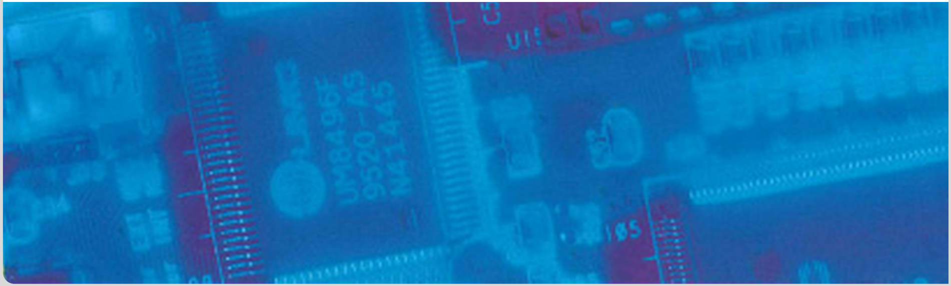
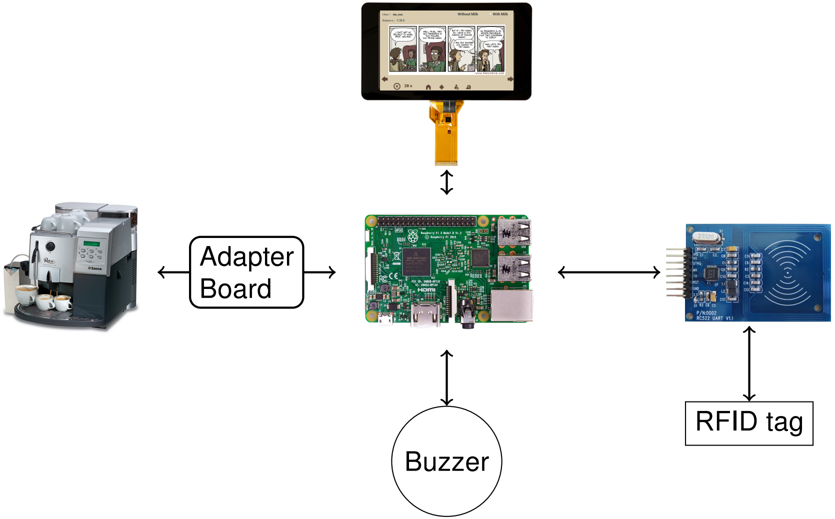
Improving Usability and Reliability of an IoT-based Controller for a Coffee Machine

Simon Korz – uhelj@student.kit.edu

CES - Chair for Embedded Systems



Introducing the Topic



Goals

Improve Usability and Reliability by

- fixing bugs
- redesigning the UI
- adding new features

Structure of this Talk

Introduction

Problem Analysis

New Design & Architecture

Results

Part 1

Introduction

Usability

"extent to which a system, product or service can be

used by specified users

to achieve specified goals

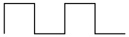
with

- effectiveness
- efficiency
- satisfaction

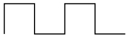
in a specified context of use"

Available Sensors

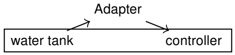
water flow

- 
- 5s-30s
- variable frequency, proportional to flow speed

grinder

- 
- less 1s
- irregular pattern

water level/blocking

- 

- input AND output

connected via GPIO

Accounting Server

Treasury, keeps record of

- Transactions (withdrawals, deposits, coffee bought)
- Users
- RFIDs

Technical:

- MySQL database
- Apache + PHP webserver

System Details

- Broadcom BCM2837, 4 core Cortex-A53 (ARMv8) 64-bit SoC @ 1.2GHz
- 1GB LPDDR2 SDRAM
- OS: Raspbian GNU/Linux (Debian 10 Buster)
- Language: Python
- GUI: PyQt5

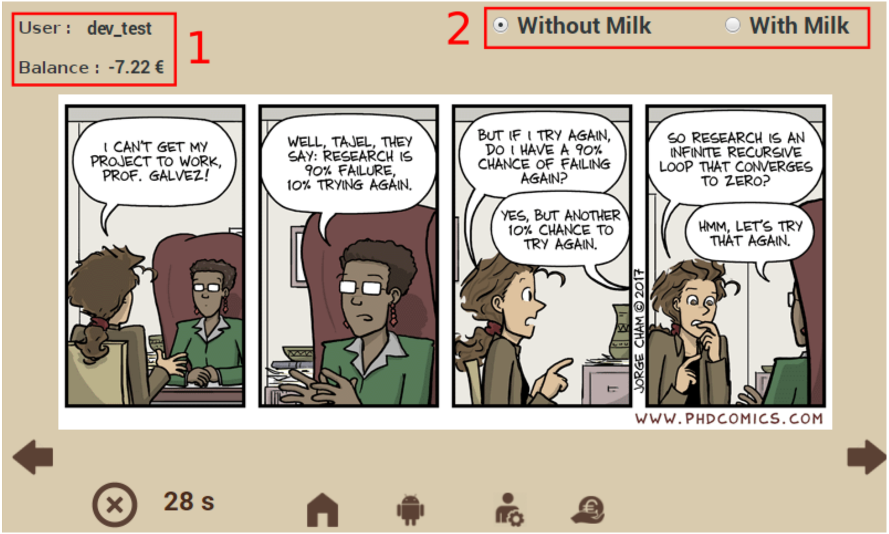
Part 2

Problem Analysis

21 Issues Identified

- GUI related
  - size of controls
  - focus on information
  - timeouts
- Bugs
  - unresponsive GUI/RFID reader
  - coffee not registered or recognized as hot water
  - offline orders not synchronized
- Missing features
  - support for new KIT-Card
  - dispensing limit
  - various buzzer sounds

GUI Problems



User : dev_test

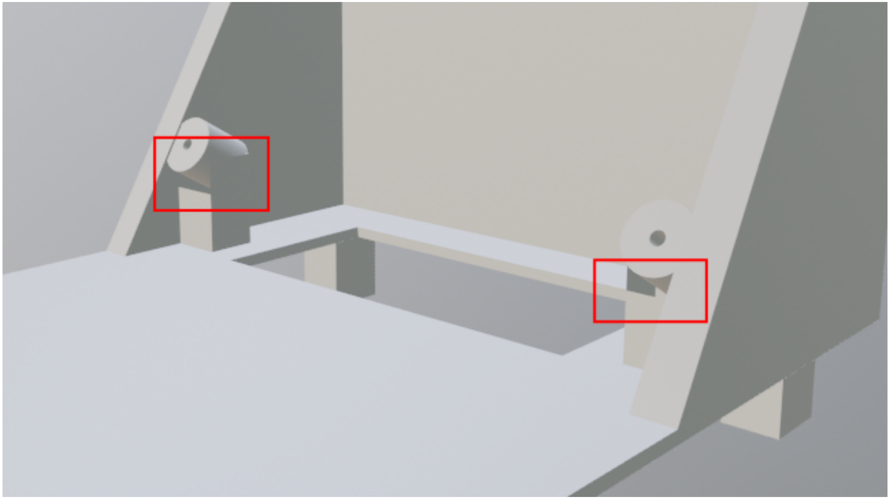
Balance : -7.22 €

[✓ Without Milk] With Milk

28 s

1
2
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

Unfinished Case



Code Quality

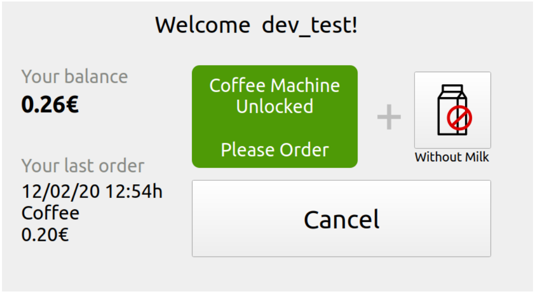
```
def printLog(array):
    ...
    if not os.path.isfile("/home/pi/CoffeeMachine/UI/maintenance.txt"):
        global previousGrinder
        if GPIO.input(grinderPin) == 0 and previousGrinder == 1 and not currentOrder == 'coffee':
            if os.path.isfile("/home/pi/CoffeeMachine/UI/order.txt") :
                with open("/home/pi/CoffeeMachine/UI/order.txt", "r") as j:
                    temp = j.readline()
                if temp == "water\n":
                    os.remove("/home/pi/CoffeeMachine/UI/order.txt")
            with open("/home/pi/CoffeeMachine/UI/order.txt", "a+") as f:
                f.write("coffee\n")
                f.close()
            with open("/home/pi/CoffeeMachine/UI/stop.txt", "a") as f:
                pass
            currentOrder = 'coffee'
            previousGrinder = 1
        elif GPIO.input(grinderPin) == 1 and previousGrinder == 0:
            previousGrinder = 1
        elif not (GPIO.input(waterFlow1Pin) == previousWaterFLow1) or not (GPIO.input(waterFlow2Pin) ==
            previousWaterFLow2):
            previousWaterFLow2 = GPIO.input(waterFlow2Pin)
            previousWaterFLow1 = GPIO.input(waterFlow1Pin)
            idleCount = 0
            if not os.path.isfile("/home/pi/CoffeeMachine/UI/order.txt"):
                with open("/home/pi/CoffeeMachine/UI/order.txt", "a+") as f:
                    f.write("water\n")
                currentOrder = 'water'
            elif not os.path.isfile("/home/pi/CoffeeMachine/UI/unlock.txt") and os.path.isfile("/home/pi/
                CoffeeMachine/UI/order.txt") and currentOrder == 'water':
                with open("/home/pi/CoffeeMachine/UI/unlock.txt", "a") as f:
                    pass
```

Part 3

New Design & Architecture

New GUI

- focus on important information
- large buttons, suitable for touch
- visual feedback for each step in ordering process

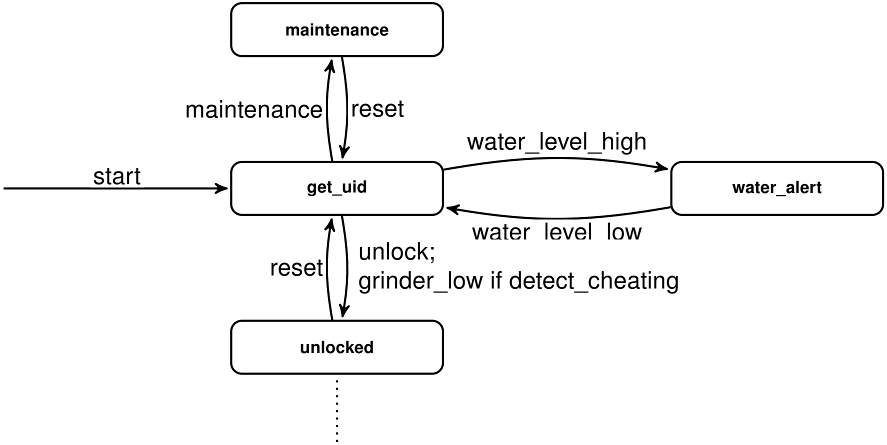


Welcome dev_test!

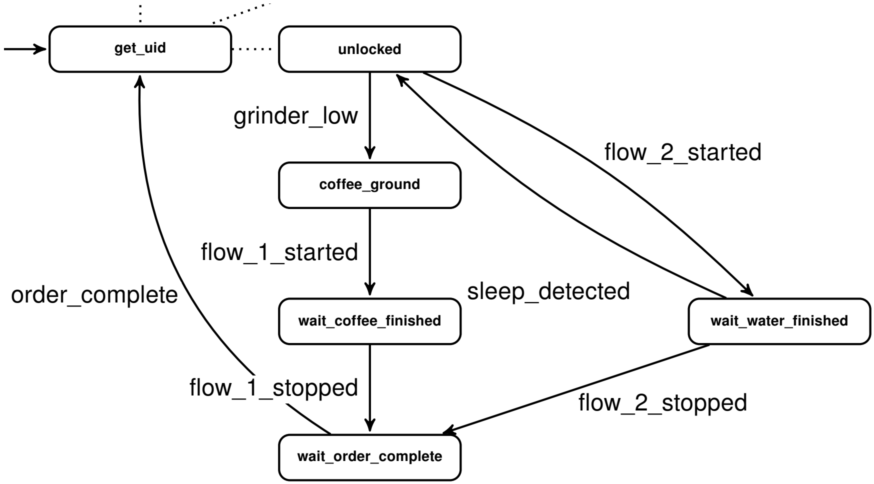
Your balance
0.26€

Your last order
12/02/20 12:54h
Coffee
0.20€

The State Machine I

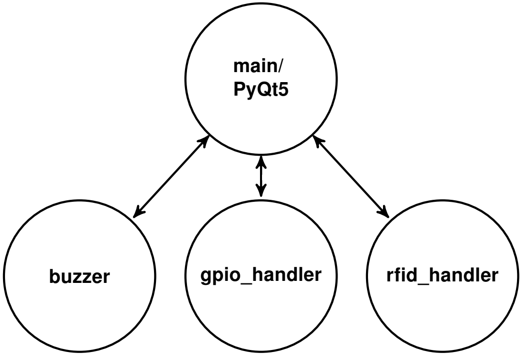


The State Machine II



Leveraging multiple CPU cores

- 4 cores available
- bidirectional pipes
- predefined messages
  e.g, CMD_PAUSE,
  CMD_RESUME,
  CMD_LOCK,
  E_GOT_ID

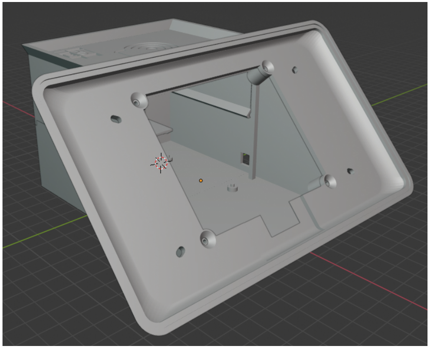


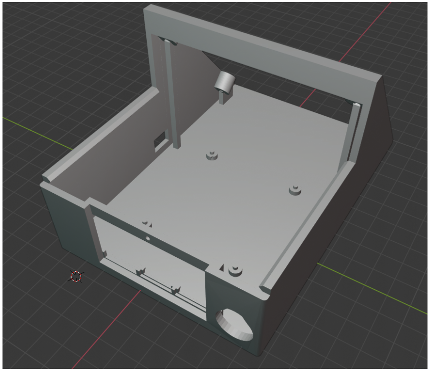
Changes in GPIO handling

- **callbacks** instead of **polling**
- Using pigpio library instead of RPi.GPIO
  - supports callbacks through hardware interrupts
  - noise filters

3D-Printed Case

- Difficulties: Non-manifold geometry, "not solid"
- Raspberry Pi held in place without screws





Part 4

Results

Printing Costs

- Resin tank: €65.45
- 1L grey resin: €160.65

$$\frac{\text{resin tank €65.45} + \text{resin €160.65}}{1000\,ml} = \text{€0.2261} \,/\, ml$$

- Top part: 90.79ml => €20,53
- Bottom part: 150.08ml => €33.93
- Screen frame: 53.60+ml => €12.12
- Total €66.58

Accumulated cost estimate €300

Reduced CPU usage

Measurements made with `ps` command one hour after boot

	%CPU	%memory	python module
Old	98.1	1.8	inputGPIO.py (GPIO & buzzer)
	12.4	2.1	inputGPIO.py (RFID)
	0.1	10.2	main.py (GUI)
	0.0	1.6	inputGPIO.py (locking)
New	5.9	0.1	pigpiod (GPIO)
	16.3	3.9	main.py (RFID)
	0.7	10.6	main.py (GUI)
	0.4	4.6	main.py (GPIO & locking)
	0.0	4.4	main.py (buzzer)

% CPU is the “cpu utilization of the process in "##.#" format. Currently, it is the CPU time used divided by the time the process has been running (cputime/realtime ratio), expressed as a percentage.",
% memory is the “ratio of the process’s resident set size to the physical memory on the machine, expressed as a percentage." [ps(1) manpage]

Improved Usability and Reliability

- Applied usability requirements
- **18/21 issues directly address usability**
- system observed to run for 1 month without restart
- positive user feedback

Questions

3D-Printed Case