

MitM Man-in-the-Middle. 17, 18, 44, 47, 48

SPI Serial Peripheral Interface. 9, 14–17, 19, 22–26, 32, 34, 40, 46

Glossary

De'Longhi ESAM 6600 A coffee maker with an integrated bean grinder and heating system as well as a milk container, predates the ESAM 6660. 22, 45–47

De'Longhi ESAM 6660 A coffee maker with an integrated bean grinder and heating system as well as a milk container. 7, 11, 13, 14, 46–48

Firmware Software that runs on the embedded microcontroller unit. 23

LCD The User-Interface Board in front of the Machine. 14–19, 22–26, 28, 29, 31, 32, 34, 37

PB The Main Logic Board inside of the Machine. 7, 14–19, 22–26, 28, 29, 31–34, 37, 48

program A specific type of coffee that the machine can brew - such as cappuccino, latte macchiato, caffeelatte. 14, 48

CHAPTER 1

Introduction

In an environment, where multiple people work in close proximity to each other, often it makes sense to share resources. Everyone can benefit from the shared resource, while it only needs to be purchased and maintained once. This ensures low cost of operation and can justify a larger initial investment to purchase a resource, more suited for this shared use.

A concrete example for sharing resources and the benefits that result from this sharing is a coffee maker that is used by multiple offices across the same organizational unit. Instead of purchasing a single coffee maker for each office, an organization may decide to create a shared space for operating the coffee maker and then provide a single machine that is intended to be shared by every office around it. This centralization offers a set of benefits for both the organization as well as all the users of the machine: Instead of everyone having to clean and service their own machine, a single coffee maker will only need to be cleaned every so often and the task can then be distributed across all users. Additionally, a single machine allows to keep a single stock of coffee beans rather than one stock per office and the machine can be placed very near to a water supply for easy refilling. Finally, by creating a coffee-kitchen, even more appliances can be centralized (such as the dishwasher) so that the effects add together.

Meanwhile, this also introduces new challenges compared to everyone having their own machine. Firstly, sharing access to a resource always introduces a need for organizing the time of access - as the coffee maker cannot brew several coffees at the same time, users need to queue up in front of the machine to receive coffee one after another. This lengthens the time each user has to wait to complete an order of coffee and can decrease the productivity of all offices. Additionally, keeping a shared coffee maker clean requires a coordinated effort by all the users of the machine - otherwise an unfair cleaning schedule would emerge leaving some users with the task of cleaning while other users would simple benefit from the clean machine. Similarly, in an environment where coffee is not supplied by the organization, the users are responsible for ensuring an uninterrupted delivery of coffee beans so that the machine has access to enough beans at all times. This is an easy problem for a single-user-machine, where the user who's solely using the machine would also be responsible for re-stocking coffee beans as soon as they are used up. In essence, there is a direct relation of coffee-consumption to coffee-delivery - the same person is responsible for both in this

scenario.

In a multi-user setup, this direct relation can not be employed realistically - every user cannot provide their own stock of beans for each coffee they brew, as the machine uses a large container for coffee beans. Instead, a scheme might be developed where the task of restocking coffee is based on a round-robin schema where each user is responsible for providing the same amount of beans one after another. While this would technically solve the problem, it would still introduce an unfair distribution of the task - a user who ordered a single coffee over a certain period of time would still need to restock the beans once empty, the same as a user ordering multiple coffees every day.

Thus it becomes clear, that the task of restocking needs to be coupled to the actual consumption of a user. This is easily implemented by tracking the number of coffees per user, billing users for their usage and purchasing new beans from the money collected through this billing. Most organizations use a simple list, requiring each user to make a mark for each coffee they ordered. By this separation of consumption and delivery in a multi-user setup, it it possible to have a constant supply of coffee beans that is paid for by all users consuming coffee. There still is a relation between the consumption and the delivery of beans - a higher number of ordered coffees will result in a higher payment and if no coffee was consumed, no payment would be required - and thus the method seems fair for everyone. If each and every person would order the same coffee (such as a short light coffee), this method would, in fact, accurately describe the consumption and thus treat each user fairly.

In fact, though, simply counting the number of coffees introduces an unfairness that originates in the fact that not every coffee is brewed the same way. A short light coffee must be treated differently from a large strong coffee yet simple counting the number of coffees consumed per user does not reflect this difference. While technically possible, tracking the type of beverage (such as light, medium, strong coffee) along with the size (short, long) and the count of beverages is not viable for a large set of users on a paper list. A simple list of tick-marks cannot solve this differentiation and thus a more elaborate method of tracking needs to be employed.

This thesis proposes to use technology to solve this problem - instead of manually tracking consumption on a list of paper by the users, the coffee machine itself is controlled and monitored by means of electronic devices and then usage is associated with the corresponding users. Freeing users from the task of tracking their own orders, the experience to get a coffee is improved and by tracking orders precisely in regards to the consumed coffee, this solution allows for a fair billing of each user. Furthermore this automated tracking also means that usage data can be processed electronically and be used for reviewing past and predicting future consumption of coffee.

In order to implement the proposed solution, an existing coffee maker will be modified such that it can be connected to and monitored and controlled by a PC. This requires the machine to be interface-able from the outside, which the machine at hand for this thesis was not. Thus, a method is devised to allow controlling and monitoring the machine, by connecting to the electronic system that is implemented inside the machine. As no documentation could be acquired from the manufacturer of the coffee maker, this thesis describes the process of gaining the required knowledge to connect to the machine's electronic control system. This process is known as reverse-engineering and results in an understanding of how the

machine works internally along with an idea of how to take control of the processes inside of the machine. Finally, this knowledge is put to use by developing a device that enables a standard PC to interface with the machine and take control of it.

This device is then used to augment the machine's capabilities through implementing a web-based billing system that ultimately solves the problem of measuring coffee consumption precisely and establishing a fair billing scheme. Due to this device, the users will also be able to operate the machine remotely which will allow for organizing spikes of usage - such as in the morning - and reduce the waiting time for each user. While each user still needs to wait the same amount of time from order to coffee delivery, this keeps the productivity intact as users don't need to queue up in front of the machine and wait for other users to finish their orders. Instead, they can submit their order from the comfort of their desk electronically and then they are notified once their order is processed.

This thesis is structured so that the reader can follow the process of analyzing the coffee maker, developing the device and validating its function. In the first chapter, the motivation to conduct this research is explained and a set of specific goals is laid out along with an analysis of related research. Then the technical background needed to follow this thesis is established. The thesis' contribution is describing the proposed solution in detail and shows how its implementation can be achieved by describing the research subject in detail along with the reverse-engineering that was conducted. This is followed by the description of the implementation where the details of how several different systems work together to provide the proposed solution are explained. Then, a set of experiments are developed and conducted to ensure the correct function of the developed device. These experiments will validate the process of reverse-engineering as the understanding acquired will need to be proven correct in showing that the device works as intended. Finally, the results of this research are evaluated and compared to the existing research, proving the significance of this research.The thesis is finished by offering ideas of how to use the provided knowledge and implemented solution in future research, introducing ideas that could not be implemented in this thesis.

1.1 Motivation

Whenever someone at the faculty of ComSys wants a coffee, they need to walk over to the coffee kitchen in order to operate the coffee maker. Due to the high number of users, chances are that either the machine is in use, or water or beans need to be refilled before a beverage can be ordered from the machine. This happens even more frequently, as the machine does not inform users about the beans being empty until the coffee is actually ordered and brewed. Then, the process stops and the user has to get beans and refill the machine. If no more beans are in stock, coffee will be unavailable until it is restocked, mostly not until the next day.

Even if all supplies are in stock and the machine is filled up completely, the order could be delayed by minutes due to the machine's warm-up time. After a period of inactivity, the machine will enter a power save state which means the heating element will shut down and will need to heat up for the next coffee.

Once a coffee is finally brewed, the process is still not finished just yet - the person needs to

remember to make a mark on a list that is located above the coffee maker and counts how many coffees everyone's had in a month. This method of keeping track of coffee consumption has inherent problems: as each coffee is billed with the same amount, it is not based on the actual amount of coffee but rather the count of beverages. A small cup of very light coffee would consume less actual beans than a large cup of a very strong brew. This discriminates everyone who likes a light blend, as they'd need to pay the same amount as someone who's ordered a strong brew that can actually contain up to twice as much actual coffee. While the amount charged per coffee is small and only meant to cover the actual cost of the coffee beans, this still unfairly distributes the cost across all users and would require a complicated tally to allow for fair billing.

Especially in a research environment, work doesn't just pause the moment a person leaves their desk - instead, they think about their work while they are waiting for the coffee machine to fulfill its purpose. And as described, the process of ordering a coffee with all the possible obstacles is all but straight-forward, making a mark can sometimes also be forgotten.

At the end of the month, the paper list has to be analyzed manually, that is each person's marks have to be counted and summed up. To collect the amounts, the person responsible will assess the list and then needs to send out emails to everyone who's consumed coffee in the past month. Again, this comes with a set of possible errors:

- marks can be illegible and counted the wrong way
- sums can be associated with the wrong person (duplicate names)
- email-addresses need to be collected for everyone, as they are not listed on the tracking list

To bring the process of ordering, consuming and billing coffee up-to-date in a time of connected appliances and the mobile web, a solution is developed that resolves most, if not all, of the described problems. This solution will streamline the ordering and automate the billing, while at the same time allowing for digital reporting of coffee consumption. Billing users precisely will make the process fair and collecting information about the orders electronically will rule out errors in the process of making and counting marks. This will ensure that ordering coffee for anyone at ComSys is pleasant and as easy as it should be, while still allowing for a monthly billing to cover the costs of purchasing coffee beans.

1.2 Goal

The goal of this thesis is to resolve the problems associated with manually keeping track of coffee consumption at the ComSys faculty. To achieve that, the existing coffee maker will be augmented and connected to a PC, thus allowing remote access and control of the coffee maker. Using a web-based billing system, users will be able to order coffee that is brewed by the coffee maker and then they will be billed precisely for their consumed coffee.

1.2.1 Billing System

The billing system should provide functions for authenticating users and allow users to control the machine by ordering their favorite beverage. Further, the system should take care of storing the orders for the actual billing and for reporting on the usage. The order process must be as easy as ordering a beverage through the user-interface of the coffee machine. That is, no additional work must be required other than placing an order and retrieving the beverage from the machine. If the machine is in any state that requires manual interaction, such as when it is missing beans or water, it is further acceptable for the user to be responsible for fixing it before receiving the ordered beverage.

This system must be accessible through both a desktop web browser and a mobile device for ordering coffees on-the-go. Users should not need to register and instead be authenticated against an existing user-database containing their ComSys-accounts.

Billing is done once a month and a designated billing user is sent a list of each user that consumed coffee in the current month. This list can be used to request payments from each user, so that coffee beans can be repurchased. During the month, each user can view their current consumption as well as the current total sum of cost. Finally, an indication should be given on how much coffee beans were consumed (in grams), so that beans can be restocked proactively without checking the actual stock and making a manual estimate.

1.2.2 Connecting Device

A way of controlling and monitoring the coffee maker through a PC is to be developed, so that it can interact with the web-based billing system. This solution must not interfere with the safety-functions built into the coffee maker and it must be safe to use in a kitchen-environment. Further, the coffee maker should not be changed visually, and every hardware addition should be contained within its original casing.

The connecting device should serve as a gateway to interface with the machine from a PC and execute and monitor the functions of the coffee maker. By providing this access, it enables to build sophisticated systems and integrate the coffee maker as an input and output device. To allow for easy implementation in other projects, the device's functions should be documented and easily extendible.

1.2.3 De'Longhi Protocol

To gain an understanding of the way the coffee maker works and how information is transmitted internally, the De'Longhi Protocol should be analyzed and reverse-engineered. The results of this should be used to implement a software that allows for **monitoring** and **modifying** the information sent via this protocol. While the hardware interfaces are to be provided by the connecting device, the protocol is needed to be understood to decode the device's internal communication and to order coffees using the billing system. For this thesis, it is not necessary to decode the meaning of every bit the protocol is comprised of. Instead, a general understanding needs to be gained so that the protocol can be generated outside of the machine, allowing for remote control of its functions. All functions the machine supports, such as brewing different kinds of coffee, producing hot water and changing

basic settings such as the taste of the coffee should be available for remote-control. This requires the protocol to be decoded to an extent where all these functions can be triggered and every sensor of the machine can be read.

1.2.4 Scope

The scope of this thesis is to develop a prototype device, along with the software running on this prototype and the web-interface needed for the billing system. Additionally, the thesis will describe a way to analyze, reverse-engineer and modify the data bus inside of the coffee machine. Finally, it will be shown how the gained understanding and the developed software and hardware can be used in an university-setting where the coffee maker will be augmented and integrated into the billing system. Designing the final hardware that needs to fulfill a set of requirements making it viable for permanent installation inside the coffee maker is not part of this thesis and will be carried out by a separate project.

1.3 Related Work

Ever since computers began being connected to one another, they were used not only to transfer knowledge and content, but also to improve peoples' lives. The following articles, theses and papers show different approaches to bring the process of coffee brewing into the digital world.

1.3.1 Connected Machines

One of the first instances is the first ever webcam being used for monitoring a coffee machine as the *Trojan Room Coffee Pot* at the University of Cambridge in 1991 [1]. Convenience, rather than billing, was the motivation behind putting a coffee pot's live image on the local network. But it still stands to show that the idea of a smarter way to brew and consume coffee dates way back to the beginning of the world-wide-web. While providing a very detailed account of the machine's current state and operation, this implementation lacked any way of controlling the machine. Users of the webcam could see if there was coffee in the pot, yet if it was empty there was no way of re-filling without going to, and interacting with, the actual machine. Finally, all users shared the same brew of coffee - there was no per-user order and as such billing would need to be calculated on an average usage without relying on any kind of measured consumption.

Researchers from Brazil took a similar approach to monitoring coffee in [2]. They employed a temperature sensor instead of a webcam and used the resulting measurements to discern whether or not freshly brewed coffee was available. Researching the healthcare aspects of sharing a coffee with other people, their focus was on knowing when a coffee was brewed to send a message through a social network. No information about either the user or the actual coffee was collected, instead it only enabled users to meet at the coffee machine once a new beverage was available.

The idea of connecting a fully automated coffee maker to a network is described in [3]. This article describes a way of sending commands to the machine via SMS, yet no monitoring of

the brewing takes place. Due to much simpler modes of operation in the article's machine, control was achieved by transmitting a set of button states via a shift-register. While the article states that remote-control is enabled through this implementation, this means that there is no way of feedback from the machine to the controlling device. Should the machine run out of water or beans, there is no way for the controlling device to know that and thus no feedback can be sent to the user.

Still, the architecture of the machine described in the article is very similar to the faculties' machine in that the system is comprised of different subsystems. One subsystem, the **User-Interface board** is used for handling user-input - it provides a set of buttons as well as a few LED-lights to show the machine's current state. The **Power-Board** represents the other subsystem - its main task is to control actors and sensors inside the machine and report state to the user-interface. This thesis will show that the De'Longhi ESAM 6660's design resembles the design of the machine in this article. Also, this article will be extended by monitoring the machine for feedback as well as keeping the user-interface in-tact rather than replacing it.

1.3.2 Automated Billing

Several researchers already made efforts to automate the billing process of of-the-shelf coffee makers. While industrial machines often come equipped with specialized vendor-specific ports for this purpose, most billing systems tend to support classic payment methods such as cash only. Interfacing with these ports to replicate billing systems then allows for controlling the machine as well as track its usage.

Maurus Rohrer from the HTW Berlin conducted a research project [4] for controlling and administering household appliances with NFC. This project focusses on implementing a NCF-controller for a Jura coffee maker. An architecture for the prototype is described which shows a modular system that is divided into a backend that communicates with the machine and a frontend that is used to interact with the users. In evaluating the options for communicating with the coffee maker, the researcher decided to leverage the vendor-specific port of the machine. The port supports the *Multidrop bus protocol (MDB)* [5] which provides functions necessary for billing and administration of the coffee maker. According to the project, MDB does not support reporting status information from the coffee maker to the billing system such as an empty water tank or missing beans. This was solved by an external webcam, monitoring the machine's display and parsing the acquired images using image recognition. In summary, this project allows users to request personalized coffee recipes using a standard NFC card, interfaces with the coffee maker through the MDB and monitors the machine's state via a webcam.

Some ideas of this project were incorporated into this thesis, such as the modular design of the system. As no vendor port is present on the machine the current thesis is based on, a different approach was chosen to communicate with the machine. This allows for removing the need for optical recognition, as an electrical connection can be used to inquire about the state of the machine. NFC connectivity is not part of this thesis, as users will be requesting coffee via a web interface only. Instead, the monitoring aspect will be emphasized by allowing users to monitor their own usage as well as be billed precisely based on the usage.

Last year, research was conducted in Munich to centralize billing of all machines in the data center of the Bavarian Academy of Sciences and Humanities [6]. A system to centrally collect billing information was designed where users could order coffee using RFID-transponders already included in their university's transponders. Special attention was given to the security aspect of the overall system, as controllers for the coffee makers were connected to the billing server using an existing network infrastructure. Communication with the coffee maker was abstracted into drivers so that different drivers could be implemented for different makes of machines. Again, the thesis leverages the Jura's proprietary communication bus and builds upon a limited set of functions [7]. In this thesis, billing was calculated on a per-product basis, so different beverages can be charged differently, making way for accurate billing of consumption. Basic monitoring of the machine was implemented using a trial-and-error approach: instead of inquiring the coffee maker for status, a coffee is ordered and then the total count of coffees brewed by the machine is read using the vendor specific interface. Should this indicate no increase in the total number of coffees, an error is assumed and the user is not charged for the order.

The thesis at hand will adapt some design decisions from this work that can help in securing the billing system. For instance, splitting up the request and delivery of a coffee into a set of transactions will allow for an error condition to result in an unmodified balance for the user. Other parts of the thesis' design can't be reused as it proposes a prepaid system rather than a postpaid system. In a setting of connecting all of the Academy's machines this is more suitable as there is not inherent control of which users are using the machine. For the setting of the ComSys machine this would be impractical as it would require users to pay in advance and take care to monitor their credits before ordering a coffee. Instead, the machine is only accessible to a limited amount of users that are known and trusted to pay the consumed coffee at the end of each month. Additionally, any communication between the billing system and the machine controller will happen through a local, trusted, serial connection. As no network is involved, the security implications and considerations for this link will be minimized in this thesis.

1.4 Background

This section explains the technical background needed to follow the thesis and references relevant material to gain a deeper understanding of the techniques and systems. Additionally, it shows how this thesis implements the described parts and explains how they are incorporated into the work.

Microcontroller Unit (MCU)

A Microcontroller Unit describes a small computer-like system that consists of all components necessary to run a program and interface with the outside world. Unlike a *microprocessor*, no further support chips (such as RAM or Math coprocessors) are needed to operate a MCU [8]. In this thesis, a MCU is used to run communication processes that have very strict real-time requirements. A usual consumer PC is processing many different tasks at the same time, such as processing input, updating the user interface and running actual programs. All of these need to happen simultaneously and thus every one of the programs

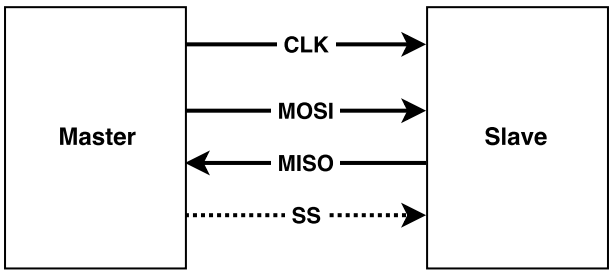


Figure 1.1: SPI bus in a two-device configuration. The arrows indicate the signals' direction.

running on a PC only has a very limited time-slot to do its processing. Off-loading tasks to a dedicated MCU allows meeting the strict timing requirements and frees the PC's resources.

Serial Peripheral Interface (SPI)

The SPI bus is an electrical interface standard, connecting several MCUs that was developed by Motorola to minimize the number of pins needed to establish a connection. This bus provides a full-duplex communication between two devices at a certain time, which means that information can be transmitted in both directions at the same time, effectively doubling the speed at which information is exchanged. It is used to connect multiple devices (a single master and multiple slaves) together in a way such that only a single additional signal-line is needed per additional slave [9].

In a two-device configuration (as shown in Figure 1.1), SPI is comprised of four signals: A Clock signal (**CLK**), a Master-Out-Slave-In signal (**MOSI**), a Master-In-Slave-Out signal (**MISO**) and an optional Slave Select signal (**SS**). The **CLK** signal is used to make sure that data is written and read from the SPI bus by all devices at the same frequency. It is provided by the master and directly influences the speed of communication - a faster clock results in a faster communication. Data from the Master to the Slave is carried by the **MOSI** signal. For each bit that is transferred, the Master toggles the **CLK** signal and sets the **MOSI** signal to the value that should be transmitted. At the same time, the Slave sets the **MISO** signal to the bit-value that is to be transferred to the Master. The **SS** signal is used to signal a particular slave that it is active and should read the data on **MOSI** and, at the same time, write to **MISO**. For a two-device configuration, this signal may be omitted by asserting the **SS** input of the slave device at all times.

The SPI standard is based on bit-by-bit transmissions, thus no opinion is given on the order or meaning of data in larger packets such as bytes - this is left to the implementation. The same holds true for the format of data exchanged, as there is no single protocol defined and each application of SPI can use a different way of transferring information. This means, that while the electrical connections are described in the standard, without further knowledge of the devices on the bus, no conclusion can be drawn regarding the exchanged data.

In this thesis, the SPI bus used by the coffee maker's manufacturer is analyzed and interfaced with, establishing a connection with the electronics that operate the machine.

Direct Memory Access (DMA)

DMA describes a method of accessing memory for a MCU. Instead of processing (slow) accesses on the CPU, the work is off-loaded to a dedicated memory controller (DMA-Controller). This effectively allows connected hardware to access the memory directly, without the CPU being involved at all. Systems lacking DMA would need the CPU to perform these operations and would block it completely for the time it takes to communicate with the connected hardware. Additionally, the CPU needs to either wait for the external periphery to signal that data is available or it needs to periodically ask for the data - both of which would consume CPU time and defer other tasks the CPU is running [10]. Using DMA, this can be optimized so that the DMA-Controller can perform the time-consuming communication and the CPU is only notified once a transfer is finished and data is available. Meanwhile, the CPU can process other tasks and is not occupied at all with the transfer.

This thesis leverages DMA to communicate with the SPI-Peripherals, allowing the CPU to take care of other tasks while the communication is ongoing. By off-loading this work, the MCU can process several tasks while the DMA-Controller takes care of sending and receiving data through SPI.

Lightweight Directory Access Protocol (LDAP)

LDAP is a protocol for accessing information stored in a directory server [11]. It allows to consolidate information in a central directory and share this common source of data with multiple clients. In this thesis, LDAP is used as a way to authenticate users without storing their login credentials outside of a trusted central server. Instead of storing sensitive information externally, the LDAP protocol is used to securely identify users with the ComSys LDAP-directory server. Using LDAP, the server is queried for a user using the credentials the user supplies. If the server finds a user for these credentials, the information returned by the LDAP server is used to identify the user and store associated data. This allows the user to change the login information at any time in a central system and every client can verify the new credential at the time of the next login.

CHAPTER 2

Thesis Contribution

This chapter will lay out the proposed solution for achieving the thesis' goals. It will describe in detail the solution that was implemented and the requirements it needs to meet. Further, a series of experiments will be conducted to validate the design and the results will be evaluated.

In the section Concept, a plan is described to build a physical device along with software that runs on the device and a PC. Also, a set of hypotheses are formulated that show the extent to which the device fulfills the requirements.

The Implementation section shows the steps taken in order to implement the concept and explains decisions made along the way. As a result of this, a working prototype is presented.

This chapter is finalized by collecting, analyzing and evaluating data in a series of Experiments using the implemented design.

This thesis focuses on the machine that was available at the ComSys, the De'Longhi ESAM 6660 Coffee Maker. While this narrows the area of research to a small subset of coffee makers, it will be shown that similar coffee makers of the same manufacturer can be reverse-engineered in the same fashion and results can be adapted. Additionally, the concept can be adapted for different machines altogether as long as a defined set of functions is provided for the billing system. Thus, even machines that only remotely resemble the research subject of this thesis may benefit from the results.

2.1 Concept

To accurately bill users for the coffee they've actually consumed, a system that keeps track of the user's orders is to be developed. As input, this system receives the orders from each user, stores them for reporting and forwards them to the coffee maker for delivery. Each user can then monitor the current month's consumption and is reminded to pay for this consumption on the end of the month. The orders are sent to the coffee maker for fulfillment by the DBC, a hardware module that is developed as part of this thesis.

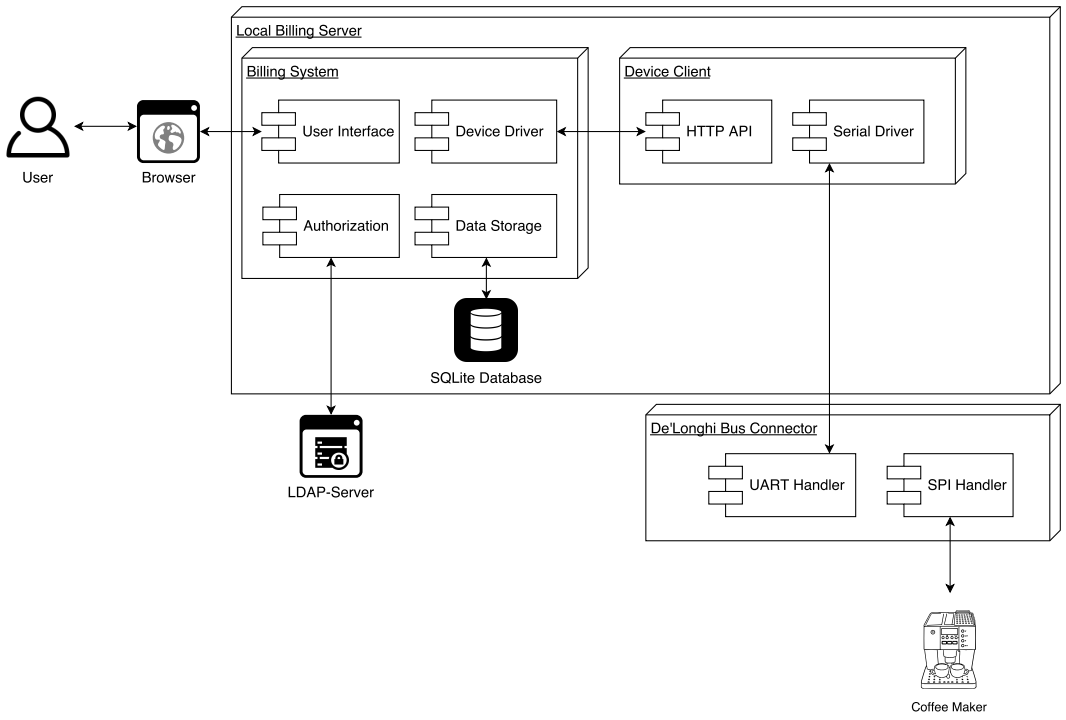


Figure 2.1: The modules the Billing System and the DBC are comprised of

2.1.1 The Billing System

For submitting orders, a web interface is created that provides an easy-to-use way to order the desired coffee. This web interface should present a list of products that can be ordered along with a price for each product. The interface should resemble a usual vending machine. When ordering a coffee, the user should be shown the current state of the machine. As the machine may need to heat up before brewing the coffee, this state may indicate for the user to wait before leaving to get the coffee. Also, any events that happen while the user's coffee is brewed should be displayed, such as an empty tank of water or coffee beans. Further, a queue will account for multiple orders arriving at the same time. If an order is not at the top of the queue and thus not delivered immediately, this is shown to the user with an indication of the waiting time required for the order. The waiting time can be indicated by showing how many orders are in front of the current order, so that the user may know when to leave for the coffee kitchen.

As soon as the order is executed and the brewing is finished, the web interface should return to normal operation so that a new beverage can be ordered.

The billing system should be accessible through a web browser for every user on the ComSys-network. Furthermore it should store a user's orders in a local database and send out a report of usage automatically every month to a dedicated billing user. The user information stored by the billing system must be minimal, that is, no personal information about the user should be stored. For authorizing and identifying users, the system is connected to the ComSys' LDAP server. This server stores the user's passwords along with other sensitive information. The billing system should only store an identifier, such as a user id or email

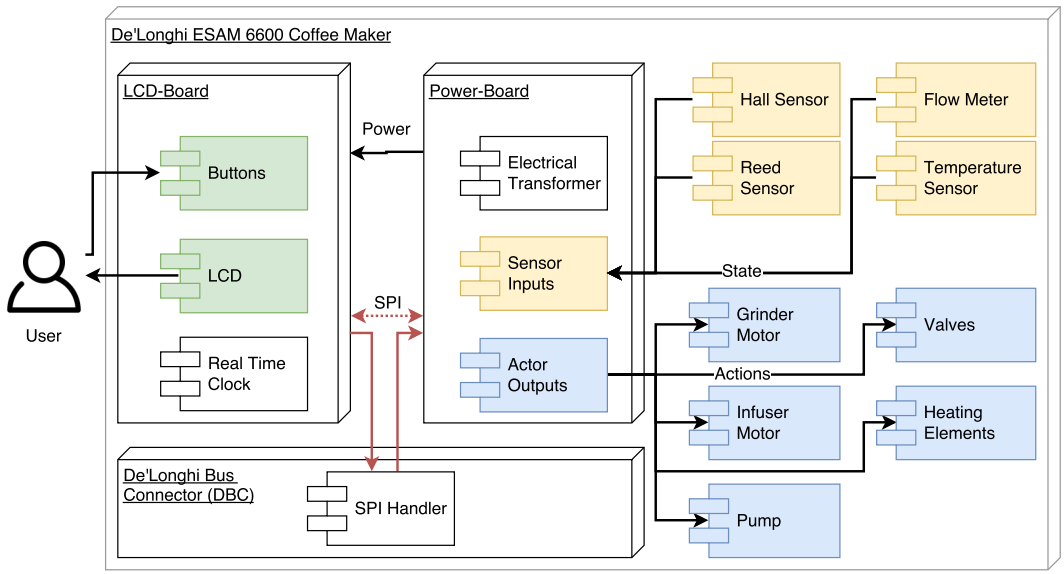


Figure 2.2: Internal separation of the De'Longhi ESAM 6660's subsystems. In green, all subsystems related to the user input are shown. All systems used for monitoring the machine's state are marked in yellow and all the systems that execute the brewing process are marked in blue. The internal communication bus is shown in red, the original connection is marked in a dotted line while the solid line shows the communication through the DBC.

from the LDAP server. This ensures that data in the billing system can always be treated as non-sensitve data and allows for relaxed security concerning the database.

The billing system should be created without making assumptions about the coffee maker's accessible functions. This means that instead of a static list of products, the billing system should allow a dynamic set of products to be configured. This list can be extended, should a different coffee machine be connected to the billing system.

To allow the billing system to be independent from the machine, an API is defined that is used to communicate between the two systems. This API is implemented in the billing system as well as the device client, which the billing system is directly connected to. This client is then responsible for translating the billing system's request into the device specific commands for the coffee maker.

2.1.2 The Coffee Maker's Internals

Internally, the De'Longhi ESAM 6660 is separated into different subsystems (Figure 2.2), each of which provides a unique function and is isolated from the rest of the system.

This separation serves several purposes: If problems arise with one part, it can be improved, while other parts stay unchanged. Also, parts can be reused across different models and new functions can be implemented on new components without the need to change existing designs. Additionally, producing the same part in larger quantities also allows for cheaper production resulting in a better return on investment for the manufacturer. This also positively affects the end-user: separated parts are cheaper to repair, as only single parts

Function	11-pin connector	8-pin connector
VCC (+5v)	3	1
GND	4	5
SPI MISO	7	6
SPI MOSI	8	7
SPI Clock	9	8

Table 2.1: Pinouts for the LCD-connector

can be replaced instead of the whole machine.

The coffee maker consists of these parts in order to fulfill its function:

- A **User-Interface**, allowing for input of orders and output of the current state (The LCD-Board)
- A **Water-tank**, that acts as a source of fresh water
- A **Bean-container**, that stores raw beans
- A **Grinder**, that grinds the beans
- A **Heater**, that heats up the water to a set temperature
- A **Pump**, that moves the heated water through the ground beans
- A **Waste-container** to dispose of the ground beans after the coffee's been extracted
- A **Power-board**, that provides power to all parts and controls the brewing process
- Several **sensors** that monitor water level, temperature and the state of case-doors

All of these parts are operated and monitored in a combined and precise manner, so that a coffee can be brewed matching the expectations of both the user and the manufacturer. This is controlled by the Power-Board's main processor, it receives all sensor inputs and takes care of opening and closing the water valves, operating the motors for grinding and the pump as well as enabling and disabling the heating elements. Through different combinations of the amounts of water, heat and coffee, all different coffee beverages that the machine supports (programs) can be brewed. Additionally, most programs support dynamic amounts of coffee (ranging from a light to strong brew) as well as water (adapting to differently-sized cups).

To connect the Power-Board and LCD-Board internally, they share a single SPI bus that is used to transmit their corresponding state to the other board. This bus is carrying all information necessary to both observe the machine's operation and to control it. Consequently, if this information is changed, the state and behavior of all components of the machine can be affected accordingly.

Using SPI allows the manufacturer to minimize the number of electrical connections that need to be made in order to transmit the inputs and outputs between the different boards. Thus they can be placed farther apart, as is evident in the De'Longhi ESAM 6660: The Power-Board is placed in the back of the machine while the LCD-Board is present at the front and center of the device. Running a large number of connecting cables through the machine would be impractical and thus an electrical bus was chosen instead, using only five

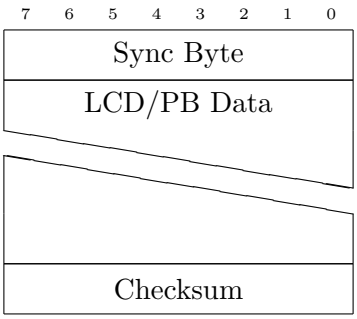


Figure 2.3: Packet Format of the De'Longhi Protocol

connections for both power and data transfer.

The SPI bus is present on the connecting cable between the two boards. Table 2.1 details the pinouts for two different styles of connectors that were encountered during this research. In both cases, **GND** inside of the machine is connected directly to the **N**-connector of the machine's power plug and there is no electrical isolation between them.

⚠ **Warning:**

When working on the insides of the machine, care must be taken to make sure the machine is plugged-in the right way around into the power-outlet, otherwise GND is tied directly to 220v instead of GND!

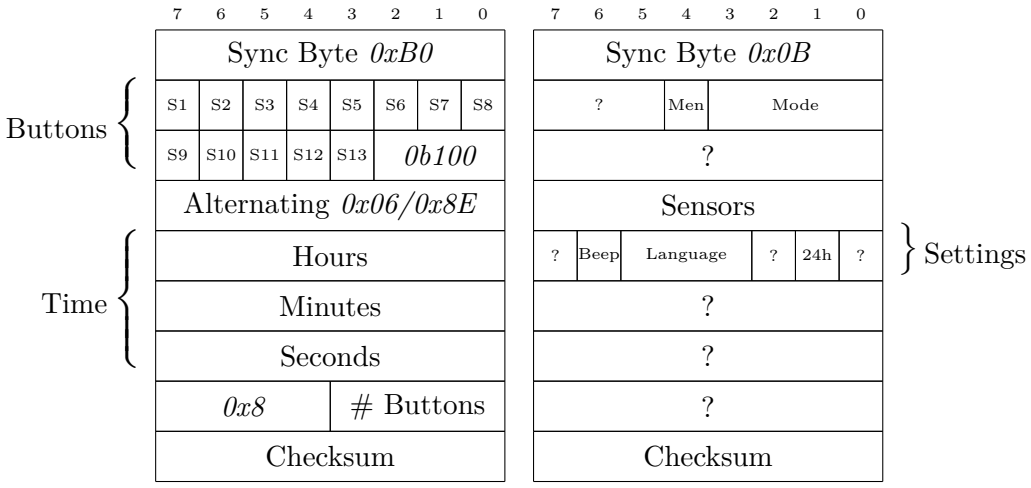
2.1.3 The De'Longhi Protocol

The coffee maker uses a proprietary protocol to transmit information over the SPI bus between the LCD-Board and Power-Board. This protocol encodes the state of each board into a binary representation that can be sent over the SPI bus. In this communication, the LCD-Board takes the role of the SPI-Master device, so it provides the clock and decides at which rate and at what time information is transferred. As SPI is a synchronous bus, the Power-Board transmits its state at the same time in the opposite direction.

The SPI bus carries information in the form of packets comprised of bytes - due to the synchronous nature of the bus, both the LCD-Board's and Power-Board's packets have to be the same length. Analysis of the communication's timing revealed that each packet consists of 9 bytes with the first and last dedicated to a sync byte and a checksum respectively. The *sync byte* differs between the LCD-Packets and PB-Packets and allows to uniquely identify each packet-type.

Each packet is ended with a *checksum* that allows to verify the received data. This is used to ignore packets that contain transmission errors, such as flipped bits. As the length of the communication cables is relatively short, the main source of transmission errors would be improper shielding from external noise. The use of a checksum mitigates this, as errors can be detected (but not corrected) and erroneous packets can be discarded by the receiver.

Figure 2.4 shows all currently known fields of both the Power-Board and LCD-Board data



a) Packet Format of the LCD Data b) Packet Format of the PB Data

Figure 2.4: Packet Formats for PB- and LCD-Data

packets. For the LCD-Data, the meaning and function of each bit is decoded as the board simply encodes the current state of all the buttons plus the current time in each packet. There is no state held inside the LCD and thus the packet's bits have the same meaning all the time. For each button on the front panel, there is a dedicated bit in the LCD-packet encoding its state. Further, at the end of the packet, a count of all pressed buttons is added. An alternating byte allows the Power-Board to differentiate consecutive packages of the LCD-Board. This allows the Power-Board to know when communication is stalled or the LCD-Board fails, so when this alternation is broken and packets are the same consecutively, the Power-Board enter a failsafe-mode and shuts down. The realtime-clock implemented on the LCD-Board is supplying the machine with a precise time that is used to wake-up the machine at set time and also to show the current time on the front display. To let the Power-Board know about the current time, it is encoded into the LCD's packets as a plain hex-number for hours, minutes and seconds.

The Power-Board does have a state that is encoded into the packet, changing the functions of certain bits according to this state. Other parts of the packet have a constant meaning, as shown in Figure 2.4. The operation of the machine is only observed in a very limited way, as only a few information is needed from the machine to know if a request for a beverage was successful. Thus, the Power-Board's packets were not fully decoded and only necessary information was analyzed and is known at the moment.

The protocol defines a set of very strict timing requirements, shown in Table 2.2. As data transfers are always initiated by the SPI master, the LCD-Board, it is responsible for making sure that the clock is run a the correct frequency and that delays between bits and bytes are respected. The SPI standard makes no assumptions about the way data is transferred so usually "a continuous clock signal and an arbitrary transfer length" is used [12]. For the De'Longhi SPI bus this does not hold true - instead of a continuous clock signal, the clock is halted after each byte that is transferred ("Delay between two bytes") and after a full packet of 9 bytes ("Delay between two packets").

Description	Measured Value
Clock frequency	125 kHz
Transfer of a single byte	60 µs
Delay between two bytes	2260 µs
Transfer of a single packet	23 ms
Delay between two packets	36 ms

Table 2.2: Timings for the SPI Bus

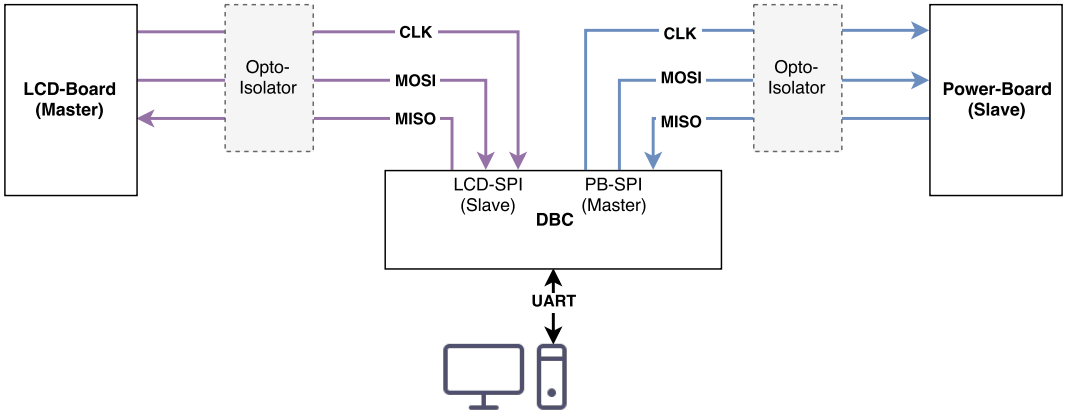


Figure 2.5: The DBC with all external connections

Hypothesis 1 (H1): *Any device connecting to the LCD-Board and Power-Board must adhere to these timings as using an uninterrupted clock signal would prevent any communication from being established.*

2.1.4 The Connecting Device

In order to gain control over the data that is sent through the internal control bus, a device is developed that is placed between the Power-Board and LCD-Board (Figure 2.5). This device consists of a microprocessor that has at least two SPI ports. On one port, the device receives data that the LCD-Board sends and transmits the data that it received from the Power-Board. On another port, it sends data to the Power-Board and forwards the response to the LCD-Board. Further, the device also provides a PC connection via a text-based serial port that is used to control the device and through it the coffee maker. Both of the SPI ports are optically isolated from the coffee maker so that any high voltage inside of the machine is not exposed through the USB to an outside PC.

This Man-in-the-Middle (MitM) approach allows for the following operations on the bus:

1. **Observing** the communication and forwarding it to a PC for logging
2. **Emulating** parts of the communication, such as the Power-Board's part, for development and testing

3. **Controlling** both directions of the communication, i.e. injecting state into both the Power-Board and the LCD-Board
4. **Reacting** to changes in the communication, such as input events on the LCD-Board

The first function will provide a way of gathering data from the device's internal bus. This data can then be analyzed on the PC and changes in the data can be associated to changes in state of the machine. As an example, input to the machine via the LCD-Board will represent a change in state.

Hypothesis 2 (H2): *User input changes the data sent on the Bus that is captured using the observe-capabilities of the device.*

For development and testing, the device will support emulating either part of the two-way communication. That is, either the Power-Board or the LCD-Board can be connected while the other part is unconnected or powered off. This way, one part of the communication can be controlled by the user, rather than the machine, and reactions by the hardware can be observed and associated to the changes.

Hypothesis 3 (H3): *Changing the data changes the hardware's state by reacting to it, such as brewing a coffee.*

Hypothesis 4 (H4): *The state defines which operation is performed by the hardware.*

The devices' normal operation is to capture the LCD-Board's data, alter it according to the PC's input, and then forward it to the Power-Board and the other way around. If no input was received from the PC, the device should not alter any data and instead act as a transparent forwarding device (*Pass-through mode*). This ensures that operation of the coffee maker is not impacted as long as the device was not instructed otherwise.

Hypothesis 5 (H5): *Operating the connecting device in pass-through mode does not impact the coffee maker's function.*

While the PC instructs the device what data to alter, it is the devices' responsibility to notify the PC of changes in data in the machine. These changes have different origins: a user input (such as pressing the power button) will change the LCD-Board's data while a changed sensor state (for example removing the water-tank) will alter the data from the Power-Board. Forwarding every packet to the PC would result in high traffic with very low information content, occupying the PC with detecting changes rather than reacting to them. Thus, a method to register for changes will be implemented so that traffic on the PC-interface is kept to a minimum. This lets the PC react in real-time, as every incoming change represents an important event that can be parsed and reacted upon.

Hypothesis 6 (H6): *The connecting device notifies the PC of any changes in state it registered for.*

2.1.5 The PC Interface

For interfacing with the MitM device from a PC, the device provides a Serial-over-USB port. A custom protocol is implemented on this port for configuration and data manipulation.

This interface is to be used to transfer the current state of the machine to the PC as well as receive state changes from the PC. Special care must thus be taken to make sure the amount of data transferred over this interface is minimal. As such, this interface needs to allow registering for changes in the state of the machine and provide a way to manipulate the traffic between the Power-Board and LCD-Board.

As the DBC is bound to strict timings on the SPI bus, packets must be manipulated on the device itself rather than the PC. Therefore, this interface needs to support setting up a set of manipulations independently from the process of receiving packages. These manipulations can be setup by the PC whenever a change in state of either the Power-Board or LCD-Board is requested.

Furthermore, this interface will represent a way for aiding in reverse-engineering other devices of the same family. It should thus be usable by humans, as well as a computer program. A text-based Command Line Interface (CLI) should be provided with a command set that can be used to explore the function of the connected coffee maker even further.

2.2 Implementation

This chapter details the implementation of the concept described before. The billing system will be described and it will be shown how it was separated into different modules, each providing an isolated function. This chapter will further depict how the gained knowledge about the coffee maker's function was utilized to develop a device that connects to the machine's internal bus and explain the software powering this device. Finally it will document a way of interfacing with this device from a PC and what functions can be accessed through this interface.

2.2.1 The Billing System

The billing system was implemented as several isolated modules. This allows the implementation to be extendible in the future and easily maintained. As part of this thesis, the following modules are developed (see also Figure 2.1)

- A web-based user interface
- An authorization module that implements the LDAP standard
- A data storage module that persists data on the billing server
- A device driver that can interface with a device client

These modules and their relation are explained in the following sections. All modules are implemented in Javascript and run on the billing server using the server-side Javascript implementation node.js as a runtime.

The Web Interface

Users interact with the billing system by ordering a coffee. To make this process seamless, an interface is developed that resembles a vending machine. A user can choose from a list of coffee products and is walked through the process of brewing. Any situation preventing the machine from brewing is displayed in the web browser and the user can take the appropriate action, such as refill water or coffee beans.

The web interface is implemented in Javascript using a framework for single-page applications, React.js. Using this framework, the web interface can react to any input immediately without making the user wait to load another page, making it work like a native PC application while staying in the browser and being compatible with almost any client device. While designing and building the interface, it was made sure that it would work well on mobile devices. This ensures that users can order coffee not only from their desk but also from anywhere they have network access using their mobile.

Integrated into the web interface is a view of the user's consumption. The consumption is calculated from each order and displayed as a table showing the date of the month along with the orders that happened on this day. This view allows every user to check on their current consumption, as the paper list would before, but clearly lists the dates and the details for each order.

Authorization

For users to login to the web interface, they need to supply the username and password associated with their ComSys-account. The login credentials are not stored in the billing system but rather sent to the ComSys-LDAP server which verifies them. Then the billing system is notified about whether or not the credentials are valid. Should they be invalid, access to the billing system is denied and the user has to login again. Otherwise, the user is shown the order page and can start ordering coffee.

No registration is necessary and a user is created in the database as soon as the first successful login takes place. This makes sure that every order placed is associated to a user known to the ComSys faculty. As billing is based on a post-paid system, that is a user can order as much coffee as desired and only pay for it at the end of the month, a certain trust is required to allow access. This trust is verified by the LDAP-Server - if a user does have an account on this server, the billing system trusts that the user will pay for the consumed coffee.

The authorization module is based on the open-source LDAP implementation ldapjs (`http://ldapjs.org`). It takes care of the low-level details of the protocol and exposes a Javascript client to interact with LDAP servers. Thus, the module could be reduced to a thin wrapper around the library that validates an incoming login-request from a user and returns the unique identifier from the LDAP server.

Data Storage

Data is stored locally on the billing server in a file-based SQLite database. This data is comprised of product data for each of the coffee maker's beverages and all users that ever logged in with their respective orders. For each user, only a unique identifier supplied by the LDAP server is stored (such as the user's email address). This allows the billing system to associate orders to a specific user while keeping the database clean of any sensitive information such as the user's password.

Using a simple file-based database such as SQLite allows to save resources on the billing server. Still, the data store is built such that it can be replaced by a more sophisticated variant should the amount of data require it. Until then, it allows for efficient storage and retrieval of all billing data and enables a pleasant user-experience.

Device Driver

The billing system includes a driver for the physical coffee maker. This abstraction of the hardware allows different coffee makers to be interfaced by the billing system, regardless of their protocol or the way they connect to the billing server. For this thesis, a single driver is implemented that connects via HTTP to a device client running on the same machine as the billing system.

The driver takes care of initializing the device client and translates state from the device client (such as changes in sensors) to the billing system. Further it queries the device client for a list of products the connected coffee maker supports and exposes this information to the billing system.

As no assumptions are made about the coffee maker itself, the driver can interface with a set of device clients as long as they implement a common API.

2.2.2 The Device Client

For this thesis, a device client was implemented that connects to the coffee machine through the DBC and communicates with the device using a serial port. This client takes care of managing the DBC's state and reports changes in state back to the billing system. These changes include both external user input at the coffee maker and changes of state internally in the coffee maker, such as low levels of water or beans. As soon as a change in state is reported to the billing system, it evaluates whether the current state would allow for orders or if it would prevent them. If so, the user is notified and an error is displayed on the machine through the device client and the DBC. As long as the state prohibits new orders, every user trying to place an order is notified accordingly in the process of ordering. When the state changes and coffee can be ordered again, the billing system will restore its normal operation and clear the display at the coffee maker.

The device client is also implemented in Javascript and runs on the billing server, using the native serial-port bindings available in node.js.

2.2.3 The De'Longhi Bus Connector (DBC)

The DBC provides the hardware-interface between the billing server's serial port and the De'Longhi ESAM 6600 coffee maker. As it needs to keep the communication on the SPI bus inside of the machine alive by arbitrating between the Power-Board and LCD-Board, it was decided to off-load this task into a dedicated MCU.

The Hardware

For the connecting device, the `STM32F407VG` MCU from STMicroelectronics was chosen. This MCU incorporates a set of features [13] rendering it ideal for the intended application:

1. A **168 MHz** CPU
2. Up to three high-speed **SPI** Ports (two of which are used) that support either master or slave mode
3. Up to four **serial** ports (one of which is used)
4. **Low-power** operation with no requirement for cooling the MCU
5. **Embedded 1 Mbyte Flash** to store the firmware
6. **Internal** oscillator and support circuitry, so it can be run with just power

The MCU is running a version of the `STM32Cube` framework provided by STMicroelectronics and the firmware is developed in C. The framework provides a set of low-level drivers that allow using the available hardware in a straightforward way. Further, it includes a configuration tool that assists in setting up hardware timers as well as disabling hardware components that are unused. Using this abstraction layer allowed focussing on using the

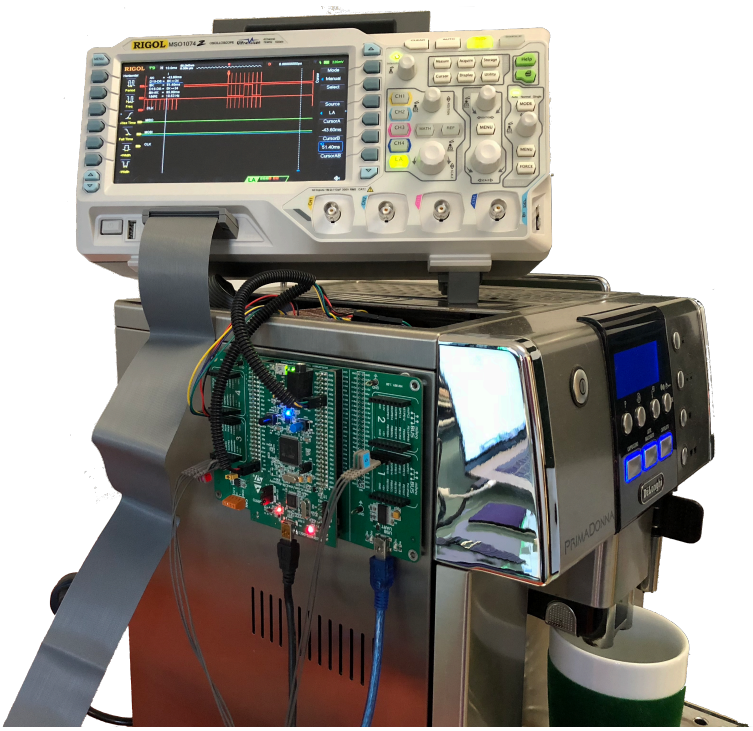


Figure 2.6: The prototype used for this thesis, using the STM32F407 evaluation board along with an breakout module for easy access of all SPI-busses, connected to the machine as well as a logic analyzer. The final board will be comprised mostly of the CPU and electrical isolation.

MCU's hardware as intended while reducing the initial amount of work required for setting up all peripherals such as the DMA controller.

The Firmware

For its main task, interfacing the two SPI busses for the Power-Board and LCD-Board, the MCU's firmware employs a single state machine (detailed in Figure 2.7). This state machine keeps track of the current state of communication by making sure each part is executed in order. Should any part of the communication fail, it is retried. If the error persists, the state machine enters an error state that is indicated to both the LCD-Board and the PC interface. The former is sent a state that shows as "General Error" on the display while the latter is sent an unsolicited error message so the billing system is informed about the error state. The billing system can then notify an administrator and prevent further orders by queueing, instead of executing them.

Before entering the main task, the connecting device needs to synchronize with both the LCD-Board and the Power-Board. First, the communication with the Power-Board is established. As the connecting device represents the SPI master in this communication, it will control and provide the SPI clock signal. Thus, the device controls when to send out data and in turn when the Power-Board will respond. Should no communication be

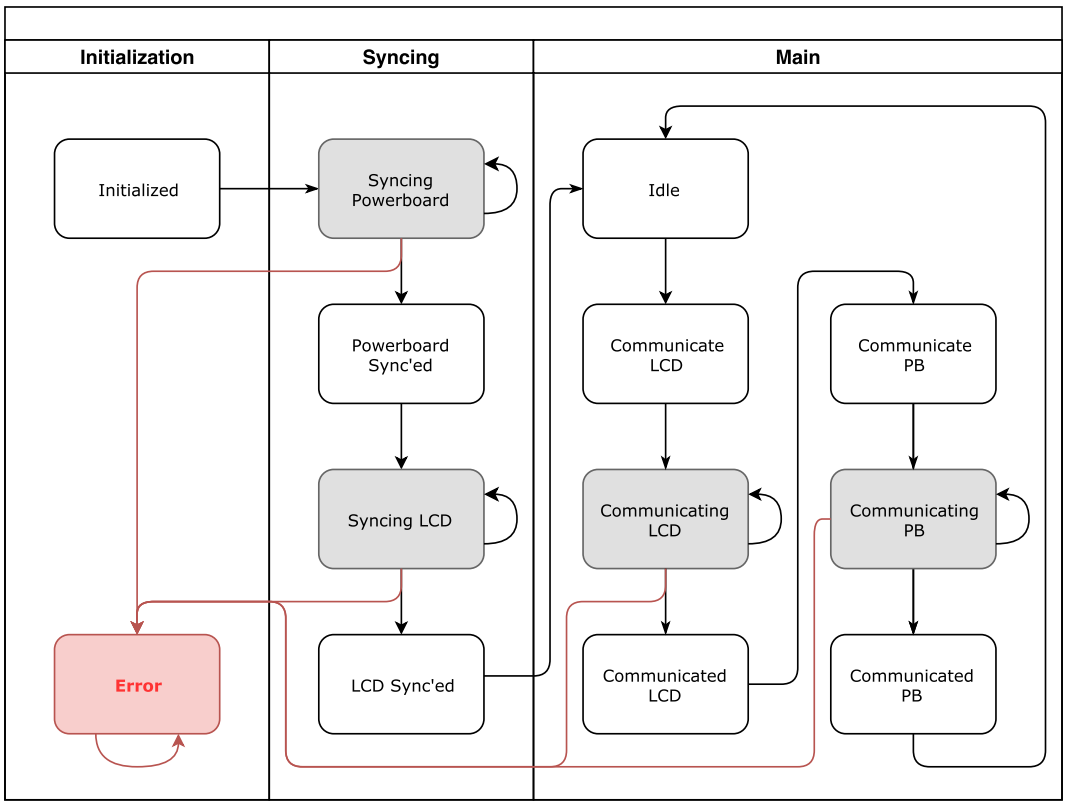


Figure 2.7: The state machine describing the DBC's operation

received from the Power-Board, then the process will not continue and the device will wait until the Power-Board responds. This ensure that the whole system is in a known state before communication between Power-Board and LCD-Board is established.

For the LCD-Board, the connecting device enters the role as the SPI slave, that means the LCD-Board will provide the clock signal. For the device to know when a transfer from the master has finished and a new one begins, it must receive the data and wait for the sync-byte. Once the sync-byte is received, the *Communicate LCD* state is entered and communication with the LCD-Board is handled through DMA.

Both the communication with the LCD-Board and the Power-Board is implemented with DMA. This reduces the processor's workload by shifting the work of transferring bits via SPI to a dedicated DMA-Controller. The DMA-Controller is configured for a certain speed and mode of SPI communication, imitating the coffee maker's communication mode, this mode is known as *SPI Mode 3*. Then, it is supplied with two regions in memory: one for receiving data (*RxBuffer*) and one for transmitting data (*TxBuffer*). Data that should be sent out is then written into the TxBuffer by the CPU. As soon as the DMA controller is instructed to transmit and receive data, control is yielded back to the CPU so that it can execute the next tasks. When the DMA transfer is completed, a callback method is called by the DMA controller. This informs the CPU that the transfer is complete and let's it update the state machine's state.

In the connecting device, this time is used for communicating with the PC and handling

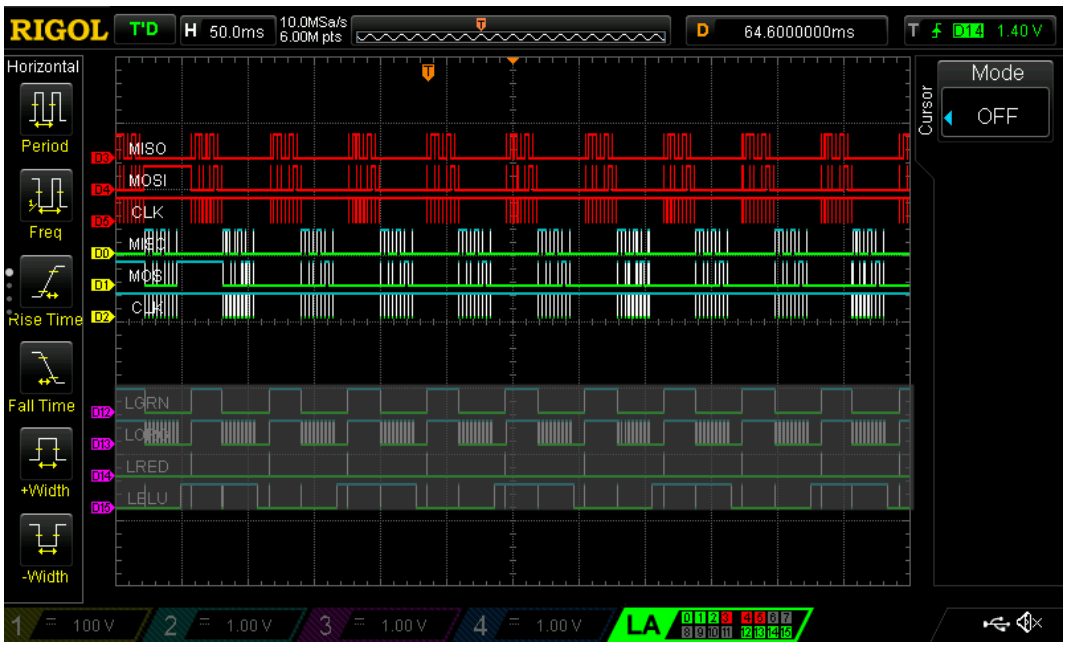


Figure 2.8: Screenshot of a logic-analyzer, showing the communication between the LCD-Board and DBC (red arrows) as well as the Power-Board and DBC (yellow arrows). The purple arrows show different duty cycles of the DBC's state machine, used for debugging the utilization of the MCU's CPU.

input through this interface. This is again implemented with DMA. In contrast to the SPI communication on the LCD-Board and the Power-Board interfaces, the PC interface is setup for serial communication. The DMA controller is setup differently as well: Instead of transmitting single packets of data, the PC interface's DMA controller is setup to write into a circular buffer. This means that a region of memory is allocated for receiving and the DMA controller is instructed to write every bit received via the serial port to this region. While writing, a pointer inside the DMA controller keeps track of the current position. Once the end of this region is reached, the DMA controller will reset the pointer and overwrite the data that was written to the beginning of the region.

It is the CPU's responsibility to make sure to read the buffer often enough to prevent loss of data from happening. The buffer is setup to hold 128 bytes of data and the CPU is interrupted to process the data every 50 µs if there's new data in the buffer. To determine whether new data is available, the CPU checks if the serial port has been idle for at least 50 µs, so as to keep a single transfer from the PC intact. Then, it parses the buffer and executes the parsed commands.

SPI Communication

Figure 2.8 shows the DBC communicating with both the LCD-Board and the Power-Board at the same time. As SPI is a synchronous bus, the DBC receives a full package from the LCD-Board before it can communicate with the Power-Board. This means that the commu-

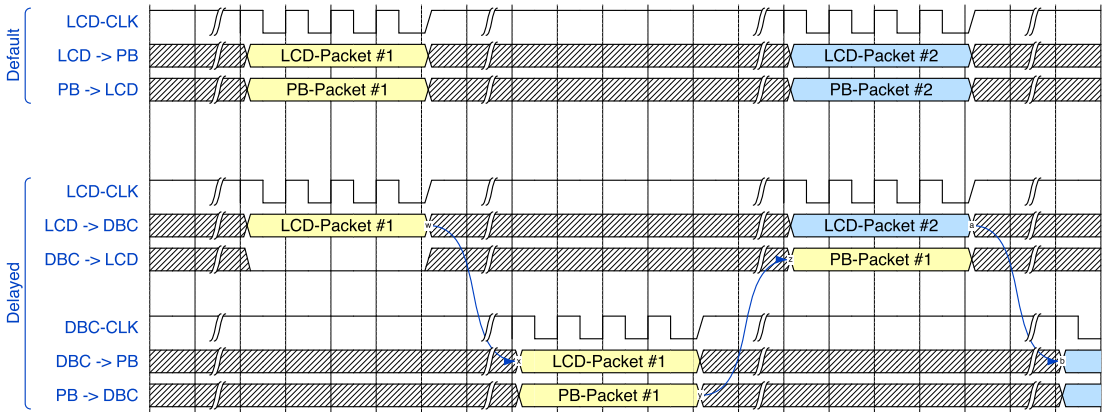


Figure 2.9: Explanation of the delay of packets due to the DBC's presence.

nication is delayed by one packet (see Figure 2.9) and that the LCD-Board will only receive the Power-Board's state one packet later than without the DBC in place. As the LCD-Board's SPI-clock cannot be halted or otherwise influenced, delaying the communication is the only way for the DBC to be able to change it on the bus.

The PC Interface

For controlling the device, a set of commands is implemented that can be executed through a serial port the MCU offers. The device can be connected to a standard USB connection and will enumerate as a serial port. These commands can be controlled both by a human using any terminal emulator or by a program taking control of the serial port. Each command is identified by its first character, followed by a number of arguments with differing lengths.

The following commands are implemented:

- `r` - *reboot* the device by resetting the MCU
- `D` - *enable* unsolicited debug output for each packet received and sent
- `d` - *disable* unsolicited debug output for each packet received and sent
- `p` - *print* the last packet received and sent
- `b<packet>` - load a packet's data into the working *buffer*
- `t<buffer>` - *transfer* the working buffer into the specified destination buffer
- `T<buffer>` - *clear* the specified destination buffer

The connecting device contains a set of buffers, the main buffers being the transmit and receive buffers used for communicating with the LCD-Board and the Power-Board. In order to allow modifying the data sent to the machine's subsystems, there are buffers for applying bitmasks to the respective transmit buffers. Before the transmit buffer is supplied to the DMA controller, both an AND- and an OR-bitmask are applied to the current transmit-buffer. This allows the connecting device to modify either just parts or all of the package that is sent out to the machine. Consequently, if these override-buffers are empty (full of `1`s for the AND- and full of `0`s for the OR-buffer), then the data is forwarded between the two devices unaltered.

In order to minimize traffic on the PC interface, a function was implemented to allow for the filtering of packets that are transmitted to the PC. As only certain bits contain information that is relevant for billing, the connecting device may only notify the PC as soon as any of these bits change. This function was implemented by keeping track of the last received package to which the log mask is applied. Using the log mask, any bits the PC is not interested in being notified about can be flipped to `0`. As a new package is received, the log mask is applied to the incoming packet and then the result is compared against the stored and masked package. If changes are detected between these two packets, an unsolicited notification is sent via the serial interface so the PC can know that the bits it is watching changed.

Data can be loaded into the connecting device via a working buffer (command `b`). This working buffer can then be transferred into any of the following buffers:

- `0` - Overwrite PB data (to be sent to the LCD), full package
- `1` - Overwrite PB data (to be sent to the LCD), AND-mask
- `2` - Overwrite PB data (to be sent to the LCD), OR-mask
- `3` - Log mask for the data received from the LCD
- `5` - Overwrite LCD data (to be sent to the PB), full package
- `6` - Overwrite LCD data (to be sent to the PB), AND-mask
- `7` - Overwrite LCD data (to be sent to the PB), Or-mask
- `8` - Log mask for the data received from the PB

Every interaction using the commands laid out is either confirmed or denied by the controlling device using a text response.

2.3 Experiments

In this section, a series of experiments is described and performed to validate the device's functions and capabilities according to the concept. It is shown that the device functions as expected and that the hypothesis that lead to this thesis proved to hold true. Further the limitations of the device as well as the coffee maker itself are presented and evaluated. Finally, a set of measurements is conducted whose results provide valuable input data for the billing system.

2.3.1 Verifying the DBC's capabilities

As part of the thesis at hand, the DBC was developed. The device provides a set of complex functions which are presumed to be working as expected. To verify the correct function of all capabilities, a set of validation-experiments is designed and executed.

Experiment Description and Expectations

This experiment comprises of a set of validations that are described in detail in the following sections. While each validation is run separately, due to their similar nature all are consolidated into a single experiment.

A **pass-through test** is conducted by connecting the DBC in-between the Power-Board and LCD-Board, so that any communication is routed through the device's software. To validate the correct function, every button on the machine's front is pressed and it is observed if they serve their original function as intended by the manufacturer. For this validation, it is expected that every function is preserved even when the communication is passed through the DBC.

For **functionally testing** the DBC's *observe*, *interception* and *injection* capabilities, a set of actions is performed on the PC rather than the coffee maker. These actions must be reflected by the coffee maker after the corresponding packets have been transmitted by the device. Every function that results in a beverage being brewed is tested and validated and the DBC's output is monitored to make sure it includes the observed state changes of the machine. This test expects that the device is capable of requesting every coffee that the machine supports and that it can monitor the machine's state accordingly.

Experiment Setup

For all validations, the machine is brought into a known state. This is achieved by resetting the machine to factory defaults according to the user manual [14].

The DBC is connected to a PC using the serial interface and a terminal emulator is launched on the PC to interact with the DBC. Using this connection, the DBC is setup to notify the PC via *filters* - the LCD-Board filter is setup to `00000000000000FF00` and the Power-Board filter is set to `FFFFFFFFFFFFFFFFFF`. These filters will notify on any input to the machine via the buttons and also on all changes of sensors and brewing-mode. Through these filters, the current state of the coffee maker can be read from the terminal emulator running on

the PC in real-time.

Execution

First, all functions of the machine are performed without the DBC installed, so the machine is unmodified. These measurements serve as a baseline to compare the machine's behavior with and without the DBC in place. During this execution, no logging data is received from the DBC and instead only the machine's reaction is evaluated for the results.

After every function is executed, the DBC is installed in **pass-through** mode. The machine's LCD-Board and Power-Board are disconnected from their shared bus and the DBC is inserted into the bus. This separates the LCD-Board and Power-Board electrically and all communication now happens through the DBC. **Pass-through** mode is the default mode after power up of DBC, and the filters described in Experiment Setup are still active. Each function is run, again from the user interface of the machine, and the response of the machine is recorded. At the same time, the DBC's output is monitored in the terminal emulator running on the PC. For every function, both the button-press in the LCD-Board's packet and the resulting change in the Power-Board's state is expected to be sent to the PC.

Next, each function is initiated from the PC by using the endpoints served by the Device Client. A set of HTTP-Requests is made to run each function and the machine is monitored to record the performed action. For a test to pass in this validation, the display must show the current state of the machine correctly, even though the state change was initiated from the PC via the DBC as the LCD-Board is not involved in starting a function.

Results

Table 2.3 shows the protocol for each tested function. Every function that resulted in the expected result, such as a coffee being brewed for a coffee order, is marked as **OK**. **N/A** marks functions, that are not available in the specific way of operation. For example, there is no way to initiate the *Warmup* state from the PC - instead the Power-Board enters this state upon power up on its own. The same applies to sensor values as they are sent by the Power-Board and cannot be changed. For functions that could not be run due to errors, **NOK** is shown in the log. The results show that ordering coffees with milk is not working in either configuration. This is caused by the machine requiring additional hardware (a milk container) in order to allow for these orders to be brewed. For the experiment, this hardware was unavailable and thus the functions could not be run successfully.

The baseline (*without DBC*) shows that all functions (except for the orders with milk) could be run and the same holds true for the DBC in *Pass-through* mode. Still, every function of the machine worked as expected and produced the desired result.

Initiating functions from the PC works for all functions that could be triggered externally, except for *operating the menu*. This is caused by the coffee client not exposing functions for operating the menu, which in turn is caused by the complex state handling needed to reflect the menu on the LCD-Board's display. As this does not influence the order process for coffee it is disregarded for this thesis.

Function	Without DBC	Pass-through	Initiated from PC	Monitored on PC
Power				
On	OK	OK	OK	OK
Power				
Off	OK	OK	OK	OK
Monitor				
Warmup	OK	OK	N/A	OK
Monitor				
Ready	OK	OK	N/A	OK
Monitor				
Empty Beans	OK	OK	N/A	OK
Monitor				
Empty Water	OK	OK	N/A	OK
Monitor				
Open Door	OK	OK	N/A	OK
Order				
One Short Coffee	OK	OK	OK	OK
Order				
Two Short Coffees	OK	OK	OK	OK
Order				
One Long Coffee	OK	OK	OK	OK
Order				
Two Long Coffees	OK	OK	OK	OK
Order				
Hot Water	OK	OK	OK	OK
Order				
Cappuccino	NOK	NOK	NOK	NOK
Order				
Latte Macchiato	NOK	NOK	NOK	NOK
Order				
Caffelatte	NOK	NOK	NOK	NOK
Order				
Hot Water	OK	OK	OK	OK
Initiate				
Cleaning	OK	OK	OK	OK
Change				
Grind-Mode	OK	OK	OK	OK
Change				
Time	OK	OK	N/A	OK
Operate				
Menu	OK	OK	NOK	OK
Disable				
Buttons	N/A	N/A	OK	OK

Table 2.3: Test protocol for the coffee makers's functions

Every function was observed on the PC, as indicated by *Monitored on PC* meaning that every change in state was communicated to the PC and can be analyzed for billing.

Evaluation

The results show that every function needed for operating the machine to receive brewed coffee is available with the DBC installed. This proves that no loss-of-function occurs due to installing the device. It also proves that the pass-through mode of the DBC works in a way that is completely transparent to the coffee maker, allowing for full access to the coffee maker's functions and confirming hypothesis H5.

By observing the changes of data sent between the LCD-Board and Power-Board, hypothesis H2 was proven to be correct: every user input event lead to a change in the data sent by the LCD-Board. When, for example, a coffee was ordered by the buttons on the front panel, it could be observed that this change in data resulted in the machine changing its state accordingly: the machine began brewing a coffee. This proves hypothesis H3 to be correct as the hardware reacts to changes in state. Ordering different kinds of beverages, it is possible to observe the hardware reacting differently. This means that the Power-Board evaluates the new state and interprets different states as orders for different products, proving H4 to be accurate.

Additionally, every function can be monitored on the PC which makes it possible to to replicate the state of the machine in software. Using the notification function by setting up filters proved H6 by showing that the DBC notifies the PC of any changes in state the latter is registered for. This enables the billing system to know the exact state of the coffee maker, i.e. all sensor states and the current mode of operation. The information can be used for interacting with the users to only allow specific functions based on the state. For instance, ordering a coffee can be delayed when the machine indicates missing water - instead, the user can be notified about this and refill the tank. As the billing system is notified about the state change by the machine, it can then schedule the user's order as soon as the tank has been refilled.

Finally, the DBC allows for *Disabling Buttons* by setting the override-filter of a button. In an office environment, the coffee maker's settings (such as water hardness or amount of water per coffee) should be immutable by the users. Without the DBC, a user could operate the coffee maker's menu without any restrictions and thus change any setting. The filtering capabilities of the DBC allow to restrict the user to a set of actions that are deemed to be safe. By disallowing the use of the menu button (P), a user can only order coffee at the machine and not change any settings. Additionally, in order to make sure the machine is always ready-to-use and not switched-off accidentally, the power-button can be filtered on the DBC so pressing it would never send a signal to the Power-Board.

2.3.2 Optimizing the SPI communication speed

Previous tests assessed the actual function of the DBC, that is the device as a whole working in a known and expected environment. As explained in section 2.2, the coffee maker's LCD-Board is responsible for the timing of all SPI communication when connected to the Power-Board. The LCD-Board will send and receive packages every 60 ms, strongly

limiting the time for the DBC to interface with the Power-Board, as the communication with the latter takes at least 32 ms. Most of this time is spent in the delay time between single bytes sent to and received from the Power-Board - in fact the actual transmission only occupies a very, very small fraction of the entire time taken for a single packet. The actual transmission takes 600 µs of 32 ms or less than 2% while the remaining time is spent waiting for the Power-Board to process the data.

Through introducing the DBC, the Power-Board is no longer connected to the LCD-Board and instead the DBC takes the role of the SPI-Master. This allows the timing to be modified, as the DBC can generate an arbitrary SPI clock to control the data flow in- and out of the Power-Board. By reducing delay times and thus speeding up the communication between the DBC and the Power-Board, the DBC can be allowed more time for processing in-between the SPI-communications.

Experiment Description

In this experiment, the DBC generates two different types of clock signals. The first is emulating the LCD's clock: between sending every byte, the DBC would wait an additional 2260 µs. Using this mode, a single packet takes about 23 ms to transfer.

The second is using the default SPI clock and does not wait between bytes, reducing the time for a single packet to about 0.6 ms. During the saved 22.4 ms, additional work could be processed by the MCU such as interfacing with the PC.

Expectations

It is expected that the Power-Board behaves like a usual SPI-slave in that it adheres to the master's clock. This means that the packet transfer is expected to work using both modes and that the Power-Board should be able to process bytes and respond using the default, uninterrupted clock signal.

Experiment Setup

For this experiment, the Power-Board is connected to the DBC while the LCD-Board is not. The DBC's firmware is modified so that no sync is established with the LCD (as it would fail). Further, it is modified such that the PB's SPI clock can be adjusted via the PC interface. No filters have been setup and every packet is logged to the PC interface.

To analyze the timings, a logic analyzer is connected to the SPI busses Clock, MISO and MOSI lines. This will allow a visualization of the lines' states at each time.

Execution

The DBC is connected to the PC and the machine is connected to power. At the PC, the communication is monitored while the DBC syncs with the Power-Board.

After the DBC successfully synchronized to the Power-Board, hot water is requested using the PC interface.

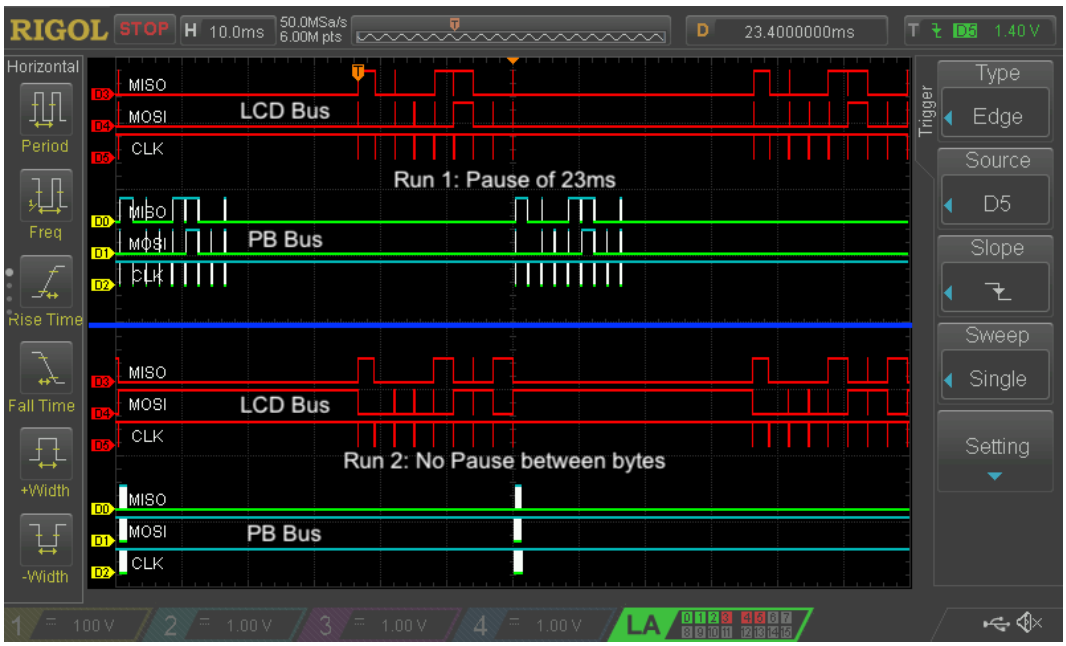


Figure 2.10: Screenshot of a logic-analyzer, showing the timing comparison between the manufacturer's clocking and continuous clocking.

Next, the DBC is reset and the the power to the coffee maker is turned off. The DBC's firmware is switched to the optimized mode and the machine is turned on again. At the PC, the synchronization process is monitored again.

Results

The DBC was able to successfully synchronize with the Power-Board using the manufacturer's timings. Additionally, commands could be sent so that the functions of the Power-Board could be operated from the PC.

Using the optimized timings, the DBC was unable to synchronize with the Power-Board. Instead of a successful communication taking place, every packet received from the Power-Board contained invalid checksums. Thus, no commands could be sent and it was not possible to order hot water at all.

Upon further analysis of the logic analyzer's data, it turned out that the Power-Board returned the same data that was sent by the DBC but offset by one byte. This meant that all packets were invalid and contained the DBC's state rather than the Power-Board's.

Evaluation

This experiment showed that the timings used by the manufacturer and documented in Table 2.2 are critical and cannot be changed, proving hypothesis H1. It is presumed that the delays are in place so that the MCU located on the Power-Board is able to process the incoming data and poll sensors in real-time (rather than keeping the state cached in

memory). As the delays are immutable, the DBC needs to leverage other techniques to make sure that enough processing time is available so that it can keep up with both the LCD-Board's and the Power-Board's SPI communication. To make sure that the PC interface's input can be processed, all SPI communication needs to be handled by the DMA-controller available on the MCU. This ensures that all timings can be met, as the CPU is freed from waiting for the SPI delays and can use this time for other processes instead.

2.3.3 Profiling the Coffee Machine

This experiment was conducted to analyze the parameters of the coffee machine. It is hypothesized that using the coffee product of an order as a basis for billing allows to precisely bill the coffee consumed by that order. This requires the coffee maker to supply the same amount of coffee per product in average. To validate this, a set of measurements is taken while the machine operates to show if the amount of coffee is constant (in a limited range) for each order.

Measuring the machine's use of coffee directly is unfeasible without interfering heavily with the machine. It would require to clean the machine's inside of every residue of coffee before a coffee is brewed. Further, it would require measuring the amount of coffee disposed by the machine in the grounds container as well as measuring all the coffee that is poured into the cup as well as disposed of in the tray. While technically possible, this would skew the amounts compared to the everyday use of the machine (where no extensive cleaning is conducted during operation) and lower the usefulness of the results obtained by this experiment.

Instead, a method to indirectly measure the consumption is devised and applied in this experiment: First, the machine's bean container is emptied and filled with a known amount of beans (100 g). Then, the time the machine's grinding mechanism is active while a coffee is brewed is taken. In the same way, the amount of water is measured by timing the water pump. This is repeated multiple times to account for deviation in timings. Finally, the weight of beans left in the container is measured by cleaning it out again and weighting the removed beans. To accurately weight the beans, a kitchen scale is used and a handheld vacuum clean makes sure that all the beans are removed from the bean container.

Experiment Description

The described method is applied to every product-configuration the machine supports. A user can choose from 5 *tastes* for a coffee, ranging from **Extra Mild** and **Mild** through **Standard** to **Strong** and **Extra Strong**. The taste influences the amount of coffee beans that is ground and brewed for the order. Similarly, the user can choose the total amount of beverage to adapt for differently sized cups. The machine can either brew **One** or **Two** coffees at the same time, and this setting directly influences the amount of water used for each order. Finally, the machine supports **Short** and **Long** coffees - this is a way for the user to adjust the amount of water for 2 different kinds of cups. While the amounts of water can be set separately for **Short** and **Long** coffees, the amount of coffee is still the same for each *taste*.

	Cups	Type	Taste
1	One	Short	Extra Mild
2			Mild
3			Standard
4			Strong
5			Extra Strong
6		Long	Extra Mild
7			Mild
8			Standard
9			Strong
10			Extra Strong
11	Two	Short	Extra Mild
12			Mild
13			Standard
14			Strong
15			Extra Strong
16		Long	Extra Mild
17			Mild
18			Standard
19			Strong
20			Extra Strong

Table 2.4: A combination of all product configurations the machine supports.

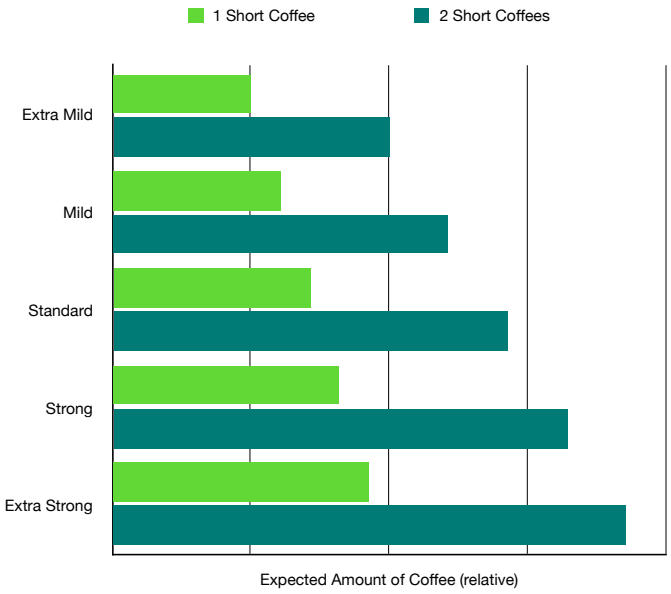


Figure 2.11: The expected amounts of ground coffee in each configuration.

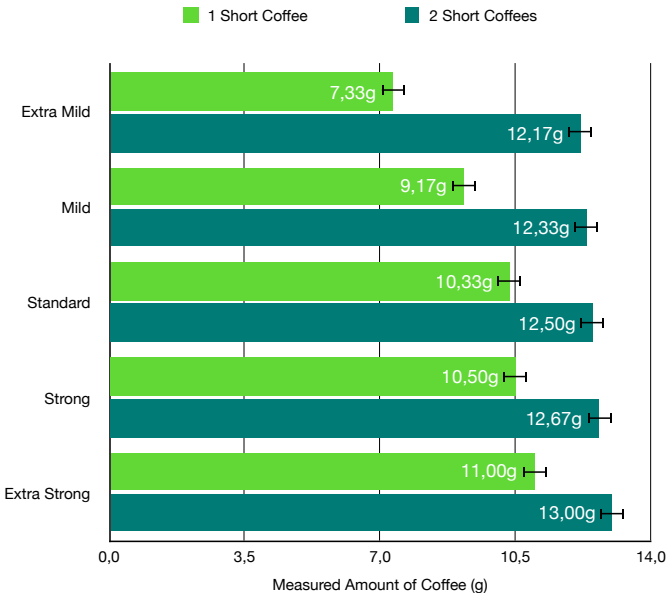


Figure 2.12: Weight of ground beans for each coffee configuration. (Achieved accuracy of measurements about ±1g)

All tastes and settings combined amount in a total of 20 different beverages (Table 2.4).

Expectations

It is expected that the machine delivers different coffees based on the configuration of each order. Figure 2.11 shows the expected relation of taste-setting with the number of cups ordered. This expectation is based on the fact that the manufacturer describes the cups-setting as a way of ordering multiple cups at the same time.

Further, for the same configuration, the amount of coffee delivered is expected to be consistent across every repetition of the order. Therefore, the coffee maker is expected to brew a constant amount of ground beans and thus run the grinder for the same amount of time. Similarly, the amount of water in each coffee and thus the pumping time is presumed to be constant for each configuration.

Experiment Setup

The amount of water is configured for each product configuration according to the user manual [14] using a measurement beaker.

The machine is setup to the following parameter settings:

Coffee Temperature: Hot

Water Hardness: 2

Amount of Water for One Short Coffee: 100ml

Amount of Water for One Long Coffee: 100ml

Amount of Water for Two Short Coffees: 200ml

Amount of Water for Two Long Coffees: 200ml

As a way to reduce the number of tests to conduct by half, the machine was setup such that the **Short** and **Long** coffees behaved exactly the same. This can be achieved by configuring the amounts of water to be the same for **One Short** coffee and **One Long** coffee as well as **Two Short** coffees and **Two Long** coffees.

The DBC is connected to the machine so that the bus can be monitored while the experiment is performed. A PC is connected to the DBC and a serial terminal client is launched on the PC.

For executing the experiment, the DBC's filtering function is leveraged. Instead of relying on manual methods of timing, such as a stop watch, the DBC can be instructed to log a message every time it detects a change in the Power-Board's state. This state includes the current brewing progress in *byte 5*. Accordingly, the following filter is constructed and sent to the DBC: `00000F000000000000`, which will then send a notification with the current state every time any bit in *byte 5* would change. At the same time, the DBC is instructed to log every keypress on the LCD-Board with the following filter: `00000000000000FF00`, so as soon as the number of pressed keys (*bytes 14 and 15*) change in the LCD's packets, another notification is sent by the DBC to the PC. This makes sure that the log consists of both the user input as well as the machine's state changes, allowing for a very precise measurement of the time, the machine takes for processing each state.

A kitchen scale is used to measure the weight of the remaining beans. As the scale is accurate to $\pm 5\,\mathrm{g}$, every measurement will inherit this accuracy. By conducting multiple repetitions of the same configuration and then weighting only the final amount of beans left in the container, this accuracy will be improved to less than $\pm 1\,\mathrm{g}$, as the sum is divided by the number of repetitions.

Execution

To conduct the experiment, $100\,\mathrm{g}$ of coffee beans is weighted and filled into the machine's bean container. The machine's grounds container is emptied and the water tank is filled with water.

A single coffee-configuration (a combination of *Cups* and *Type*, see Table 2.4) is ordered **6** times. Each time, the process of brewing the coffee is awaited and then the next repetition is started.

After the repetitions are finished, the bean container is emptied and the leftover beans are weighted. Another coffee-configuration is chosen and the experiment is restarted until all configurations are run.

After running the experiment, the data is imported into a spreadsheet and adapted to be summarized. As the DBC logs absolute timestamps instead of the length of time the machine stayed in a state (see Listing A.1), this data is converted and relative times are calculated. At the same time, each log entry is associated with the coffee-configuration and the repetition-number. Repetitions are identified in the log by the button presses to select

	Cups	Type	Taste	Average Brewing Time (ms)	Measured Coffee Amount (g)
1			Extra Mild	4333	7,3
2			Mild	5014	9,2
3	One	Short	Standard	5756	10,3
4			Strong	6015	10,5
5			Extra Strong	6453	11,0
6			Extra Mild	6828	12,2
7			Mild	6769	12,3
8	Two	Short	Standard	6933	12,5
9			Strong	7183	12,7
10			Extra Strong	7314	13,0

Table 2.5: Average Brewing Time and Measured Coffee Amount for all tested product configurations

a different configuration.

To calculate the weight of coffee for each configuration, the measured weight of leftover beans is subtracted from the initial amount (100 g) and then divided by the number of runs (6). This weight is also associated with the run and the configuration and then appended to the spreadsheet. Listing A.1 shows the resulting data measured for **One Short Coffee - Extra Strong Taste**.

Results

This section shows and explains the results. During the execution, a large set of data was acquired and then analyzed and summarized for this section. The raw data is available upon request and reflects every measurement made, thus all graphics are based on the raw measurements.

Table 2.5 shows a summary of all data collected. For each tested configuration, the average brewing time (over all repetitions) is shown as well as the measured amount of coffee consumed per repetition. Different orders lead to different amounts of coffees - Figure 2.12 shows in detail the amount per coffee configuration. While the expected distribution of coffee amounts (Figure 2.11) is not reflected in the actual data, the amounts per coffee differ vastly based on the ordered configuration. When ordering **Two Coffees**, the range of different amounts is very small, as indicated by the fact that all values range between 12.1 g and 13.0 g - well within the margin of error of the scale used to weight the beans.

For each repetition, Figure 2.13 shows the time the machine spent grinding coffee (dots). Each line represents the average for corresponding *taste* across all repetitions. The figure shows that in different runs the machine spent a varying amount of time grinding the beans for the same coffee-configuration. The deviation is smaller than the differences in average between each configuration (represented by the lines' distances). Still, this fact implicates that internally the machine does not brew every coffee the same way. Figure 2.19 illustrates this by showing that the same coffee-configuration (color of the dots) is consuming different

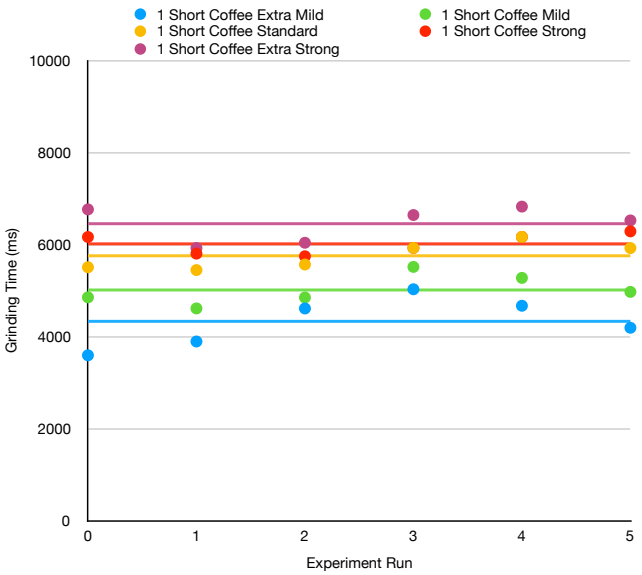


Figure 2.13: Grinding times for each *taste* of *One Short Coffee*

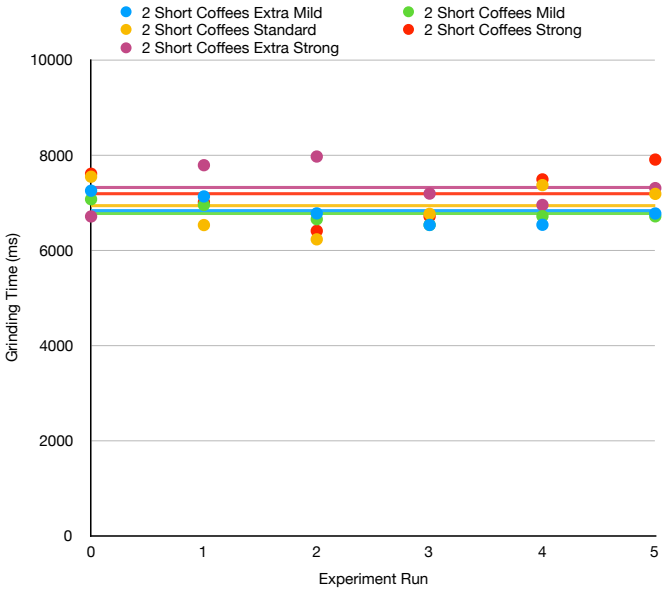


Figure 2.14: Grinding times for each *taste* of *Two Short Coffees*

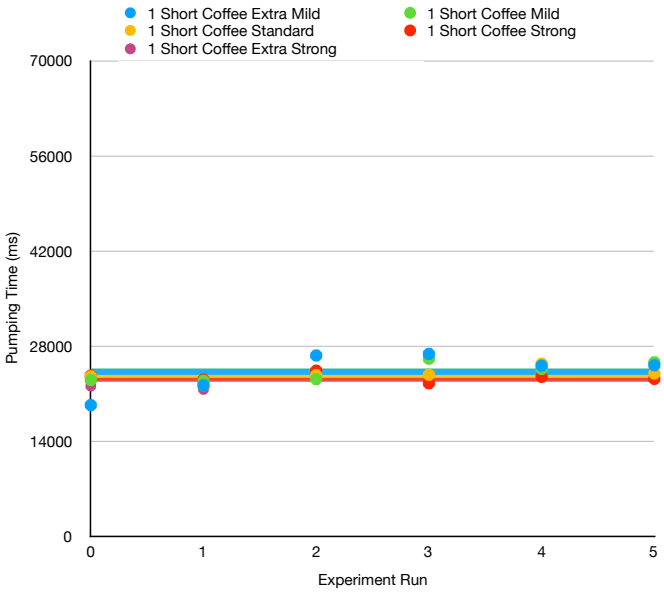


Figure 2.15: Pumping times for each *taste* of *One Short Coffee*

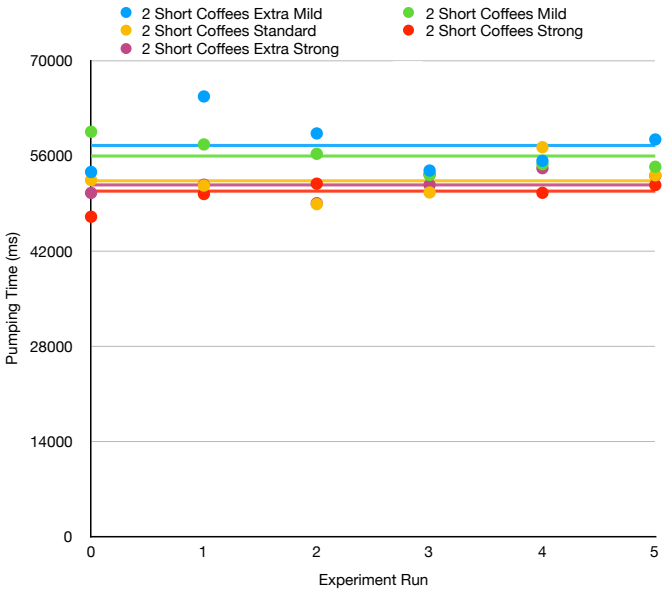


Figure 2.16: Pumping times for each *taste* of *Two Short Coffees*

amounts of coffee and water (x and y-positions respectively).

Measurements conducted to assess the accuracy of time-measurements (Figure 2.17) prove that the timing is very accurate and not responsible for varying grinding times. The length of two constant processes are plotted in relation to each other: the time the machine delays after a program was selected (between 2090 ms and 2105 ms) and the time the machine leaves the ground coffee to brew in the brewing chamber (between 1140 ms and 1280 ms). These times are not affected by the order of the user at all and instead they are constant so the small deviation is caused by the delays between two packets on the SPI-bus and the limited accuracy of the DBC's internal clock.

For the amount of water, measured by the pumping time, it was expected that each cup-setting would be different but measurements would be consistent for different tastes. Figure 2.15 shows that for **One Coffee**, both expectations can be assumed to be true. The machine consistently activates the pump for the same amount of time regardless of the taste.

For **Two Coffees**, the machine evidently behaves differently, as shown in Figure 2.20. Pumping times differ by up to 17.6 s which results in vastly different amounts of water.

Evaluation

The most important question this experiment should answer is whether or not billing based on the product would result in a fair way to bill actual coffee usage. The data suggests that while the machine does not brew the same product the same way every time, it still does so with a large-enough consistency that billing based on the product will outperform billing based on the count of coffees.

As shown by the results, ordering **Two Coffees** at the same time results in a slightly higher coffee consumption compared to **One Coffee** (12.5 g versus 10.3 g for a **Standard** coffee). For the current way of billing, the user would need to pay for two coffees (as indicated by the order), yet consume only about 20% more coffee beans. This proves that the current system is unfair and overemphasizes the number of coffees rather than the amount of beans consumed. The billing system mitigates this problem, as orders can be billed precisely based on the coffee that is ordered instead of the count.

Furthermore, the measured weights of beans and the amount of water for each coffee-configuration will be employed in the billing system to measure the total consumption of the machine. These values can then be shown in the web interface to get an estimate of the number of coffees that can be ordered before refilling the machine becomes necessary. This solves the problem that the machine does not know how much coffee is left before brewing a coffee.

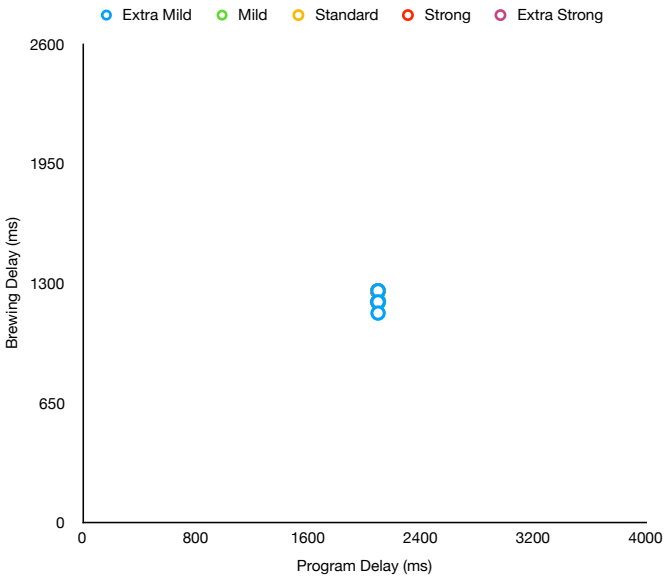


Figure 2.17: Two constant delays plotted in relation to each other, illustrating the accuracy of the measurements taken with the DBC. Figure 2.18 shows an enlarged version of this chart.

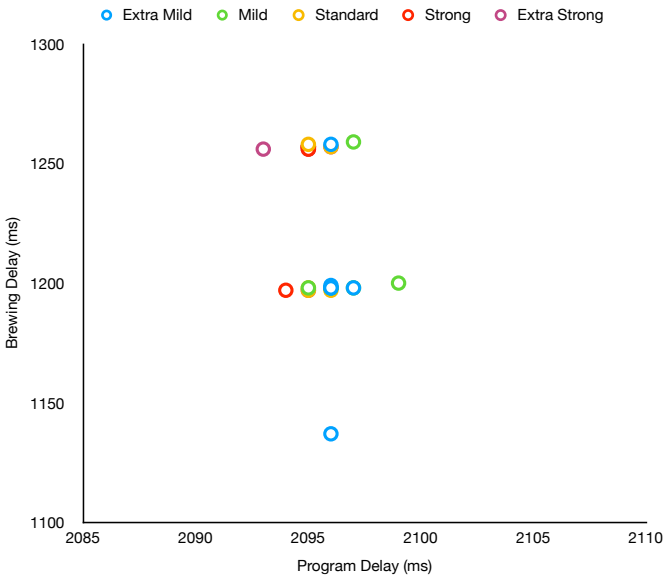


Figure 2.18: Two constant delays plotted in relation to each other and enlarged to allow for differentiating between the configurations and repetitions of the measurements.

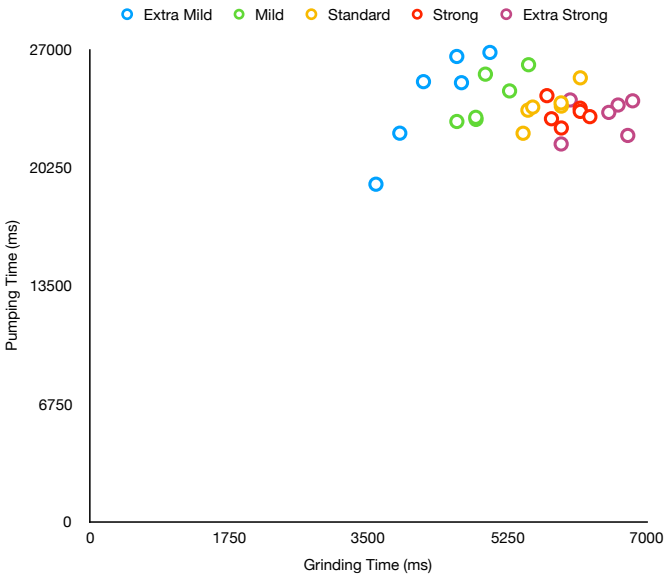


Figure 2.19: Broad distribution of grinding and pumping times for the same product configuration, illustrating the randomness of the coffee maker's grinding timings.

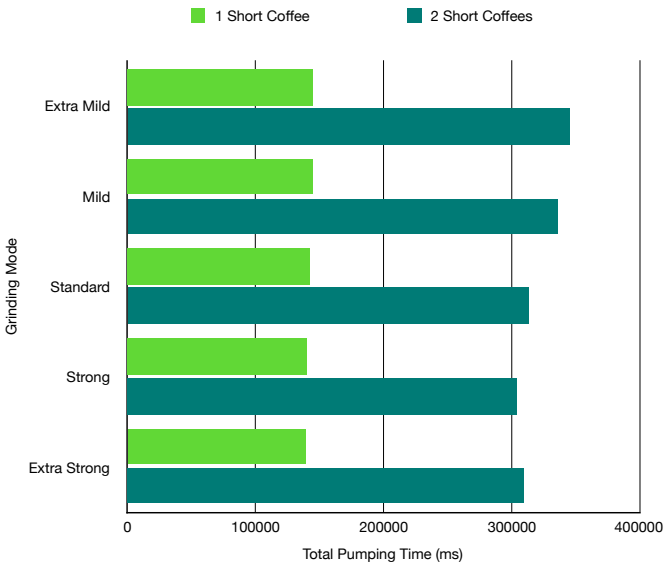


Figure 2.20: Total pumping times for each grinding mode and cup count, showing that the coffee maker's pumping times are almost constant.

CHAPTER 3

Thesis Outcome

This chapter focuses on the Evaluation of the prototype that was designed and implemented. The data collected during experiments is discussed and it is used to verify or falsify hypothesis formulated in the Concept.

Then, a Conclusion is drawn from the collected data and the hypothesis. It is presented how the device can be implemented into the workflow of ComSys and which problems can be solved with it. Also, this section will provide an analysis of the shortcomings of this design and how they may impact the implementation.

This chapter is concluded by describing how the design may be improved or extended in Future Work. Ideas are presented that came up while working on the topic, but were out of scope for this thesis. An effort is made to discuss these ideas and how they may resolve shortcomings inherent in this design.

3.1 Evaluation

This thesis was motivated by the problems associated with keeping track of coffee consumption manually and by the fact that the coffee maker exists as an isolated appliance with no way of external communication. Reverse-engineering the De'Longhi SPI protocol lead to a way into the internal communication of the coffee machine. In doing so, an understanding was acquired about the processes and modules that make up the machine and how they interact with each other. This understanding was then put to use by implementing a new hardware device that interprets the protocol. Finally, a billing system was implemented, using the device as a way to interface with the coffee maker. This system solves the problems outlined in the motivation by providing an easy-to-use web interface and precise billing based upon the actual consumption. The conducted experiments were able to prove all hypotheses to be correct, which stands to show that the understanding developed by this thesis does match the real-world functionality and behaviour the coffee maker provides. The DBC's functional tests show that the device fulfills the requirements as per the goals of this thesis. A device was created that can:

1. **intercept** the machine's internal communication

2. **interact** with the machine in a fashion such that commands can be issued
3. **monitor** the state of the machine and report changes back to the PC

It was proven through experiments that the DBC supports all functions of the coffee maker that are needed for day-to-day operation and to order a coffee. Even more so, it extends upon these functions and allows for previously impossible setups, such as a restricted menu access.

The implementation of the device shows that the protocol was reverse-engineered well enough to both receive and transmit information on the internal bus, including signing the information with a checksum. This proves, that the protocol deducted from observing the communication matches the implementation of the manufacturer. Further experiments prove that timing-requirements for sending and receiving are met under all circumstances which leads to an error-free communication with the device. This means the MitM device's function is transparent for the coffee maker and it is not impacted due to the addition of the device. As a result, any error handling that's implemented in the coffee maker is unaffected and safety routines still work as intended. This makes the coffee maker safe to operate in any environment, which allows its continued use in the ComSys's faculty kitchen.

Using the DBC as a way to interact with the machine, a web interface was created that exposes the functions of the coffee maker and makes them accessible on the local network. Included in this web interface is a way to identify users and associate orders, which in turn allows for reporting on consumption per user. This interface replicates the functions the machine offers at its front panel and adds to that functions that are needed to coordinate multiple users at the same time. Leveraging this interface, users don't need to stand in front of the machine anymore and can order their coffee from the comfort of their desk instead.

Meanwhile, the integration of a billing system based on the actual coffee consumption allows for precise billing and thus a fair distribution of coffee-cost across all users. Instead of having to pay the same amount each time a user orders a coffee, they can now decide which taste they prefer and they are billed for the amount of coffee actually consumed.

Figure 3.1 illustrates how long it would take ordering each of the Short Coffee variants to consume the same amount of coffee beans as ordering One Short Standard Coffee once every (working) day for a whole month. This combination is used as the reference, to show how unfair the current billing setup treats different beverages. Currently, a user that orders a Short Extra Mild Coffee every day would pay the same amount after 20 days as a user that orders Two Short Extra Strong Coffees. In reality, though, their orders would only need to be billed the same after 28 days and 16 days of continuous ordering respectively. With the billing based on weight this thesis proposes and implements, this unfair billing can be corrected so that users enjoying mild coffees will actually save 30% in cost compared to users ordering standard coffees.

Additionally, users are now just ordering their coffee with their own user account and don't need to worry about keeping track of their orders anymore. The outdated paper list hanging above the coffee maker can be replaced by the monthly billing email sent by the billing system.

This increases comfort and acceptance across all relevant user groups: For the end-user, ordering coffee becomes more convenient and they are billed accurately based on consump-

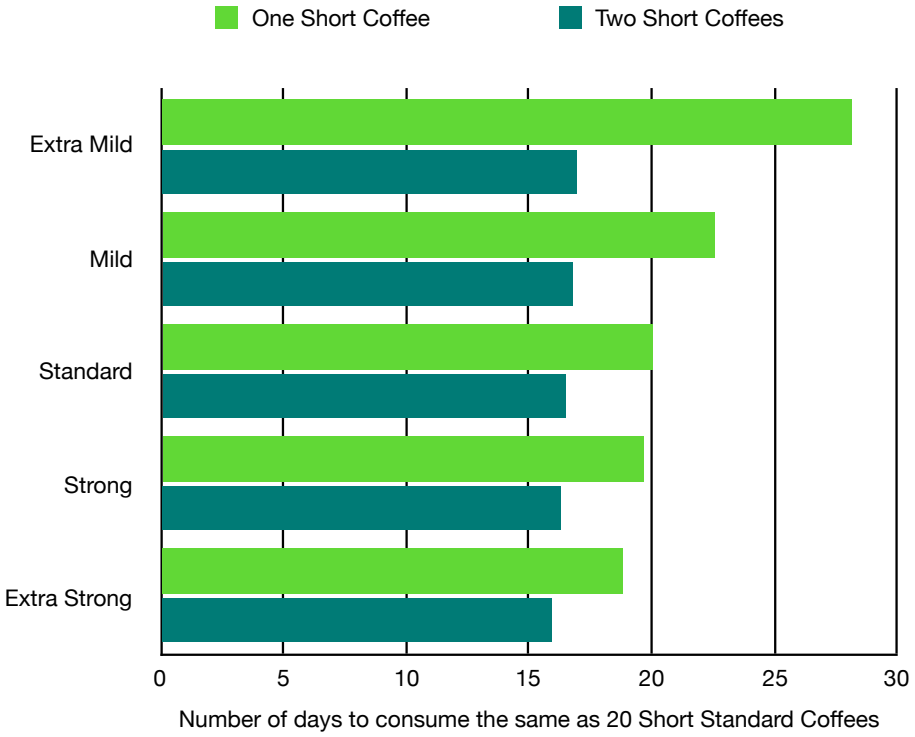


Figure 3.1: Deviation of consumed Coffee beans, when consuming a single coffee each day for one month, using One Short Standard Coffee as the reference.

tion. For the billing-user, the new system allows for an electronic way of collecting the data and sending reports by mail. And for the machine's owner the DBC's filtering function assures that no unauthorized access to the devices' settings menu can happen.

By publishing the results of the reverse-engineering as well as source code and documentation for the DBC (`https://bitbucket.org/0ff/iot-coffee-machine`), this thesis establishes a way to interact with the coffee machine through a well-defined way. This will enable future research where the machine can be integrated and the functions and responsibilities of the DBC improved upon. With the DBC as a gateway, the De'Longhi ESAM 6600 is now connectable to any device that can communicate using a serial port and so it can be integrated into the Internet of Things (IoT). Other devices can leverage the functions provided without requiring a deep understanding of the machine's inner communication and interact with the coffee maker through an easy-to-use API.

Limitations

The current way of interfacing with the coffee maker does come with a set of limitations. While some can be resolved using the billing system, others could not be addressed fully in this thesis.

The biggest limitation is that still manual processes are required, as no automation of the actual brewing and coffee maintenance was part of this thesis. Whenever the coffee maker is used to brew a beverage, it needs to be made sure that a coffee cup is placed under the

outlet of the coffee maker. Otherwise, the fresh coffee would simply be dispensed by the machine, as no sensors are available on the machine to check for the existence of a coffee cup. After the brewing, the ground coffee is dispensed into a waste container. This container fills up over time and again, the user needs to manually interact with the machine in order to make sure that the next order can be processed successfully. While this thesis tries to work around these problems by queueing coffee orders until they can be processed, the limitations of the coffee maker still surface after a short time of use and show the potential for further automation.

The lack of customizable user in- and output at the site of the coffee maker is another limitation that could not be solved in this thesis. This means that currently, the developed solution is dependent fully on the coffee maker's input and output capabilities. While these are designed to aid in the process of ordering a coffee and provide functions to interact with a user for this particular request, they don't fully support the developed extension to the coffee maker's operation. In particular, communication with the user has to happen through the coffee maker's display which only supports a very limited set of about 100 messages. They are all related to different states of the coffee maker such as error messages describing which state caused the error or information about different steps in the process of ordering a coffee. This thesis does require a way to identify users before coffee is brewed and the coffee maker itself is not built for this scenario. While conducting research on how to solve this shortcoming, it was established that identification would need to happen through a new way of inputting data into the billing system. For now, this is limited to the web interface where a user can login and thus authorize with the billing system. Every action executed after login can then be attributed with the user. Still, this requires a change in the behavior of the coffee maker's users: instead of ordering beverages at the machine, it is now mandatory to use the web interface.

Another limitation is the lack of generality achieved through implementing the DBC and the corresponding device client. While it is shown that different machines of the same kind (i.e. the De'Longhi ESAM 6600 and the De'Longhi ESAM 6660) can be connected to the billing system successfully, no further research was conducted on applying the gained information about the De'Longhi protocol to other machines. This is due to the lack of access to related machines and the fact that connecting to the internal SPI bus of the machines requires hardware modifications. While implementing all software components, focus was laid on abstracting all the devices capabilities and it was made sure not to rely on the current devices functions. Instead, well-defined interfaces were introduced that allow to change the actual coffee maker and to broaden the set of machines that can be integrated into the billing system.

3.2 Conclusion

Through the implementation of the device and the billing system, users can now order beverages from the comfort of their desks and will be notified as soon as the beverage is brewed. As the consumption is now tracked based on the beverage's actual coffee content, consuming light coffees in small cups is now considerably cheaper than ordering a large cup of strong coffee. This means that billing is now fair in regards to the actual usage.

As tracking and billing now happens automatically, users cannot forget to make a mark anymore and the current state of usage is always up-to-date. As a result, the tracking list besides the coffee maker was retired and instead the space was used to show monthly statistics of which beverage was preferred and which users ordered the most amounts. The monthly process of analyzing the tracking lists and counting the marks was retired as well. Using the statistics the billing system provides, billing is now as easy as sending a mail to each user that ordered a coffee in the past month and requesting the amount of money calculated precisely by the billing system.

As a result of the thesis, an understanding of the machine's internal communications bus and the protocol used on that bus was developed. This understanding is published in this thesis in the form of a protocol definition as well as a verification algorithm to validate the received data. The published description can be used to implement similar MitM devices as well as in repairing broken coffee makers of a similar kind as the De'Longhi ESAM 6660. Other machines of the same family are using an adapted protocol, because they may have different functions and as such need different bits in the protocol. As part of this thesis, another member of the ESAM 66xx, the De'Longhi ESAM 6600, was analyzed and the protocol for this machine was found to be different from the original machine. While the electrical interface stayed the same, this older machine would not support certain functions and thus the protocol was using shorter messages to transfer the state. Key parts of the protocol (such as sync bytes as well as the checksums) stayed the same, though sped up the process of adapting the protocol a lot. As the De'Longhi ESAM 6600 predates the De'Longhi ESAM 6660, it is assumed that other machines of the same family will be using similar protocols. Publishing the methods of reverse-engineering these protocols, it is trusted that other machines can be enabled to work with the device and the corresponding PC-client, allowing for a much wider audience for the billing system. This would permit different institutions and organizations to monitor their machines and bill their users in a fair and precise manner. Even when no billing is intended as the coffee is offered for free, this thesis enables a way to report on the usage and brings convenience into the process of ordering a coffee.

Finally, the software implementation of both the MitM device and the PC-client exposing the control-API is published under an open-source license. These can be used to replicate the experiments and results of this thesis as well as improve upon the functions these components provide.

Comparing this thesis to other research in the field, a new way of interacting with the coffee machine was found, allowing very easy access to the machine's functions and status information. Other researchers relied on manufacturer-provided interfaces (such as [6]) or implemented complex vision-based systems to observe the machine (such as [4]). Given that both the De'Longhi ESAM 6600 and the De'Longhi ESAM 6660 lack any manufacturer-port entirely, this thesis provides the first easy-to-use and -integrate interface to integrate these machines into a computer controlled environment.

3.3 Future Work

A framework for communicating with the De'Longhi ESAM 6660 has been established and can now be leveraged to extend the functions of the machine. In the future, it is possible to design and implement further methods of requesting coffee and other beverages without the need to reverse-engineer any of the machine's parts again. For instance, as a way to combine a beverage configuration (such as *a strong coffee*) with the user's identity, a device that reads NFC or RFID tags applied to the coffee cup would be feasible. This device would then contact the billing system and submit an order along with an identifying token, such as a unique id per tag, which the billing system would associate with the user. A user would be able to register different tags for different cups and any time a certain cup is registered by the machine, the machine is instructed to brew the coffee exactly the way, the user wishes. Building upon the foundation laid by this thesis, work can be focused on the implementation of the NDC/RFID device and the machine can be controlled via an open API.

Due to the fact that the MitM device supports every function that the machine exposes through its user interface, different programs can now be combined into a single beverage. This would allow, for example, to fill even bigger cups of coffee or even cans to be placed in meetings or gatherings. Now, this would require the user to manually order the same beverage multiple times and wait for completion every time. In the future, a single order could be implemented that automatically brews the same (or even different) beverages until a predefined amount of beverage is reached.

In fact, the whole built-in configuration of which beverage is brewed with what amount of water or coffee might be replaced using the device. The machine's Power-Board supports a mode that allows full control over the sensors, valves and motors. While it was not explored during this thesis, in the future this could enable a custom type of coffee altogether. No restrictions in terms of programs would be placed by the machine and it could be used to brew entirely custom recipes.

The billing system, in its current form, is used only for monitoring coffee consumption and taking orders for beverages. In the future, it could be extended to cover the complete billing process established in the faculty. That is, in addition to the current data, orders of coffee beans could be tracked and billing could be calculated based on actual bean prices rather than fixed prices per beverage. Also, the machine's utility could be monitored in such a way that the billing system would know in advance when to re-stock on coffee beans and other consumables. This would then extend the usability from monitoring past-events to predicting the future usage of the machine and may help in reducing storage space for consumables as well as eliminate out-of-coffee situations.

Similarly, the web interface could be extended by functions such as favorite products and custom products. Each user could then customize the list of products so that products often ordered are listed at the top of the list. The billing system may even sort the products based on the user's order history, employing the data stored about the user to customize the user's experience.

Further analysis of the consumption-data will allow to get an even better understanding of the habits and needs of the coffee maker's users. Improved reporting on the data as well as visualization may provide additional insights into how much coffee users consume in average

and when most coffee is consumed. This would allow for shutting down the machine for most of the day and only powering it up when a time of high usage is reached. While also preserving energy, this will help to extend the life of the machine by reducing stress on the heating-element and consequently reduce the cost of operation for the coffee maker.

Bibliography

[1] Quentin Stafford-Fraser. *The Trojan Room Coffee Pot*. May 1995. URL: http://www.cl.cam.ac.uk/coffee/qsf/coffee.html.

[2] Tiago Franklin R. Lucena et al. "Augmenting Object with IoT to Enhance Elders' Social Life". In: *eHealth 360°: International Summit on eHealth, Budapest, Hungary, June 14-16, 2016, Revised Selected Papers*. Ed. by Kostas Giokas, Laszlo Bokor, and Frank Hopfgartner. Cham: Springer International Publishing, 2017, pp. 36–41. URL: https://doi.org/10.1007/978-3-319-49655-9_6.

[3] *World's first text-enabled espresso machine*. June 2013. URL: https://www.zipwhip.com/blog/zipwhip-text-enabling-of-the-delonghi-magnifica-espresso-machine/.

[4] Jürgen Sieck and Maurus Rohrer. *Bedienung und Verwaltung von Haushaltsgeräten mittels NFC am Beispiel eines Kaffeevollautomaten*. Tech. rep. Hochschule für Technik und Wirtschaft Berlin, Sept. 2012.

[5] MDB Protocol and Multi-Drop Bus. "Internal Communication Protocol". In: *MDB/ICP, National Automatic Merchandising Association (NAMA), Version* 3 ().

[6] Leon Busse. "Implementierung eines Debit-Abrechnungssystems für Jura-Kaffee- vollautomaten". Bachelor's Thesis. Ludwig–Maximilians–Universität München, Nov. 2017.

[7] Leon Busse. *CoffeeSystem - Technische Dokumentation*. Tech. rep. Ludwig–Maximilians–Universität München, Sept. 2017.

[8] Stan Augarten. *The Most Widely Used Computer on a Chip: The TMS 1000. State of the Art: A Photographic History of the Integrated Circuit (New Haven and New York: Ticknor & Fields)*. Tech. rep. 1983.

[9] Jacob Beningo. "HAL Design for SPI". In: *Reusable Firmware Development: A Practical Approach to APIs, HALs and Drivers*. Berkeley, CA: Apress, 2017, pp. 201–217. URL: https://doi.org/10.1007/978-1-4842-3297-2_8.

[10] Manuel Jiménez, Rogelio Palomera, and Isidoro Couvertier. "Embedded Peripherals". In: *Introduction to Embedded Systems: Using Microcontrollers and the MSP430*. New York, NY: Springer New York, 2014, pp. 299–382. URL: https://doi.org/10.1007/978-1-4614-3143-5_7.

[11] Kurt Zeilenga. *Lightweight directory access protocol (ldap): Technical specification road map*. 2006.

[12] Frederic Leens. "An introduction to I2C and SPI protocols". In: *IEEE Instrumentation Measurement Magazine* 12.1 (Feb. 2009), pp. 8–13.

[13] *RM0090: STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM®-based 32-bit MCUs*. 15th ed. July 2017. URL: http://www.st.com/resource/en/datasheet/stm32f407vg.pdf.

[14] *FULLY AUTOMATIC COFFEE CENTER ESAM6600 - IMPORTANT INSTRUCTIONS*. 2006. URL: http://www.delonghi.com/Global/InstructionManuals/GB/GB-5732147800.pdf.

Appendix

A.1 Experiment Results

```
1  [Delonghi] LCD:RX=B040208E10372C81E7 -> PB:TX=B040208E10372C81E7  PB:RX=0
       B0700200701040093 -> LCD:TX=0B0700200701040093 TS=16991
2  [Delonghi] LCD:RX=B000208E10372C80A6 -> PB:TX=B000208E10372C80A6  PB:RX=0
       B0701200701040094 -> LCD:TX=0B0701200701040094 TS=17111
3  [Delonghi] LCD:RX=B000200610372E8020 -> PB:TX=B000200610372E8020  PB:RX=0
       B0702200701040095 -> LCD:TX=0B0702200701040095 TS=19206
4  [Delonghi] LCD:RX=B00020061037318023 -> PB:TX=B00020061037318023  PB:RX=0
       B070320070504009A -> LCD:TX=0B070320070504009A TS=22796
5  [Delonghi] LCD:RX=B000208E10373580AF -> PB:TX=B000208E10373580AF  PB:RX=0
       B0704200701040097 -> LCD:TX=0B0704200701040097 TS=26567
6  [Delonghi] LCD:RX=B000200610373B802D -> PB:TX=B000200610373B802D  PB:RX=0
       B0705200701040098 -> LCD:TX=0B0705200701040098 TS=33330
7  [Delonghi] LCD:RX=B000208E103800807B -> PB:TX=B000208E103800807B  PB:RX=0
       B0706200701040099 -> LCD:TX=0B0706200701040099 TS=34348
8  [Delonghi] LCD:RX=B000200610380480F7 -> PB:TX=B000200610380480F7  PB:RX=0
       B070720070304009C -> LCD:TX=0B070720070304009C TS=38120
9  [Delonghi] LCD:RX=B000200610380680F9 -> PB:TX=B000200610380680F9  PB:RX=0
       B0708A0070304001D -> LCD:TX=0B0708A0070304001D TS=40036
10 [Delonghi] LCD:RX=B000200610380780FA -> PB:TX=B000200610380780FA  PB:RX=0
       B070920070104009C -> LCD:TX=0B070920070104009C TS=41114
11 [Delonghi] LCD:RX=B000200610380A80FD -> PB:TX=B000200610380A80FD  PB:RX=0
       B070A20070104009D -> LCD:TX=0B070A20070104009D TS=44108
12 [Delonghi] LCD:RX=B000208E10380B8086 -> PB:TX=B000208E10380B8086  PB:RX=0
       B070BA00703040020 -> LCD:TX=0B070BA00703040020 TS=45365
13 [Delonghi] LCD:RX=B00020061038218014 -> PB:TX=B00020061038218014  PB:RX=0
       B070D2007030400A2 -> LCD:TX=0B070D2007030400A2 TS=67450
14 [Delonghi] LCD:RX=B00020061038258018 -> PB:TX=B00020061038258018  PB:RX=0
       B070E2007030400A3 -> LCD:TX=0B070E2007030400A3 TS=71399
15 [Delonghi] LCD:RX=B000200610382C801F -> PB:TX=B000200610382C801F  PB:RX=0
       B070F2007050400A6 -> LCD:TX=0B070F2007050400A6 TS=77860
```

The Log, the DBC produces for a single repetition of the experiment for **One Short Coffee - Extra Strong Taste**

```
exp run weight cups taste pb-state timestamp runtime
0 0 36 1-short xstrong 0 16991 120
0 0 36 1-short xstrong 1 17111 2095
0 0 36 1-short xstrong 2 19206 3590
0 0 36 1-short xstrong 3 22796 3771
0 0 36 1-short xstrong 4 26567 6763
0 0 36 1-short xstrong 5 33330 1018
0 0 36 1-short xstrong 6 34348 3772
0 0 36 1-short xstrong 7 38120 1916
0 0 36 1-short xstrong 8 40036 1078
0 0 36 1-short xstrong 9 41114 2994
0 0 36 1-short xstrong A 44108 1257
0 0 36 1-short xstrong B 45365 22085
0 0 36 1-short xstrong D 67450 3949
0 0 36 1-short xstrong E 71399 6461
0 0 36 1-short xstrong F 77860 3770
```

The data extracted from the DBC's Log for **One Short Coffee - Extra Strong Taste**