Automated Card Payment Testing

A.J. de Graaff
W. Zirkzee

TU Delft

Automated Card Payment Testing

by

A.J. de Graaff
W. Zirkzee

to obtain the degree of

Bachelor of Science
in Computer Science

at the Delft University of Technology,

Student number:	4012534	4398858
Project duration:	April 24, 2017 – July 3, 2017	
Coach:	Dr. RangaRao Venkatesha Prasad	
Client:	bunq B.V.	

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TUDelft

abstract

Every development roll-out, there needs to be certainty everything works. Especially when it comes to the banking domain. People care about their money and want it to be handled with care. Currently this means some tests, including making a payment using a Maestro debit card, need to be performed manually. In order to further testing, this project aims to automate these Maestro payments. This has been achieved by constructing a robotic device to perform the physical actions associated with creating a payment request, and create a piece of software to verify if the payment was accepted. The main software is based around the public API bunq B.V. offers. Using the API it is possible to retrieve information related to your bank accounts, including payments made. The robotic device is consists of a combination of solenoids and servos to operate the keypad and to present a bunq Maestro debit card. Additionally the system has extra functionality such as self assessment if the Maestro workflow is still functioning. As a result there should no longer be a need to physically take a terminal home and complete payments. Even though the delivered system does not have functionality to work with a phone's NFC, the system should be easily upgraded to support this functionality.

Preface

This report documents our bachelor project, at bunq B.V. working on a solution for testing of physical MasterCard payments. It has been written in order to receive a bachelors degree in Computer Science at the Delft University of Technology. The project and report have been written and completed from April 24 to July 3 2017.

The project has been undertaken at the request of bunq B.V and would not have been possible without the following people, for this we would like to thank:

RangaRao Venkatesha Prasad as our supervisor from Delft University of Technology, guiding us during the project.

Wessel Van for being our manager and helping us find our way at bunq B.V.

Ali Niknam for having projects available and hiring us as interns to complete the project.

Furthermore we would also like to thank all of the colleagues at bunq B.V. who have helped us in any way.

We have enjoyed our time at bunq B.V. and it has been an interesting and wonderful new experience.

A.J. de Graaff
W. Zirkzee
Delft, June 2017

Contents

1 Introduction

Everyone uses it and there is no way around it, but many are not satisfied with the current system. We are talking about banking. Ali Niknam is one of those people, but his new mission is to fix the flaws in banking.[5] Bunq is Ali's new project, a new online bank, or as they like to say themselves an IT company with a banking license. [11]

As part of the bachelor project bunq has employed Wouter Zirkzee and Arend Jan de Graaff in order to complete a project they would like to see realized. The project is to develop an automated card payment testing system. Currently after each new development rollout, that happens every tuesday night, a DevOps employee has to take a payment terminal home and manually complete a transaction in order to make sure the system still functions correctly. By automating this payment cycle it can easily be tested at any moment, from anywhere.

Within this report, you will find how we have achieved our goal. Chapter 2 starts with describing the problem and an analysis of what the client expects. The third chapter will provide an overview of technologies that can be used. Followed by a chapter that will provide a description of the design that has been created for the final product. Chapter 5 documents the result of the three development sprints during the project, documenting what has been completed during the sprints. Finally we will conclude our thesis by evaluating what we have achieved, looking back at set goals and results, give recommendations for future work, and talk about ethical issues we have encountered during our project.

2 Problem

2.1. Description

Every development rollout, usually Tuesday night, bunq manually checks if card payments with a bunq card still work. In the current situation this is done by having a DevOps employee take a pin terminal home, and make 4 payments of €0,01 to confirm that the payments are still processed.

The goal of this project is to develop a system that relieves the DevOps employees from the need to take the pin terminal home and complete the transactions. Not having to transport the device means the chance of breaking or losing the machine is reduced. As specified in the companies project description (appendix A), this project consists of both a software and hardware part as a physical payment has to be made.

2.2. Analysis

In order to understand the problem from the clients perspective, a list of requirements that have to be completed in order for to achieve a satisfactory result at the end of this project was composed.

2.2.1. Requirements (MoSCoW)

This section lists the requirements for the final product. Using the MoSCoW method the requirements are separated based on their priority. Must haves are the Minimum Usable Subset of functionality. Should haves are included if possible with regard to time. Could haves are requirements that are nice to have (i.e. increased user experience), but not necessary.

2.2.2. Must Have(s)

- **Control Touch Screen**
  The system must be able to control the touch screen as this is required to set up a payment on the terminal.

- **Control Numeric Pad**
  The system must be able to press the buttons on the terminal's numeric pad in order to set an amount, confirm and enter the PIN code.

- **Use Contact EMV Card**
  The system must be able to insert and retract a PIN card with EMV chip to continue with the payment.

- **Verify Payment**
  The system must be able to verify the payment has been successfully completed by reading and confirming the payment in the back-end after the payment is accepted on the terminal.

- **Start on demand**
  The system must be able to start a testing cycle on the demand of bunq.

- **Web Interface**
  The machine should be run a server and display a website. This website must give information about the status and the ability to start a new test.

2.2.3. Should Have(s)

- **Use Wireless EMV Card**
  The system should have the ability to make payments using a wireless EMV card by presenting it to the terminal's NFC reader, in addition to contact EMV.
- **Self start**
  The system should be able to detect when the Maestro payments might not function correctly, if this occurs the system should autonomously start a new test.

2.2.4. Could Have(s)

- **Counter**
  For team spirit and confidence in the system it would be nice to have a counter, keeping track of sequentially successful completed payments made.
- **Optical Feedback**
  The system could have optical feedback (e.g. webcam pointed at the terminal) to monitor if any errors might occur.
- **Individual Control**
  The system should allow to control the terminal with each keypress. For example, only press one number on the keypad instead of run the pre-designed test.

3 Background

This chapter describes the equipment and techniques that will be used during the project and play a role in designing the solution. Finally in section 3.4 an overview is presented of similar solutions that are already commercially available.

3.1. Payment Terminal

At bunq B.V. a VX 680 GPRS payment terminal [20] equipped with CVV software is provided. This is a portable battery powered point of sale terminal which has an integrated numeric pad, multi-color touchscreen, triple-stack magnetic stripe card readers, EMV smart card reader, contactless payment features, internal thermal printer, one port for supporting different peripherals, and a connector for docked charging. The device is also equipped with a SIM Card for GPRS communication to allow payment without limitations of its location. To connect peripherals the following cables and adapters are available:

- USB Host Cable
- Multi-port Adapter
- Modem Dongle
- Serial Dongle
- USB Serial Dongle
- Full Featured Base with multiple ports

The multi-port adapter provides connections for power, USB, mini USB, RJ45, and COM1. This cable is designed for development and deployment purposes. However as the device already is a few years old, most support has been dropped making further documentation and resale of these items very hard to find.

3.2. Payment Methods

In order to conduct a sales transaction on the VX 680 the following actions need to be executed: first 'payment' has to be selected on the touch screen of the payment terminal. A new screen is shown and the amount should be entered using the numeric pad, in an appending manner. End the payment setup by pressing 'OK', and start the transaction. The payment terminal will show the amount requested and ask to insert a card.

Using the VX 680 payment terminal multiple payment methods are accepted: Europay, MasterCard and Visa (EMV), Near Field Communication (NFC), or magnetic stripe cards. However since magnetic stripe cards are deprecated, this technique will not be taken into account.

3.2.1. EMV

Transactions using EMV are the global standard for chip payments using chip technology. Currently the EMV specifications are managed by EMVCo, an organization which consists of American Express, JCB, Discover, MasterCard, UnionPay and Visa. The most recent statistics report that in Q2 of 2016 42.4% of the transactions were completed using EMV technologies. Up from the previous years 2015 and 2014, 35.8% and 23.9% respectively. One major exception is The United States with only increased with 0.26% from Q3 2014 to Q2 2015 and 7.20% from Q3 2015 to Q2 2016. [1]

EMV transactions can be split up into two categories, contact and contactless. Contact transactions require to physically insert the card into the reader, often these transactions are referred to as Chip and PIN, because verification that the chip belongs to the cardholder is done using a PIN entry. Contactless uses the ISO 14443-3 communication protocol and only requires the card to be in short range (approx. 4cm).

3.2.2. NFC

NFC is a technology enabling short-range wireless communication between an initiator and a target. It builds on High Frequency (HF) radio-frequency identification (RFID) using mainly the same standards: ISO 14443 Type A&B and ISO 18000-3 [13]. The initiator is always considered active generating a radio-frequency field (RF). The target can be either passive or active. Passive targets have to be powered by the initiator, which makes it simple to create unpowered stickers, cards, etc. Active targets create their own RF which makes it possible to have peer-to peer-communication. In order to function the initiator sends out a signal to the target at 13.56MHz. Targets within 4cm can receive and respond with the requested information. For more sensitive information, such as payments, a secure channel is established and information send encrypted.

3.3. Transaction

As mentioned before the terminal has two categories, contact and contactless. Although they both use the EMV protocol only the contact-payment has a 'secure' transfer. Contact payment in this case uses the cryptogram key on the chip in combination with the payment details -amount, random number, currency, date- to generate a cryptographic key, and sends this as an authorization request. The issuer (terminal) then follows the same steps to create its own cryptographic key. If all the data was correct on both sides, the key should be the same. The issuer now uses these data elements, combined with the cryptogram key of the bankcard and a response code, to generate a new cryptogram. The new cryptogram and the response code are send back to the chip. The chip then uses its information to cryptogram key, response code and the cryptogram code it sent to the issuer to generate a cryptogram key which should be equal to the key it just received from the terminal.

When the terminal receives a contact payment it will perform both a chip validation and sends the cryptogram to the backend, where the backend will respond with an issuer validation. However when it receives a contactless paymement it will also send a chip validation, however when the backend sends back the issuer validation, the bankcard will no longer be in contact.

3.4. Commercial off-the-shelf

There are multiple point of sale testing devices already commercially available. Commercial solutions include, but are not limited to:

- UL Terminal test station
- Rhiscom
- GETECSA Robot PinPAD
- GrupoHDI
- Abrantix AG

While all of these would be more than sufficient to solve the client's need, there is a common flaw rendering all of these unavailable. These solutions are very expensive, not all of the devices have a listed resale price, but after seeking contact with quotes of an 'excess of €100k for this type of solution' have been received.

4 Design

This chapter will describe the requirements of the final system and the choices that are made during the process of getting the final design.

4.1. Concept

In this section a high level design is described based on the background information in section 3 that fulfill the requirements listed in the previous section 2.2.1.

The first part of the requirements is to setup the transaction for which there are two possibilities, electronic or physically. To set up a payment electronically there should be a way to send a signal to the device, either using an adapter or a new terminal that has additional connectivity. However due to limited availability of the required adapters and related documentation there was not enough confidence in this option. The second way is to build a robotic system that physically presses the right buttons to set up a payment.

In order to complete the transaction a card or device has to be physically presented by either inserting or kept near wireless connectivity, and in case of contact EMV a PIN code has to be entered.

When evaluating two options of the design concept, the physical approach has been chosen as there already was a need for robotics to complete the transaction using a card and entering a PIN code. Once the payment has been made on the terminal, the transactions should be monitored in the back-end and make sure the transaction is processed as expected.

4.1.1. Budget

As bunq B.V. could still be considered a start-up company, they do not have a large budget. While the client did not give us a specific budget, they want a decent solution while keeping the cost down as much as possible.

4.2. Hardware

In this section an overview is provided of the available hardware to build the physical system, deciding between pre-built options or the components that would be used for a self built solution.

4.2.1. Direct control

Controlling the terminal could be done directly by removing the buttons of the keypad from the terminal. This way it is possible to attach cables directly to the contacts of the keypad. Then there will be no need to use a robot to press buttons since a simple 5V signal can imitate the button presses. However, the terminal has tampering protection. This means that when tampering is detected the terminal becomes useless. One can assume that the keypad has this kind of protection on it, since some people might otherwise want to use it to find PINs from the customers. Because it is likely that removing the keypad and soldering wires on it will block the terminal, it is a bad idea to remove the keypad.

4.2.2. Pre-built robotic arms

Initially pre-built robotic arms were considered. Prices of these arms can range from as low as €30 to well over €10.000. Deciding which one would fit the project is based on two main criteria: accuracy, repeatability, duty cycle and torque. Accuracy is the minimum amount of degrees the motor has to turn. Repeatability is how accurate it can perform the same task repeatedly. Duty cycle is the amount of time the signal is active during an on-off cycle, usually expressed as a percentage. This percentage also relates to the time it can be turned on before overheating. Torque specifies the amount of pressure the arm is able to exert fully extended, when it is at its lowest. Robotic arms with minimal specifications for this system were priced above €400.

4.2.3. Controllers

The following solutions will not work with software input, but just by purely using the terminal and using robotics and micro-controllers to do the same.

- **Mindstorms**
  The Lego Mindstorms series are kits containing software and hardware to create Lego robots with educational intentions. The strong point about these kits is that Lego has a proven track record for being very durable and that it is very easily assembled and expanded. The Mindstorms series uses its specially designed sensors and motors. While in other designs the amount of motors and sensors can be easily ordered extra from any kind electronics shop, with Lego it is limited to the continuation of the (Mindstorms) series and the amount of motors and sensors in the box. However, spare and extra parts can be ordered from certain dedicated stores. But at the moment of this report these motors and sensors are in the range of 25—40€a piece. Compared to normal motors the costs of these are very high. Furthermore the starter kit starts at 350€. The combined costs of the motors and the kit will therefore be too high compared to the other alternatives discussed below.

- **Arduino**
  As micro controller board an Arduino can be used, its strength lays in controlling electronics that the machine uses to input the card and press the buttons on the keypad. The Arduino does require another extra shield to be connected to the internet. A couple of Arduinos could be suitable for this project:

  – Arduino UNO (AVR(16MHz), 8bit, 5V, 20€)
  – Arduino Yun (WiFi, Ethernet, AVR(16MHz), MIPS(400MHz), Linux, 8bit, 5V, 65€)
  – Arduino TIAN (WiFi, Ethernet, ARM(48MHz), MIPS(533MHz), Linux, 32bit, 3.3V, Bluetooth, 85€)

  The Arduino micro controller boards are in general used for non-complex projects like giving a start signal to a dc-motor. Because the robot has to be connected to the internet or it should represent its data in a more complex way—like an user interface—, Which is possible, but may make it unnecessarily tougher with an Arduino. Not to mention that the Arduino models that have internet and video output are rather expensive compared to their Raspberry PI equivalent.

- **Raspberry PI**
  Another possibility for a micro controller is the Raspberry PI. Its main strength lays in the connection to monitors, servers, peripherals and its higher processing speed. Since the Raspberry PI is basically a mini computer connected to the internet it can use many tools designed for PC's and web browsers. Possible uses could be a remote desktop tool. Making the sharing of video and data a lot easier since this could just shared with a desktop sharing tool instead of being send via a custom made live stream. Possible Raspberry PI models could be used, these are:

  – Raspberry PI A+ (ARM(0.7GHz), 32bit, 256MB RAM, 30€)
  – Raspberry PI 2 (Ethernet, ARM(0.9GHz, 32bit), 1GB RAM, 35€)
  – Raspberry PI 3 (WiFi, Ethernet, Bluetooth, ARM(1.2GHz, 64bit), 1GB RAM, 35€)

From this list it is quite clear that with multiple internet and wireless connections, a higher clock speed and a 64-bits system that the Raspberry PI-3 is the clear winner.

4.2.4. Actuators

To physically press the buttons on the payment terminal something is needed that can touch and let go of a button, can be used for a long time, can be controlled by a Raspberry Pi and needs to be precise. Possible solutions are: servo's, solenoids and linear actuators.

- **Servos** have a low energy use and are very precise. The disadvantage of servos is that they are quite large and require another large area to turn their arm. Servos have a low cost
- **Solenoids** use an electromagnet to quickly push or pull a slug of metal in or out a casing. Furthermore it can be placed fully vertical, meaning it will just require vertical space. Solenoids also have a low cost.
- **Linear actuators** use a small motor to turn a gear that will move a rod in and out. They are also small and move vertically. However they are relatively expensive.

There could be two possibilities for testing payments via the payment terminal. The least complex, cheapest and most reliable way would be to cover three buttons on the keypad (1, OK and Cancel), the 'start payment' button and a mechanism to hold the card over the NFC antenna. This however does not match the wishes of the client, and therefore not requirements set for the project. The final product however will implement all 15 buttons on the keypad, 'start payment', insert card (chip) and 'insert' card (NFC). These different movements for product would need their own kind of motor.

Start payment
To start the payment, the screen needs to be pressed by something. It would make sense to immediately jump to the solenoid. However, during testing it turned out that this approach would lead very probable to damage to the screen. Another problem is that extra support would be needed to keep the strength of the solenoid in check. It is therefore decided that it would be most useful to go for another option. The next choice is the servo. A servo would be able to use a rod to push the screen at a low speed, therefore it would not damage the screen that easily. Another advantage is that the servo can be placed further away from the terminal. This allows for more space above the screen which makes sure that NFC payments can be performed as well.

Enter Amount/PIN
When the payment is started it is required to enter the amount to pay and providing the PIN after the card has been inserted. To do this, it is required to push the buttons on the keypad. There could be multiple ways to push these buttons. The most reliable way is to have a matrix of solenoids on top of the keypad. A matrix of servos would be rather difficult due to the size of the servos.

Insert/Remove Card
During and after the payment the card will have to be inserted and removed. Considering the servo and solenoid possibility the solenoid would jump out since it is very good in pushing or pulling. However that is also its weakness. The solenoids tested here have a spring to put them back in position. This spring would then have to pull the card out afterwards. If the spring were to be powerful enough to pull out the the card that would mean that this the solenoid should have twice the power of this spring. Those solenoids will likely become very powerful, and therefore very expensive. Servo's on the other hand are precise, and have a lot of torque to push a bankcard into the slot. Since in this case only the chip is required in the payment, a hole could be drilled into the bankcard to attach it to the servo.

NFC Payment
The main difference with the NFC payment over the chip payment is the lack of a PIN requirement. In this transaction the card only needs to be held over the screen. Tests pointed out that at very certain locations the card could be put perpendicular to the screen, however keeping it horizontal made payments a lot easier. The NFC antenna's has more range than expected, about 3cm from the edge of the screen payments could still be made, so a machine that has a relatively long range would be needed here. The best and easiest solution should therefore be a servo with an arm connected to it with on the end the bankcard. This card should either be glued or put in some sort of clamping device to prevent damage to the antenna.

4.3. Software

The following section discusses the reasoning behind the elemental software choices that will be implemented over the course of the project.

4.3.1. States

The payment terminal follows a certain pattern. The tests should follow this pattern to perform a successful transaction. Therefore the machine should be programmed in the way of the state diagram shown in figure 4.1. The machine should start in its start state S_0. Then it will press the display of the terminal S_1. This will then allow the amount to be input using the numpad. The machine should input a small amount to be paid. So it should always be able to make a transaction. This amount will have to be confirmed by pressing the ”OK” button S_2. At this moment the card should be inserted S_3. The terminal will ask for the PIN of the card, which the machine will input and confirm with OK S_4. If everything went properly, the payment has been done and the machine will wait for the back-end to confirm the payment S_5. This means the machine has successfully completed the test and will reset itself to prepare for the next test. If the confirm did not come for a certain amount of time the machine will return that the test failed and will reset itself.

4.3.2. Operating System

There are multiple operating systems available for the Raspberry PI. The most discussed are Raspbian and Ubuntu Mate. Raspbian is a dedicated OS for the Raspberry PI. Raspbian includes all libraries supported by the Raspberry PI community[3]. And Raspbian is the Pi Foundation’s official supported Operating System[2]. Ubuntu Mate is an Ubuntu version —a Linux distribution— specialized on being used on devices with limited processing power[19]. Ubuntu Mate is compiled in a way to specifically work on the Raspberry PIs one and two, this means that it will be just this bit faster than Raspbian. But it also means that it will not work on any other Raspberry PI. Ubuntu Mate does not support all of the libraries from the Raspberry yet[19]. This could make the project unnecessarily harder, since speed is not a real hard requirement. Therefore the project will be running with the Raspbian OS. The Raspbian OS has two versions, one with a desktop environment and one without. The advantage of not having a desktop environment is the amount of space saved (300 MB compared 1.3 GB). The advantage with a desktop environment is that every application is just shown on the desktop.

The Raspberry PI’s native language is Python, a widely used high-level programming language for general-purpose programming. Furthermore Python has a lot of libraries and online support for a lot of applications, like hardware and server control[3].

4.3.3. Web server

The Raspberry PI should house a web server and run a web interface on this. As the Raspberry PI will run using a Linux distribution, there is a wide range of available options where Apache and Nginx are the two most popular. [14] [15] Apache has been chosen due to native ability to process PHP and the extensive amount of first- and third-party documentation as a result of being a popular option for a long time, over 20 years.

4.3.4. High-level-design

The software to be run should have of a controller for the hardware, a connection to the back-end, a web interface and a way to create tests inputted by the user from the web interface. An high level description is shown in figure 4.2. A mainController object will be created, which will connect to the back-end, the web-interface creates an hardwareController object. The main controller has the ability to extract data from the back-end, send/receive data to the web-interface and give/take the data for the hardware controller.

4.3.5. Programming languages

As this is considered a side project by the client, the code base can be written in any programming language it deems fit. The team has previous experience with numerous languages, but mostly Java. However the client’s back-end of is written in objective oriented PHP. Due to this using PHP has a big advantage if any core dependency might be needed. A downside is that neither of the team members have previous experience with PHP. The team agrees the two pro’s outweigh the lack of experience,

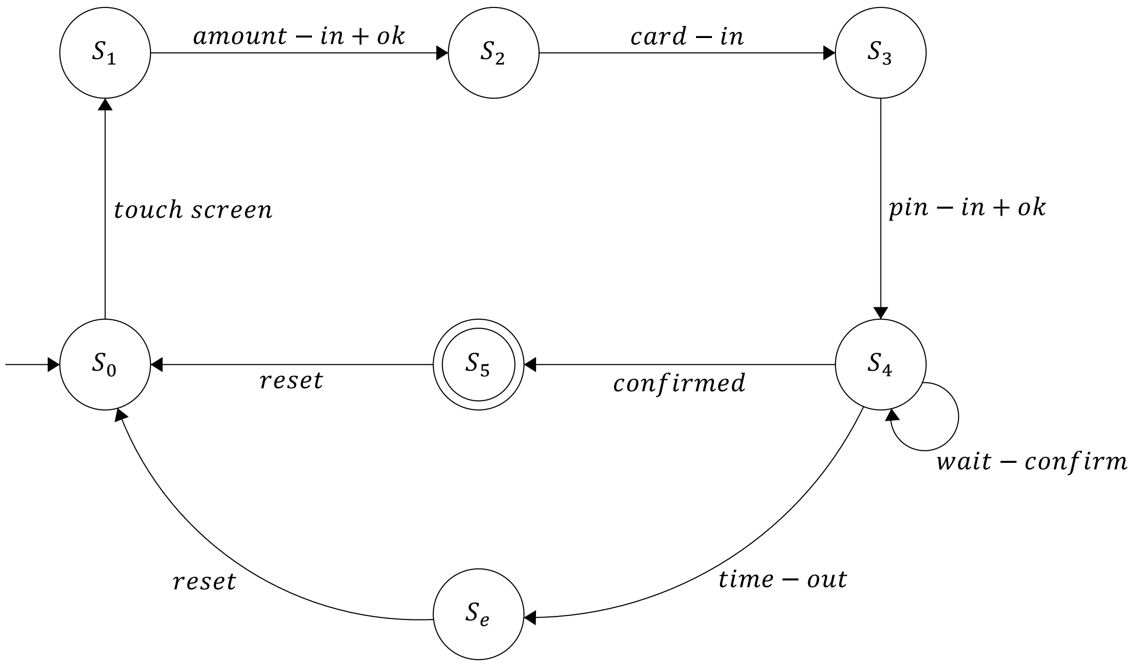


Figure 4.1: Payment terminal state diagram

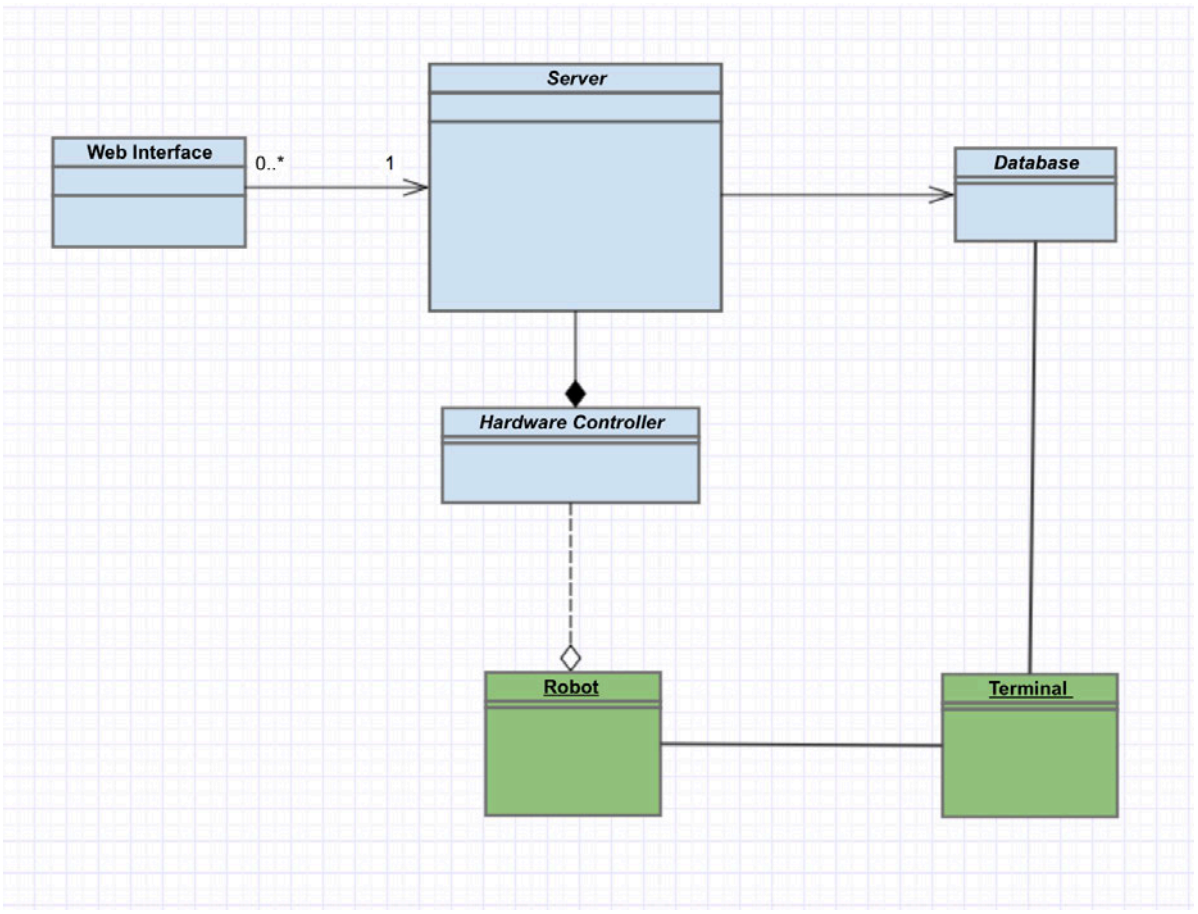


Figure 4.2: High level software design

as the team feels they will be able to learn PHP quite fast.

4.4. Final Design

The research on both the hard- and software have lead to the following design. For hardware a rigid base will be created around the payment terminal, with a solenoid mounted above each button on the numeric pad and the screen. The solenoids will be controlled by a Raspberry Pi to achieve the desired states according to the state diagram (figure 4.1). The Raspberry Pi will run Raspbian with an Apache web server which will serve as the interface and input of the system, as it should display the current status and statistics, and allow to manually start a test when desired.

5 Development

This chapter documents the project's development throughout the assignment. First the desired development process is described in section 5.1, all following sections will document the progress that has been made per sprint.

5.1. Development Process

During the development of this project, an agile software development methodology is used. Agile software development is a way of project management that allows software developers to quickly adapt when requirements and solutions change over the course of the project. This project has been divided into three sprints of two weeks. During each sprint new features need to be implemented and deployed keeping a functioning version available at all times. For this project sprints will last two weeks, too have enough time to get familiar with new techniques, implement and test. The first goal is to create an minimal viable product, once this has been achieved extra functionality will be added to improve the system during the following two sprints.

5.2. Sprint 1: minimal viable product

The first sprint of the development is focused on getting a minimal viable product running to get most of the hardware issues out of the way. This means the robot has to be build, and be able to complete the actions to make a payment. This payment has to be verified and displayed on a web interface. During this first sprint the team will split up the hardware and software to have a minimal viable product running as soon as possible.

5.2.1. Web Interface and Database

The web server is hosted on a Raspberry Pi 3, running an Apache 2 server. A web interface has been created using HTML, PHP, jQuery and AJAX. The interface contains a single button and a placeholder to display the history of results. The button will send an AJAX request to start a new test. The placeholder will be periodically refreshed by continuous AJAX calls injecting a table with an entry for each result in the database. They are ordered on descending date, showing the newest, most important results first.

5.2.2. Database

The results gathered from the tests will be stored in a database. The client already has a MySQL database in production. Further as there are no special requirements for the database in this system, any general-purpose database management system will suffice. For those reasons this system will also use a MySQL database.

The preliminary design only contains one table, with two columns. One column for the result of the test (type VARCHAR 250), and one for with the date (type DATETIME) and time to identify the results.

5.2.3. Payment Verification

Whether a payment is accepted or denied for any number of reasons (e.g. insufficient balance), processed in the back-end and stored in the database. A company such as bunq. B.V. is quite protective of their database, and would prefer to have as little interactions with the database as possible. After looking in to alternative ways this could be done have chosen for a safe approach using the public API. In order to use bunq's public API, first there are a few setup requests needed before being able to request any actual data.

The payment verification using bunq's public API[6] consists of a total of 5 API calls. The first 3 calls are part of a setup to open a session. After this two calls are required to retrieve the actual information we need to verify the payment.

Setup

In order to receive any meaningful API response a few steps are required. First a public and private key need to be generated. The public key needs to be shared with the server using

```
POST /installation
```

endpoint. The server will respond with a token and the server's public key to be used to verify future responses are from the bunq API. All following requests need to be signed using the private key and all the correct headers using the SHA256 algorithm.

All calls need to be made from a registered device. The endpoint

```
POST  /device-server
```

registers to the IP from which it is send, or any others provided in the body. The body also has to contain the API key associated with the account you want to use, and a name to identify the device. The API key will be bound to the IP address of the newly registered device, so it is important the IP address does not change.

Finally a session needs to be opened at the following endpoint.

```
POST /session-server
```

The session response will contain another token, required for all requests send over the session. Once this session has been established it will stay available for an amount of time set by the auto logout time in the user account, with a maximum of one week.

Requests

Now that a session has been opened requests can be send to fully control the account linked to the API. Every IBAN number is linked to a monetary account. So by using the endpoint

```
GET /monetary-account
```

a list of all monetary accounts belonging to the user is returned. By scanning through the accounts until the the correct IBAN number is matched the MonetaryAccountId can be found.

Finally a request can be made a list of all MasterCardActions from that account with

```
GET /monetary-account/id/mastercard-action
```

These MasterCardAction responses will contain information such as the decision if the payment has been accepted or denied. If it is denied a textual explanation of why it was denied. This decision is stored in the database, with the current date and time in order to keep track of when each result was obtained.

5.2.4. Structural Components

The structure needs to hold the solenoids, servos and payment terminal. The first iteration will test the ability of the structure to hold the terminal and the solenoids. While keeping a manual button press for the start of the payment. This way less parts need to be cut which will save money on the amount of objects to be cut. Since there is very little space between the solenoids the structure will have to be cut out very precisely. The cheapest way for very precise cutting is laser cutting. Laser cutting will therefore be used to obtain the parts. Appendix E.1 shows how this is cut out of $1mm$ acrylate, and the second drawing, which is 50% it's original size, is cut out of $3mm$ acrylate. In the drawings of the first design the structure is made up of two length elements which are connected by multiple width elements. The terminal will be put in between. The structure will elevate the pin terminal so that the terminal's keyapd is horizontal under the solenoid assembly. This will also make sure that the structure will not be pushed up instead that the button will be pushed down. As mentioned the material chosen for the structure was acrylate, because it is a see through material this would add to simplifying construction, making it easier to spot issues and make the structure more aesthetically pleasing.

5.2.5. Electronic hardware

To control the electrical flow to the solenoids a control circuit is required. As this is just a switch sprint one will create a circuit that drives the solenoids. Because the Raspberry Pi GPIO (general purpose input output) pins can only safely provide $3.3V, 16mA$[4] while the solenoids run on higher power $19V, 1A$ they can not be directly controlled by the Pi. Therefore they have to be controlled by an external power source and switched on and off with a switch. The Pi will send the up signal for the switch. This makes the switch conduct current through it. A transistor (appendix C.2) will be used as a switch. In the control circuit (figure 5.1a) a flyback diode is shown parallel to the solenoid. Flyback diodes are used to protect the circuit from sudden power surges. Because the solenoids create a power surge upon activation this diode is needed to catch this peak. A simplified version of the design of the full circuit can be found in figure 5.1b.

5.2.6. Pi control

The control of the solenoids via their transistors should be the easiest to do. Therefore the focus of the Pi control in sprint one is to get the solenoids working via their transistors. Python has the ability to easily import all the libraries required to control the GPIO pins of the Pi. The control of the transistors is relatively easy. All the Pi has to do is send out an high or low signal to the designated pin. This makes the transistor switches to respectively 'on' or 'off'. When switched on, the transistor will let the current flow to the solenoid making it extend the metal rod. When switched off, the transistor will block the current flow, allowing the solenoid to retract. The following code is part of the Solenoid class. This method instructs to send an high or low signal —where self.piPin is the pin on the Pi of this solenoid object—

```
GPIO.output(self.piPin, GPIO.HIGH)
GPIO.output(self.piPin, GPIO.LOW)
```

The control function of the solenoid is contained in the solenoid's own class, this function gives either a high or low signal.

5.3. Sprint 2: extra functionality

In the second sprint software focused on implementing extra functionality, and improving the code from the previous sprint. Hardware had faced some setbacks, which have led to taking longer than expected to produce a working system. For this reason the hardware minimal viable product development has been extended into the second sprint.

5.3.1. Configuration File

To create a minimal functioning version some settings have been hard coded. To increase the ease of making simple adjustments (e.g. new IBAN number or API version) to keep the system functioning, it should make use of a configuration file. A configuration file containing settings that are susceptible to change have been stored in JSON file format. The server will load all settings from JSON format into a mapped array. This array is to be accessible in order retrieve and use the desired settings. This utility will have to be extended as new features are added.

5.3.2. Styling

To increase both convenience and the look of the web interface, styling has been added by making use of the Bootstrap library. This library has been chosen due to the ease of use, which will save time compared to creating a stylesheet from scratch. After applying and tweaking the library to fit, the web interface has a clean and user friendly look. The most important difference is that the results are displayed in a color coded table: successful payments are shown in light green while a denied payment show up red, making it easy to recognize the results in a single look and spot if a test has failed. See figure 5.2.

5.3.3. Real time Testing

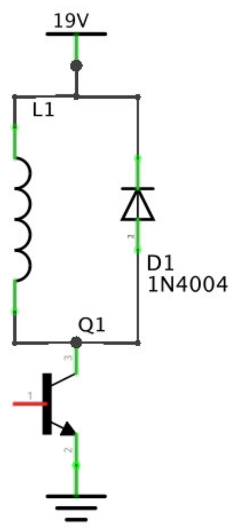
One of the features the client would like to see is for the system to start a test by itself. Leaving the machine to run continuously is not an option, both the physical machine will wear/burn out and there is a cost attached to running a test. A payment has to be made with a minimum value of €0,01. While this cost is payed to bunq, there is a transaction fee associated to making the payment as it relies upon MasterCard services. While the cost seems low, constantly making payments can add up to a running cost of €1000 per year, besides shortening the lifespan of multiple parts of the device itself.

The system needs to evaluate if the MasterCard workflow still works as expected. If it does not, the system should start a new test automatically. The idea is that if costumers are successfully making PIN transactions, there is no need to run a test. To accomplish this a real time data feed of recent payments would be needed.

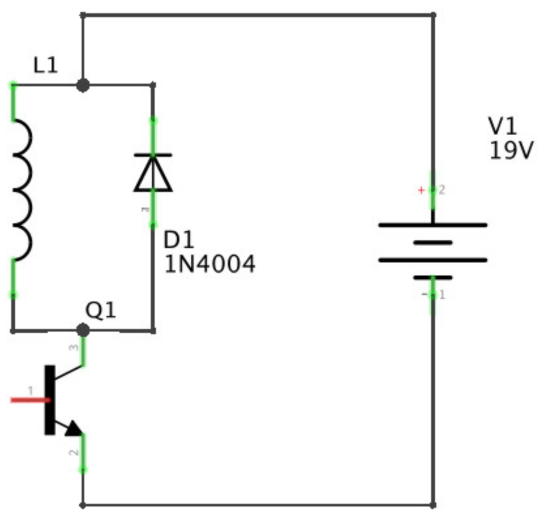
Bunq B.V. already feeds time-series data into Graphite, an monitoring tool.[8] One of these data data entries consists of the amount of successful payments made. Graphite is a monitoring tool to store and track numeric time-series data. Using Graphite Render URL API [9] this data can be retrieved, based on relative or absolute time indications, in a JSON response.

In order to determine if the payments per time unit are too low, there needs to be a baseline in order to compare the current payments. Using the graphite API two datasets have been retrieved to create plots, a set for one day containing values of every minute (figure 5.3) and a set over 2 weeks containing average values of every hour (figure 5.4). The vertical axis shows the number of successful MasterCard transactions, and the horizontal axis shows the time in UTC(+0) instead of local Amsterdam UTC(+2) time.

[redacted]

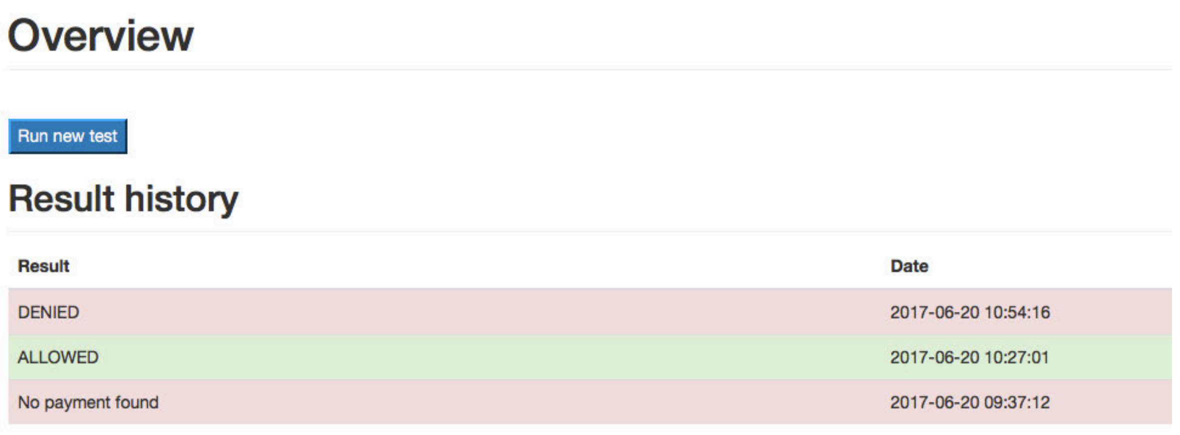


(a) The basic design for the solenoid control, the red wire indicates the connection with the Pi



(b) Simplification of the circuit, the red wire indicates the connection with the Pi

Figure 5.1: circuits of sprint 1



Overview

Result history

Result	Date
DENIED	2017-06-20 10:54:16
ALLOWED	2017-06-20 10:27:01
No payment found	2017-06-20 09:37:12

Figure 5.2: Snapshot of web page after styling

Figure 5.3: Payment history of one full day

Figure 5.4: Hourly average payments over 2 weeks

[redacted]

5.3.4. Refactor bunq API classes

After growing to be a bit more comfortable with the PHP language, the first written code seemed to be of lower quality than the rest. A big reason of this was due to the use of cURL for the execution of API requests. After implementing the Graphite requests (section 5.3.3) using Guzzle [10] which was a lot cleaner, the bunq requests have been refactored to use Guzzle to have a higher readability, more reusable code, higher testability and finally for the sake of uniformity. After the refactor the complexity of the code, especially the execution method was reduced.

5.3.5. Structural Components

After assembly of the structure designed in sprint one, a couple of issues were found. The first issue found was during the assembly. The structure was extremely hard to assemble because the structure would only be firmly into place when all parts were connected. Before that the structure was very loose and the person who would assemble the structure had a very hard time. Even when working with multiple persons this still was an issue. It was also already found that the acrylate was a fragile material and forcing the material would easily break at their weak areas. The third problem was that a lot of the spacers were designed to be shifted in from the underside. Because all pieces are not a tight fit, some parts would easily fall out. This was especially difficult because the PIN-terminal had to be placed from the front. To do this the front spacer had to be removed. When not done carefully, other parts would fall out. The final problem was no longer an assembling problem. When the solenoids would press the buttons on the PIN-terminal the bottom of the structure would bend down. This meant that although the right button was "hit", the button was not pressed.

The mentioned problems in the design have been revised according to drawing E.3. The sides were redesigned to hold the spacers on top instead of on the bottom. And extra parts were added to hold the the structures of the solenoids into position even before they were placed between the side structures. The final change was done to increase the thickness of the pieces that supported the PIN-terminal from $6mm$ to $19mm$. Since the front side of the structure had not been designed in sprint one the front servo assembly was added as well.

5.3.6. Circuit testing

To test the design of sprint one an Arduino was used as a way of controlling and powering the circuit. Since its IO ports can give 5V no external power source was needed. The circuit for the solenoids as designed in the first sprint had a couple of hiccups.

Transistor burn-out

During testing the circuit for the solenoids worked perfectly, however within 30 cycles of 1 second on and 1 second off (duty cycle 50%) the transistor burned out. The reason for the burn-out of the transistor was easily found. Looking at its spec sheet (appendix C.2) the transistor has a maximum collector current of $200mA$. Measurements during testing pointed out that the solenoid spikes to around $1750mA$. The solution for this would be to reduce the solenoids peak current or to have transistors with higher maximum collector currents. Since the solenoids were already quite small it was unlikely the current could be brought down. As solution new, more heavy duty MOSFET transistors were ordered.

MOSFET transistors (appendix C.3) have the ability to switch really fast, which would be useful if one would want to apply PWM to the solenoids.

Pi problems

During the installation of the Pi and its GPIO software the Pi often lost its connection to the network and had some other unexpected glitches. To test how the Pi would handle itself over a longer period of time —since it should be on 24/7— the Pi was left on over the weekend. Unexpectedly, at a Monday morning the Pi was no longer working. A reboot did not help the problem, which made sense, since no indicator LED was blinking anymore. It has been concluded that the Pi may had burned out due to some static electricity or that some cables or another conducting object caused a short circuit.The reason for the problems of the Pi were a lot harder to pin-point because it had 'just' stopped working over the weekend. The office in which the electronics are build has a large tendency to charge people statically, electric static discharges(ESDs) are quite common. Another possibility may be the wires that were connected to 5V and ground. Overly enthousiastic cleaners might have helped us by removing the dust, but may also have burned-out the Pi. To prevent this possibility in the future, diodes from the Pi ports to the circuit will be added in the next design.

A simple version of the revisioned circuit is shown in figure 5.5

5.3.7. Floating gate issue

The circuit designed in sprint 2.0 did not work, for unknown reasons. Testing the MOSFET transistor with LEDs did show an ability to switch, however switching off happened slowly and when tested with multiple switches the LEDs had different levels of brightness. When tested with the solenoids they would either immediately switch on and switch off after a while or would not switch on at all. After meeting with dr. Verhoeven[24] showed that the problem here would be a so called "floating gate". This means that the electric flow can not dissipate when the signal switches from high to low. Another issue is that when the software of the Pi initiates the pin as a GPIO output it already sends a low value signal, thereby opening the gate. Furthermore, the diode does have actual resistance and which lowers the current of the GPIO port. The floating gate problem could easily be solved by placing a high value resistor in front of the gate. This would allow the current to flow back to ground. According to this information a new circuit has been designed (figure 5.6)

5.3.8. Pi control

With the solenoid software working, the control software for the servo could be designed. Even although the servo has not yet been included within the circuit, the software for its control could already be designed. The control of a servo is a bit more complicated than that of the solenoid. The servo uses an analog signal to turn to a given the position. If it is not yet in that position it will move to it. When it is already in that position it will stay there. Since the Pi is not able of sending analog signals it will imitate an analog signal via a special method. Using a certain amount of high and low digital signals the Pi can simulate an analog signal. This way of controlling is called pulse width modulation (PWM). In figure 5.7 an example of a PWM signal is shown. Our servo runs at $50Hz$ and has a center position of $1520\mu s$ this means that at its center position it receives a high signal $1520\mu s$ long every $20ms$. The servo has a range of $180\circ$ with a difference of $400\mu s$. The Pi GPIO library translates this domain to run from 2.5 to 12.5. The Pi has a certain order to run a servo. First it needs to make an object on which it set a pin to use a PWM signal.

```
    p = GPIO.PWM(self.piPin, 50)
```

Followed by setting this object's pin as an output signal.

```
    GPIO.setup(self.piPin, GPIO.OUT)
```

From there it needs to know at what signal length it needs to start. In other words, the start position of the servo. Let's say it starts at its mininum value.

```
    p.start(2.5)
```

Since it needs some time to reach this position the execution of the code should wait for let's say 1 sec.

```
    time.sleep(1)
```

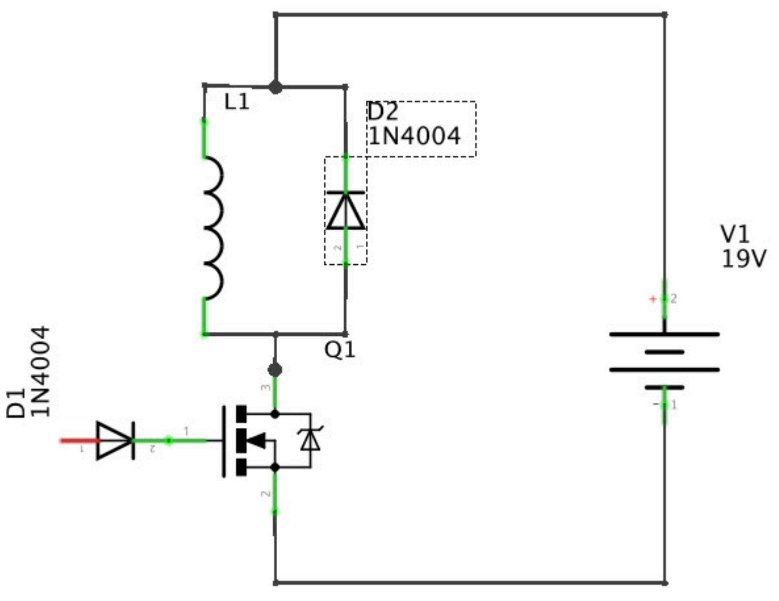


Figure 5.5: Second revision of circuit

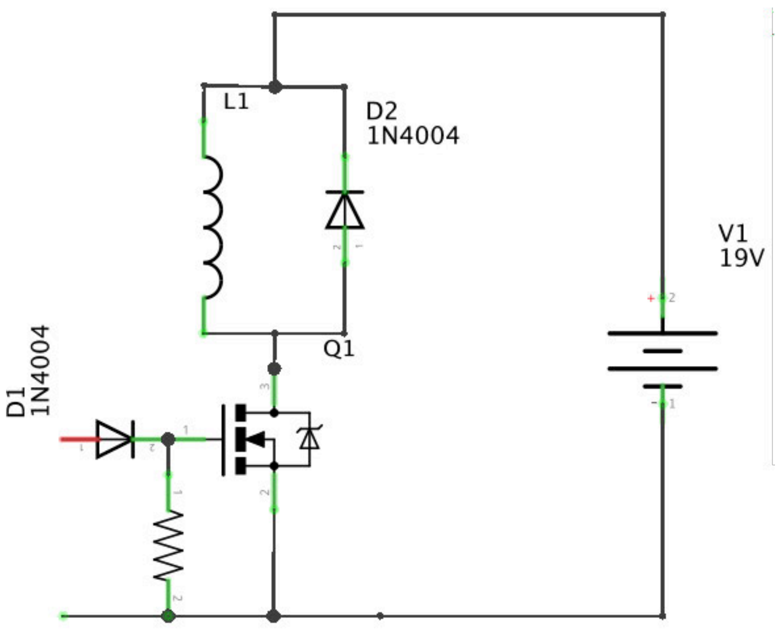


Figure 5.6: Third revision of circuit

From there it can be set to a different position, and the code will have to wait again.

```
p.ChangeDutyCycle(12.5)
time.sleep(1)
```

Finally the connection needs to be closed and the object be terminated. When doing this the servo will remain in its last set position.

```
p.stop()
```

5.4. Sprint 3: extra functionality

The third and final sprint was again focused on extra functionality, but as the deadline was closing in there was also some attention for preparing to deliver the system and real world integration testing. During the previous sprint the hardware had made some advances, leading to this sprint pulling everything together to create a working device.

5.4.1. Global settings

PHP does not keep state, therefore every request is unique. However as there is only one payment terminal testing setup, only one test can be running at a time. In order to accomplish not being able to start a new test while one is already running, one extra table is created in the database, containing global settings. This table contains a single row that can only be updated. Besides only keeping track of when a test is running, there are multiple API calls to setup the connection and get the right information (i.e. private/public keys, session token) that currently get repeated for every test. By saving these values it is sufficient to only run the setup calls once initially and renew the values when they have expired. This will decrease the amount of API calls on average from 5 to 1, increasing the speed, load, and reducing the cost associated with API calls.

While there already is a configuration file, the database is used to store these global settings. The key difference between the configuration and global settings is that the configuration values are required to be set before starting, while everything in the the global settings can be set dynamically by the program.

5.4.2. SIG code feedback processed

During this sprint our code has been submitted to the Software Improvement Group (SIG) for feedback. In the feedback the code received an average rating of 3 out of 5 stars, due to lower scores for Unit Interfacing and Module Coupling. The reason for the lower Unit Interfacing score is because of too many parameters on the bunq API request classes, suggesting to introduce a datatype to store related parameters together. A former refactor (see section 5.3.4) already reduced these parameters on average from 5 to 3. The reduced amount of parameters and the fact that all parameters belong together, instead of part of the parameters, made the team feel the current implementation would be sufficiently maintainable.

The low Module Coupling was also related to the bunq API Request class. The class was invoked on 36 different occasions, it was likely the class carried too much logic. Removing the execute method and moving this to a new class RequestExcutor abstracted this logic from the original class, according to the 'low coupling, high cohesion' principle.

5.4.3. API

To interact with the system besides through the web interface, a simple RESTful API will be implemented. The provisional design of the API only has two endpoints. One to retrieve the results and one to start a new test.

```
GET  /results
POST /test
```

This allows the monitoring environment that is currently in place to retrieve previous results in JSON form. Additionally it is possible to start a test without having to press the button, giving the option to automatically run a test each development roll-out.

There are several popular frameworks available, including: Slim[17], Silex[16] and Lumen[12]. While the documentation looks very promising, Laravel has its own syntax which will take some time to learn and the framework is quite difficult to integrate into an already existing project. Slim is a microframework, where some might say it is too minimal, but has very thorough documentation and is very lightweight. Silex based on the Symfony2[18] framework, which makes it extensible and easy to test. The downside is that the documentation is less extensive, compared to Slim. As the design only plans for a few endpoints and the lightweight Slim framework is more favorable.

5.4.4. Authentication
To restrict access from all employees to the ones who will need it, authentication will be required to interact with both the web interface and the API.

Basic authentication transits the credentials as $"username : password"$ in base64 encoding in a standard field of the HTTP header. This can be retrieved on the server to validate the submitted username and password[23]. The downside of this approach is leaving it vulnerable to attacks such as packet sniffing. For this reason many dismiss basic authentication, however this can be easily disputed by combining basic authentication by securing the connection with a secured tunnel, such as HTTPS. Given proper ciphers and server certificate are used HTTPS will ensure protection against man-in-the-middle attacks, solving the weakness of basic access authentication.

Oauth2.0[21] is an often used authorization protocol, with 4 separate models. The protocol relies on a separate authorization server and resource server. A third-party application (e.g. Facebook) as authorization servers to grant access to a service. The user credentials are only presented to the authorization server which will be exchanged for an access token, that will only be send over SSL. The resource server will use the token to authenticate and properly authorize the user, without learning the user credentials as these have effectively been replaced with a token.

The main difference between the two is that in basic authentication the resource server and authentication server are combined into one, asking the credentials directly from the user.

This system will be run internally, accessible over the clients VPN. This decreases the need for strong security as only employees will have access, which led us to use basic authentication without the use of a secure tunnel. However this could be implemented if there should be a wish to increase the security later on. The submitted username and password combination will be checked against a username and hashed version of the password stored in the database table.

5.4.5. Production account
Until this point testing has been done using the API sandbox and mocked responses. Since MasterCardActions are only generated for transactions where MasterCard is involved, and no pin cards are linked to the sandbox a real world setting has not been tested yet. To make this possible a live production account and pin card have to be used, which will involve costs. The cost of this system will be funded by one of the internal accounts of the client. However this will create fake revenue, to prevent this from being measured by analytics in finance the following has been agreed upon. The finance department will provide the card with an amount of money, that can be used for transactions, the system owner will be responsible for this amount. If the system owner leaves bunq B.V. the used money needs to be refunded, to refund the start amount in full to finance. The system owners successor will take over and repeat this process.

Once the financial aspect was in order, the system was tested with a production account. The first test failed as the bunq's public API documentation seemed to be incomplete, returning a different response than stated. After adjusting the code for the different response, everything worked as expected. Multiple tests have been run, returning both true positives and true negatives.

5.4.6. Structural Components
Upon arrival of the second iteration of the laser cut structure a major flaw was discovered. As laser cutting is very precise, the structure had been made to fit very precise. While this was no problem in the structure designed in sprint one, the new laser cut structure's material thickness deviated by 10% on one side compared the other. Making one side around $3,5mm$ while the other side was the required $3mm$. Since the cut areas were exactly $3mm$ because of the laser cut the structure did not fit as expected. When still trying to assemble the structure it broke several parts. Filing out large parts of the structure was required which took a lot of valuable time. Another problem that arose was that two redundant parts were forgotten to be added on the drawing. Although these parts were redundant they were required because of the damage from the forced construction.

A problem that arose during testing was the movement of the solenoids. Some parts of the solenoids got stuck in the assembly. The solution to this was drilling the holes larger by hand.

Finally the redundant parts and the servo holder were replaced by sawing out a redesigned piece (appendix E.4). Due to time constraints a quick fix needed to be found for the elements that need to press the screen, hold the card and push the card in and out. These were cut out of very strong cardboard and reinforced with steel bolts. And finally screwed onto their servo. Due to their difference in style compared to the rest of the robot the marketing colours of our client will be used to make the cardboard more aesthetically pleasing and giving it a more personal flavour for the company. The final design of the laser cut structure can be found in the final drawing of appendix E.5.

5.4.7. Removing MOSFETs

With the diode added to the circuit it unfortunately still did not work as expected. Measuring the circuit showed that all voltage levels were correct throughout the circuit. Yet the MOSFET did not switch as expected, when given an high signal the measurements showed a closed circuit. This was noticed due to the voltage dropping from $43V$ (open circuit voltage) to a closed circuit voltage of $19V$ as given on the specs on the side of the voltage source. However the voltage measured over the solenoid was $0V$.

The MOSFET kept giving trouble in the circuit. In the first sprint it turned out that the NPN-transistors were effective but just not sturdy enough. The MOSFETs were removed from the design and the design (figure 5.1b) from the first sprint was used again but this time with new transistors. The focus of these transistors was that they were the type of transistor used in sprint 1, but that they would have higher specifications. The chosen BD139 transistors (appendix C.4) are rated for $80V, 1500mA$ and allow short peaks up to to $3000mA$. The reason the MOSFET transistors were chosen first was because the MOSFETs are rated for $16000mA$ and were considered safer for long time use. Another option would have been using a driver circuit for the MOSFETs[22] as seen in figure 5.8b. Due to the extra complexity and time constraints this solution will not be implemented. The option chosen is using the circuit from sprint 1 found in figure 5.1b.

5.4.8. Voltage regulators

With the solenoid circuit operational the servo circuit could be created. Some of the group members had experience with servo's and connecting them required just a $5V$ supply and the control signal could be directly connected to the Pi. The way to connect the servo into the circuit can be found in figure5.8a. The only issue with connecting the servo into the circuit is that the servo runs on $5V$ while the rest of the circuit runs on $19V$. The voltage can be brought down to $5V$ by using a voltage controller. An issue with voltage controllers is that they reduce the excess power into heat. for example, if a $15V$ circuit uses a voltage regulator to run something at $5V - 2A$ the excess $10V$ (at $2A$) will be released as heat. In this case that would mean that $5V * 2A = 10W$ would need to be dissipated. For a small voltage controller without a very large heat-sink this would be impossible. Fortunately the servo's run at a much smaller current. According to their spec sheet (found in appendix C.5) the servo's run in the order of magnitude of about $10mA$. The amount of heat that would need to be dissipated is $14V * 0.01A = 0.14W$. Not only is $0.14W$ a small amount, the maximum amount of time that a servo will be actually running is in the order of 10 seconds. This way the circuit is fully functioning. The final circuit design can be found in appendix D.1.

5.4.9. Pi control

With the electronics finished all that remains is controlling them to create tests. The pins of the Pi are 'hard-coded' to match with the object that they are connected to. The Pi control will turn this list into a list containing servo and solenoid objects. Each of them have their name and their designated pin as attribute so that their name can be called to find their pin. To know which pin needs to give what kind of signal we need to follow the state diagram 4.1 from the design chapter. The Pi control needs to make a sequence in which all actions will be performed. These actions depend on the given input of payment type, PIN number and amount to be debited. Furthermore the sequence needs to be able to get the PIN-terminal into the start position. For the tests to pass it is essential that the PIN-terminal is in the begin state. Although it seems not very scientific, the best and safest way to do this is to press the OK and Cancel buttons twice. Therefore the first part of the sequence will be set to do just that.

```
self.seq.append(”cancel”)
self.seq.append(”ok”)
self.seq.append(”cancel”)
self.seq.append(”ok”)
```

Then the robot has to press the screen to start the payment.

```
self.seq.append(”servscreen”)
```

The PIN-terminal will now request the amount to be debited. This is more complex, because the sequence list requires a string value, while the argument is a double. To solve this the double will be multiplied by 100 and parsed to an integer follwed by a parse to a string. From this string the sequence will take the first to last character and add these to the sequence.

```
xs = []
strNum = str(int(self.payAmount * 100))
for i in range(0, len(strNum), 1):
    xs.append(strNum[i])
return xs
```

If the payment type is a chip payment, the card will need to be inserted, the PIN entered, OK pressed and the card extracted. The way to get the PIN follows the same way as the one from the amount to be debited, just without the multiplication by 100 and the parse to an integer. Leading to the following code:

```
self.seq.append(”servchipon”)
self.seq.extend(self.pinSequence())
self.seq.append(”ok”)
self.seq.append(”servchipoff”)
```

In case of an nfc payment, the command is a lot easier. Because there is no need for a PIN number all the robot has to do is move the card towards the screen, wait a bit and pull it back. For the sequence this just means:

```
self.seq.append(”servnfc”)
```

The PIN-terminal will now confirm the successful payment, or report a payment failure. To continue from this state OK will need to be pressed.

```
self.seq.append(”ok”)
```

The PIN-terminal used has a receipt printer. This printer is not used because of the mess that it would create by printing a receipt with every test. Not to mention that it would need to be replaced multiple times. Therefore the receipt paper has been removed. The terminal will give an error on this however, this error can be dismissed by pressing cancel. Followed by a confirmation that the terminal can be given back to the owner. This screen is passed by pressing the ”OK” button. This would give the final addition to the sequence will be:

```
self.seq.append(”cancel”)
self.seq.append(”ok”)
```

The controller will use this sequence to send a ’pulse’ to the object with its matching icon on the keypad. So ”OK” is a solenoid, which means it will make a push movement on the ”OK” button. The servo’s will perform their own action when given a pulse. By creating a sequence for the names linked to the GPIO pin numbers a test sequence can be run.

It turns out that the solenoids become very hot when running at $19V$. This means that the solenoids will have to run on at low duty cycle. The solenoids need about $20s$ to cool when they have been on for $1s$. So they have a duty cycle of $\frac{1}{1+20} \approx 5\%$. Experimentation showed that the solenoid only needed to be on for just $0,1s$ to press the button. Then the solenoid will only have to cool down for $2s$. As the heating of the solenoid is a large issue in this project also another way is applied to reduce heat. By using PWM, like with the servos, the energy requirement can be brought down even further. The solenoids seem to work best at a frequency of $1000Hz$ with a pulse length of $0,05ms$. This way the solenoids seem to be on for $0,1s$ while they are actually just on for $0,05s$. Combined with the previously mentioned cooldown, the solenoids now have a duty cycle of $\frac{0,05}{0,05+2} \approx 2,5\%$.

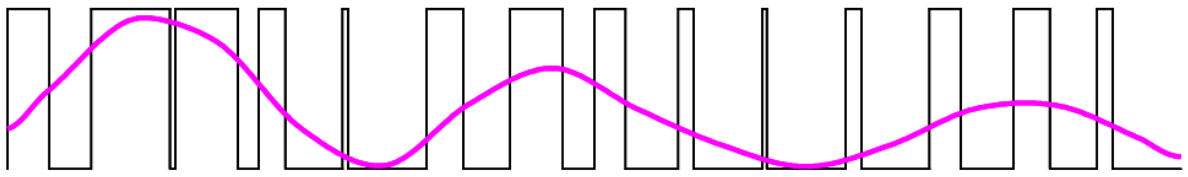
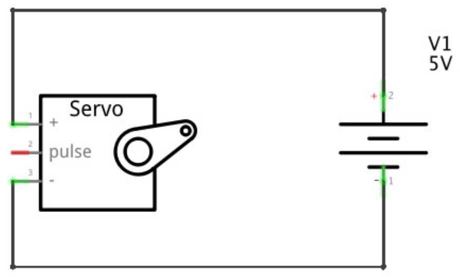
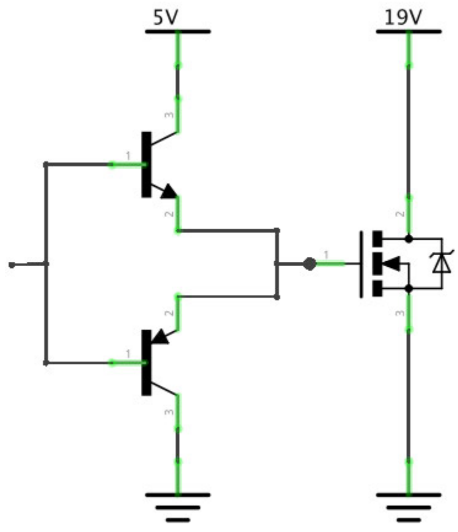


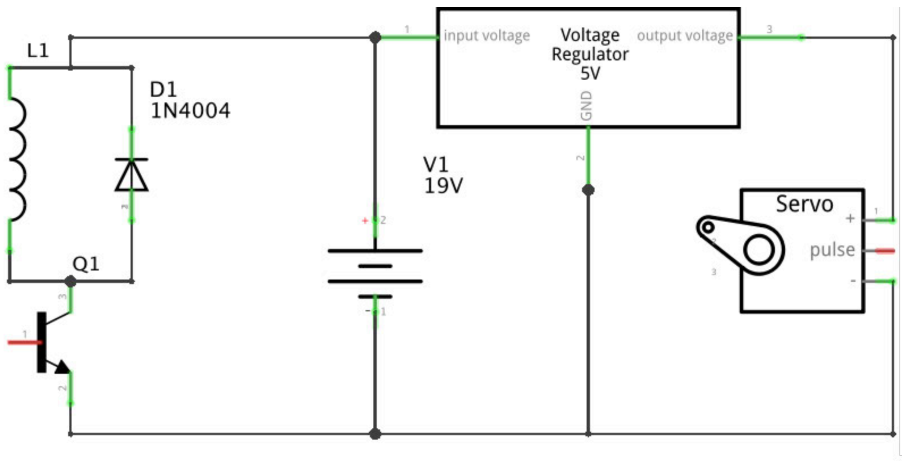
Figure 5.7: example of an PWM signal



(a) Servo connect in the circuit, the red wire leads towards the Pi



(b) Driver circuit for the MOSFET, the left wire leads towards the Pi



(c) Final circuit design, the red wires lead towards the Pi

Figure 5.8: circuits of sprint 3

6 Conclusions

This chapter will conclude the project giving an evaluation of the final product and measure it based on the requirements set at the start of the project (see section 2.2.1) and mention the achievements during the projects, but also the mistakes and how these could have been prevented.

6.1. Evaluation

The final product satisfies many of the requirements. The robotic solution is capable of running and completing test sequence using both contact and wireless EMV. This test can be started and verified using a web interface or by using, an unplanned additional feature, a RESTful API. Sadly not all of the requirements have been fulfilled. The system can start a payment cycle, however the pin robot is not yet able to push the 4,5,6,*,0 and # buttons. This means that tests will need to be run without PIN numbers containing these numbers. As main additional feature the system is also able to autonomously asses if the Maestro workflow is still operational, and run a test if it is not.

Due to unforeseen time delays within the hardware part of the project (floating gates, deviating material thickness, etc), hardware related requirements like: A counter, optical feedback, and individual control of the keypad have unfortunately not been met. However these were classified as could haves, to improve the overall user experience not having these features is not a critical issue.

Even though not all of the must have requirements are fully implemented, the client is still satisfied as the system is capable of performing PIN transactions without being able to press every single button. As the unsuccessful requirements concern the structure of the robotics, further work on the structure would help meet all the requirements.

To ensure that all developed features work as is expected, there has been made sure there is a full test coverage for all features.

6.2. Conclusion

In conclusion, the project was a success. A solution that can test payments has been delivered to the client. The project did come with some unfortunate setbacks. The first setback was met quite soon into the project, one of the original three team members was conditionally allowed into work on the project. However, his conditions to continue with the project were not met and he was forced to resign from the team. Having just two team members left the workload of the project needed to be rescheduled to two people, reducing the amount of features that would be implemented during the project. Due to the project consisting of hardware and software, the group split up at the start to maximize the team's efficiency and create a minimum viable product. This led to a point where it was hard to fully understand what the other person was working on. This became a apparent when one of the group members fell ill close to the deadline. Better communication would have prevented this problem. For the hardware part there was not a lot of experience yet, but this was thought to be good enough to be able to get a working product. The product was eventually realized, but went over budget and took longer than expected. Looking back it can be concluded that it would have been useful to request assistance in the time. To conclude, the project successfully implemented a product that is capable of testing payments remotely. Although the project planning could have been improved it allowed us to complete a working prototype in time.

7 Discussion

Finally this chapter will give some recommendations for future work if one might want to improve the current system. It will also discuss ethical issues that have been encountered during the project.

7.1. Recommendations

To improve the system in the future, this section will discuss some recommendations.

First of all the software, as mentioned in section5.3.3 the current implementation is likely not very future proof. As bunq B.V. has plans to expand internationally, this will almost certainly change patterns that have been noticed in the current datasets of accepted Maestro payments. Due to the limitations of the stored data in Graphite and not enough time to gather and save this data during the time of this project, adjustments have been made and created a time sensitive model. A recommendation would be to create a time invariant model that would be more likely to persist as the patterns change.

Future work on the hardware side would be an improvement of the frame and a more elegant way to operate the solenoids. It would also recommendable to let some of the extra implementations be added. As the basic hardware (i.e. basic electronics, structure) has been completed the extra features can be added. Other work could be improvements of the structure to handle more features. The raspberry Pi provides libraries to easily support camera functions. Another recommendation would be to implement PWM on the solenoids to decrease the amount of energy required and the amount of heat produced. At this area, a more solid frame could be connected to the solenoids to immediately provide an heatsink to them. As the electronics are at the moment placed on a breadboard it would be more elegant to solder these to a PCB making them sturdier and giving the whole board a cleaner look.

7.2. Ethical Issues

Our project's ethical issue relates to the safety of the tests. It is important that the tests are performed thoroughly. If the tests are not performed thoroughly, or shortcuts are taken, it is possible that flaws in a software update are not discovered. Since there is a lot of private bank account data it is essential that these remain a secret. Our client's business plan is providing a service that other banks do not. Our client promises to keep all data secret, and never sell it to any third-party.[7]. To work within this promise it has been important to use as much anonymized data as possible during this project.

In case our system malfunctions by returning a false positive it is possible that a flawed update might be rolled out. This could cause inconvenience for the clients customers, resulting in bad PR and even losing costumers. This meant testing had to have a high overall priority, as such mistakes can not be permitted.

Bibliography

[1] Emvco. URL https://www.emvco.com/about_emvco.aspx?id=202. [Online; accessed 2-May-2017].

[2] Pi download page. URL https://www.raspberrypi.org/downloads/. [Online; accessed 26-June-2017].

[3] Pios. URL https://www.raspbian.org/. [Online; accessed 26-June-2017].

[4] Power supply. URL https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md. [Online; accessed 21-June-2017].

[5] 'banken maken dingen die de klanten niet willen'. URL https://fd.nl/ondernemen/1120680/banken-maken-dingen-die-de-klanten-niet-willen. [Online; accessed 15-June-2017].

[6] bunq api documentation, . URL https://doc.bunq.com. [Online; accessed 12-May-2017].

[7] Questions, questions, questions, . URL https://www.bunq.com/en/news/questions-questions-questions. [Online; accessed 21-June-2017].

[8] Graphite documentation, . URL http://graphite.readthedocs.io/en/latest/. [Online; accessed 5-June-2017].

[9] The render url api, . URL http://graphite.readthedocs.io/en/latest/render_api.html. [Online; accessed 5-June-2017].

[10] Guzzle, php http client. URL http://guzzle.readthedocs.io/en/stable/. [Online; accessed 7-June-2017].

[11] De bankier van de toekomst is een it-nerd. URL http://nos.nl/nieuwsuur/artikel/2075991-de-bankier-van-de-toekomst-is-een-it-nerd.html. [Online; accessed 15-June-2017].

[12] Lumen. URL https://lumen.laravel.com/docs/5.4/configuration. [Online; accessed 10-June-2017].

[13] Tag type technical specifications. URL http://nfc-forum.org/our-work/specifications-and-application-documents/specifications/tag-type-technical-specifications/. [Online; accessed 2-May-2017].

[14] Usage of web servers broken down by ranking, . URL https://w3techs.com/technologies/cross/web_server/ranking. [Online; accessed 4-May-2017].

[15] October 2015 web server survey, . URL https://news.netcraft.com/archives/2015/10/16/october-2015-web-server-survey.html. [Online, accessed 4-May-2017].

[16] Documentation - silex. URL https://silex.sensiolabs.org/doc/2.0/. [Online; accessed 10-June-2017].

[17] Slim framework. URL https://www.slimframework.com. [Online; accessed 10-June-2017].

[18] Symfony. URL https://symfony.com. [Online; accessed 10-June-2017].

[19] Pi download page. URL https://www.raspberrypi.org/downloads/. [Online; accessed 26-June-2017].

[20] *User Manual VX680*. CCV. CID088A/03012013.

[21] D. Hardt. The oauth 2.0 authorization framework. RFC 6749, RFC Editor, October 2012. URL `http://www.rfc-editor.org/rfc/rfc6749.txt`. `http://www.rfc-editor.org/rfc/rfc6749.txt`.

[22] RangaRao Venkatesha Prasad. personal communication.

[23] J. Reschke. The ’basic’ http authentication scheme. RFC 7617, RFC Editor, September 2015.

[24] Chris Verhoeven. personal communication.

A
Original Project Description

A.1. Project Description

Although we are a mobile bank, debit cards and subsequently card payments are still very important for us and our users. Every time something changes in our backend, we need to test if our debit card service is still up and running. We currently do this by making an actual card transaction using a debit card and card terminal. This means we're dependent on our terminal and its location. Wouldn't it be great if we could test card payments and make card transactions from any place we want?!

We would like to have a permanent setup with a card terminal in our office that can be accessed online. This would allow us to test card payments from any distance using an internet connection. How this setup would look like is up to you! For example, it could include a 'robot arm' that pushes the buttons on the card terminal and a camera that records the results on the terminal's screen.

A.2. Company Description

bunq is not an ordinary bank. Instead of making more money, we want to reinvent money itself with mobile technology. That's why we built our own banking system from scratch, including an app that fits your entire bank in your pocket. And that's just the beginning! We're working non-stop on futuristic payment methods and other innovations to make money and banking as easy, transparent, and fun as possible.

B Info Sheet

B.1. General Information

Title of the project: Card Payment Testing
Name of the client organization: bunq B.V.
Date of the final presentation: July 3d, 2017

B.2. Description

Every development roll-out, there needs to be certainty everything works. Especially when it comes to the banking domain. People care about their money and want it to be handled with care. Currently this means some tests, including making a payment using a Maestro debit card, need to be performed manually. In order to further testing, this project aims to automate these Maestro payments. This has been achieved by constructing a robotic device to perform the physical actions associated with creating a payment request, and create a piece of software to verify if the payment was accepted. The main software is based around the public API bunq B.V. offers. Using the API it is possible to retrieve information related to your bank accounts, including payments made. The robotic device is consists of a combination of solenoids and servos to operate the keypad and to present a bunq Maestro debit card. Additionally the system has extra functionality such as self assessment if the Maestro workflow is still functioning. As a result there should no longer be a need to physically take a terminal home and complete payments. Even though the delivered system does not have functionality to work with a phone's NFC, the system should be easily upgraded to support this functionality.

B.3. Members of the project team

Name: Arend Jan de Graaff
Interests: Gaming, Traveling, Watching Shows, Low-level programming
Contribution and role: Hardware, embedded software

Name: Wouter Zirkzee
Interests: Computer Science, Music, Finance, Sports
Contribution and role: front-end & back-end developer and tester

B.4. Client

Name: Wessel Van
Affiliation: bunq B.V.

B.5. Coach

Name: RangaRao Venkatesha Prasad
Affiliation: Embedded Software

B.6. Contact

Otto Visser
Huijuian Wang
bep-ewi@tudelft.nl

The final report for this project can be found at: `http://repository.tudelft.nl/`.

C
Tech sheets

Figure C.1: Solenoid technical data sheet

十 年 沉 淀　专 业 领 先
Professional Solenoid Technology

众恒电器
ZONHEN ELECTRIC APPLIANCES
电磁铁 电磁阀 电磁泵
SOLENOID VALVE PUMP

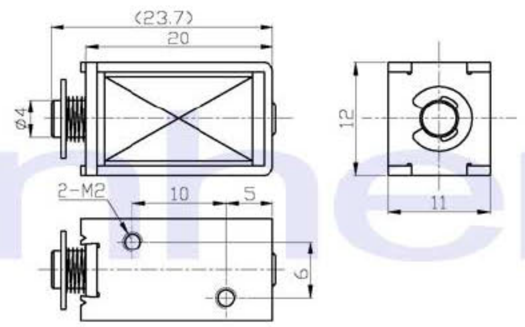
框架式ZHO

ZHO-0420L/S Open Frame Solenoid

ISO/TS16949 ISO9000 ISO14000
RoHS REACH CE UL VDE TÜV

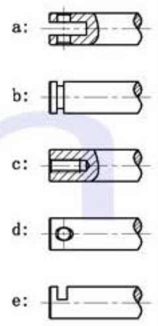
总重量 Total Weight: 13g
应用范围: 保险柜锁, 医疗仪器, 玩具等
APPLICATION: Safe Lock, Medical Device, Toy etc

类型: 拉式 TYPE: PULL

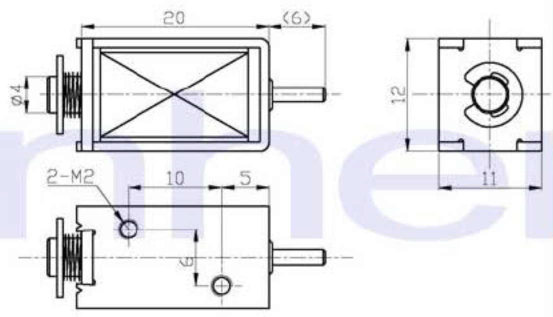


单位:mm Units:mm　通电状态 Shown Energized

滑杆外部与负载连接的典型结构
plunger type:

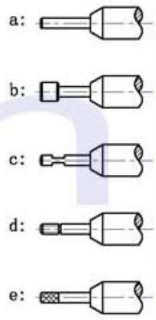


类型: 推式 TYPE: PUSH

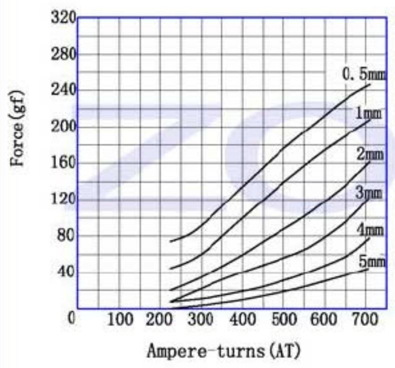


单位:mm Units:mm　通电状态 Shown Energized

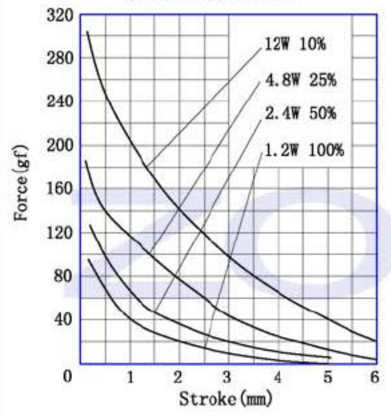
顶杆外部与负载连接的典型结构
push shaft type:



安匝/吸引力
AMPERE-TURNS Vs FORCE



行程/吸引力
STROKE Vs FORCE



COIL DATA线圈资料

工作周期(%) = 工作时间 / (工作时间+断开时间) X100% Duty cycle(%) = "ON"time / ("ON"time+"OFF"time) X100%	连续 Continuous 100%	间断 Or less 50%	间断 Or less 25%	间断 Or less 10%
最大工作时间(秒) MAX."on"time in seconds	∞	50	18	3
瓦特(W)(20℃) Watts at 20℃	1.2	2.4	4.8	12
安匝数AT(20℃) Ampere-turns at 20℃	243	338	477	737

型号 Type no.	阻值(20℃) Resistance (Ω)±10%	线圈匝数 No. Turns	电压(VDC) VoltS DC			
ZHO-0420L-XX A XX	7.5	609	3	4.2	6	19
L 拉Pull / S 推Push	30	1193	6	8.5	12	19
电压Voltage(V)	120	2385	12	17	24	38
系列号Serial No.: A, B, C... / 电阻Resistance(Ω)	480	4656	24	34	48	76

http://www.zonhen.com　20　http://www.solenoid.cn

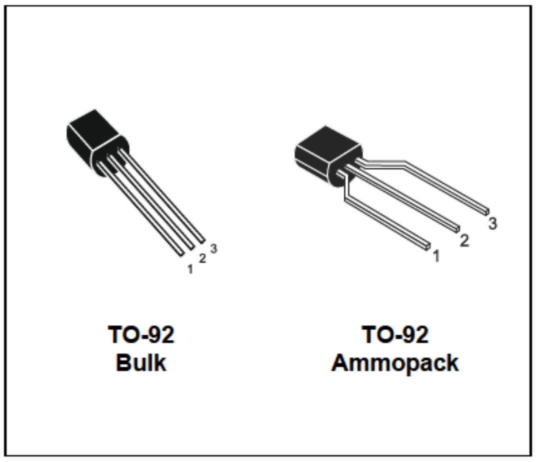
Figure C.2: 2N3904 NPN-Transistor technical data sheet

2N3904

SMALL SIGNAL NPN TRANSISTOR

PRELIMINARY DATA

Ordering Code	Marking	Package / Shipment
2N3904	2N3904	TO-92 / Bulk
2N3904-AP	2N3904	TO-92 / Ammopack

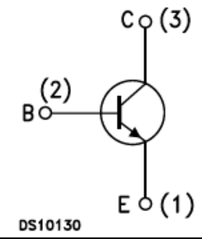


- SILICON EPITAXIAL PLANAR NPN TRANSISTOR
- TO-92 PACKAGE SUITABLE FOR THROUGH-HOLE PCB ASSEMBLY
- THE PNP COMPLEMENTARY TYPE IS 2N3906

APPLICATIONS

- WELL SUITABLE FOR TV AND HOME APPLIANCE EQUIPMENT
- SMALL LOAD SWITCH TRANSISTOR WITH HIGH GAIN AND LOW SATURATION VOLTAGE

INTERNAL SCHEMATIC DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CBO}	Collector-Base Voltage ($I_E = 0$)	60	V
V_{CEO}	Collector-Emitter Voltage ($I_B = 0$)	40	V
V_{EBO}	Emitter-Base Voltage ($I_C = 0$)	6	V
I_C	Collector Current	200	mA
P_{tot}	Total Dissipation at $T_C = 25$ °C	625	mW
T_{stg}	Storage Temperature	-65 to 150	°C
T_j	Max. Operating Junction Temperature	150	°C

Figure C.2: 2N3904 NPN-Transistor technical data sheet

THERMAL DATA

$R_{thj-amb}$ •	Thermal Resistance Junction-Ambient Max	200	°C/W
$R_{thj-case}$ •	Thermal Resistance Junction-Case Max	83.3	°C/W

ELECTRICAL CHARACTERISTICS (T_{case} = 25 °C unless otherwise specified)

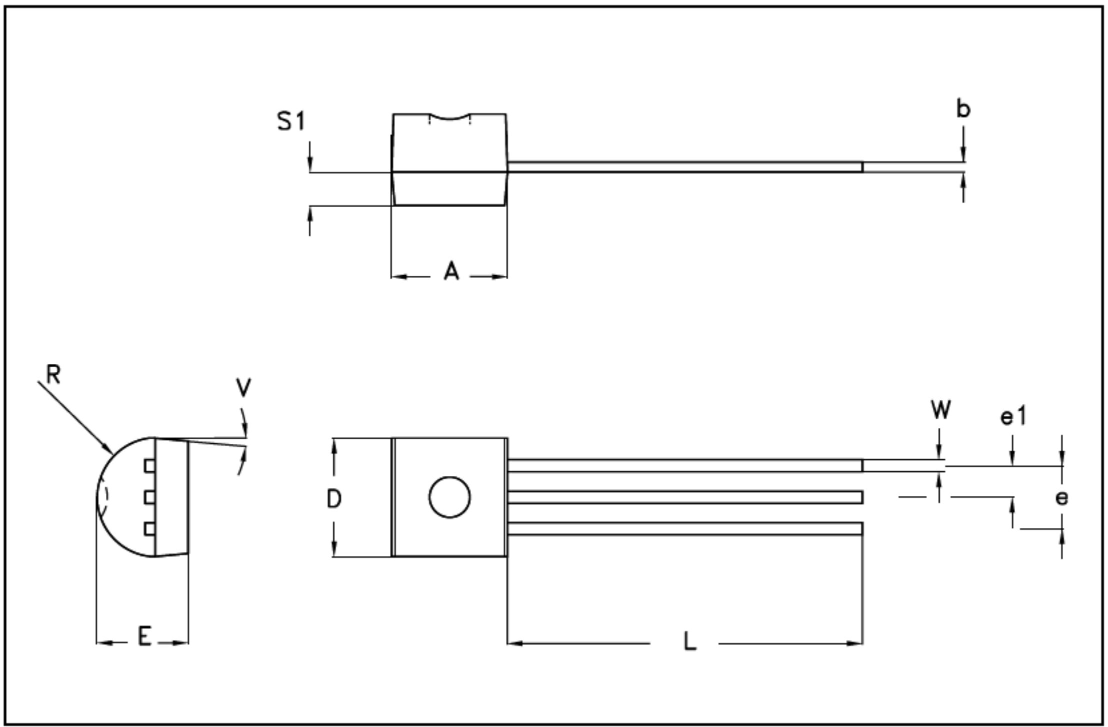
Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
I_{CEX}	Collector Cut-off Current (V_{BE} = -3 V)	V_{CE} = 30 V			50	nA
I_{BEX}	Base Cut-off Current (V_{BE} = -3 V)	V_{CE} = 30 V			50	nA
$V_{(BR)CEO}$*	Collector-Emitter Breakdown Voltage (I_B = 0)	I_C = 1 mA	40			V
$V_{(BR)CBO}$	Collector-Base Breakdown Voltage (I_E = 0)	I_C = 10 μA	60			V
$V_{(BR)EBO}$	Emitter-Base Breakdown Voltage (I_C = 0)	I_E = 10 μA	6			V
$V_{CE(sat)}$*	Collector-Emitter Saturation Voltage	I_C = 10 mA I_B = 1 mA I_C = 50 mA I_B = 5 mA			0.2 0.2	V V
$V_{BE(sat)}$*	Base-Emitter Saturation Voltage	I_C = 10 mA I_B = 1 mA I_C = 50 mA I_B = 5 mA	 0.65		0.85 0.95	V V
h_{FE}*	DC Current Gain	I_C = 0.1 mA V_{CE} = 1 V I_C = 1 mA V_{CE} = 1 V I_C = 10 mA V_{CE} = 1 V I_C = 50 mA V_{CE} = 1 V I_C = 100 mA V_{CE} = 1 V	60 80 100 60 30		 300 	
f_T	Transition Frequency	I_C = 10 mA V_{CE} = 20 V f = 100 MHz	250	270		MHz
C_{CBO}	Collector-Base Capacitance	I_E = 0 V_{CB} = 10 V f = 1 MHz		4		pF
C_{EBO}	Emitter-Base Capacitance	I_C = 0 V_{EB} = 0.5 V f = 1 MHz		18		pF
NF	Noise Figure	V_{CE} = 5 V I_C = 0.1 mA f = 10 Hz to 15.7 KHz R_G = 1 KΩ		5		dB
t_d t_r	Delay Time Rise Time	I_C = 10 mA I_B = 1 mA V_{CC} = 30 V			35 35	ns ns
t_s t_f	Storage Time Fall Time	I_C = 10 mA I_{B1} = $-I_{B2}$ = 1 mA V_{CC} = 30 V			200 50	ns ns

∗ Pulsed: Pulse duration = 300 μs, duty cycle ≤ 2 %

Figure C.2: 2N3904 NPN-Transistor technical data sheet

TO-92 MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A	4.32		4.95	0.170		0.195
b	0.36		0.51	0.014		0.020
D	4.45		4.95	0.175		0.194
E	3.30		3.94	0.130		0.155
e	2.41		2.67	0.095		0.105
e1	1.14		1.40	0.045		0.055
L	12.70		15.49	0.500		0.609
R	2.16		2.41	0.085		0.094
S1	1.14		1.52	0.045		0.059
W	0.41		0.56	0.016		0.022
V	4 degree		6 degree	4 degree		6 degree



TO-92 AMMOPACK SHIPMENT (Suffix"-AP") MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A1			4.80			0.189
T			3.80			0.150
T1			1.60			0.063
T2			2.30			0.091
d			0.48			0.019
P0	12.50	12.70	12.90	0.492	0.500	0.508
P2	5.65	6.35	7.05	0.222	0.250	0.278
F1,F2	2.44	2.54	2.94	0.096	0.100	0.116
delta H	-2.00		2.00	-0.079		0.079
W	17.50	18.00	19.00	0.689	0.709	0.748
W0	5.70	6.00	6.30	0.224	0.236	0.248
W1	8.50	9.00	9.25	0.335	0.354	0.364
W2			0.50			0.020
H	18.50		20.50	0.728		0.807
H0	15.50	16.00	16.50	0.610	0.630	0.650
H1			25.00			0.984
D0	3.80	4.00	4.20	0.150	0.157	0.165
t			0.90			0.035
L			11.00			0.433
I1	3.00			0.118		
delta P	-1.00		1.00	-0.039		0.039

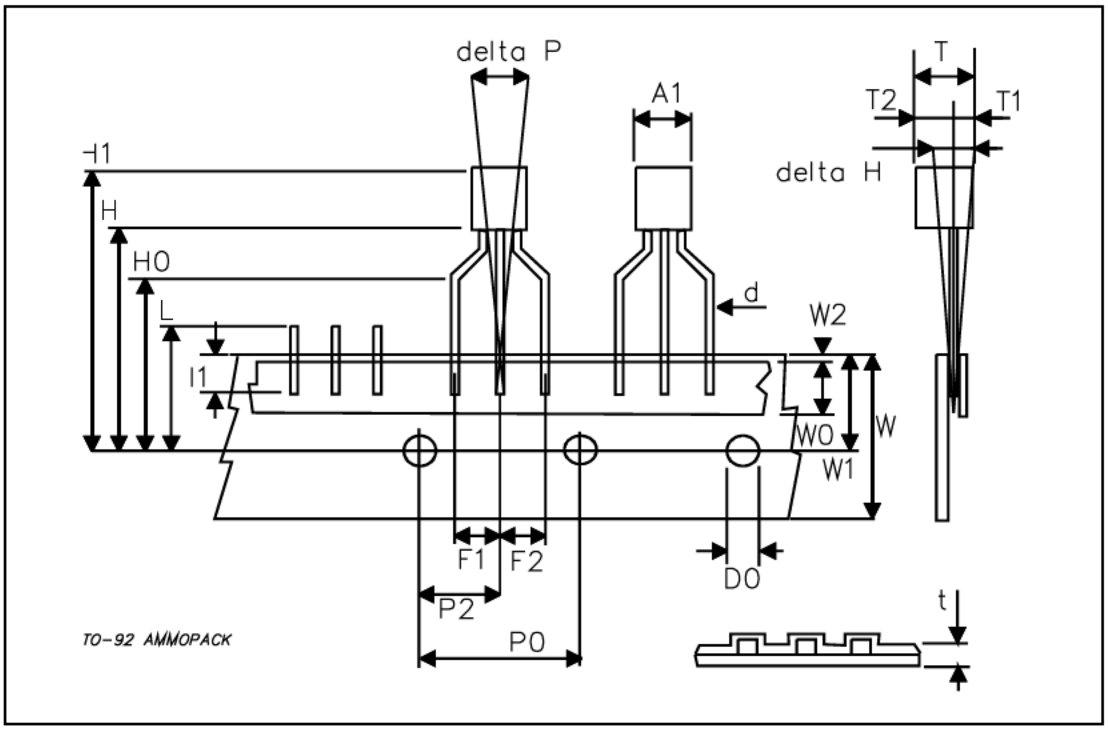


Figure C.2: 2N3904 NPN-Transistor technical data sheet

Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specification mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

The ST logo is a trademark of STMicroelectronics

© 2003 STMicroelectronics – Printed in Italy – All Rights Reserved

STMicroelectronics GROUP OF COMPANIES

Australia - Brazil - Canada - China - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States.

http://www.st.com

Figure C.3: IRF530 MOSFET-Transistor technical data sheet

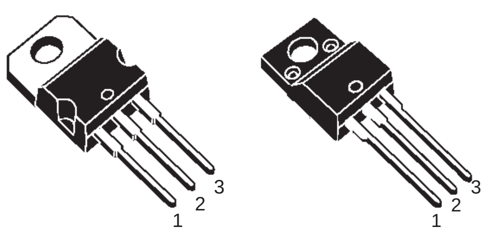
SGS-THOMSON MICROELECTRONICS

IRF530
IRF530FI

N - CHANNEL ENHANCEMENT MODE POWER MOS TRANSISTOR

TYPE	V_{DSS}	$R_{DS(on)}$	I_D
IRF530	100 V	< 0.16 Ω	16 A
IRF530FI	100 V	< 0.16 Ω	11 A

- TYPICAL $R_{DS(on)}$ = 0.12 Ω
- AVALANCHE RUGGED TECHNOLOGY
- 100% AVALANCHE TESTED
- REPETITIVE AVALANCHE DATA AT 100°C
- LOW GATE CHARGE
- HIGH CURRENT CAPABILITY
- 175°C OPERATING TEMPERATURE
- APPLICATION ORIENTED CHARACTERIZATION

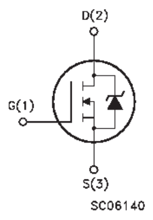


TO-220 TO-220FI

APPLICATIONS

- HIGH CURRENT, HIGH SPEED SWITCHING
- SOLENOID AND RELAY DRIVERS
- DC-DC & DC-AC CONVERTER
- AUTOMOTIVE ENVIRONMENT (INJECTION, ABS, AIR-BAG, LAMP DRIVERS Etc.)

INTERNAL SCHEMATIC DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value		Unit
		IRF530	IRF530FI	
V_{DS}	Drain-source Voltage (V_{GS} = 0)	100		V
V_{DGR}	Drain- gate Voltage (R_{GS} = 20 kΩ)	100		V
V_{GS}	Gate-source Voltage	± 20		V
I_D	Drain Current (continuous) at T_c = 25 °C	16	11	A
I_D	Drain Current (continuous) at T_c = 100 °C	11	7.8	A
I_{DM}(•)	Drain Current (pulsed)	64	64	A
P_{tot}	Total Dissipation at T_c = 25 °C	90	40	W
	Derating Factor	0.6	0.27	W/°C
Viso	Insulation Withstand Voltage (DC)	-	2000	V
T_{stg}	Storage Temperature	-65 to 175		°C
T_j	Max. Operating Junction Temperature	175		°C

(•) Pulse width limited by safe operating area (1) $I_{SD} \leq 11$ A, di/dt ≤ 200 A/µs, $V_{DD} \leq V_{(BR)DSS}$, Tj ≤ T_{JMAX}

Figure C.3: IRF530 MOSFET-Transistor technical data sheet

THERMAL DATA

			TO-220	TO220-FI	
$R_{thj\text{-}case}$	Thermal Resistance Junction-case	Max	1	3.75	$^oC/W$
$R_{thj\text{-}amb}$	Thermal Resistance Junction-ambient	Max	62.5		$^oC/W$
$R_{thc\text{-}sink}$	Thermal Resistance Case-sink	Typ	0.5		$^oC/W$
T_l	Maximum Lead Temperature For Soldering Purpose		300		^oC

AVALANCHE CHARACTERISTICS

Symbol	Parameter	Max Value	Unit
I_{AR}	Avalanche Current, Repetitive or Not-Repetitive (pulse width limited by T_j max, $\delta < 1\%$)	16	A
E_{AS}	Single Pulse Avalanche Energy (starting $T_j = 25\ ^oC$, $I_D = I_{AR}$, $V_{DD} = 50$ V)	100	mJ

ELECTRICAL CHARACTERISTICS ($T_{case} = 25\ ^oC$ unless otherwise specified)

OFF

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
$V_{(BR)DSS}$	Drain-source Breakdown Voltage	$I_D = 250\ \mu A$ $V_{GS} = 0$	900			V
I_{DSS}	Zero Gate Voltage Drain Current ($V_{GS} = 0$)	V_{DS} = Max Rating V_{DS} = Max Rating $T_c = 125\ ^oC$			1 10	μA μA
I_{GSS}	Gate-body Leakage Current ($V_{DS} = 0$)	$V_{GS} = \pm 20$ V			± 100	nA

ON (∗)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
$V_{GS(th)}$	Gate Threshold Voltage	$V_{DS} = V_{GS}$ $I_D = 250\ \mu A$	2	3	4	V
$R_{DS(on)}$	Static Drain-source On Resistance	$V_{GS} = 10V$ $I_D = 8$ A		0.12	0.16	Ω
$I_{D(on)}$	On State Drain Current	$V_{DS} > I_{D(on)} \times R_{DS(on)max}$ $V_{GS} = 10$ V	16			A

DYNAMIC

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
g_{fs} (∗)	Forward Transconductance	$V_{DS} > I_{D(on)} \times R_{DS(on)max}$ $I_D = 8$ A	5	8		S
C_{iss} C_{oss} C_{rss}	Input Capacitance Output Capacitance Reverse Transfer Capacitance	$V_{DS} = 25$ V f = 1 MHz $V_{GS} = 0$		950 150 50	1300 270 70	pF pF pF

Figure C.3: IRF530 MOSFET-Transistor technical data sheet

ELECTRICAL CHARACTERISTICS (continued)

SWITCHING ON

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
$t_{d(on)}$ t_r	Turn-on Time Rise Time	V_{DD} = 50 V I_D = 8 A R_G = 4.7 Ω V_{GS} = 10 V		12 20	16 28	ns ns
Q_g Q_{gs} Q_{gd}	Total Gate Charge Gate-Source Charge Gate-Drain Charge	V_{DD} =80 V I_D =16 A V_{GS} = 10 V		32 9 13	44	nC nC nC

SWITCHING OFF

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
$t_{r(Voff)}$ t_f t_c	Off-voltage Rise Time Fall Time Cross-over Time	V_{DD} = 80 V I_D =16 A R_G = 4.7 Ω V_{GS} = 10 V		11 12 25	15 17 35	ns ns ns

SOURCE DRAIN DIODE

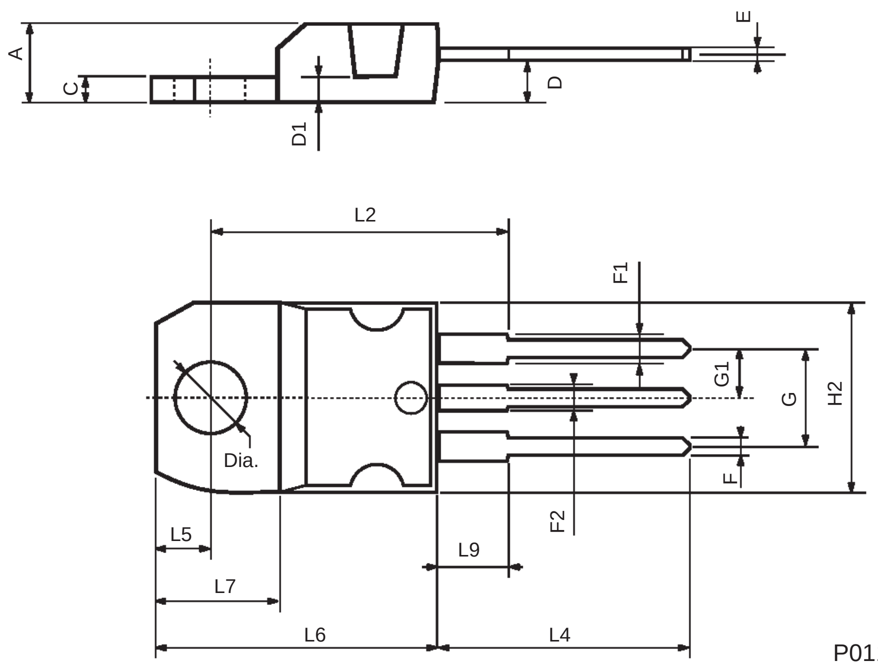
Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
I_{SD} I_{SDM}(•)	Source-drain Current Source-drain Current (pulsed)				16 64	A A
V_{SD} (∗)	Forward On Voltage	I_{SD} = 16 A V_{GS} = 0			1.6	V
t_{rr} Q_{rr} I_{RRM}	Reverse Recovery Time Reverse Recovery Charge Reverse Recovery Current	I_{SD} =16 A di/dt = 100 A/µs V_{DD} = 30 V T_j = 150 ^oC		150 0.8 10		ns µC A

(∗) Pulsed: Pulse duration = 300 µs, duty cycle 1.5 %
(•) Pulse width limited by safe operating area

Figure C.3: IRF530 MOSFET-Transistor technical data sheet

TO-220 MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A	4.40		4.60	0.173		0.181
C	1.23		1.32	0.048		0.051
D	2.40		2.72	0.094		0.107
D1		1.27			0.050	
E	0.49		0.70	0.019		0.027
F	0.61		0.88	0.024		0.034
F1	1.14		1.70	0.044		0.067
F2	1.14		1.70	0.044		0.067
G	4.95		5.15	0.194		0.203
G1	2.4		2.7	0.094		0.106
H2	10.0		10.40	0.393		0.409
L2		16.4			0.645	
L4	13.0		14.0	0.511		0.551
L5	2.65		2.95	0.104		0.116
L6	15.25		15.75	0.600		0.620
L7	6.2		6.6	0.244		0.260
L9	3.5		3.93	0.137		0.154
DIA.	3.75		3.85	0.147		0.151



P011C

Figure C.3: IRF530 MOSFET-Transistor technical data sheet

ISOWATT220 MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A	4.4		4.6	0.173		0.181
B	2.5		2.7	0.098		0.106
D	2.5		2.75	0.098		0.108
E	0.4		0.7	0.015		0.027
F	0.75		1	0.030		0.039
F1	1.15		1.7	0.045		0.067
F2	1.15		1.7	0.045		0.067
G	4.95		5.2	0.195		0.204
G1	2.4		2.7	0.094		0.106
H	10		10.4	0.393		0.409
L2		16			0.630	
L3	28.6		30.6	1.126		1.204
L4	9.8		10.6	0.385		0.417
L6	15.9		16.4	0.626		0.645
L7	9		9.3	0.354		0.366
Ø	3		3.2	0.118		0.126

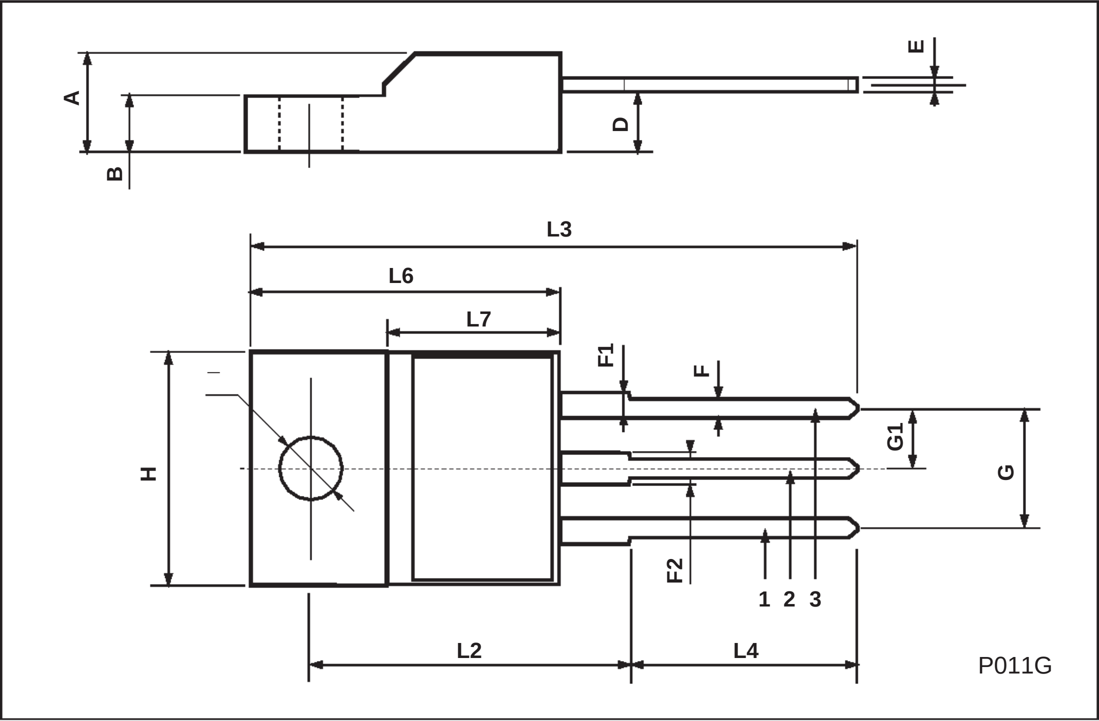


Figure C.4: BD139 NPN-Transistor technical data sheet

BD135 - BD136
BD139 - BD140

Complementary low voltage transistor

Features

- Products are pre-selected in DC current gain

Application

- General purpose

Description

These epitaxial planar transistors are mounted in the SOT-32 plastic package. They are designed for audio amplifiers and drivers utilizing complementary or quasi-complementary circuits. The NPN types are the BD135 and BD139, and the complementary PNP types are the BD136 and BD140.

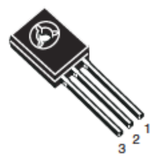


Figure 1. Internal schematic diagram

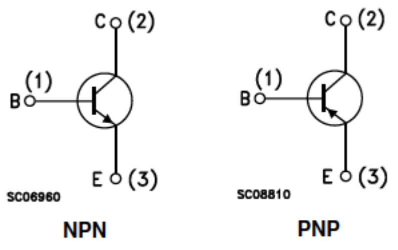


Table 1. Device summary

Order codes	Marking	Package	Packaging
BD135	BD135	SOT-32	Tube
BD135-16	BD135-16		
BD136	BD136		
BD136-16	BD136-16		
BD139	BD139		
BD139-10	BD139-10		
BD139-16	BD139-16		
BD140	BD140		
BD140-10	BD140-10		
BD140-16	BD140-16		

Figure C.4: BD139 NPN-Transistor technical data sheet

Contents

Figure C.4: BD139 NPN-Transistor technical data sheet

1 Electrical ratings

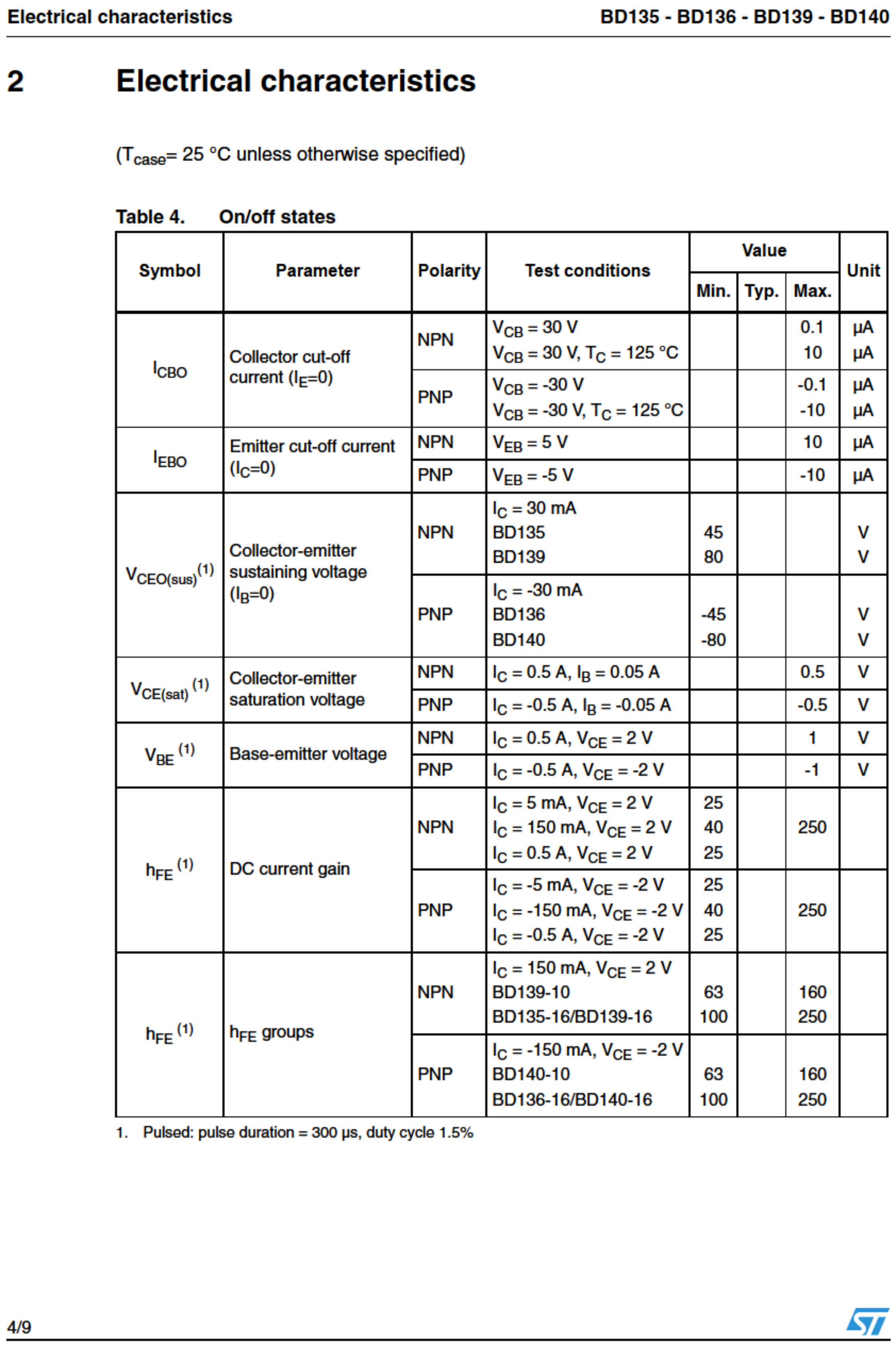
Table 2. Absolute maximum ratings

Symbol	Parameter	Value				Unit
		NPN		PNP		
		BD135	BD139	BD136	BD140	
V_{CBO}	Collector-base voltage ($I_E = 0$)	45	80	-45	-80	V
V_{CEO}	Collector-emitter voltage ($I_B = 0$)	45	80	-45	-80	V
V_{EBO}	Emitter-base voltage ($I_C = 0$)	5		-5		V
I_C	Collector current	1.5		-1.5		A
I_{CM}	Collector peak current	3		-3		A
I_B	Base current	0.5		-0.5		A
P_{TOT}	Total dissipation at $T_c \leq 25$ °C	12.5				W
P_{TOT}	Total dissipation at $T_{amb} \leq 25$ °C	1.25				W
T_{stg}	Storage temperature	-65 to 150				°C
T_j	Max. operating junction temperature	150				°C

Table 3. Thermal data

Symbol	Parameter	Max value	Unit
$R_{thj\text{-}case}$	Thermal resistance junction-case	10	°C/W
$R_{thj\text{-}amb}$	Thermal resistance junction-ambient	100	°C/W

Figure C.4: BD139 NPN-Transistor technical data sheet



2 Electrical characteristics

(T_{case}= 25 °C unless otherwise specified)

Table 4. On/off states

Symbol	Parameter	Polarity	Test conditions	Value			Unit
				Min.	Typ.	Max.	
I_{CBO}	Collector cut-off current (I_E=0)	NPN	V_{CB} = 30 V			0.1	µA
			V_{CB} = 30 V, T_C = 125 °C			10	µA
		PNP	V_{CB} = -30 V			-0.1	µA
			V_{CB} = -30 V, T_C = 125 °C			-10	µA
I_{EBO}	Emitter cut-off current (I_C=0)	NPN	V_{EB} = 5 V			10	µA
		PNP	V_{EB} = -5 V			-10	µA
$V_{CEO(sus)}$ (1)	Collector-emitter sustaining voltage (I_B=0)	NPN	I_C = 30 mA				
			BD135	45			V
			BD139	80			V
		PNP	I_C = -30 mA				
			BD136	-45			V
			BD140	-80			V
$V_{CE(sat)}$ (1)	Collector-emitter saturation voltage	NPN	I_C = 0.5 A, I_B = 0.05 A			0.5	V
		PNP	I_C = -0.5 A, I_B = -0.05 A			-0.5	V
V_{BE} (1)	Base-emitter voltage	NPN	I_C = 0.5 A, V_{CE} = 2 V			1	V
		PNP	I_C = -0.5 A, V_{CE} = -2 V			-1	V
h_{FE} (1)	DC current gain	NPN	I_C = 5 mA, V_{CE} = 2 V	25			
			I_C = 150 mA, V_{CE} = 2 V	40		250	
			I_C = 0.5 A, V_{CE} = 2 V	25			
		PNP	I_C = -5 mA, V_{CE} = -2 V	25			
			I_C = -150 mA, V_{CE} = -2 V	40		250	
			I_C = -0.5 A, V_{CE} = -2 V	25			
h_{FE} (1)	h_{FE} groups	NPN	I_C = 150 mA, V_{CE} = 2 V				
			BD139-10	63		160	
			BD135-16/BD139-16	100		250	
		PNP	I_C = -150 mA, V_{CE} = -2 V				
			BD140-10	63		160	
			BD136-16/BD140-16	100		250	

1. Pulsed: pulse duration = 300 µs, duty cycle 1.5%

Figure C.4: BD139 NPN-Transistor technical data sheet

2.1 Electrical characteristics (curves)

Figure 2. Safe operating area

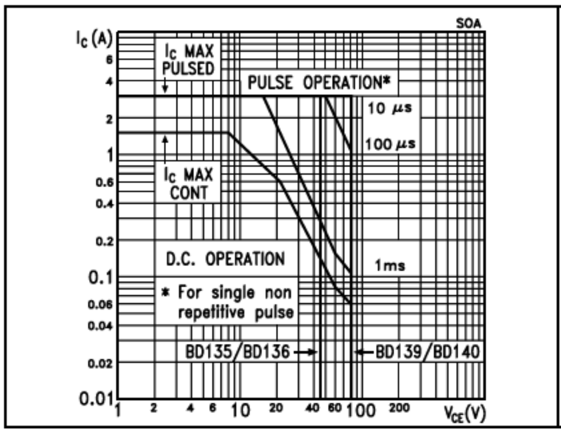


Figure 3. Derating

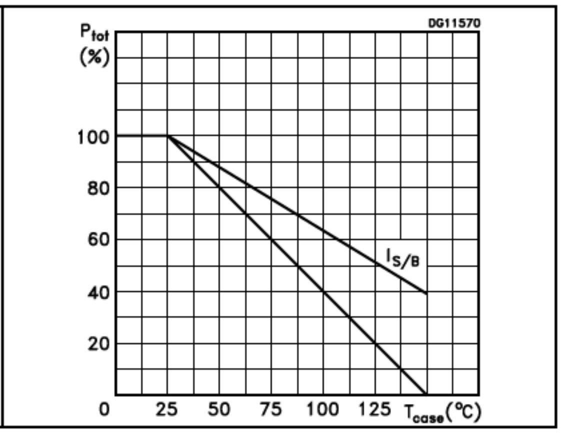
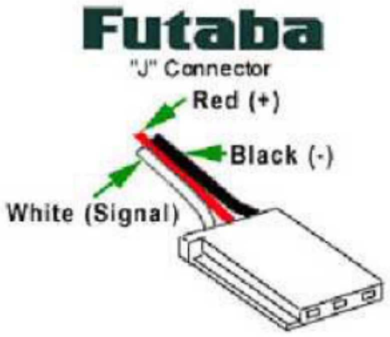


Figure C.5: Servo technical data sheet

S3003 FUTABA SERVO



...S3003 FUTABA SERVO...

Detailed Specifications

Control System:	+Pulse Width Control 1520usec Neutral	Current Drain (4.8V):	7.2mA/idle
Required Pulse:	3-5 Volt Peak to Peak Square Wave	Current Drain (6.0V):	8mA/idle
Operating Voltage:	4.8-6.0 Volts	Direction:	Counter Clockwise/Pulse Traveling 1520-1900usec
Operating Temperature Range:	-20 to +60 Degree C	Motor Type:	3 Pole Ferrite
Operating Speed (4.8V):	0.23sec/60 degrees at no load	Potentiometer Drive:	Indirect Drive
Operating Speed (6.0V):	0.19sec/60 degrees at no load	Bearing Type:	Plastic Bearing
Stall Torque (4.8V):	44 oz/in. (3.2kg.cm)	Gear Type:	All Nylon Gears
Stall Torque (6.0V):	56.8 oz/in. (4.1kg.cm)	Connector Wire Length:	12"
Operating Angle:	45 Deg. one side pulse traveling 400usec	Dimensions:	1.6" x 0.8"x 1.4" (41 x 20 x 36mm)
360 Modifiable:	Yes	Weight:	1.3oz. (37.2g)

D

final circuit drawing

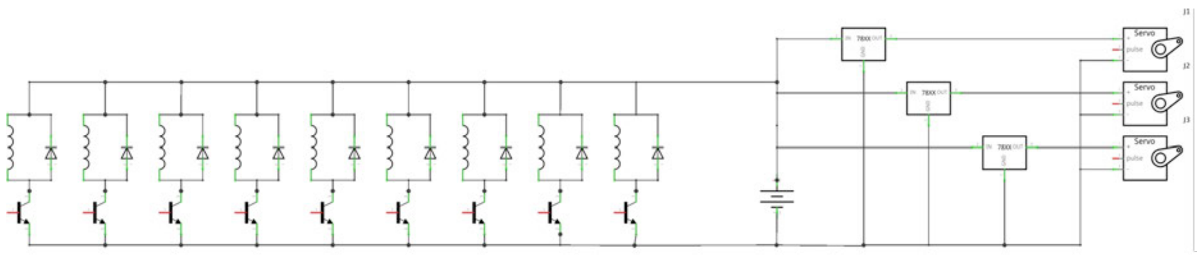


Figure D.1: final design of the electrical circuit

E
Structure drawings

Figure E.1: drawing for 1st iteration 1mm thick parts, scale 1:2

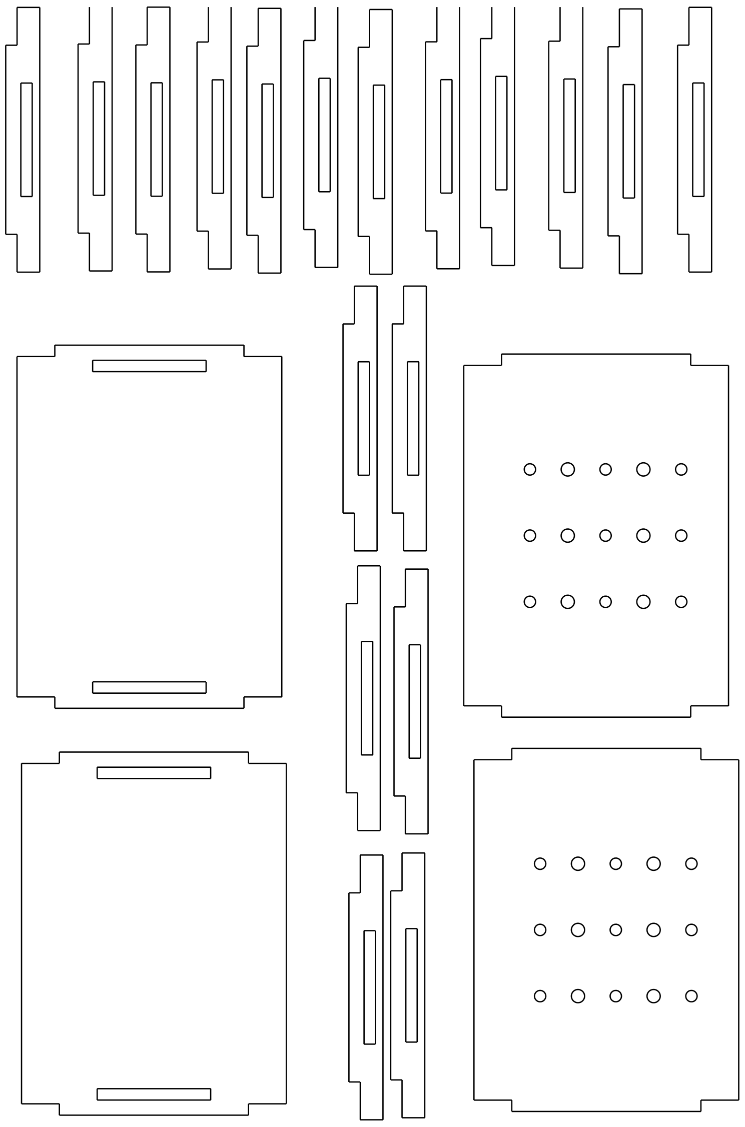


Figure E.2: drawing for 1st iteration 3mm thick parts, scale 1:2

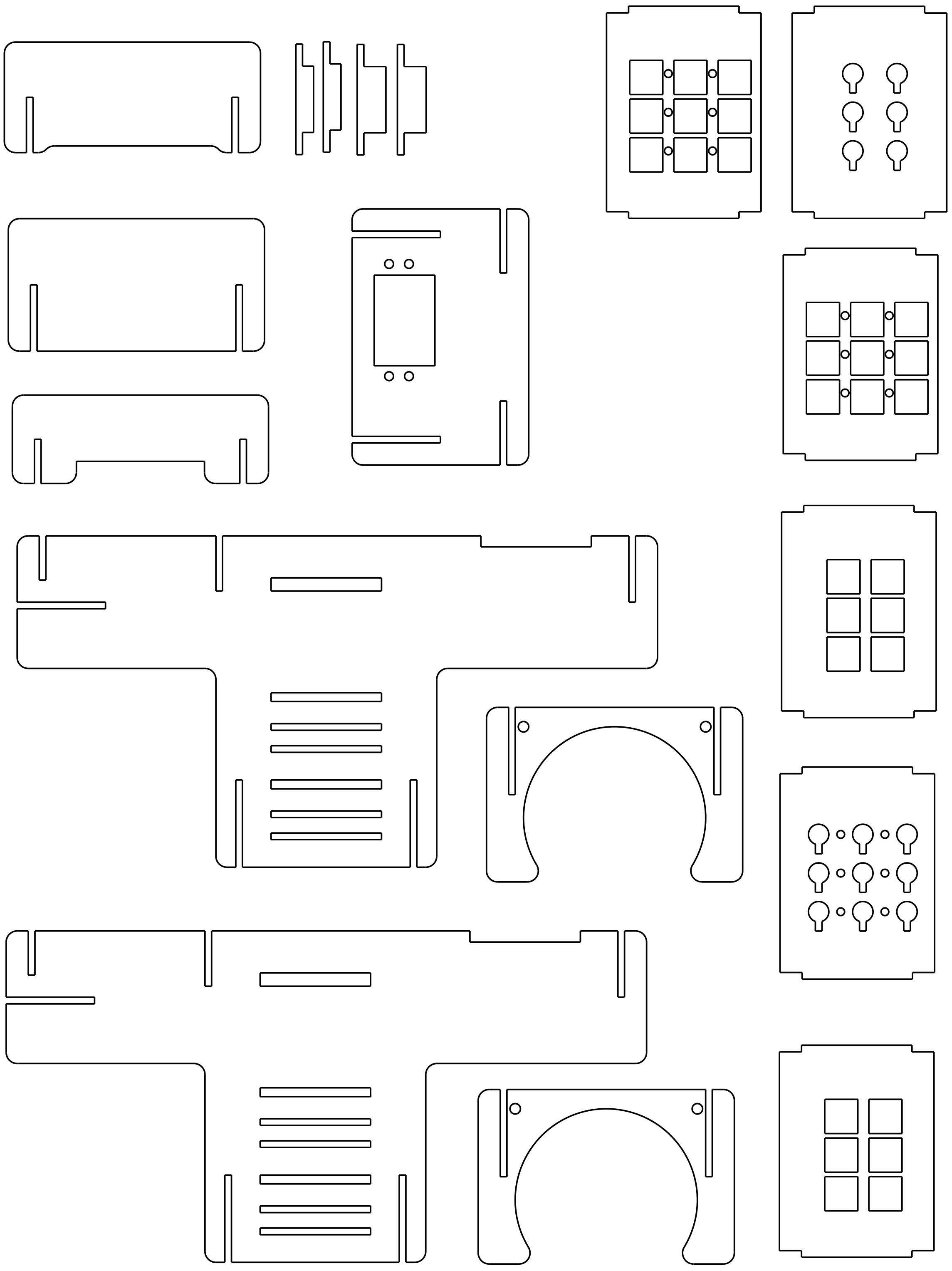


Figure E.3: drawing for 2nd iteration 3mm thick parts, scale 1:2

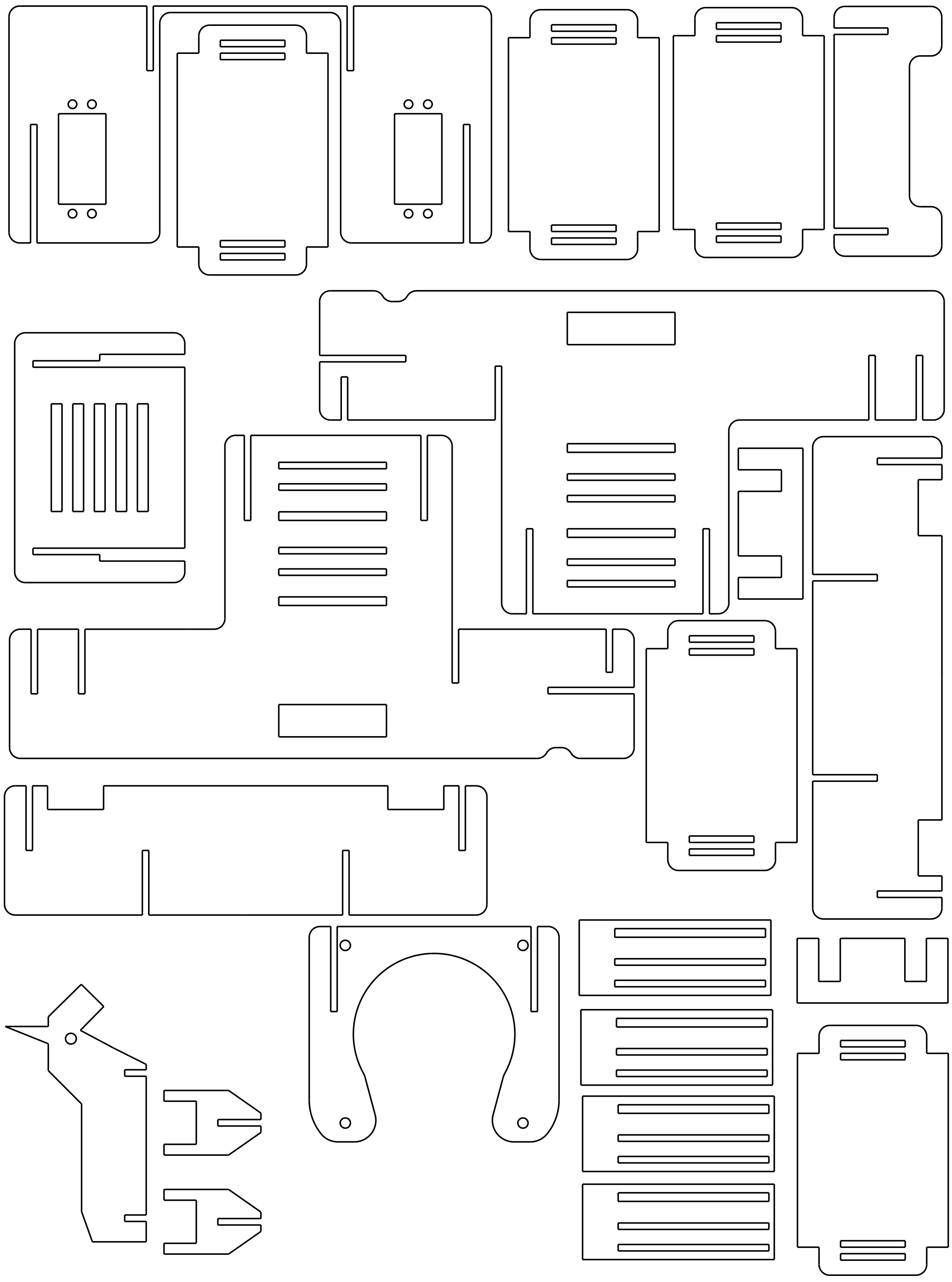


Figure E.4: drawing for the servo holder, scale 1:2

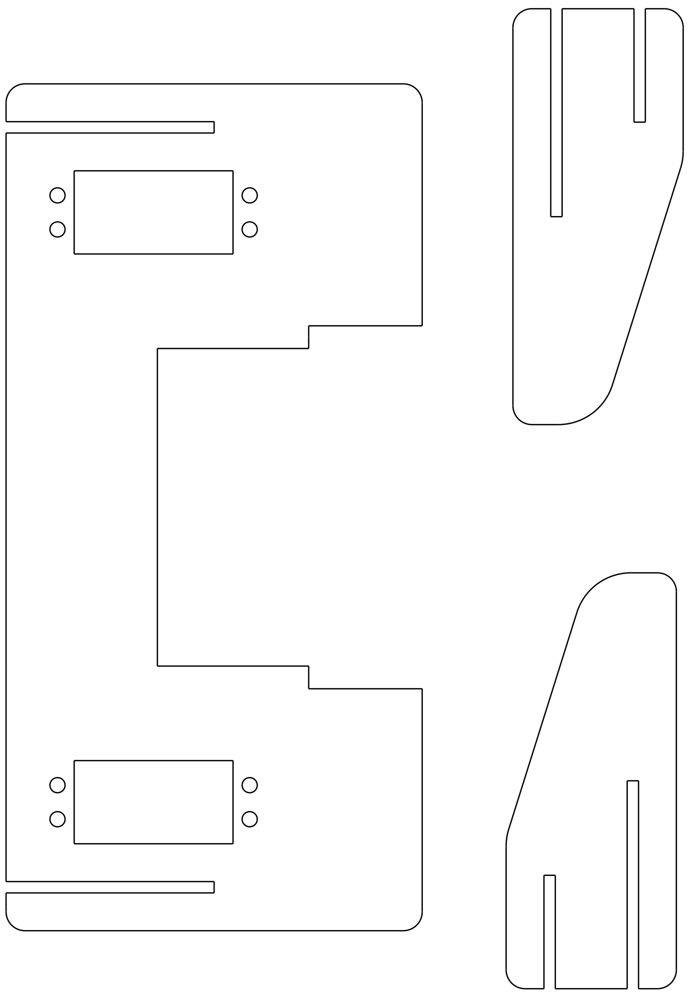
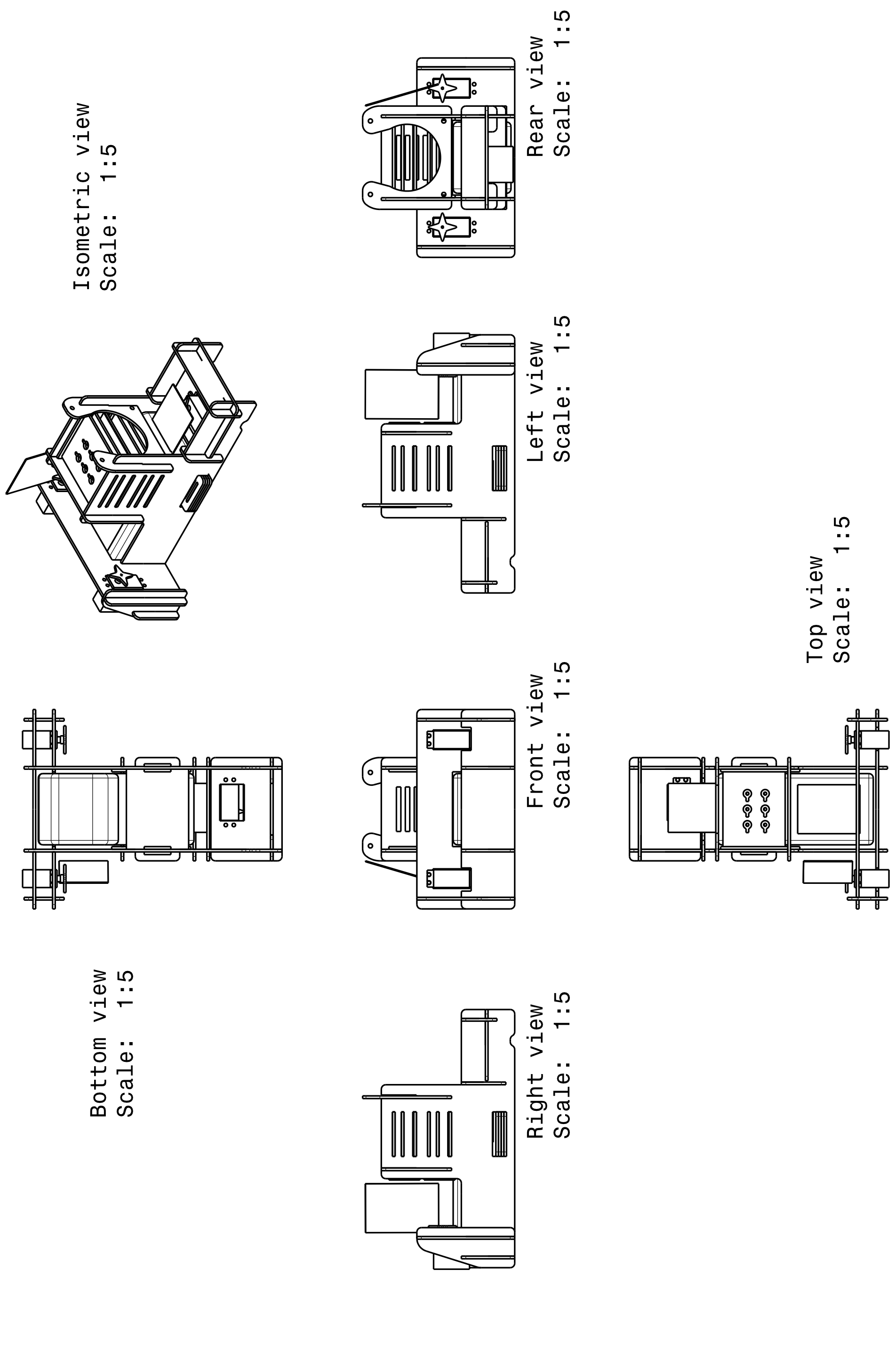


Figure E.5: scale isometric view, scale 1:5 reduced to 1:8



F Software Improvement Group

During the project there have been two upload moments to submit our code to the Software Improvement Group. The feedback we received from the first upload has been taken into account when improving the code for the second submission. You can read about this in section 4.2

F.1. First analysis (Dutch)

De code van het systeem scoort 3 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code gemiddeld onderhoudbaar is. De hoogste score is niet behaald door lagere scores voor Unit Interfacing en Module Coupling.

Voor Unit Interfacing wordt er gekeken naar het percentage code in units met een bovengemiddeld aantal parameters. Doorgaans duidt een bovengemiddeld aantal parameters op een gebrek aan abstractie. Daarnaast leidt een groot aantal parameters nogal eens tot verwarring in het aanroepen van de methode en in de meeste gevallen ook tot langere en complexere methoden.

Jullie hebben een aantal contructors die dezelfde parameters gebruiken (API key, token, private key). Als deze parameters logischerwijs bij elkaar horen is het beter om er een apart datatype voor te introduceren. Op die manier verhoog je ook het abstractieniveau van de code, waardoor deze beter leesbaar blijft als in de toekomst de hoeveelheid functionaliteit gaat groeien.

Voor Module Coupling wordt er gekeken naar het percentage van de code wat relatief vaak wordt aangeroepen. Normaal gesproken zorgt code die vaak aangeroepen wordt voor een minder stabiel systeem omdat veranderingen binnen dit type code kan leiden tot aanpassingen op veel verschillende plaatsen.

In dit systeem wordt de class 'Request' op 36 verschillende plaatsen aangeroepen. Daarnaast is deze class vrij fors. Je zou dit kunnen verbeteren door de datastructuur van request van het uitvoeren van het request te splitsen. Met andere woorden: de methode execute kan in een andere class RequestExecutor (of iets dergelijks). Op die manier voorkom je dat alle kernlogica centraliseerd wordt in één class.

De aanwezigheid van test-code is in ieder geval veelbelovend, hopelijk zal het volume van de test-code ook groeien op het moment dat er nieuwe functionaliteit toegevoegd wordt.

Over het algemeen scoort de code dus gemiddeld, hopelijk lukt het om dit niveau nog wat te laten stijgen tijdens de rest van de ontwikkelfase.

F.2. Second analysis (Dutch)

In de tweede upload zien we dat zowel de omvang van het systeem als de score voor onderhoudbaarheid is gestegen. Van alle groepjes zien we bij jullie de grootste verbetering tussen de eerste en de tweede upload. Jullie zijn gestegen naar 4 sterren, wat echt een presentatie is in zo'n korte tijd. De eerder genoemde verbeterpunten, Unit Interfacing en Module Coupling, zijn ook de grootste stijgers op het niveau van de deelscores.

Ook is het goed om te zien dat jullie naast nieuwe productiecode ook aandacht hebben besteed aan het schrijven van nieuwe testcode.

Uit deze observaties kunnen we concluderen dat de aanbevelingen van de vorige evaluatie zijn meegenomen in het ontwikkeltraject.