

**360 video**

**Debbaut Sara**

**De Roover Mathias**

**Van Audenaerde Jasper**

|  |
| --- |
| **Projecten 2** |
| **Academiejaar 2013-2014** |
| **Professionele Bachelor ICT** |

Inhoud

[1. Onderwerp 2](#_Toc378569583)

[2. Realisatie 2](#_Toc378569584)

[Gebruikte software 2](#_Toc378569585)

[Visual Studio 2013 2](#_Toc378569586)

[Idle 2](#_Toc378569587)

[Gebruikte hardware 3](#_Toc378569588)

[Kinect 3](#_Toc378569589)

[Raspberry Pi 3](#_Toc378569590)

[Raspicam 3](#_Toc378569591)

[Servo HS-311 4](#_Toc378569592)

[3. Functionaliteit 5](#_Toc378569593)

[4. Code 6](#_Toc378569594)

[Client programma 6](#_Toc378569595)

[Raspberry Pi 10](#_Toc378569596)

[Python code 10](#_Toc378569597)

[5. Demo 12](#_Toc378569598)

[6. Conclusie 12](#_Toc378569599)

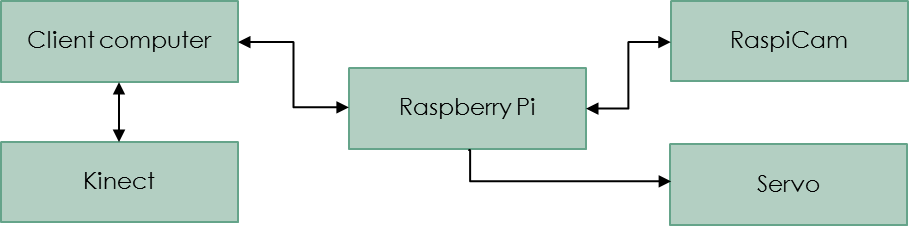
[Kinect 12](#_Toc378569600)

[Raspberry Pi 12](#_Toc378569601)

[Raspicam 12](#_Toc378569602)

# Onderwerp

Het project bestaat uit een spraak/bewegingssensor, de Kinect. Deze sensor registreert handelingen die door de gebruiker uitgebeeld worden, en stuurt ze vervolgens door naar het client-programma. Dit client-programma zal vervolgens deze handelingen omzetten in signalen bestemd voor de Raspberry Pi. De Raspberry pi zal hierna de servomotor aansturen zodat de camera-module kan roteren. De beelden van deze roterende camera worden via de Raspberry Pi teruggestuurd naar het client-programma en uiteindelijk getoond op het beeldscherm. Dit alles gebeurd over een intern netwerk.



# Realisatie

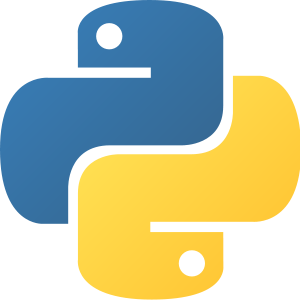
## Gebruikte software

### Visual Studio 2013

Visual Studio 2013 is een krachtige tool van Microsoft die een complete set ontwikkelingstools om computerprogramma’s, in diverse programmeertalen te ontwikkelen, aanbiedt. Van de beschikbare talen is voor dit project C# gebruikt omdat de Kinect van Microsoft dit vereist.

Downloaden op: <http://www.visualstudio.com/>

### Idle

Idle is een standaard programmeeromgeving om programma’s in Python 2.6 te schrijven, dat standaard op het Debian Wheezy besturingssysteem van de Raspberry Pi staat.

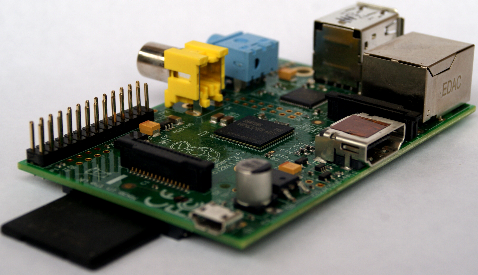
## Gebruikte hardware

### File:KinectSensor.pngKinect

De Kinect, een sensor ontwikkeld door Microsoft om de Xbox 360 zonder controller te besturen, alsook Windows applicaties. Deze bestaat uit een camera, infraroodsensor en een microfoon.

De software van Kinect is in staat tot complexe gebaarherkenning, spraak- en gezichtsherkenning. Het is mogelijk het 'skelet' van maximaal vier personen tegelijk te herkennen. Hierbij kunnen niet alleen hand en armgebaren herkend worden, maar kan volledige lichaamsbeweging gedetecteerd worden. Afhankelijk van de afstand tot de 'camera' kunnen zelfs individuele vingers herkend worden.

### Raspberry Pi

Dit is een single-board computer. Dit wil zeggen dat je een complete computer hebt die gebouwd is op één printplaat, ook zijn dit meestal kleine printplaten. Aangezien het een complete computer is heb je alle eigenschappen van een computer. CPU, geheugen, input/output. De Raspberry heeft deze dus ook. Hierdoor kan je hem inzetten als NAS,HTPC,server…

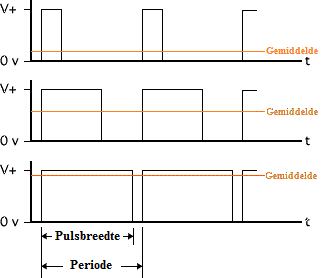
Door de kleine formfactor is de raspberry is de hardware dan ook beperkt in perfomantie. CPU: 750 MHz. RAM: 512MB. Dit wil zeggen dat er bij veel/zwaar rekenwerk toch wel enige wachttijd is eer deze is uitgevoerd. De I/O pinnen die aanwezig zijn op de printplaat kunnen worden gebruikt om externe componenten aan te sturen of in te lezen.

### Raspicam

Dit is een 5MP camera die geconnecteerd is met de raspberry pi via een CSI connector. Dit is momenteel de enige ondersteunde camera die via de CSI connector kan worden aangesloten. De enige andere optie is om een usb camera te gebruiken. De RaspiCam kan wat een webcam ook kan: foto’s trekken en filmen. De camera wordt aangestuurd via de shell van de Raspberry Pi. Dit via de commando’s: raspistill(foto) en raspivid(video).

Er kunnen extra parameters aan de commando’s worden meegegeven zodat Raspicam zich anders gedraagt.(filters, hoogte en breedte, fps…).

### Servo HS-311

Dit is een motor die gebruikt maakt van het servo principe met 3 draden(plus pool, min pool, stuursignaal). De servo vergelijkt het stuursignaal met een referentie signaal. Bij een verschil signaal zal de motor links of rechts draaien. Het signaal dat gebruikt is een PWM signaal. Hierbij wordt er een blok golf (enkel hoge of lage spanning) gegenereerd met een vaste frequentie. Enkel de duty cycle wordt aangepast, dit is de tijd dat het signaal 1(hoog) of 0(laag) is. De HS-311 heeft een voeding nodig van 5 volt en een stuursignaal die een blokgolf van 3 tot 5 volt betreft. De puls breedte voor het stuursignaal moet tussen 2400µs(rechts) en 600µs(links) liggen. De servo is aangesloten op de Raspberry Pi deze stuurt een PWM signaal naar de servo. Je kan de servo voeden door deze te connecteren met de Raspberry Pi maar helaas zal deze dan soms de raspberry laten vastlopen. Dit komt omdat de servo te veel stroom trekt. De oplossing hier voor is om de servo te voeden met een extra power supply.

# Functionaliteit

Het project bestaat uit een samenvoeging van bovenstaande onderdelen, eens de gebruiker het IP adres heeft ingesteld op de Raspberry Pi en in het client programma kan hij de te gebruiken poort kiezen. Deze kan gewijzigd worden indien gewenst of mag onveranderd blijven. Een gedetailleerde beschrijving van hoe je dit moet doen vind je in de technische documentatie.

Eens dit gebeurd is kan de gebruiker het programma eenvoudig opstarten, waarna het programma automatisch de stream zal ophalen die gestart word wanneer men de Raspberry Pi aanzet.

De aansturing van de servo, om de camera te laten draaien, gebeurt via onderstaande woordenlijst of de beschikbare lichaamsbewegingen:

Spraak:

* Links: left, view left, turn left.
* Midden: center, middle, focus.
* Rechts: right, view right, turn right.
* Sluiten: close, exit, stop.
* Minimaliseren: switch, minimalize, tab.

Lichaamsbewegingen:

* Links: je linkerhand een halve meter naar links bewegen op schouderhoogte.
* Rechts: je rechterhand een halve meter naar rechts bewegen op schouderhoogte.

# Code

## Client programma

Het client-programma is een WPF project en bestaat uit 4 klassen.  
**MainWindow.xaml** is waar alles begint, in het design vinden we de volgende code:

1. **<WindowsFormsHost** x:Name="winform" HorizontalAlignment="Left" Height="446" Margin="11,53,-56,-476" Grid.Row="3" VerticalAlignment="Top" Width="928"**/>**

Deze bovenstaande code vormt een venster op het xaml-form, dit venster vullen wij op met een VLC-plugin in **MainWindow.xaml.cs.**

1. vlc = **new** AxVLCPlugin();
2. winform.Child = vlc;

5. **string** uri = "http://192.168.1.26:8090";
6. vlc.addTarget(uri, **null**, AXVLC.VLCPlaylistMode.VLCPlayListAppendAndGo, 0)
7. vlc.play();

Als deze **MainWindow.xaml.cs** ingeladen wordt gebeuren er verschillende dingen:

1. TelnetConnection tc = **new** TelnetConnection("192.168.1.26", 6667);

We maken een connectie met de raspberry pi. Deze connectie zullen wij doorgeven aan elke functie die oproepen zal worden.

1. SenseMovement startSensing = **new** SenseMovement(tc);
2. startSensing.Start();

We starten het zoeken naar beweging met de kinect. Zoals vermeld geven wij onze telnetconnectie mee als parameter.

1. **using** (var memoryStream = **new** MemoryStream(Encoding.ASCII.GetBytes(Properties.Resources.SpeechGrammar)))
2. {
3. var g = **new** Grammar(memoryStream);
4. speechEngine.LoadGrammar(g);
5. }
7. speechEngine.SpeechRecognized += SpeechRecognized;

We laden een XML-file in met verschillende nodes, deze nodes worden omgezet naar een String in de custom functie **Properties.Resources.SpeechGrammar**. Waarna deze string wordt ingelezen als te herkennen woordenboek.

De **XML-file** met onze spraakwoordenboek ziet er als volgt uit:

1. **<grammar** version="1.0" xml:lang="en-US" root="rootRule" tag-format="semantics/1.0-literals" xmlns="http://www.w3.org/2001/06/grammar"**>**
2. **<rule** id="rootRule"**>**
3. **<one-of>**
4. **<item>**
5. **<tag>**CLOSE**</tag>**
6. **<one-of>**
7. **<item>** close **</item>**
8. **<item>** exit **</item>**
9. **<item>** stop **</item>**
10. **</one-of>**
11. **</item>**
12. **<item>**
13. **<tag>**SWITCH**</tag>**
14. **<one-of>**
15. **<item>** switch **</item>**
16. **<item>** minimise **</item>**
17. **<item>** tab **</item>**
18. **</one-of>**
19. **</item>**
20. **<item>**
21. **<tag>**CENTER**</tag>**
22. **<one-of>**
23. **<item>** center **</item>**
24. **<item>** middle **</item>**
25. **<item>** focus **</item>**
26. **</one-of>**
27. **</item>**
28. **<item>**
29. **<tag>**LEFT**</tag>**
30. **<one-of>**
31. **<item>** left **</item>**
32. **<item>** turn left **</item>**
33. **<item>** view left **</item>**
34. **</one-of>**
35. **</item>**
36. **<item>**
37. **<tag>**RIGHT**</tag>**
38. **<one-of>**
39. **<item>** right **</item>**
40. **<item>** turn right **</item>**
41. **<item>** view right **</item>**
42. **</one-of>**
43. **</item>**
44. **</one-of>**
45. **</rule>**
46. **</grammar>**

De **<tag></tag>** woorden zijn de woorden waarop we verder bouwen in onze code. De woorden onder **<item></item>** zijn de mogelijke voice-commands.

CLOSE zal het programma afsluiten  
SWITCH minimaliseert alles  
CENTER centreert de camera SM.TelnetChar("M");  
RIGHT stuurt de servomotor 1stap naar rechts. SM.TelnetChar("R");   
LEFT stuurt de servomotor 1stap naar links. SM.TelnetChar("L");

In de **senseMovement.cs** file zullen wij de zoeken naar een armbeweging op 50cm van de heup. Als deze beweging 15 seconden net terug gevonden wordt zal de camera zich bewegen naar zijn startpositie.

1. **public** SenseMovement(TelnetConnection tc)
2. {
3. **this**.tc = tc;
4. send = **new** SendMovement(tc);
5. }

Bij het aanmaken van deze klasse wordt er direct een nieuwe klasse aangemaakt om de movenment naar de raspberry pi te sturen. Deze connectie loopt over de telnetconnectie uit onze main klasse.

1. Thread t2 = **new** Thread(**delegate**()
2. {
3. send.TelnetChar("L");
4. });
5. **if** (handLeft.X <= hipCenter.X - 0.4)
6. {
7. t2.Start();
8. lastHit = DateTime.Now;
9. mSend = **false**;
10. }

De bovenstaande code bekijkt of het linkerhand zich 40cm van het midden van de heupen bevind. En zorgt ervoor dat in de komende 15 seconden de camera zicht niet automatisch centreert. Vervolgens zal dit code fragment een “L” versturen naar onze raspberry pi.

In de **sendMovement.cs** klasse versturen we het ontvangen charter over de ontvangen connectie. Ook loggen wij alles dat gebeurt zodat wij een overzicht hebben over alles dat gebeurt. Zo kunnen wij rapper fouten lokaliseren of onze output bekijken.

1. **public** SendMovement(TelnetConnection tc) {
2. **this**.tc = tc;
3. }
5. /// <summary>
6. /// Telnet connection
7. /// </summary>
8. **public** **void** TelnetChar(**string** charToSend)
9. {
10. **string** prompt = "";
11. // while connected
12. **if** (tc.IsConnected)
13. {
14. prompt = charToSend;
15. tc.WriteLine(prompt);
16. **string** temp = tc.Read();
17. }
19. **using** (StreamWriter w = File.AppendText("logFile.txt"))
20. {
21. Log(charToSend + " send to raspberry through telnet.", w);
22. }
23. Thread.Sleep(100);
25. //  500 is 5sec
26. //  25 is 0.25sec
27. }
29. **public** **static** **void** Log(**string** logMessage, TextWriter w)
30. {
31. // w.Write("\nLog Entry : ");
32. w.WriteLine("\n{0}", DateTime.Now);
33. w.WriteLine("\r : {0}", logMessage);
34. }

**Telnetinterface.cs** is waar de connectie maken met de socket op de rasperry pi.

1. **public** **void** Write(**string** cmd)
2. {
3. **if** (!tcpSocket.Connected) **return**;
4. **byte**[] buf = System.Text.ASCIIEncoding.ASCII.GetBytes(cmd.Replace("\0xFF","\0xFF\0xFF"));
5. tcpSocket.GetStream().Write(buf, 0, buf.Length);
6. }

## Raspberry Pi

### Python code

Omzetten van percentages naar puls breedte.

1. **def** calculatePulse(percentage):
3. **return** 700 + (1800\*percentage/100)

Aansturen van de motor. Men geeft als parameter in deze functie de richting mee.(0=links,1=rechts,2=midden) berekent aan de hand van deze parameter een nieuwe puls lengte uit.

1. **def** controlMotor(direction):
2. **global** percentage
4. **if** direction == 0:
5. percentage-=3
6. **elif** direction == 1:
7. percentage = 50
8. **elif** direction == 2:
9. percentage+=3
10. **if** percentage <0:
11. percentage = 0
12. **elif** percentage >100:
13. percentage = 100
14. pulse = int(round(calculatePulse(percentage)/10.0))\*10
15. servo.set\_servo(18,pulse)
16. **print**(pulse)
17. **return**

De client zal karakters naar de raspberry sturen. Dit gebeurt via LAN. Je ontvangt de data van de client. Hierna vergelijk je de data die de client stuurt met de vooraf ingestelde karakters voor het aansturen van de motor. Als de data gelijk is aan een vooraf ingesteld karakter dan zal de motor aangestuurd worden. Als de client geen data stuurt dan zal de connectie met de client afgesloten worden.

Opstarten van de stream aan de hand van een shell commando. Dit wil zeggen dat de stream draait in de shell.

1. **def** startStream ():
3. commando = "sudo -u pi bash -c \"raspivid -o - -t 0 -w 640 -h 460 -hf -vf -fps 25|cvlc â€“vvv stream:///dev/stdin --sout '#standard{access=http,mux=ts,dst=:8090}' --no-audio :demux=h264\""
4. call ([commando], shell=True)
5. #Function for handling connections. This will be used to create threads
6. **def** clientthread(conn):
7. #Sending message to connected client
8. conn.send('Welcome to the server. Type something and hit enter\n\r') #send only takes string
9. #infinite loop so that function do not terminate and thread do not end.
10. **while** True:
12. #Receiving from client
13. data = conn.recv(1024)
14. data = data.rstrip('\r\n')
16. **if** data == "L" **or** data == "l":
17. reply = '\n\rje hebt L gestuurd\n\r'
18. controlMotor(0)
19. **elif** data == "R" **or** data == "r":
20. reply = "\n\rje hebt R gestuurd\n\r"
21. controlMotor(2)
22. **elif** data == "M" **or** data == "m":
23. reply = "\n\rje hebt M gestuurd\n\r"
24. controlMotor(1)
25. **else** :
26. reply = '\n\rserver:' + data + ' is geen geldige invoer\n\r'
28. **if** **not** data:
30. **break**
32. conn.sendall(reply)
34. #came out of loop
35. **print** 'Connection closed'
36. conn.close()

De socket zal wachten tot een client connectie maakt. Als er een connectie gemaakt word zal er een nieuwe thread gestart worden voor de data te ontvangen en zo de motor aan te sturen.

1. **while** 1:
2. #wait to accept a connection - blocking call
3. conn, addr = s.accept()
4. **print** 'Connected with ' + addr[0] + ':' + str(addr[1])
5. #start new thread takes 1st argument as a function name to be run, second is the tuple of arguments to the function.
6. start\_new\_thread(clientthread ,(conn,))

# Demo

U kan een demo bezichtigen via onderstaande link.

<https://www.dropbox.com/s/znq5n32g4fa2bud/demo.mp4>

# Conclusie

Na het beëindigen van het project kunnen we enkele conclusies trekken over de gebruikte onderdelen het werken van deze onderdelen apart en met elkaar. De verschillende onderdelen zijn vrij innovatief en nieuw, waardoor het een leuke uitdaging is om er een project rond te maken.

## Kinect

De sensor is een krachtig en veelzijdig apparaat dat voor tal van applicaties kan worden ingezet en gebruikt. Wij hebben hiervan maar een tipje van de ijsberg gebruikt. De basis voor spraak en bewegingen hebben we nu onder de knie, al was dit niet zo simpel als in eerste instantie was gedacht. Vooral de spraak juist in het programma krijgen heeft voor ons voor kopzorgen gezorgd maar eens dit erin zit was het snel uit te breiden met verschillende commando’s.

## Raspberry Pi

De computer op kredietkaart formaat kan bijna alles zoals een standaard computer, al heeft de Pi zijn beperkingen op gebied van snelheid. Voor CPU intensieve taken dien je geduld te hebben, vaak zagen we tijdens het browsen of compileren de load balk van de processor naar 100% schieten.

Maar dit belet je niet op de Pi te gebruiken in tal van toepassingen waarvoor een normale computer veel te krachtig zou zijn.

## Raspicam

De kleine camera is zeer handig om te gebruiken in combinatie met de Raspberry Pi. Door zijn formaat en mogelijkheden, HD filmen en foto’s nemen, is deze camera zeer handig in combinatie met de Raspberry Pi.

Deze camera kent wel een groot nadeel, tijdens het project is een camera er plots mee opgehouden. Dit had te maken met zijn gevoeligheid voor elektrostatisch schok.