Karlsruhe Institute of Technology

Institute für Technische Informatik
Chair for Embedded Systems

Design and implementation of an IoT-based control device for an off-the-shelf coffee machine

Bachelorarbeit
von

Karim Ben Ammar

an der Fakultät für Informatik

Betreuer:	Prof. Dr. Jörg Henkel
Betreuende Mitarbeiter :	Dr.-Ing Lars Bauer
Betreuende Mitarbeiter :	M.Sc Farzad Samie
Betreuende Mitarbeiter :	M.Sc Marvin Damschen

Tag der Anmeldung:	31.1.2017
Tag der Abgabe:	30.05.2017

Chair for Embedded Systems CES

Selbstständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig angefertigt und mich keiner fremden Hilfe bedient habe. Alle Stellen, die wörtlich oder sinngemäß veröffentlichtem oder unveröffentlichtem Schrifttum entnommen sind, habe ich als solche kenntlich gemacht.

Contents

List of Figures

Acronyms

RFID Radio-Frequency Identification

NFC Near Field Communication

UID Unique Identifier

SPI Serial Periphal Interface

FSM Finite-state machine

GPIO General-purpose input/output

1 Introduction

We live in a world evolving at an incredible pace. New technologies to facilitate our daily life are being developed : Smart homes where a person can interact with his house, fridge lights. . .
Having an interactive environment around us is not a luxes anymore, but a necessity to improve our daily tasks. The main idea behind this project is to give a second life to our not so "interactive" instruments. We want to control them and monitor them in a modern way.

This work is an enhancement of an older project, which upgraded an automatic coffee machine. The main idea was to add an interactive user control, and facilitate the payment method. Our goal is to clarify some dark corners left by the previous work which was too complex. We want to know our machine better, how it works and if it has any unusual behavior, all through a user friendly interface.

1.1 Requirements

The final product should present the following features :

- Only authenticated users should be allowed to manipulate the coffee machine (Order coffee, maintenance,~~. . .~~)
- Our authentication system is based on RFID tokens. Every card has a unique identifier (UID), making the user management task easier. Besides, every KIT employee/students owns such card which prevents additional charges. Finally, every user can have multiple RFID tokens.
- For this work, we will use a preexisting accounting system. The user should be able to order his drink at any time For both cases where an Internet connection is provided or not)
- The user should be able to order coffee with the main application (raspberry pi) or his android Smartphone, using a user friendly and intuitive interface
- Our project should be portable : The user can plug it to the sensors of any coffee machine without any major modification to the apparatus.
- The system should be able to monitor the coffee machine and log its behavior ~~for a given period of time at a given interval~~. This data will be used to define the different states of our machine and detect any unusual behavior.
- The system should be easy to use and upgrade

1.2 Structure of this work

In chapter 2, the preexisting ~~H~~ardware and software used in this project are discussed. Chapter 3 presents the main features of this work, the upgraded hardware, the new software, how they were designed and implemented. Chapter 4 summarizes this work, and gives an outlook.

2 Foundations

In this chapter we will introduce the foundations of the project.

2.1 Coffee machine

2.1.1 Overview

For this work, we will ~~user~~ a Saeco Royal Professional (SUP016R) coffee machine (see figure 1) as our main device. This is a “top-quality fully automatic espresso machine coffee which allows the brewing of either espresso or normal coffee, using both bean and pre-ground coffee”. This machine also offers hot water and steam if needed.

Figure 1: Saeco Royal Professional[1]

In order to fulfill the requirements described above, the coffee machine needs to be locked until an authorized user presents logs in. A better understanding of the machine (grinding, brewing cycles . . .) is required if we want to create a good monitoring system. The different sensors are located on the right side of the coffee machine. They can be easily accessed after removing the case. A better upgrade would be to control the coffee machine directly, unfortunately the button board is secured on the inaccesible side of the board, and therefore we used the senors as an ordering mechanism in this work.

[1]https://www.segafredo.com.au/product/saeco-royal-cappuccino/

2.1.2 Workflow

In order to provide an optimal user experience, we need to have a better understanding of the coffee machine. This can be devided in two main processes : Preparation phase and the order phase. This describes the coffee ordering mechanism.

Preparation phase :

First the gear motor is initialised and moves to home position followed by the activation of the water heater for one and a half minute. This would heat the water to the desired temperature coninuously for 60sec and then is alternated for the remaining period fo time.

Ordering phase :

After pressing the start button, the grinder is triggered for 5.5 sec, followed by two consecutive activations of the doser. The gears change to brewing position, and the pre-brewing begins followed by the brewing process. The duration depends on the coffee selected. Finally, the gears move back to home position. A summary of the whole process can be viewed in figure 2

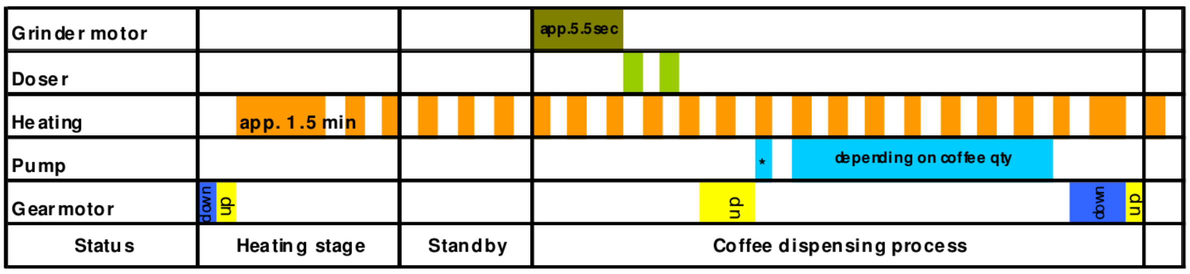


Figure 2: Coffee ordering process from

2.1.3 Electrical connections

As depicted in figure 3, the coffee machine has multiple sensors. For this project, we will use the input of last four sensors :

- Flow meter system 1 : For water usage detection
- Flow meter system 2 : For water usage detection
- Reed switch water level : For low water level detection
- Doser switch : For coffee detection

Flow meter sensors :

The flow sensors have each three pins, used as follows :

Wire	Type
Green wire	5V
Brown wire	Ground
Yellow wire	Output

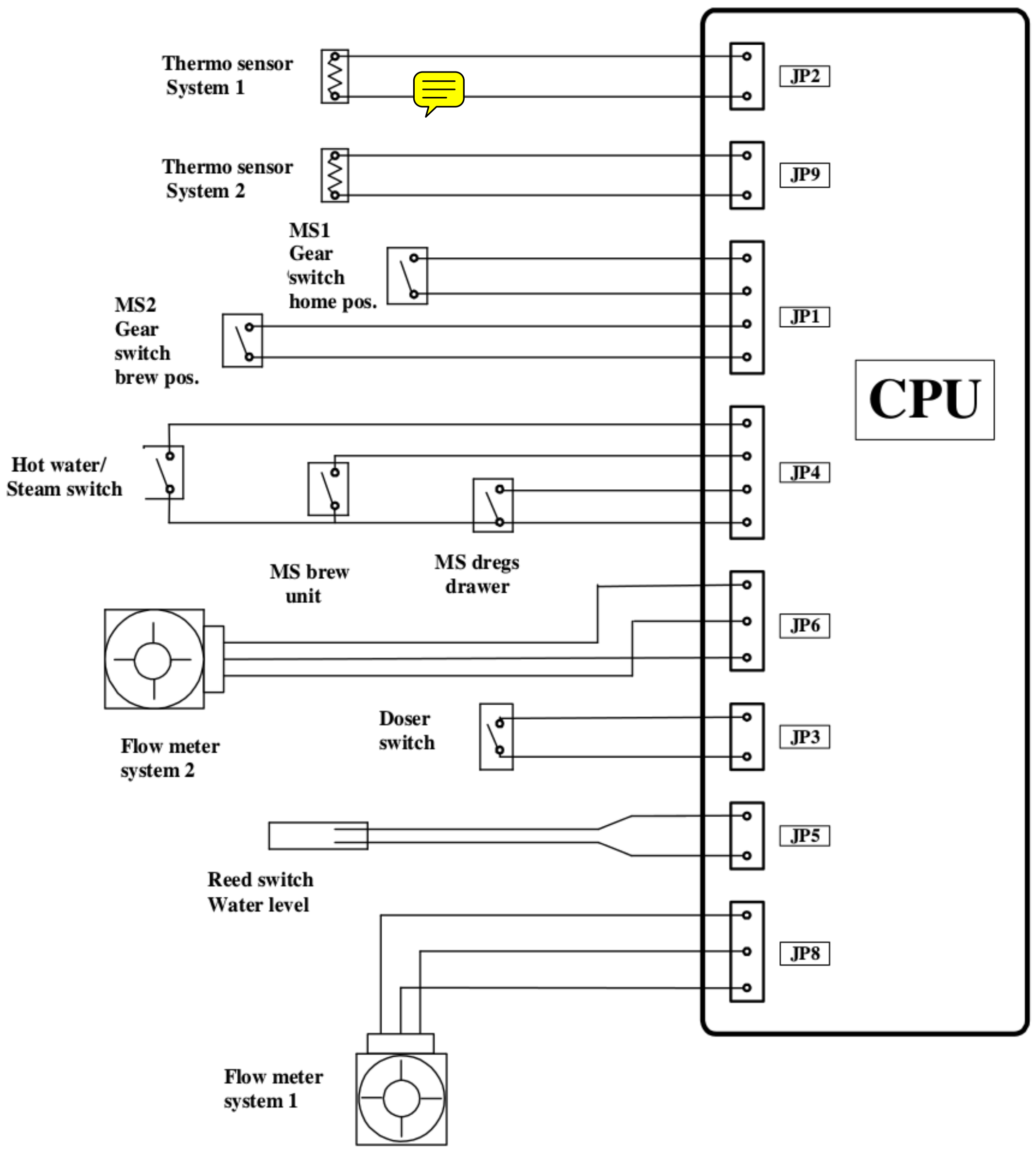


Figure 3: Coffee machine electrical board from

These sensors monitor the flow rate: They control the water quantity delivered for the different beverages. When a transaction is detected, the system will inspect the flow meter if it is turning. According to , if no signal is detected within 10 seconds, the cycle is stopped, and a message is displayed asking the user to de-aerate.

Water level/Doser switches :

As pictured in figure 4, both switches will pull the pin to ground when activated. For water level detection, a reed switch is used. The switch's mechanism is made from two seperate components. The first one is the magnet. It is floating inside a small chamber in the water tank. This provides him the necessary freedom to follow the water level. The second part is the actual switch placed at the same level with the magnet. Whenever the water drops under a certain threshold, the switch would be no longer controlled by the magnet, and trigger the low water level signal. This will lock the coffee machine temporarly, until the water tank is refilled.

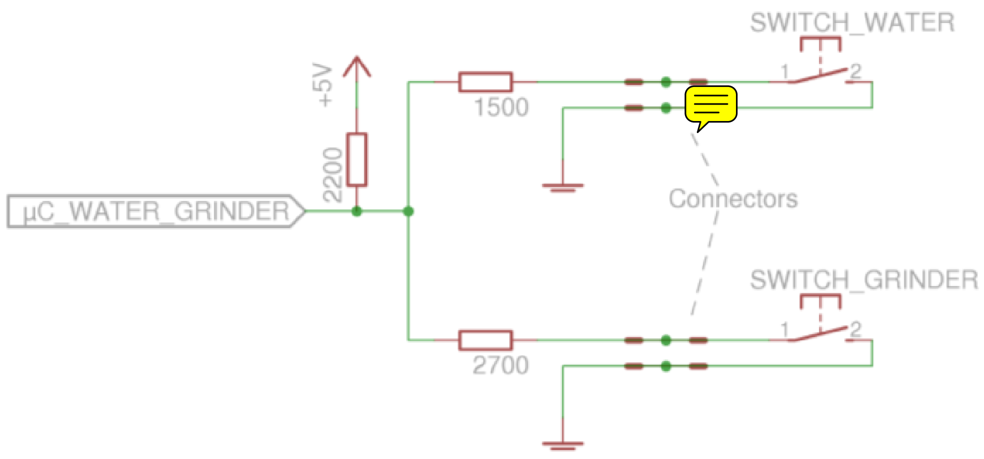


Figure 4: Electrical representation of the water and doser switches from

2.2 Accounting system

The order transactions are managed by a preexisting accounting system. The user has to register and add money to his account in order to use the provided system. After ordering his beverage, the user has to inform the system about his transaction. Different methods were implemented for this purpose. The currently used procedure consists of printed tally sheets. The user has to write down his transaction by adding a mark in front of his name. The sheets are checked periodically, and the corresponding accounts are updated accordingly. The second method consists of a web interface. In this project, we will use this approach. Every user is identified by a unique ID, which will be used in every transaction. This method gives the administrator more control over the transactions (Less counting mistakes, and set a limit for negative balances). Plus, the user has the possibility to consult his consumption statistics and history after each order.

3 Design/Implementation

3.1 User Interaction

The developed system has to ensure the workflow pictured in figure 5. Any user has the possibility to consult the entertainment content without the need to identify himself. The coffee machine has to remain locked, preventing any order, until the user presents his RFID token.
If the user is registered, and his balance is above a certain threshold, the coffee machine will be unlocked, otherwise the user will be asked to register his RFID token before retrying. This can be done by the coffee administrator. When an invalid token is detected, a custom window with the user's UID should popup. This will facilitate the registration task for the admin.
Once the coffee machine is unlocked, the user can either press the coffee button or the water button. A distinct button has to be provided to the user regarding his milk preferences. The system should automatically detect the chosen beverage and charge the user accordingly. The accounting page should be visible only to valid users. The machine has to return to the locked state as soon as the user received his order.

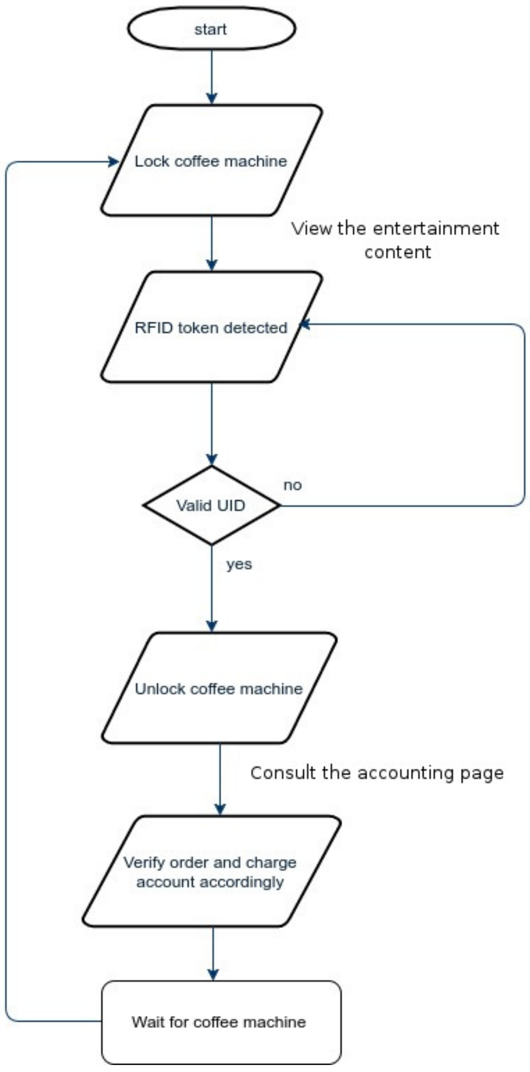


Figure 5: User interaction workflow

3.2 Hardware

In order to provide the best user experience with this project, we have to take into consideration some criteria when choosing the hardware. Over the past years, a new trend of mini-computers appeared : NanoPi 2 Fire, Raspberry pi, NanoPC-T3 . . .
For our system, we will use a raspberry pi, the official pi 7” touchcscreen (see figure 6) and the Joy-IT RFID Modul.

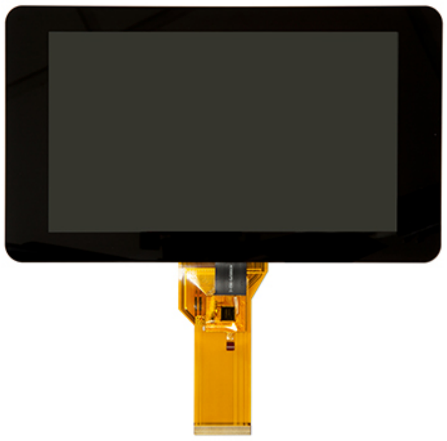


Figure 6: Official raspberry pi 7~~”~~ touchscreen

3.2.1 Raspberry pi

In this project, we will use a raspberry pi 3 model B (see figure 7) as our main controller for several reasons :

- Affordable
- Provides a GPIO interface
- Characteristics : Despite ~~it's~~ size, the raspberry pi is considered as a mini-computer
- Linux based operating system (RASPBIAN)
- No need for extra hardware for internet connectivity

The currently used raspberry pi features a 1.2GHz 64-bit quad-core ARMv8 CPU, 1GB RAM, 4 USB ports, and 40 GPIO pins. It also provides a Bluetooth 4.1 interface that can be used for future upgrades to the system. For a better user experience and due to the lack of display, we attached a 7” LCD touchscreen to the raspberry pie.
A custom PCB was made to permit the communication between the raspberry pi and the coffee machine. This will prevent eventual damages due to the difference between voltages.
With this solution, we should be able to receive/send data from/to the coffee machine.

[2]Source : https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

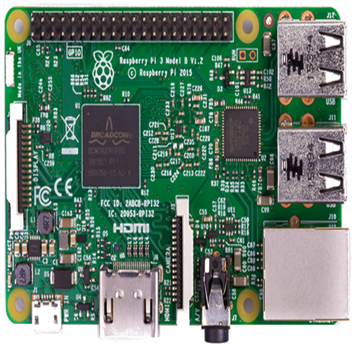


Figure 7: Raspberry pi 3 model B[2]

3.2.2 RFID Reader

For authentication purpose, we will use RFID tokens. They can be extracted from any RFID emitter ex. KIT employee/student card. In order to read the token's UID, we will use a Joy-IT RFID Modul (see figure 8). This works perfectly with the raspberry pi. The RFID reader is directy connected to the GPIO Board on the raspberry pi using the SPI as a communication interface.

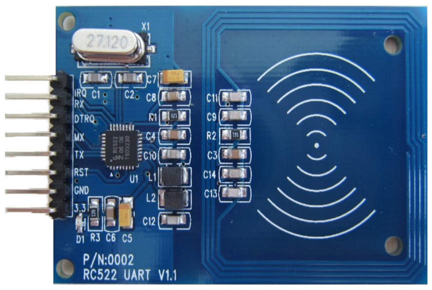


Figure 8: Joy-IT RFID-RC522 Modul[3]

3.2.3 Adapter Coffee Machine/Raspberry pi

Due to the difference in voltage between the raspberry pi and the coffee machine, a custom PCB has been created. It uses optocouplers to transfer the coffee machine's output to the raspberry pi and the locking signal in the other direction. This isolates the two systems preventing any accidental damage to the components. The adapter is made from two seperate parts connected by 11 pins cable. The board on the coffee machine's side uses two four-channel Avago ACPL-247 optocouplers and one single-channel Avago ACPL-217 for the nine different ouputs from the sensors. A second single channel optocoupler is used to unlok the coffee machine. As shown in figure the raspberry-facing part will be directy plugged on top of the raspberry pi. This prevents unexpected connection problems.

[3]Source: http://4.bp.blogspot.com/–ABDuudJrms/VmCqSnUZtdI/AAAAAAAAJfs/Jcp702CzI80/s1600/RC522.jpg

3.3 Software

In order to provide a flawless user experience, we used state of the art technologies. Our system is divided in two main sub systems : Back-end and Front-end.

The Back-end manages the communication with the server, the off-line transactions, the monitoring system and the entertainment material.

The Front-end is what the user interacts with. Our main goal was to provide an interactive, intuitive and beautiful interface. This part is implemented for the raspberry pi and Android Smartphones.

The main problem with the previous project was it's complexity, and lack of portability. To tackle these problems and create an adaptive and extensible software, we divided our Back-end in two main parts :

- User interaction manager
- Monitoring and debugging system

3.3.1 User interaction manager

The raspberry pi is the core of our system :

- It should be able to lock/unlock the coffee machine
- Manage the RFID input
- Communicate with the accounting system
- Provide entertainment content
- Intercept transactions from the android application
- Monitor the behavior of the coffee machine

This part should be implemented in python. Due to thread scheduling problems in the raspberry pi, our system was divided in multiple subprocesses :

- Tornado server
- lock/unlock the coffee machine
- Intercepting RFID input

UID detection :

The communication with the RFID-RC522 over SPI, is implemented using the python library MFRC522. The method read() in figure 9 verifies every 0.01 second the reader. If a valid RFID emitter is detected and no user is already logged in, a custom QR code image containing the read UID is generated and the identifier is sent to the view class. If an android transaction was detected before the user put his card. The read UID is ignored and a message is displayed informing the user that the machine is already in use.

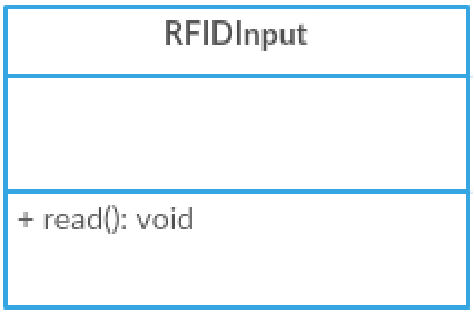


Figure 9: RFID class diagram

Communication with the server :

In this project, we will use a preexisting accounting system to manage the transactions. After detecting the user's order, an HTTP request is sent to `http://i80misc01.itec.kit.edu/coffee/buy.php?rfid=(1)&black=(2)&water=(3)`
where (1) is the user's UID, (2) and (3) represent respectively the coffee and water choice. If (2) is set to "true", the user doesn't want milk with his coffee. The prices are defined by the server.

One other service provided by the server is getting user information. After launching a request to `http://i80misc01.itec.kit.edu/coffee/getuser.php?rfid=(1)` with (1) corresponding the the user's personal UID, a response with the user's name, token, and balance. The name and balance are displayed in the order page. The token is used to display the user's personal transactions history in the accounting section. The different views will be analyzed in the User interface section. If the internet connection is missing, the system will use the local database for the transactions as explained in the Offline Handler section

Offline Handler:

The standard behaviour of the system is to communicate with the server for each user's session. Unfortunately, this may not be always the case. The system is using a Mysql database as backup for these scenarios. The coffee_time database has two different tables as shown in figure 10

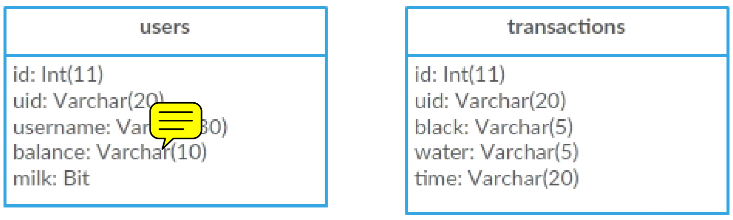


Figure 10: Tables used for the backup database

The users table contains the different required information about each user : UID, Name, and Balance. On the other hand, the transactions table saves all offline transactions. For each order, the user's UID, the milk choice, water choice, and a time stamp are recorded in the database. Both tables are periodically updated by the system.

Tornado Server:

One of the new implemented features in this system, is the possibility to order coffee from a smartphone. For this purpose, the raspberry pi should be constantly listening for incoming HTTP requests using the tornado library. As depicted in figure 11, the tornado class has two main methods: get() and post(). If a request is detected, the script verifies that no user is already connected. If a user is already logged in, a warning message is sent via the post() method to the android application asking the user to retry after few minutes. If the machine is free, the uid is sent to the RFID reading script where it will be processed as explained in the UID detection section.

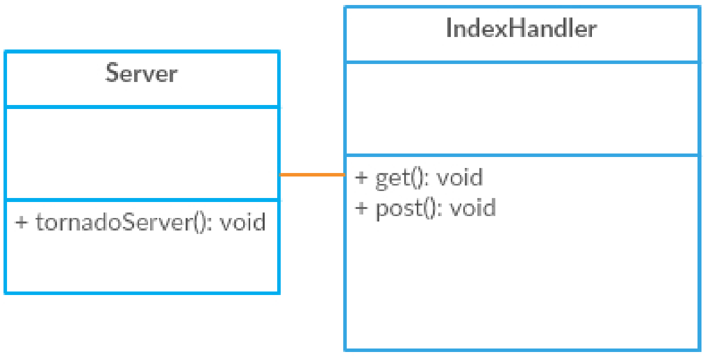


Figure 11: The tornado server class diagram

Locking mechanism:

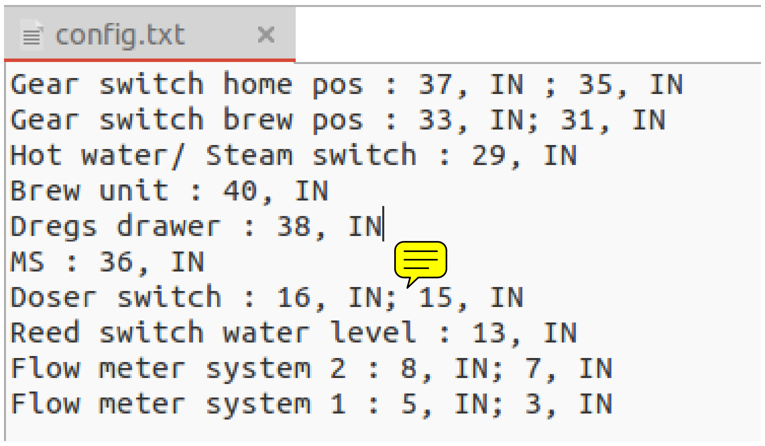
In order to prevent unauthorized users from using the coffee machine, a locking mechanism was developed. The coffee machine provides serveral locking singals : Preventing the motor from going to brew position, stopping the grinder..., but all these signals prevent only the coffee process. The best way to also prevent water usage, is to simulate the water level sensor. A low water will prevent any order from the coffee machine. The only disadvantage of this mechanism, is the "Fill Water Tank" message displayed on the coffee machine, which can be misleading for some users. If the sensor is pulled down during a transaction, the coffee machine would think that there is enough water for another order, and stays unlocked. In order to prevent such unwanted behaviour, the coffee machine is locked at the end of the transaction regardless of the view locking.

3.3.2 Monitoring system

In this section, we will introduce a new feature implemented in this project : The Monitoring system. The main role of this process is to manage the states of the whole program while logging all inputs from the coffee machine.

Logging System:

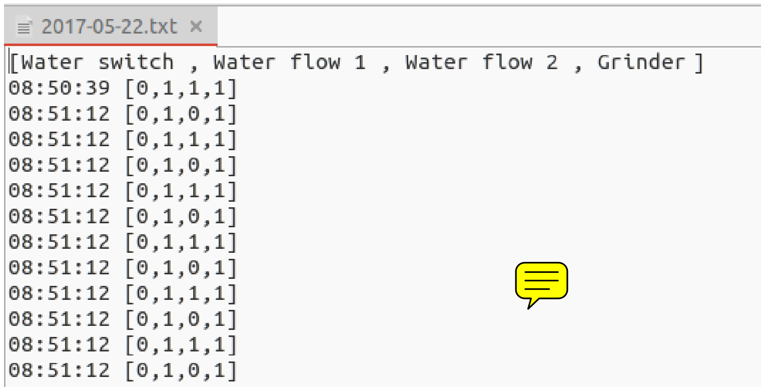
The logging system is in charge of parsing the coffee machine's input and delivering it to the state machine. The logging system is tuned by a config file. Figure 12 is an example of the current setup.



```
Gear switch home pos : 37, IN ; 35, IN
Gear switch brew pos : 33, IN; 31, IN
Hot water/ Steam switch : 29, IN
Brew unit : 40, IN
Dregs drawer : 38, IN
MS : 36, IN
Doser switch : 16, IN; 15, IN
Reed switch water level : 13, IN
Flow meter system 2 : 8, IN; 7, IN
Flow meter system 1 : 5, IN; 3, IN
```

Figure 12: Example of config file for the logging system

The general format of a config line would be as follows :
(Sensor's name) : (Pin 1 number), (Type : In/Out) ; (Pin 2 number), (Type : In/Out)
When the logging system starts, the config file is read and parsed. The sensor names are stored in a List and printed at the beginning of the log file. The pins are stored next to each other starting from the first sensor's first pin. This list will be used setup the GPIO pins according to their type (IN/OUT), and will be used in every iteration to read the inputs of every sensor, which will be stored in the log file with a timestamp as pictured in 13. With the oscilloscope, we could determine the frequency of the water flow sensors (100 Hz). This helped to determine the suitable reading frequency (1KHz).



```
[Water switch , Water flow 1 , Water flow 2 , Grinder ]
08:50:39 [0,1,1,1]
08:51:12 [0,1,0,1]
08:51:12 [0,1,1,1]
08:51:12 [0,1,0,1]
08:51:12 [0,1,1,1]
08:51:12 [0,1,0,1]
08:51:12 [0,1,1,1]
08:51:12 [0,1,0,1]
08:51:12 [0,1,1,1]
08:51:12 [0,1,0,1]
08:51:12 [0,1,1,1]
08:51:12 [0,1,0,1]
```

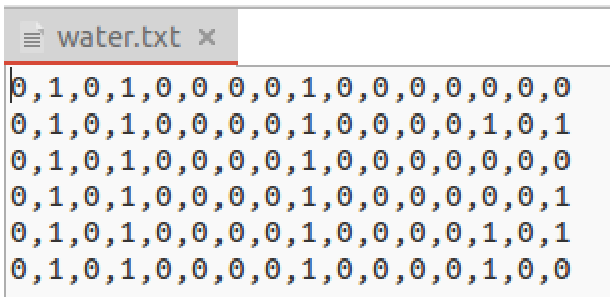
Figure 13: Example of a log file

State Machine:

In order to provide a portable system, our FSM is dynamically generated. Therefore, the user will have to proceed as follows :

- Step 1 : The user creates a file for each state (water, coffee, . . .). Each file's name corresponds to the state's name.
- Step 2 : The user copies then the logs corresponding to each action (Order coffee, order water, . . .) from the log file to each file created in the previous step

Figure 14 shows such files. Each line corresponds to an input from the coffee machine, where every number is the state of the different sensors. When stating our application, the system will read the different states and generate the corresponding Finite State Machine. If a new input is detected, the system will move to the unkown state. All inputs in this state will be written to a new file. As soon as the idle input is detected, the system will change to the idle state. The user can then lable the new generated file with the corresponding state, which will be added the the FMS after restarting the application.



```
0,1,0,1,0,0,0,0,1,0,0,0,0,0,0,0
0,1,0,1,0,0,0,0,1,0,0,0,0,1,0,1
0,1,0,1,0,0,0,0,1,0,0,0,0,0,0,0
0,1,0,1,0,0,0,0,1,0,0,0,0,0,0,1
0,1,0,1,0,0,0,0,1,0,0,0,0,1,0,1
0,1,0,1,0,0,0,0,1,0,0,0,0,1,0,0
```

Figure 14: Sample of the water state log file

As shown in figure 15, our State Machine can bedivided in 5 different states : Idle state, coffee state, water state, no water state, unkown state. The current state is defined by the coffee machine.

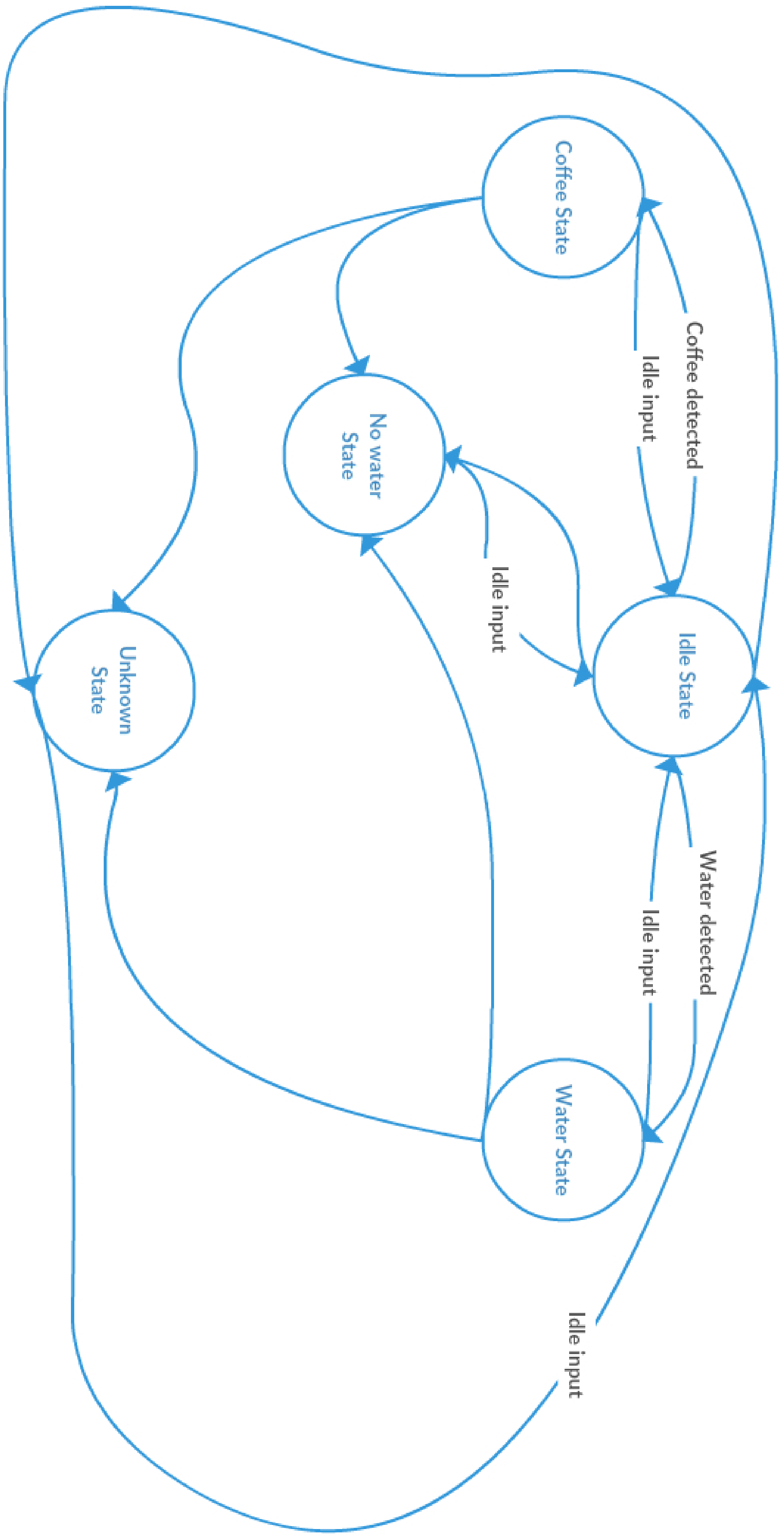


Figure 15: The State Machine graph

After the FMS is generated, every output from the coffee machine is converted to a function's name using the convertToFunc method. The next step is to call the method in the state machine's instance. Depending on the input, the FMS will move to the corresponding state. If an uncommon input is detected, the FMS moves to the unkown state.

```
try:
    moveToState = getattr(coffeeMachine,
                          coffeeMachine.convertToFunc(s))
    moveToState()
except AttributeError as e:
    print(str(e))
except MachineError :
    coffeeMachine.toUnkown()
```

Before or after state transition, a custom method is called as pictured in 16. Depending on the current state, the FMS has to ensure the corresponding behaviour. In the idle state, the coffee machine has to remain locked, and the RFID reading script is continuously called. When a valid card is detected, the coffee machine is unlocked, and the RFID reading script is paused. If a coffee/water input is detected, the FMS moves to the corresponding state, saving the order in a list. When the coffee machine enters the idle state for longer than 3 seconds, the system set the ¨done ¨flag to True and the corresponding transaction is transmitted to the server.

The unkown state or the noWater state can be accessed from any other state. In order to leave the unkown state, the coffee machine has to send the corresponding idle output. When the water tank is refilled, the FMS will be back to the idle state.

During the different states, the logging system will write the input with a timestamp to a file if the new value differs from the previous one. This ensures a compressed log of the daily behavior. Everyday at midnight, a new file is created with the corresponding date to contain the daily logs, and a cleaning script is started. The main role of this script is to ensure that a log file is not saved for longer than a week.

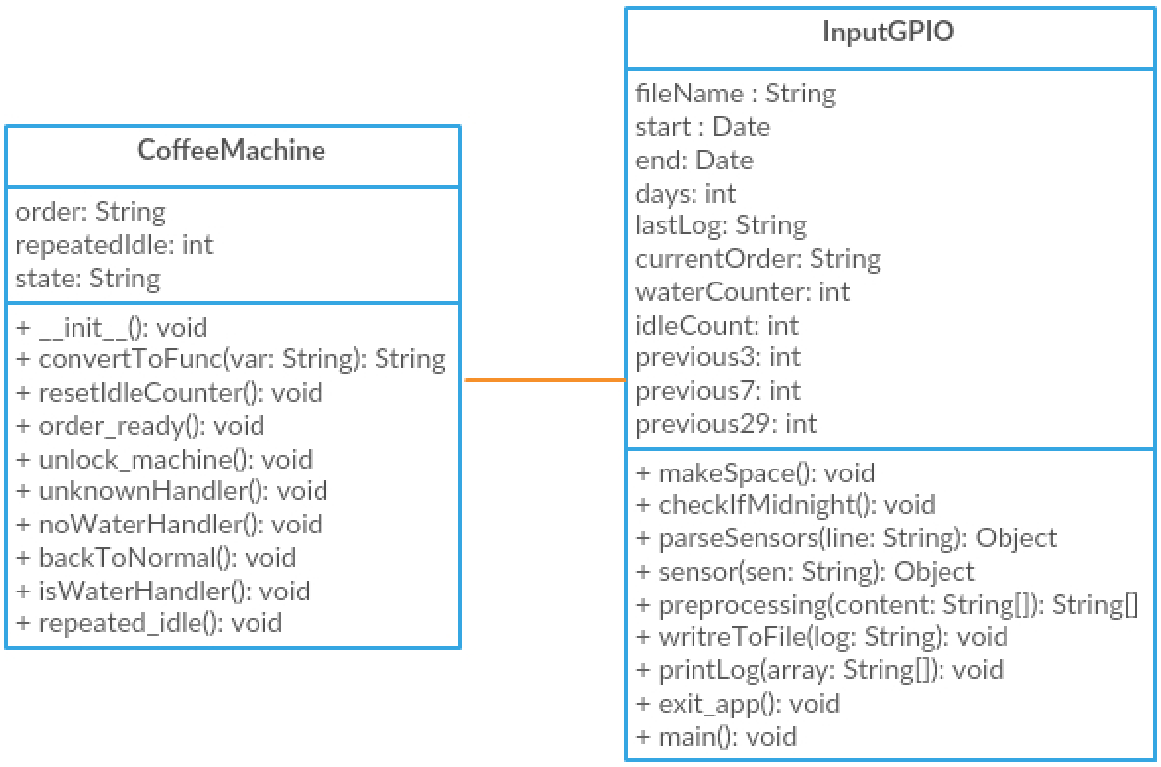


Figure 16: The State Machine class

3.3.3 Raspberry pi Interface

Our Front-end is divided in two main interfaces. The raspberry pi's interface is the main view used for interaction with the coffee machine. It should provide a responsive user interface and manage the communications between the different subprocesses. This is the core of our system, and therefore it should be carefully architected. In order to provide a robust system, we will use the python library PyQT for the different views and threads management. As depicted in figure 17 the architetural pattern used is the model-view-controller pattern

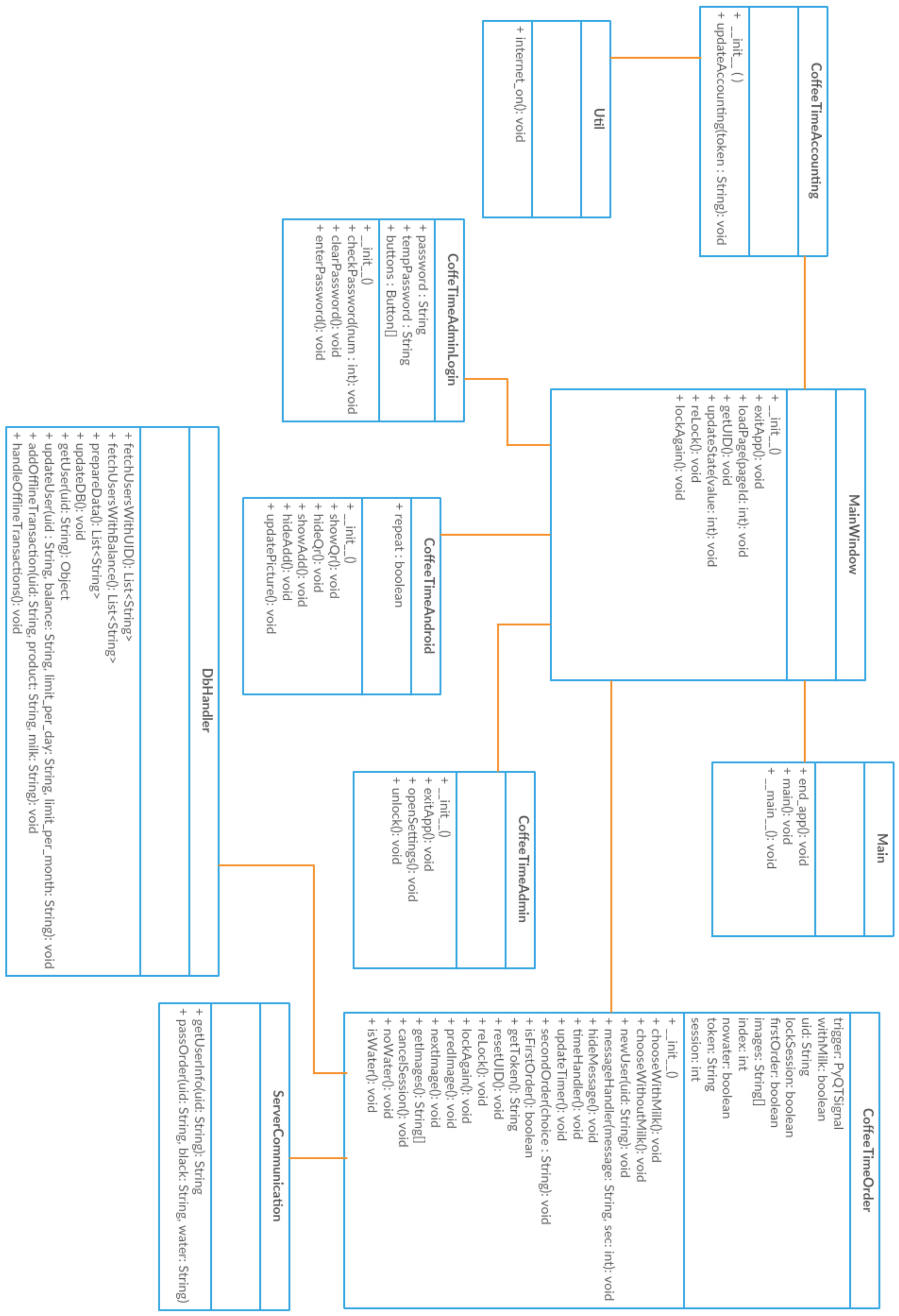
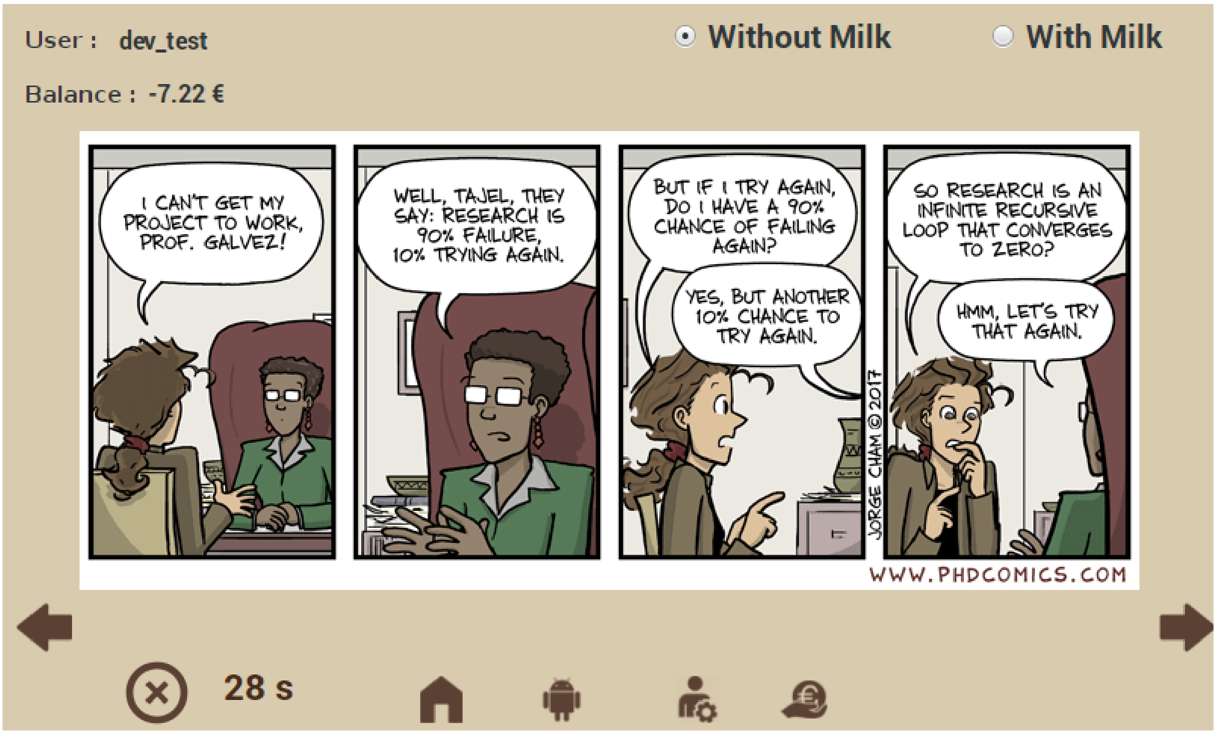


Figure 17: Class diagram of the user interface

Our interface is divided in 4 different pages. In the main page (see figure 18), the user has the possibility to view his balance, and choose if he wants milk with his coffee.


User : dev_test

Balance : -7.22 €

[✓ Without Milk] With Milk

28 s

I CAN'T GET MY PROJECT TO WORK, PROF. GALVEZ!
WELL, TAJEL, THEY SAY: RESEARCH IS 90% FAILURE, 10% TRYING AGAIN.
BUT IF I TRY AGAIN, DO I HAVE A 90% CHANCE OF FAILING AGAIN?
YES, BUT ANOTHER 10% CHANCE TO TRY AGAIN.
SO RESEARCH IS AN INFINITE RECURSIVE LOOP THAT CONVERGES TO ZERO?
HMM, LET'S TRY THAT AGAIN.
JORGE CHAM © 2017
WWW.PHDCOMICS.COM

Figure 18: The ordering page

The server offers daily new entertainment content which can be accessed under `http://i80misc01.itec.kit.edu/coffee_entertain/linklist`.
A python script is triggered everyday at 7 A.M to access the link above, parse it's content, get the respective links for the different images and download them. The downloaded pictures can be viewed in the main page, or in the saving screen. Images are saved up to 2 weeks then deleted. Due to the pictures's high resolution, every image is scaled to the screen's hight or width depending on its dimensions. The class reponsible for the entertainment content is the downloadPictures as shown in figure 19. The method getLinks returns a list with the different links to each image as explained previously. The method makeSpace, is triggered when the script is started. It will look for files older than two weeks and delete them. Due to the lack of file extension in the links, the library BeautifulSoup is used to find the corresponding type of each image. All the downloaded pictures are stored under the Pictures folder, where they can be accessed by the view class or the screen saver.

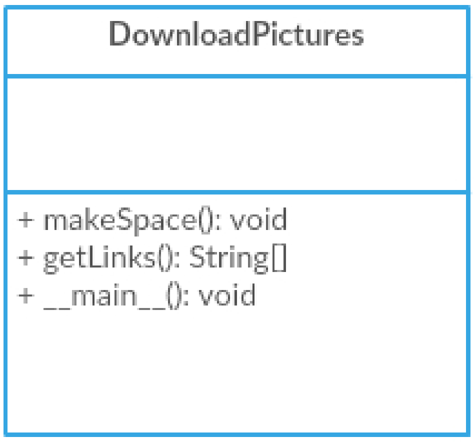


Figure 19: The entertainment manager class

The second window is the app center. As pictured in figure 20 in this page, the user can either download the app by pressing the download button and scanning the given QR code, or add his UID to the android app. This can be done by placing the RFID identifier and scanning the generated QR code after pressing the "Add User" button.

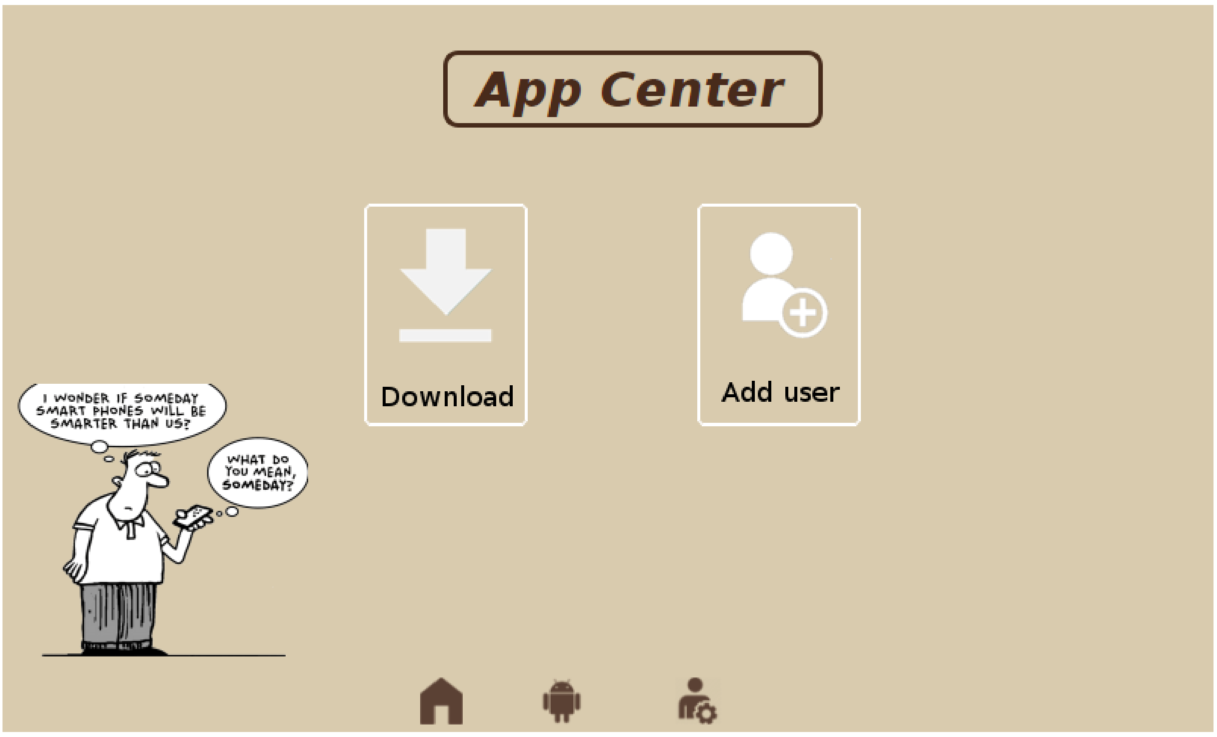

App Center
Download
Add user
I WONDER IF SOMEDAY SMART PHONES WILL BE SMARTER THAN US?
WHAT DO YOU MEAN, SOMEDAY?

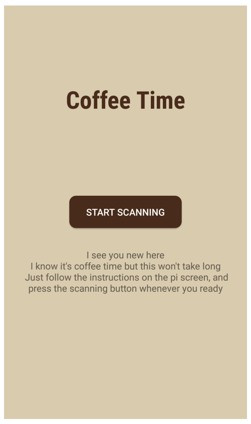
Figure 20: The App center page

The accounting page, provides an up-to-date statement of all accounts. This page is updated after each transaction. It can be viewd under `http://i80misc01.ira.uka.de/coffee/kasse.php?token=COFFE`

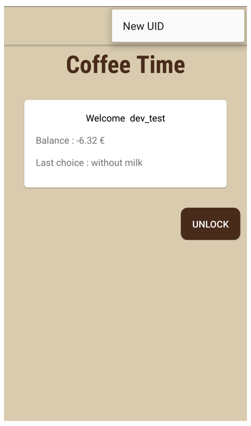
The last page, is the admin page. This section is password protected. After entering the right combination, the user has the possibility to unlock the coffee machine (for maintenance), enable/disable ssh connections, or exit the application.

3.3.4 Android Application

After downloading the application from the server, the user is presented with a welcome screen(Figure 21a) where he can scan his UID. This window is presented only the first time the user opens the application.


Coffee Time
START SCANNING
I see you new here
I know it's coffee time but this won't take long
Just follow the instructions on the pi screen, and
press the scanning button whenever you ready

(a) Android login page



(b) Android main page

Figure 21: Android application windows

When accessing the main window, the user can consult his balance, and unlock the coffee machine from distance. If the user changed his physical RFID, he can update his new UID by pressing the New UID button (Figure 21b).
This feature opens new possibilities : Offering coffee points to a friend as a gift . . .

3.4 Challenges

In this section, we will discuss the different problems we encountered during this upgrade. One of the main difficulties we experienced was a variant output from the coffee machine. The idle state would be different from day to day. This was caused by a bad connections with the PCB. The problem was solved after the installation of the final pcb and increasing the reading Frequency (From 10ms to 1 ms).

The main difference between the previous setup and this new upgrade is the number of devices managing the system. Due the different processes running concurrently with high frequencies, we encountered random misbehaviours. A custom scheduler was developed to improve the response time and reduce the chances of having two high priority jobs processed at the same time. This improved the responsiveness of the system, providing a better user experience.

Another reported problem was the RFID reader not responding or has a long response time. The issue could be hardware related (Faulty cable . . .), or software related. We first isolated the Hardware possibility by replacing the cables and testing the components seperately. Concluding that the problem wasn't hardware related, we started the reading code in a seperate script with higher CPU priority. Due the uncommonness of this problem, this solution is still not confirmed. If the problem persists, a radical action may be taken such as automatic reboot.

In order to improve the user experience and provide a faster response time, the suboptimal techniques used for the communication between the different processes were replaced by more complex and efficient methods. Each file was replaced by a flag in the shared memory. In order to prevent undesirable behaviour, semaphore is used to ensure that a value can be accessed by only one process at a time.

3.5 Costs

The different hardware parts cost 138,92 €as mentioned in detail in Table 1.

Item		
Component	Description	Price(€)
Raspberry pi 3	Model B	33,61
Touchscreen	LCD touch display 7”	67,22
RFID Reader	Joy-IT RFID Modul MFRC-522	8,39
Netzteil		10,92
Total		138,92

Table 1: Costs of the different Hardware parts

The other cost that must be taken under consideration is the power consumption. According to ... a raspberry pi 3 consumes around 3W in our case. The value of the kWh is around 0.2916 €as mentioned by ... This leads to a total cost of 7.47 €per year.

4 Conclusion

This chapter summarizes the work done during this thesis. The upgrade described above was developed in order to provide a solid solution to the daily problems encountered with a basic setup. The new system helps the coffee administrator by providing a secure and immediate connection between the user and the accounting system. This prevents endless tally sheets counting and accounts having a negative balance, plus no user can pretend forgetting to write his order. The upgrade provides :

- A decent authentication mechanism using RFID tokens, preventing unauthorized users from using the coffee machine.
- The possibility to use a smartphone as an authentication device
- Daily entertainment content

One of the faced problem was to discern the waking up state from a usual water order. A better approach would be to control the coffee machine directly, this would prevent any unwanted behaviour, or changing the logging mechanism. A sound based detection system can provide better results than the sensor's input, providing a better overview of the system.

References

[1] Tobias Modschiedler. *Upgrading an off-the-shelf coffee machine with RFID-based access control.* 2014.

[2] Saeco International Group. *saeco royal teh list : Service manual.* 2003.