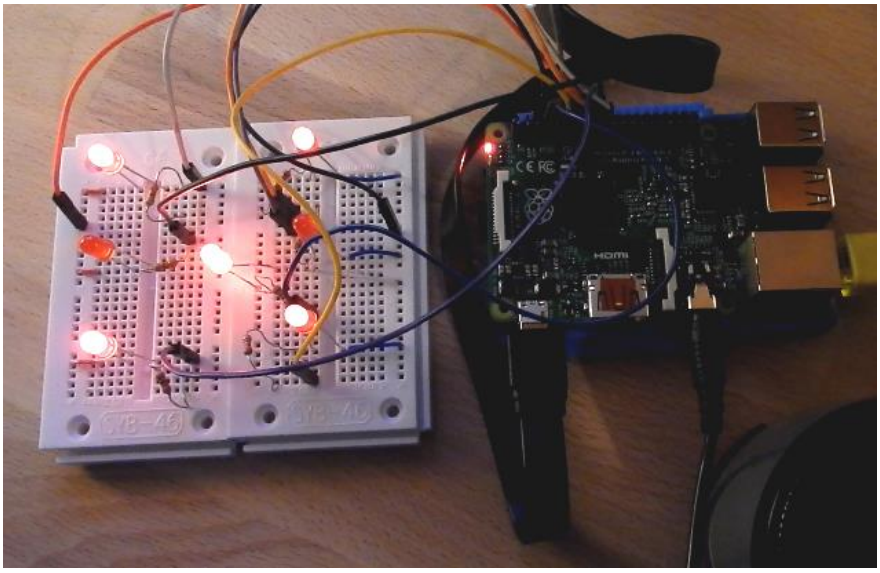
DOBBELSTEEN 2.0

Stap-voor-stap instructie



1. Project

Een dobbelsteen met LED-jes en gesproken tekst, aangestuurd door de Raspberry Pi, geprogrammeerd met Python 2.7.

2. Benodigdheden

- Raspberry Pi (2 model B / 3 model B)
- Luidspreker met 3,5mm plug
- 7 x LED
- 7 x weerstand 220Ω
- 2 x breadbord (bijv. SYB-46)
- 9 x GPIO jumper wire male/female
- 9 x U-shaped jumper kabeltjes male/male (voor breadboard)

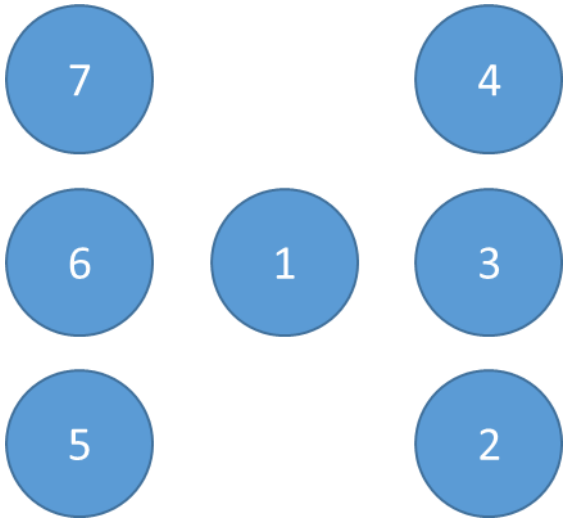
3. Belangrijkste vereisten

De Raspberry Pi moet voorzien zijn van Tekst To Speech software. Zie hier voor een stap-voor-stap instructie:

http://lekkerprutsen.nl/laat-je-raspberry-pi-praten/

4. Bouw de hardware

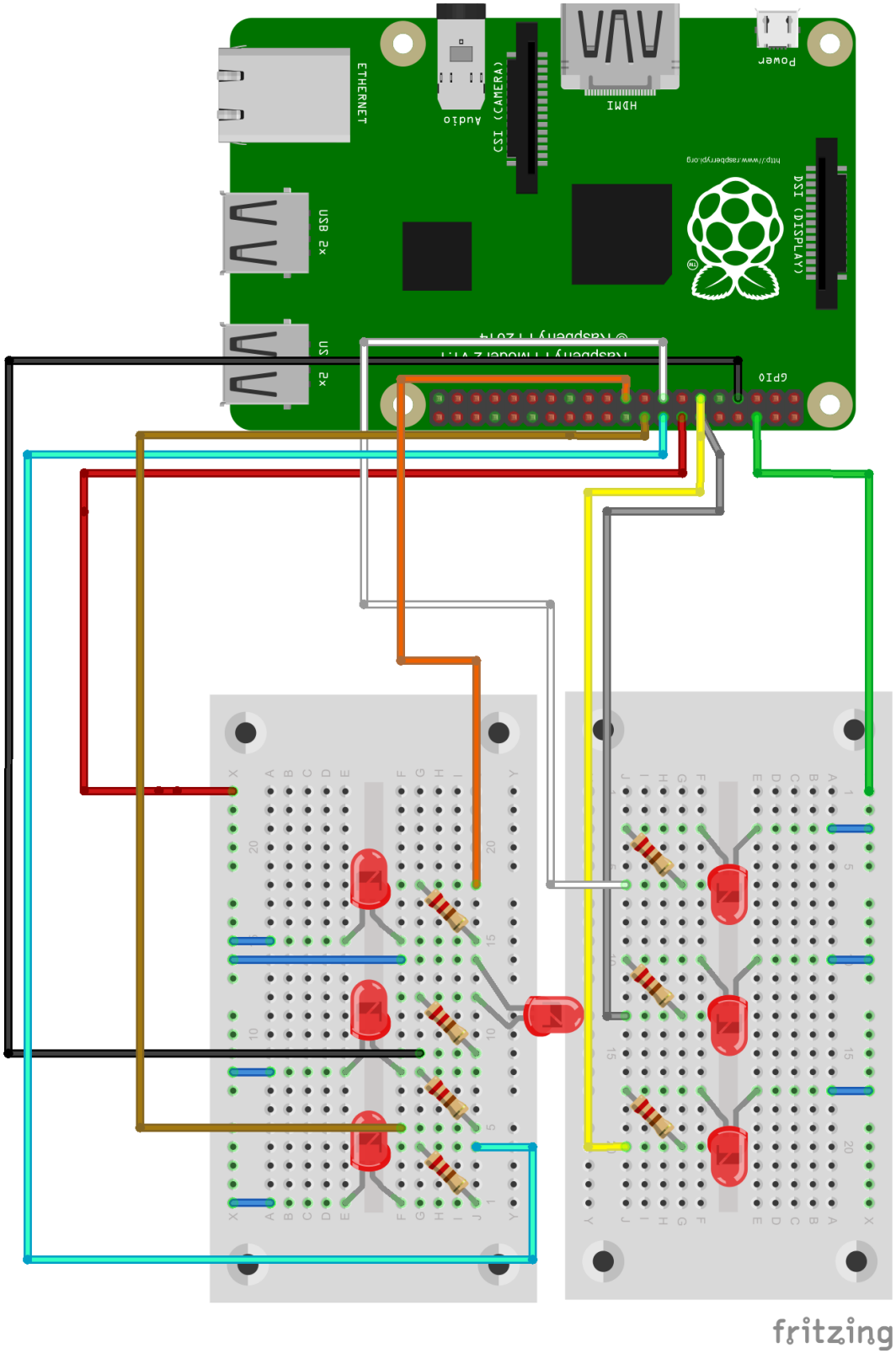
De dobbelsteen bestaat uit 7 LED's die als volgt geplaatst worden op het breadboard:



De LED's worden op de volgende GPIO poort aangesloten:

LED	GPIO
1	4
2	17
3	18
4	22
5	23
6	24
7	10

De componenten worden volgens onderstaand schema op de breadboards geplaatst en aangesloten op de GPIO poorten van de Raspberry Pi:



Figuur 1 Aansluitschema gemaakt met FRITZING (www.fritzing.org)

5. Code

```
# -*- coding: utf-8 -*-

import pyttsx
engine=pyttsx.init()

import pygame, sys, random
from pygame.locals import *
pygame.init()

import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)

LED = [4,17,18,22,23,24,10]
for i in LED:
    GPIO.setup(i, GPIO.OUT, initial=False)

FIELD = pygame.display.set_mode((320,320))
pygame.display.set_caption("dobbelsteen")

YELLOW = (255,255,0); GREEN = (0,117,30)
p1 = ((160,160)); p2 = ((60,60)); p3 = ((160,60));
p4 = ((260,60))
p5 = ((60,260)); p6 = ((160,260)); p7 = ((260,260))
mainloop = True

print "Druk een willekeurige toets in, om de dobbelsteen te gooien, [esc] beendigt het spel"

while mainloop:
    for event in pygame.event.get():
        if event.type == QUIT or (event.type == KEYUP
and event.key == K_ESCAPE):
            mainloop = False
        if event.type == KEYDOWN:
            FIELD.fill(YELLOW)
            SCORE = random.randrange (1,7); print SCORE

            if SCORE == 1:
                for i in LED:
                    GPIO.output(i, False)

                pygame.draw.circle(FIELD,GREEN,p1,40)
                GPIO.output(LED[0],True)

                engine.say('One')
                engine.runAndWait()

            if SCORE == 2:
                for i in LED:
                    GPIO.output(i, False)

                GPIO.output(LED[6],True)
                GPIO.output(LED[6],True)

                pygame.draw.circle(FIELD,GREEN,p2,40)
                pygame.draw.circle(FIELD,GREEN,p7,40)

                GPIO.output(LED[6],True)
                GPIO.output(LED[1],True)

```

```
61                  engine.say('Two')
62                  engine.runAndWait()
63
64
65              if SCORE == 3:
66                  for i in LED:
67                      GPIO.output(i, False)
68
69                  GPIO.output(LED[0],True)
70                  GPIO.output(LED[3],True)
71                  GPIO.output(LED[4],True)
72
73                  pygame.draw.circle(FIELD,GREEN,p1,40)
74                  pygame.draw.circle(FIELD,GREEN,p4,40)
75                  pygame.draw.circle(FIELD,GREEN,p5,40)
76
77                  engine.say('Three')
78                  engine.runAndWait()
79
80              if SCORE == 4:
81                  for i in LED:
82                      GPIO.output(i, False)
83
84                  GPIO.output(LED[1],True)
85                  GPIO.output(LED[3],True)
86                  GPIO.output(LED[4],True)
87                  GPIO.output(LED[6],True)
88
89                  pygame.draw.circle(FIELD,GREEN,p2,40)
90                  pygame.draw.circle(FIELD,GREEN,p4,40)
91                  pygame.draw.circle(FIELD,GREEN,p5,40)
92                  pygame.draw.circle(FIELD,GREEN,p7,40)
93
94                  engine.say('Four')
95                  engine.runAndWait()
96
97              if SCORE == 5:
98                  for i in LED:
99                      GPIO.output(i, False)
100
101                 GPIO.output(LED[1],True)
102                 GPIO.output(LED[3],True)
103                 GPIO.output(LED[4],True)
104                 GPIO.output(LED[6],True)
105                 GPIO.output(LED[0],True)
106
107                 pygame.draw.circle(FIELD,GREEN,p1,40)
108                 pygame.draw.circle(FIELD,GREEN,p2,40)
109                 pygame.draw.circle(FIELD,GREEN,p4,40)
110                 pygame.draw.circle(FIELD,GREEN,p5,40)
111                 pygame.draw.circle(FIELD,GREEN,p7,40)
112
113                 engine.say('Five')
114                 engine.runAndWait()
115
116
117             if SCORE == 6:
```

```
118                 for i in LED:
119                     GPIO.output(i, False)
120
121                 GPIO.output(LED[1],True)
122                 GPIO.output(LED[2],True)
123                 GPIO.output(LED[3],True)
124                 GPIO.output(LED[4],True)
125                 GPIO.output(LED[5],True)
126                 GPIO.output(LED[6],True)
127
128                 pygame.draw.circle(FIELD,GREEN,p2,40)
129                 pygame.draw.circle(FIELD,GREEN,p3,40)
130                 pygame.draw.circle(FIELD,GREEN,p4,40)
131                 pygame.draw.circle(FIELD,GREEN,p5,40)
132                 pygame.draw.circle(FIELD,GREEN,p6,40)
133                 pygame.draw.circle(FIELD,GREEN,p7,40)
134                 engine.say('Six')
135                 engine.runAndWait()
136
137     pygame.display.update()
138
139 pygame.quit()
140 GPIO.cleanup()
141
```

6. Kijk, zo werkt het!

`3 import pyttsx` `4 engine=pyttsx.init()`	In dit project combineren we meerdere functionaliteiten. Met de regel *`import pyttsx`* importeren we de pyttsx bibliotheek die nodig is voor "Text To Speech" functionaliteit. Met *`engine=pyttsx.init()`* initialiseren we deze functie. Daarmee maken we de functie klaar voor het ontvangen van opdrachten.

`6 import pygame, sys, random`
`7 from pygame.locals import *`
`8 pygame.init()`
	De pygame bibliotheek stelt ons in staat om grafische elementen toe te voegen aan ons programma. Tevens laden we de system en random bibliotheek. Die laatste is nodig voor het genereren van de random score van de dobbelsteen.

`10 import RPi.GPIO as GPIO`
`11 import time`
`12 GPIO.setmode(GPIO.BCM)`
	Om de GPIO poorten van de Raspberry Pi aan te sturen, moet de GPIO bibliotheek geladen worden met de regel import *`RPi.GPIO as GPIO`*. Met *`GPIO.setmode(GPIO.BCM)`* geven we aan dat we gebruik maken van de BCM methode om de GPIO poorten mee aan te duiden.

`14 LED = [4,17,18,22,23,24,10]`	Met *`LED=(4,17,18,22,23,24,10)`* maken we een lijst aan waarin de GPIO's en daarmee de aangesloten LED's in de juiste volgorde staan. We houden daarbij de volgorde aan uit de tabel op pagina 2.

15 for i in LED: 16 GPIO.setup(i, GPIO.OUT, initial=False)	Met deze *for* lus wordt voor elke waarde in de lijst LED (dus voor alle 7) de GPIO poort geïnitialiseerd. Met de waarde *false* zetten we dus alle LED's uit bij aanvang van het programma.
18 FIELD = pygame.display.set_mode((320,320)) 19 pygame.display.set_caption("dobbelsteen")	Hiermee maken we het scherm *FIELD* aan met dat 320 bij 320 pixels groot is. Het scherm krijgt de naam "dobbelsteen".
21 YELLOW = (255,255,0); GREEN = (0,117,30)	In het programma willen we gebruik maken van de kleuren Yellow en Green. Hier stellen we deze kleuren in met behulp van een RGB code.
22 p1 = ((160,160)); p2 = ((60,60)); p3 = ((160,60)); 23 p4 = ((260,60)) 24 p5 = ((60,260)); p6 = ((160,260)); p7 = ((260,260))	De ogen van de dobbelstenen kennen 7 mogelijke posities in het scherm *FIELD*. Met deze regels geven we deze posities aan. Bijvoorbeld `p4 = ((260,60))` geeft aan dat het middelpunt 4 ligt op 260 pixels naar rechts en 60 pixels naar beneden.
25 mainloop = True	Deze regel geeft bij aanvang van het spel de hoofdlus de waarde *true*.
27 print "Druk een willekeurige toets in, om de dobbelsteen te gooien, [esc] beendigt het spel"	Hiermee wordt de instructie voor de gebruiker op het scherm getoond.
29 while mainloop: 30 for event in pygame.event.get(): 31 if event.type == QUIT or (event.type == KEYUP 32 and event.key == K_ESCAPE): 33 mainloop = False	Met *while mainloop* begint de hoofdlus die dus altijd de waarde True, heeft behalve als: - de gebruiker het speelscherm sluit → `event.type == QUIT` - of als de gebruiker op de escape toets drukt → `event.key == K_ESCAPE` ..dan krijgt de hoofdlus de waarde *false* en stop het programma.
34 if event.type == KEYDOWN: 35 FIELD.fill(YELLOW) 36 SCORE = random.randrange (1,7); print SCORE	`if event.type == KEYDOWN:` → als de gebruiker een willekeurige toets drukt (m.u.v. esc) dan.. `FIELD.fill (YELLOW)` → maak het speelscherm geel `SCORE = random.randrange (1,7)` → bepaal dan random een score die ligt in de reeks van 1 tot 7 en… `print SCORE` → toon de score op het scherm
65 if SCORE == 3:	Het programma geeft voor elke mogelijke score aan welke actie er moet volgen. Als voorbeeld bekijken we de acties die volgen als de `SCORE == 3`
66 for i in LED: 67 GPIO.output(i, False)	Hiermee zetten we alle LED's uit. Bij aanvang van het programma staan ze al uit, maar na elke dobbelbeurt moet dit opnieuw gebeuren.
69 GPIO.output(LED[0],True) 70 GPIO.output(LED[3],True) 71 GPIO.output(LED[4],True)	Bij score 3 moeten de LED's 1, 4 en 5 (zie schema p2) aangaan. Dat zijn dus de eerste, vierde en vijfde waarde in de reeks LED. Echter, in een reeks wordt begonnen met de

	waarde 0. We hebben dus de waarde 0, 3 en 4 uit deze reeks nodig.
73 pygame.draw.circle(FIELD,GREEN,p1,40) 74 pygame.draw.circle(FIELD,GREEN,p4,40) 75 pygame.draw.circle(FIELD,GREEN,p5,40)	In het speelscherm moet bij score 3 de ogen 1, 4 en 5 getoond worden. Met *pygame.draw.circle(FIELD, GREEN,p1,40)* tonen we een circle in het scherm FIELD, met de eerder gedefinieerde kleur GREEN, op de eerder gedefinieerde postitie p1 met een radius van 40 pixels.
77 engine.say('Three') 78 engine.runAndWait()	Met *engine.say('Three')* laten we de Text To Speech functionaliteit het woord "Three" uitspreken. Met *engine.runAndWait()* wordt de spraakfunctie gestart en andere processen tijdelijk gestopt.
137 pygame.display.update()	Met deze regel aan het einde van de lus zorgen we ervoor dat de ogen van de dobbelsteen getoond getoond worden.
139 pygame.quit() 140 GPIO.cleanup()	Aan het eind van het programma wordt met *pygame.quit()* het speelscherm gesloten. Met *GPIO.cleanup()* worden de geopende GPIO poorten gesloten.