Documentatie Pinautomaat

Bytegroep 10

20 juni 2018

Samenvatting

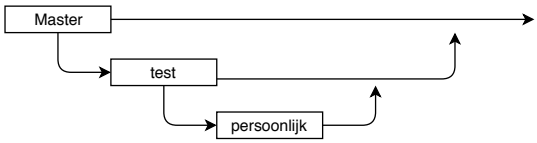
De opdracht voor dit project was om een werkende geldautomaat te maken. In dit verslag zijn onderzoek analysen met adviezen terug te vinden. Wij hebben aan de hand van deze adviezen hierna ontwerpen gemaakt en gerealiseerd. Wij moeten tijdens de loop van het project letten op vijf onderdelen: *beheren, analyseren, adviseren, ontwerpen en realiseren.*

Inhoudsopgave

1 Version control

Voor dit onderdeel moesten wij kunnen werken met Git en Version control. Hier is veel onderzoek over gedaan zodat het goed toegepast kan worden dit project. In figuur 1 is het model wat wij gebruiken op onze Git pagina, Project4Bankalicious. Ieder groepslid werkt op zijn individuele branch, en als je klaar bent met jouw onderdeel, dan wordt hij in test gemerged. Als er geen problemen lijken te zijn op de test beslissen wij in een groep of wij test naar de master kunnen schrijven. Alle documentatie is terug te vinden op Github.

- Issues on Github



Figuur 1: git model

2 Samenwerkingscontract

Financiën De kosten voor het gezamenlijke deel van de opdracht(en) worden gezamenlijk betaald, iedereen betaald even veel.

Schades Als er iets kapot gaat, worden de kosten van het vervangen gedeeld door de groep tenzij er opzettelijk of onverantwoordelijk mee om is gegaan. In dat geval zullen de kosten voor de persoon die de schade heeft aangericht zijn.

Github afspraken

- Niet op de master werken. Je werkt altijd op een eigen kopie van de test branch.
- Gelieve geen word bestanden te uploaden zonder ook een pdf te uploaden.
- Commits moeten duidelijk beschrijven wat er is aangepast. Liever een paar woorden teveel dan te weinig.
- De repository van project 4 is geen zandbak om Git te testen.
- Directories zijn met kleine letters en underscores, en hebben een duidelijke naam.
- Alle benamingen en commits moeten in het Nederlands.
- Branch namen zijn met kleine letters en woorden zijn verbonden met een min teken.

Te laat bij afspraken Al onze afspraken worden in goed overleg gepland daarom wordt er van groepsleden verwacht dat ze op tijd zijn bij een afspraak (zowel deadlines als overleg) als een groepslid zonder goede 1 reden meer dan 5 minuten te laat is dan staat daar een consequentie tegen over. Deze consequentie is of 20 chicken McNuggets of een taart (let op: cake is geen taart!), als er meerdere mensen te laat zijn moeten zij allebei iets mee nemen.

Afwezigheid bij afspraken Al onze afspraken worden in goed overleg gepland daarom wordt er van groepsleden verwacht dat ze bij afspraken aanwezig zijn. Als een groepslid zonder goede reden afwezig is bij een overleg staat daar een consequentie tegenover. Deze consequentie is of 40 chicken McNuggets of twee taarten. Deze reden zal met de groep besproken moeten worden voordat het een geldige reden is.

Communicatie Communiceer goed, dat betekent duidelijk en met respect durf anderen aan te spreken. Vraag hulp als je hulp nodig hebt. Geef op tijd aan als je wil stoppen met de studie en draag alle onderdelen van het project tijdig over aan een ander groepslid.

Rolverdeling De rolverdeling staat netjes gedocumenteerd op Github, en ieder teamlid is gediend deze zo goed mogelijk uit te voeren.

- Paul H. Organisator
- Paul W. Verbinder
- Gerard Expert
- Merijn Documentatie manager, onderzoeker
- Aron Tester
- Boas Kwaliteit bewaker
- Floor Designer, expert
- Mohammed Designer

2.1 Kwaliteitseisen

Aan het begin van het project hebben we een paar eisen beschreven waar ons eindproduct aan moet voldoen, ze staan hieronder.

2.1.1 Code eisen

- De code moet leesbaar zijn voor iedereen dus ook voor mensen die het niet geschreven hebben. Het doel is dat al onze ouders ook kunnen begrijpen wat er staat, een onderdeel hier van is commends.
- De code moet zo efficiënt mogelijk zijn. Variabele moeten een goede en duidelijk naam hebben en een vaste structuur volgen, namen van klasse beginnen met een Hoofdletter, namen van andere variabelen beginnen met een kleine letter.
- Als een variabele bestaat uit twee woorden begint het tweede woord met een Hoofdletter. Aka camelcase bv. camelCase.
- Splits zoveel mogelijk code op in classes en methodes. Stop bijbehorende code in een pakkage.
- Maak eerst een klasse diagram voor dat je gaat programmeren.
- In bestandsnamen mogen geen * worden gebruik dit vind Windows namelijk niet leuk.

2.1.2 Product eisen

- De geldautomaat moet biljetten van tenminste vier verschillende waarden uit kunnen geven
- De gebruiker kan niet, zonder een pin-opdracht te geven, geld uit de automaat halen
- De geldautomaat geeft altijd het juiste bedrag
- De geldautomaat geeft alleen geld als het saldo toereikend is
- De gebruiker kan zelf selecteren welke biljetten hij/zij wil ontvangen
- De gebruiker kan geen biljetten kiezen die niet aanwezig zijn in de geldautomaat
- De geldautomaat is robuust (kan zelfstandig staan en valt niet om/uit elkaar tijdens gebruik)
- De biljetten in de geldautomaat mogen maximaal de dikte van een speelkaart hebben

- Na het pinnen wordt er een bon geprint met een bonprinter. Op deze bon staat in ieder geval hoeveel geld er is opgenomen en bij welke (lokale of individuele) bank dit is gebeurd
- Er moet encryptie gebruikt worden tussen de geld automaat en de server en tussen de Arduino en de computer.

2.1.3 Eindgebruiker eisen

- Een mooie en overzichtelijke GUI voor de pin automaat, het moet de oma test kunnen doorstaan (als er tijd is gaan we het echt proberen).
- Het moet niet zomaar uitvallen.
- De pin automaat moet duidelijk terug communiceren wat de automaat verwacht.
- Als er geld afgeschreven wordt komt er het zelfde bedrag uit de automaat.
- Als er lange tijd niks gebeurd wordt de gebruiker automatiche uitgelogd.

2.1.4 Git eisen

- Niet op de master werken. Je werkt altijd op een eigen kopie van de test branch.
- Gelieve geen word bestanden te uploaden zonder ook een pdf te uploaden.
- Commits moeten duidelijk beschrijven wat er is aangepast. Liever een paar woorden veel dan te weinig.
- De repository van project 4 is geen zandbak om Git te testen.
- Directories zijn met kleine letters en underscores, en hebben een duidelijke naam.
- Alle benamingen en commits moeten in het Nederlands.

2.1.5 Git eisen

Zie sectie 2 voor de eisen voor onze git pagina.

3 Analyse & advies

3.1 Dispenser

Wij hebben onderzoek gedaan naar twee manieren van het uitgeven van biljetten. Wij hebben vooral onderzoek gedaan bij de tweede jaars. Optie één is om een zuigpomp te gebruiken en de biljetten naar boven te zuigen. De tweede jaars raden dit af, omdat zij hier zelf geen goede ervaring mee hadden.

Manier twee is om de biljetten uit te werpen met een elektromotor. De tweede jaars hadden hier goede ervaring mee, dit ding namelijk minder vaak fout bij het uitwerpen. De elektromotor zal de kaarten uit de dispenser rollen.

Voor het uitgeven van biljetten is het efficiënter om een elektromotor te gebruiken. Hier is er meer kans dat het juiste aantal biljetten uit de automaat komen.

3.2 Veiligheid

Netwerk encryption Er is veel onderzoek gedaan naar de beveiliging van het dataverkeer. Voor netwerken beveiliging heb je een encryptie nodig over het netwerk. Wij raden aan om hiervoor een bestaand protocol toe te passen. SSL wordt professioneel toegepast bij bedrijven. Het kan gewoon gebruikt worden als hij goed toegepast wordt met de meest recente versie.

lokale encryption Voor lokale encryption zou je de hele database kunnen encrypten, en decrypten wanneer je data nodig hebt. Dit kost echter een hoop rekenkracht, en je gegevens zullen op een paar plaatsen onversluiteld zijn.

Een oplossing hiervoor is om gevoelige gegevens te hashen, en vervolgens in de database op te slaan. Als je bijvoorbeeld wilt controleren of een pincode klopt, zal deze eerst gehasht worden en vervolgens vergeleken met wat er in de database staat. Een veelgebruikt hashing algoritme voor java en c++, de talen waar wij in programmeren, is bijvoorbeeld md5.: Het is beter om een hashing algoritme te gebruiken wat iets zwaarder is dan md5, omdat je een hash makkelijk terug kunt rekenen. Een beter veelgebruikt algoritme is bijvoorbeeld SHA1. Deze adviseren wij te gebruiken.

$$\text{pincode} \xrightarrow{hash} \text{hashed pincode} \xrightarrow{vergelijk} \text{Databasemethashes}$$

Afgesloten hardware Het is heel cruciaal voor de automaat dat buitenstaanders geen toegang hebben tot de logica van de machine en de biljetten. Om dit te voorkomen is het het handigst om de hardware aan de binnenkant van de automaat te houden. Wij raden aan om een kast om de functionele onderdelen heel te bouwen, zodat deze niet meer toegankelijk zijn.

3.3 Biljetten

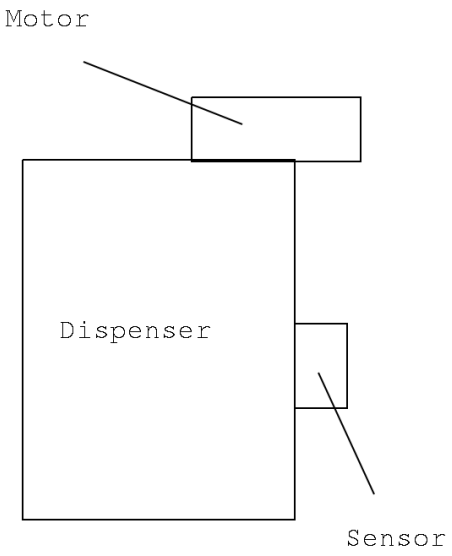
Wij hebben eerst aan de tweede jaars gevraagd wat voor soort materiaal wij het best konden gebruiken voor de biljetten. Zij kwamen met twee opties.

- Je kan zelf biljetten maken, om ze zo op je model af te stellen.
- Je kan speelkaarten gebruiken, omdat deze gemaakt zijn om met een dispenser uit te werpen.

Wij raden aan voor het gebruik van speelkaarten vanwege de dispencer. De manier hoe de dispencer gemaakt is zorgt ervoor dat alleen de speelkaarten geschikt zijn voor biljetten. Wij kunnen direct aan te slag met de dispenser als wij speelkaarten gebruiken, sinds wij deze al hebben. Er zijn 4 dispencers waarvan elk een bepaald type biljet bevat. Eén dispencer bevat biljetten van €5, een ander van €10, een ander weer van €20 en één van €50.

3.4 Sensoren

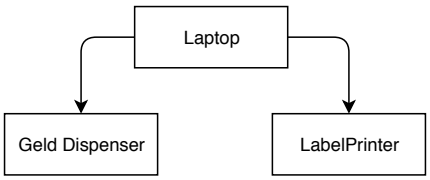
In een ATM kan er maar een bepaald hoeveelheid van biljetten. In de ATM moet er bijgehouden worden hoeveel biljetten er zijn uitgegeven. Om de uitgave ban biljetten te meten gaan wij gebruik maken van sensoren. Er zullen 4 sensoren gebruikt worden bij elke dispencer. Elke sensor kan bijhouden hoeveel biljetten van zijn aangewezen dispencer zijn uitgegeven. In het systeem van de ATM wordt dit geregistreed.



Figuur 2: Zijaanzicht Dispenser Sensor

3.5 Communicatie

De laptop zal opdrachten sturen naar de Labelprinter en de geld dispenser. Zie de netwerk diagram in figuur 3



Figuur 3: Netwerk Diagram

Hieronder is in pseudocode te zien welke opdrachten de laptop kan versturen naar de verbonden apparaten

```
public class App{
  new Printer("anaam", "vnaam", "atmNaam", "bedrag";

  new Dispenser;
  dispenseMoney("aantal5euro", "aantal10euro", "aantal20euro
      ", "aantal50euro");
}
```

Voor de communicatie tussen de klassen in de GUI van de automaat zijn in een diagram weergegeven. Om deze te bezichten, zie de bijlage.

3.6 Bonnen

De Obama.cf bank
Powered by:
Wondel en de Hekkies
wondelendehekkies.cf

klant: Merijn Plagge
EURO: 900
ATM: ATM@roffa
datum: 2018-06-13
tijd: 12:13:42

Figuur 4: Bon ontwerp

Voor het uitgeven van bonnen hebben wij een labelprinter gebruikt. In figuur 4 is de bon te zien die de printer uitprint. Hieronder is een link naar de sourcecode.

- Code labelprinter

Pseudocode uitleg

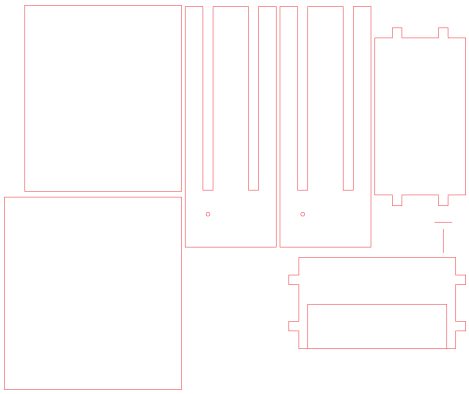
```
1  public class Printerclass {
2    String string;
3    // contructor
4    public PrinterClass(voornaam, achternaam, atmNaam, bedrag)
         {
5      string = inhoud wat geprint moet worden.
6
7      Printerjob pj = voorbereiden voor printen.
8      pageformat pf = het formaat waarin geprint word.
9      pf.setOrientation(PageFormat.PORTRAIT); zet de
           orientatie naar portrait
10     pf.setPaper(paper); zet hoe groot het papier is zodat
           alles op de bon komt.
11     pj.setPrintable(new MyPrintable(), pf);
12         try {
13             pj.print();
14         } catch (PrinterException ex) {
15             ex.printStackTrace();
16     }
17     Bovenstaande stukje zorgt ervoor dat het daadwerkelijk
           geprint word en dat een error word opgevangen.
18   }
19 }
```

3.7 Materiaal

Er zijn hier een paar opties. We kunnen bijvoorbeeld onderdelen 3d printen, maar dat zou erg duur zijn. Het meest voor de hand liggende materiaal is triplex hout. Dit is een redelijk goedkoop, en makkelijk verkrijgbaar materiaal. Wij kunnen dit in het *Tesla Lab* ophalen en daar direct uitsnijden.

4 Ontwerp & realisatie

4.1 Ontwerp proces dispensers



Figuur 5: Dispenser ontwerp

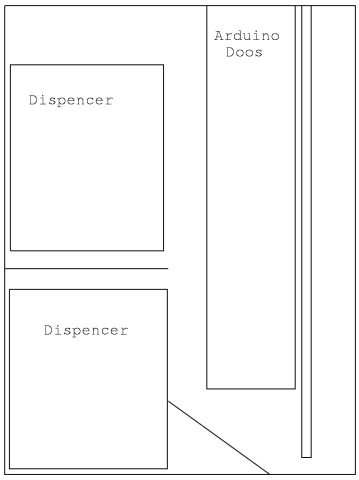
Ontwerp De automaat moest vier dispensers hebben. Wij zijn begonnen met een prototype te maken voor een van deze dispensers. In figuur 5 is een bouwtekening te zien die wij hebben uitgesneden en daarna in elkaar gezet.

Tijdens het vormgeven bleek dat elastiekjes het makkelijkste werkten om het plateau waar het geld in ligt naar boven te trekken. Ook bleek dat de as van de motor het best van lego gemaakt kon worden. Hiervoor was het idee om de as van karton te maken, maar dit bleek niet sterk genoeg te zijn en moeilijk te maken. Na het testen met de elastiekjes en de as hebben wij de meeste onderdelen aan elkaar gelijmd.

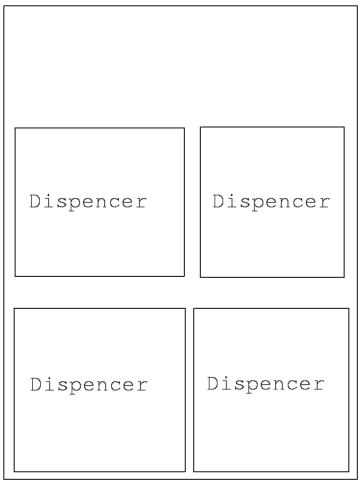
Ontwerp 2.0 Hierna hebben we op de millimeters nauwkeurig gekeken wat de beste afstanden zouden zijn voor de uitgaven van de kaarten. Aan de hand hiervan hebben wij een nieuwe print gemaakt en opnieuw alles getest. Deze nieuwe print wijkt enkele millimeters af van het eerste ontwerp Deze bleek goed, en wij hebben *Ontwerp 2.0* dus uiteindelijk drie keer extra geprint.

4.2 Bouwtekening ATM

Figuur 6: Vooraanzicht ATM

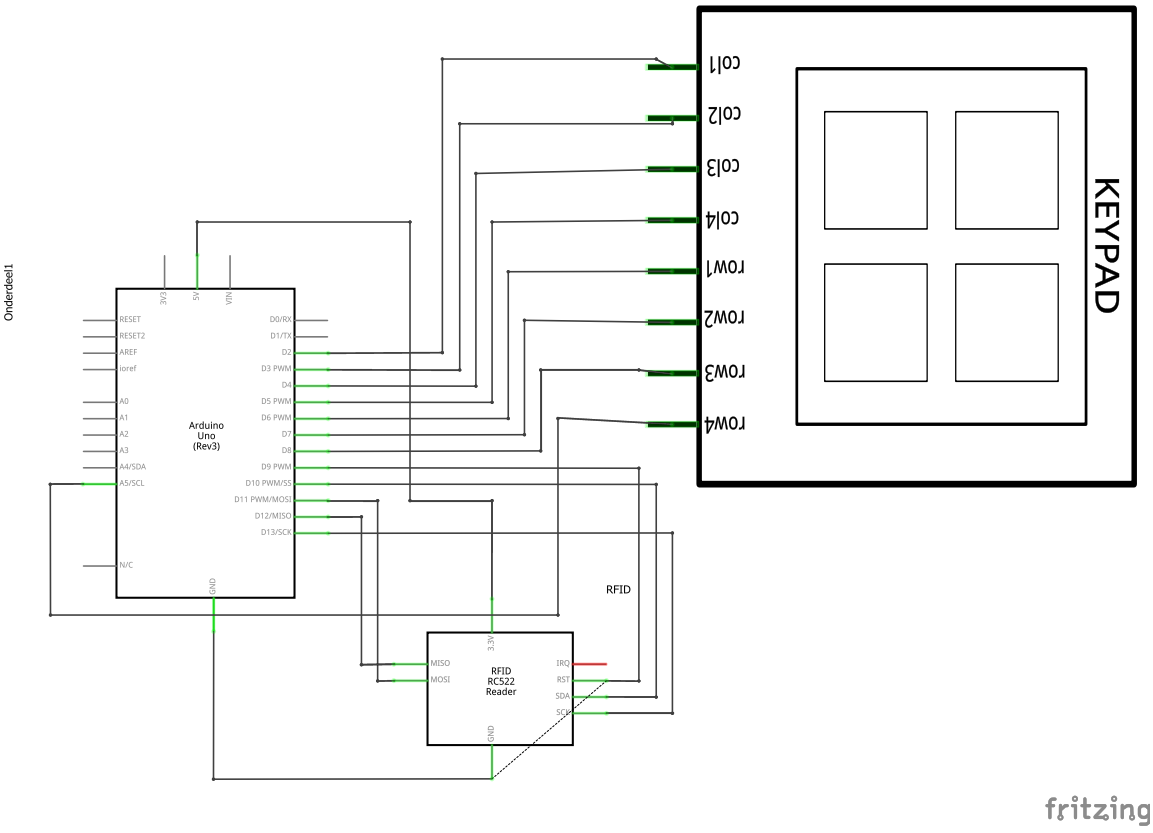


Figuur 7: Zijaanzicht ATM

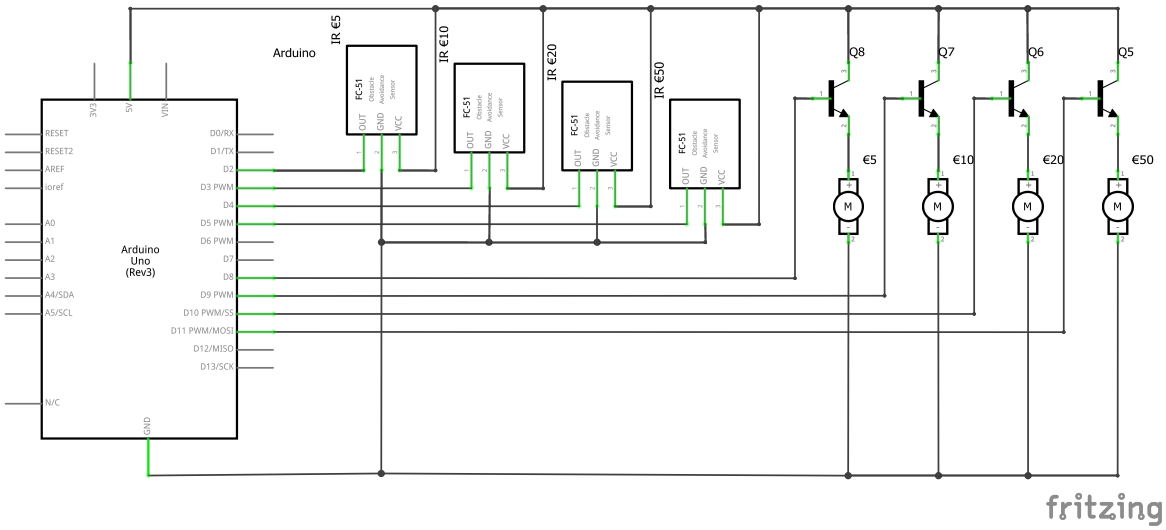


Figuur 8: Achterkant ATM

4.3 Electrisch schema



Figuur 9: Keypad en RFID



Figuur 10: Dispencer Schema

4.4 Toepassen kwaliteitseisen

In deze paragraaf worden de de kwaliteids eisen die we eerder hebben opgesteld vergeleken met het eindproduct.

De code moet leesbaar zijn voor iedereen dus ook voor mensen die het niet geschreven hebben. Het doel is dat al onze ouders ook kunnen begrijpen wat er staat, een onderdeel hier v an is commends. We denken niet dat onze ouders de code kunnen begrijpen, de code is wel lees en begrijpbaar door anderen mensen met programeer kennis. We hebben wel overal netjes code ingesprongen.

De code moet zo efficiënt mogelijk zijn. De code is redelijk efficent, er zit alleen wel een while(true), daarom is het programma vrij zwaar op het systeem. Een mogelijkheid om de code te verbeteren is het vervangen van wile loop's door interupts.

Variabele moeten een goede en duidelijk naam hebben en een vaste structuur volgen, namen van klasse beginnen met een Hoofdletter, namen van andere variabelen beginnen met een klei ne letter. Dit hebben we goed gedaan.

Als een variabele bestaat uit twee woorden begint het tweede woord met een Hoofdletter. Aka camelcase bv. camelCase. Dit hebben we ook goed gedaan.

Splits zoveel mogelijk code op in classes en methodes. Stop bijbehorende code in een pakkage. We hebben redelijk veel verschillende classes en methodes, dit had meer kunnen zijn maar is over het algemeen goed. We hebben geen pakkage's gebruikt.

Maak eerst een klasse diagram voor dat je gaat programmeren. Het grootste gedeelte van de code was al klaar voordat we met dit project begonnen dit is daarom onrelevand.

In bestandsnamen mogen geen * worden gebruik dit vind Windows namelijk niet leuk. We hebben een * gebruikt in bestands namen.

4.4.1 Product eisen

De geldautomaat moet biljetten van tenminste vier verschillende waarden uit kunnen geven Dat kan

De gebruiker kan niet, zonder een pin-opdracht te geven, geld uit de automaat halen Het is niet mogelijk om geld uit de automaat te halen zonder pin opdracht.

De geldautomaat geeft altijd het juiste bedrag Meestal geeft de automaat het juiste bedrag, er zijn allen momenten dat er per ongeluk een extra biljet uit komt.

De geldautomaat geeft alleen geld als het saldo toereikend is De automaat cheked eerst het saldo voordat hij geld uitgeeft.

De gebruiker kan zelf selecteren welke biljetten hij/zij wil ontvangen Dit kan, mits er genoeg biljetten in de automaat zitten.

De gebruiker kan geen biljetten kiezen die niet aanwezig zijn in de geldautomaat De automaat houdt bij hoeveel biljetten er nog in de automaat zitten.

De geldautomaat is robuust (kan zelfstandig staan en valt niet om/uit elkaar tijdens gebruik) De automaat is stefig genoeg om te blijven staan en niet uit elkaar te vallen, als er geweld word gebruikt gaat hij wel kapot.

De biljetten in de geldautomaat mogen maximaal de dikte van een speelkaart hebben We gebruiken speelkarten als biljetten, die zijn dus niet te dik.

Na het pinnen wordt er een bon geprint met een bonprinter. Op deze bon staat in ieder geval hoeveel geld er is opgenomen en bij welke (lokale of individuele) bank dit is g ebeurd De bonnetjes printen, op de bonnetjes staat ht loge van de bank, de naam van de klant, het opgenomen bedrag, bij welke automaat de pintransactie is gedaan, de datum en de tijd.

Er moet encryptie gebruikt worden tussen de geld automaat en de server en tussen de Arduino en de computer. Dit hebben we niet gedaan.

4.4.2 Eindgebruiker eisen

Een mooie en overzichtelijke GUI voor de pin automaat, het moet de oma test kunnen doorstaan (als er tijd is gaan we het echt proberen). We hebben de oma test niet geprobeerd maar vinden de GUI zelf erg duidelijk.

Het moet niet zomaar uitvallen. Tenzij de gebreuker het actief kapot maakt blijft het werken.

De pin automaat moet duidelijk terug communiceren wat de automaat verwacht. Dit is over het algemeen goed maar kan op sommige momenten nog net iets beter.

Als er geld afgeschreven wordt komt er het zelfde bedrag uit de automaat. Dit gaat meestal goed, soms komt er allen iets te veel geld uit.

Als er lange tijd niks gebeurd wordt de gebruiker automoties uitgelogd. Dit gebeurt niet.

4.4.3 Git eisen

Niet op de master werken. Je werkt altijd op een eigen kopie van de test branch. Ja dit hebben we allemaal gedaan.

Gelieve geen word bestanden te uploaden zonder ook een pdf te uploaden. We hebben alle word bestanden ook geüpload als .pdf.

Commits moeten duidelijk beschrijven wat er is aangepast. Liever een paar woorden te veel dan te weinig. De meeste commits zij vrij duidelijk, sommige hadden beter gekund.

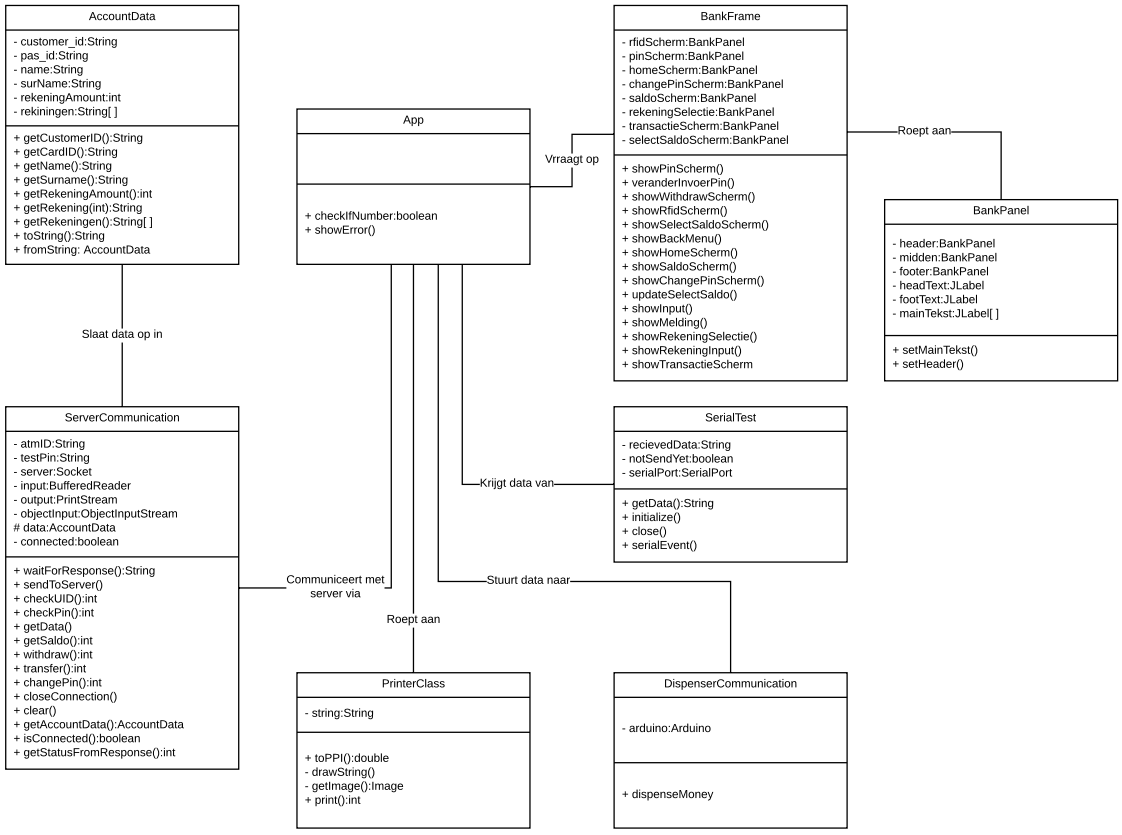
De repository van project 4 is geen zandbak om Git te testen. We hebben de project 4 repository niet gebruikt als zandbak.

Directories zijn met kleine letters en underscores, en hebben een duidelijke naam. Alle directories voldoen aan deze eisen behalve die van Floor.

Alle benamingen en commits moeten in het Nederlands. Het grootste deel van de commits is volledig in het Nederlands, sommige computer termen zijn wel in het Engels.

5 Bijlagen

5.1 Klassen diagram GUI



Figuur 11: Klassediagram GUI

6 Risico log

Hieronder is het risico log gedocumenteerd. Hier staan de problemen die wij tijdens het project hebben behandeld. Er staat hier ook hoe en of ze zijn opgelost.

Tabel 1: Risico log

	Beschrijving risico	**Kans**	**Impact**	**Risico**	**Maatregel**	**Status**	**Beschrijving status**	**Da tum**
1	Computer teamlid crasht	1	2	2	G it gebruiken	N	Alles staat op Git	018-5-9
2	Groepslid verlaat het team	1	4	4	T Taken overdragen	O	Hierover hebben wij afspraken	018-5-16
3	Iemand breekt de Git pagina	3	4	12	Version control leren met git updates geven wat	O	Iedereen kent Git nu	018-5-16
4	Groepslid is vaak afwezig	2	3	6	U we hebben gedaan	O	Aron en Floor zijn vaak afwezig	018-5-16
5	Klant eis vervalt	5	5	25	Back up taak nodig	O	Paul W verloor zijn taak	018-6-20

Tabel 2: Tabel beschrijving

Afkorting	**Beschrijving**
N	Nog geen probleem
O	Opgelost probleem
B	Nu een Bezig probleem
NO	Probleem niet kunnen oplossen
Rows	**Aantal**
Impact	[1-5]
Kans	[1-5]
Risico	[Impact*Kans]
Datum	Datum toegevoegd