

Internet of Things

Surveillance camera
met motion detection
en notificaties

New media development 2020-2021

Team

Dante Weverbergh

Jonas Di Dier

Kobe Devillé



inhoudsopgave

Discover	5
- briefing	5
- uitleg concept	5
Define	6
- Functionele vereisten	6
- Niet functionele vereisten	6
- Noodzakelijke hardware	6
- Noodzakelijke software	6
Design	7-9
- adobe Xd	7
-mockup	8-9
Develop	10-21
- Woordje uitleg	10
- frontend	10-13
- database	14-17
- backend	18-21
- hardware	21
Deliverables	22-25
- handleiding	22-25
- timesheet	25

Discover

Briefing

De student moet een IoT creatie maken, gebruik makende van aangeleerde en nieuwe / trending technologieën.

Deadline werkstuk?

maandag 7 juni 20:00

Uitleg concept

Ons project bestaat uit een (bewakings)camera die een notificatie met foto stuurt wanneer beweging gedetecteerd wordt. Na detectie van beweging start een videoopname, die beëindigd en geüpload zal worden zodra de beweging stopt. Via het dashboard kan de gebruiker de tijdlijn bekijken van alle opnames. Het dashboard kan ook gebruikt worden om de camera in of uit te schakelen, en om de instellingen aan te passen.

Define

We bieden de nodige code en documentatie aan zodat een gebruiker zelf een surveillance camera systeem kan opstellen. Hiernaast is er ook een webapplicatie voorzien die de gebruiker met een eigen Firebase project moet verbinden, om zo het dashboard te kunnen gebruiken.

Functionele vereisten

- Beweging detecteren met camera
- Foto's en video's maken
- Foto's en video's uploaden naar Firebase
- Push notificaties versturen
- Bestanden ouder dan een week automatisch verdwijnen
- Camera & led in en uit kunnen schakelen via het dashboard
- Tijdslijn met opnamen kunnen bekijken
- Bestanden markeren die niet gewist mogen worden.
- Instellingen aanpassen (gevoeligheid, tijd tussen groepen)

Niet functionele vereisten

- Beweging detecteren in een omgeving met weinig licht
- responsive applicatie
- snelle laadtijden van foto's en video's in het dashboard
- Eenvoudige user interface.

Noodzakelijk hardware

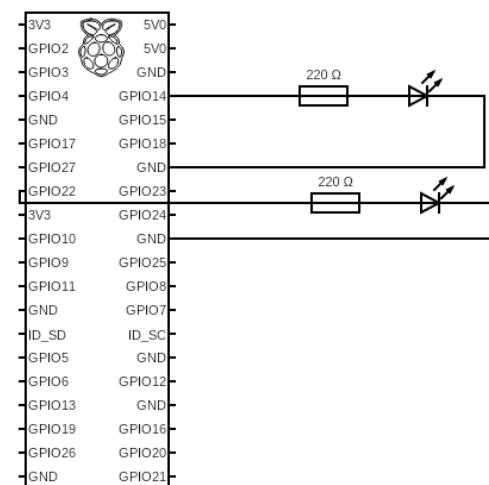
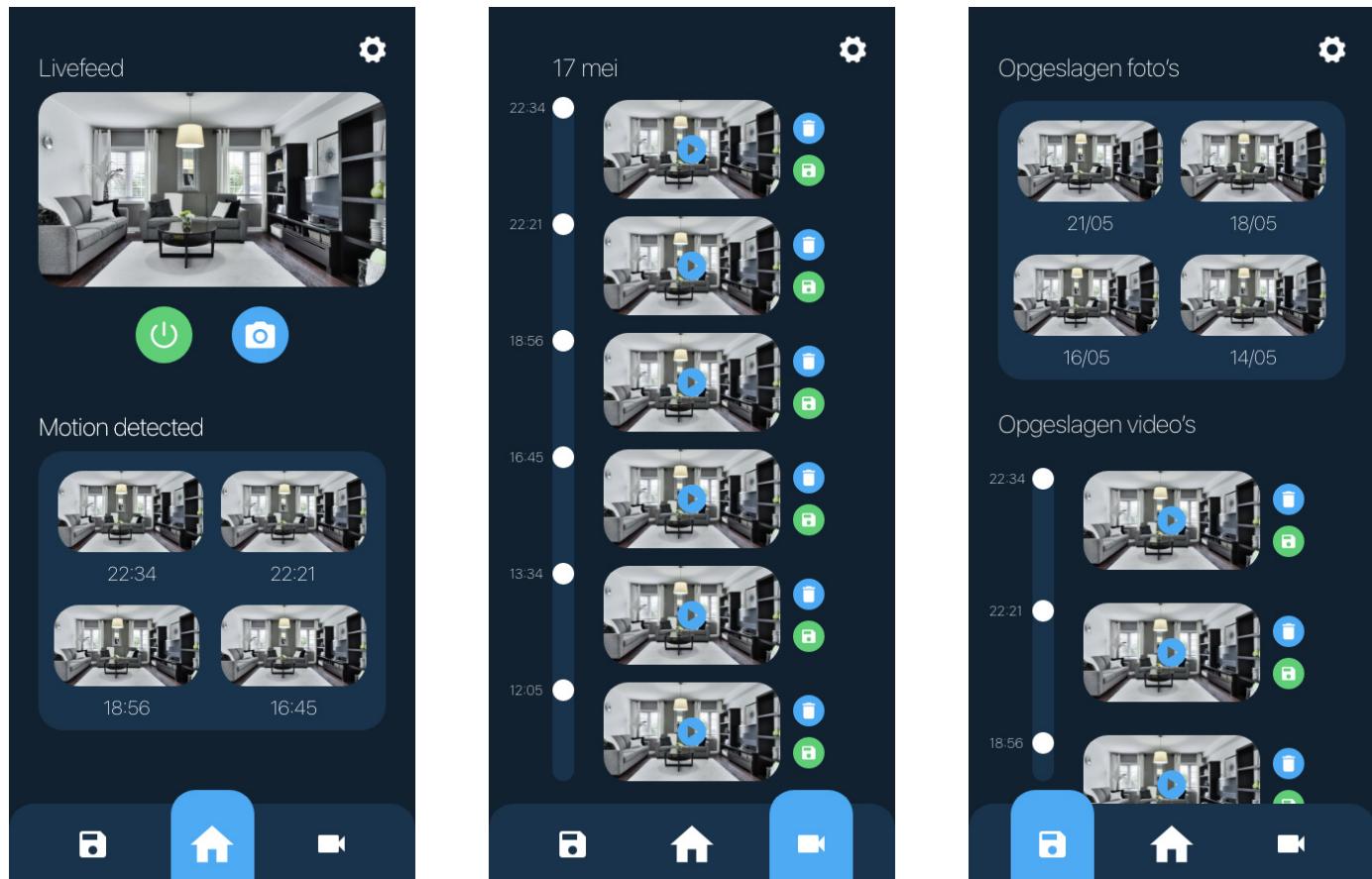
- Raspberry Pi
- Camera module
- Led lichtje
- breadboard met jumper kabels en weerstand(en)
- weerstand
- smarthpone

Noodzakelijke software

- RaspberryOs
- Python 3
- moderne browser met Push API support (<https://caniuse.com/push-api>)
(Voor de ios gebruikers, notificaties werken niet binnen safari)

Design

Design in ADOBE XD + mockup



Design

Design in applicatie

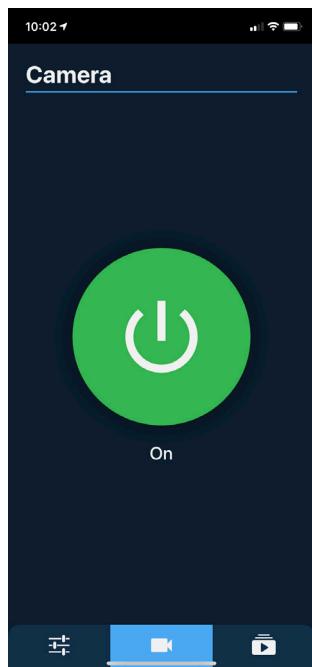


Inlogscherm

Inloggen met volgende gegevens

Email: *user@iot.com*

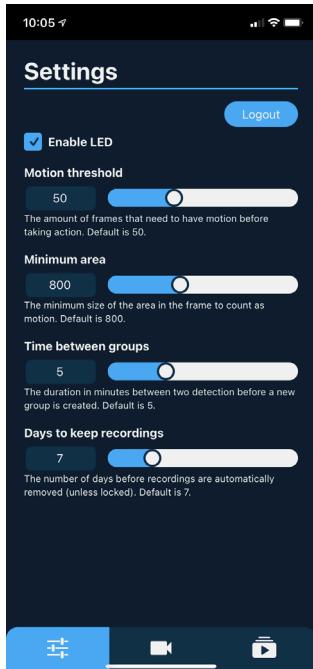
Password: *password*



Home dashboard

Hier kan de camera in en uit geschakeld worden.

Design



Settings pagina

Hier kunnen de instellingen worden aangepast.

Zoals:

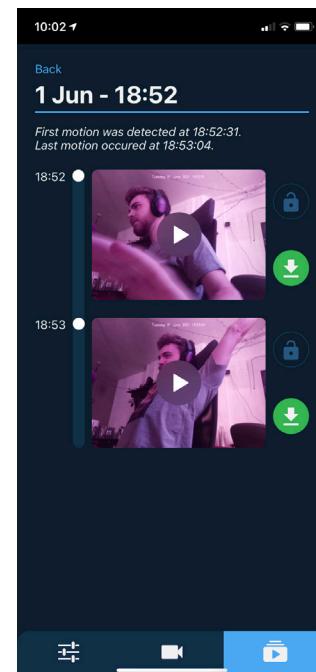
- *Led aan en uitzetten*
- *Motion threshold aanpassen (aantal frames beweging voor de camera begint te recorden).*
- *Minimum area (Hoeveel oppervlakte van het beeld moet bedekt zijn voor de camera gaat recorden).*
- *Time between groups (Tijd tussen 2 recordings voor er een nieuwe groep vormt).*
- *Days to keep recording (dagen dat recordings worden behouden.)*



Tijdslijn

Hier kunnen de beelden bekijken worden.

Je kan de beelden hier ook downloaden via de groene knop. Of vastzetten op de tijdlijn zodat ze niet automatisch verwijderd worden wanneer ze ouder zijn dan 1 week.



Develop

Frontend

Het dashboard bestaat uit een React PWA (progressive web app) en maakt gebruik van de React Router en React Query library. Onderstaand fragment is een voorbeeld van hoe data opgehaald en weergegeven wordt in een component.

```
const TimelineDetail = () => {
  const { id } = useParams();
  const { data: motionMoment, ...query } = useMotionMoment(id);

  if (query.isLoading) {
    return <Spinner />;
  }

  if (query.isError) {
    return <Alert color="danger">{query.error.message}</Alert>;
  }

  return (
    <>
    {/* ... */}

    <Title>{formatDate(motionMoment.id, 'D MMM - H:mm')}</Title>

    {/* ... */}

    <section className="videos">
      {motionMoment.recordings.map((recording) => (
        <VideoCard
          key={recording.id}
          recording={recording}
          motionId={motionMoment.id}
        />
      )));
    </section>
  </>
);
};
```

Develop

De webapplicatie wordt gehost via Firebase Hosting en integreert Firebase Authentication, Firestore, Storage, en Cloud Messaging. Firebase Authentication wordt gebruikt om een vooraf ingestelde gebruiker te authenticeren, zodat de applicatie niet toegankelijk is voor het publiek.

Voorbeeld Login component:

```
const handleSubmit = async (e) => {
  e.preventDefault();

  const formData = new FormData(e.target);
  const email = formData.get('email');
  const password = formData.get('password');

  try {
    await firebase.auth().signInWithEmailAndPassword(email, password);
  } catch (error) {
    setError(error);
  }
};
```

Als database hebben we de Firestore van Firebase gebruikt, de Firestore bevat al de documenten van gedetecteerde bewegingen, alsook een document waar alle instelling van de applicatie worden opgeslagen.

Voorbeeld React Query custom hook:

```
const getSettings = async () => {
  const db = firebase.firestore();
  const settings = await db.collection('app').doc('settings').get();
  return settings.data();
};

const useSettings = () => {
  return useQuery('settings', getSettings, { staleTime: 5 * 1000 });
};
```

Develop

Alle foto's en video's worden opgeslagen in Firebase Storage. De documenten in Firestore verwijzen naar deze locatie.

Voorbeeld React Query custom hook:

```
const getStorageURL = async (path) => {
  return path ? firebase.storage().ref(path).getDownloadURL() : '';
};

const useStorageURL = (path) => {
  return useQuery(
    [path?.split('/')[0], path],
    () => {
      return getStorageURL(path);
    },
    { staleTime: Infinity }
  );
};
```

Develop

Firebase Cloud Messaging wordt gebruikt om push notificaties te sturen naar de gebruiker. Bij het openen van de applicatie wordt indien nodig toestemming gevraagd, waarna de verkregen token opgeslagen wordt in Firestore.

Voorbeeld custom hook:

```
const useRegistrationToken = () => {
  // ...

  useEffect(() => {
    const getToken = async () => {
      // Get permission
      const permission = await Notification.requestPermission();

      if (permission === 'granted') {
        try {
          // Get registration token
          const messaging = firebase.messaging();
          const currentToken = await messaging.getToken({
            vapidKey: 'xxxx-xxxx-xxxx-xxxx-xxxx',
          });

          if (currentToken) {
            // ...

            // Send token to firestore
            const db = firebase.firestore();
            const settingsRef = db.collection('app').doc('settings');
            await settingsRef.get();
            settingsRef.update({
              registrationTokens:
                firebase.firestore.FieldValue.arrayUnion(currentToken),
            });
          } else {
            // ...
          }
        } catch (err) {
          // ...
        }
      } else {
        // ...
      }
    };
  });

  getToken();
}, []);

// ...
};
```

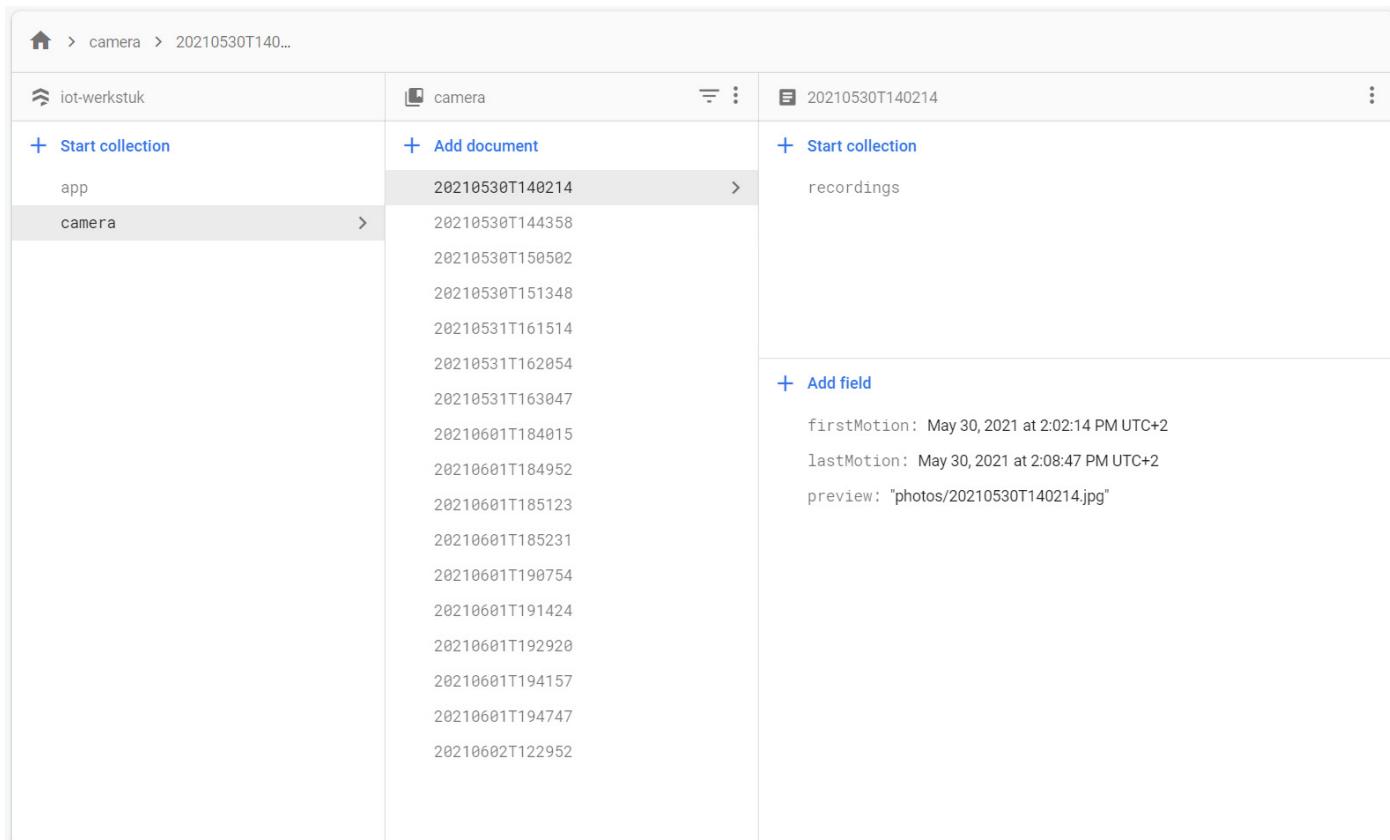
Develop

Database

Wanneer beweging gedetecteerd wordt, zal dit opgeslagen worden in een firebase document. Dit document bevat de velden

- *firstMotion*,
- *lastMotion*
- *preview*

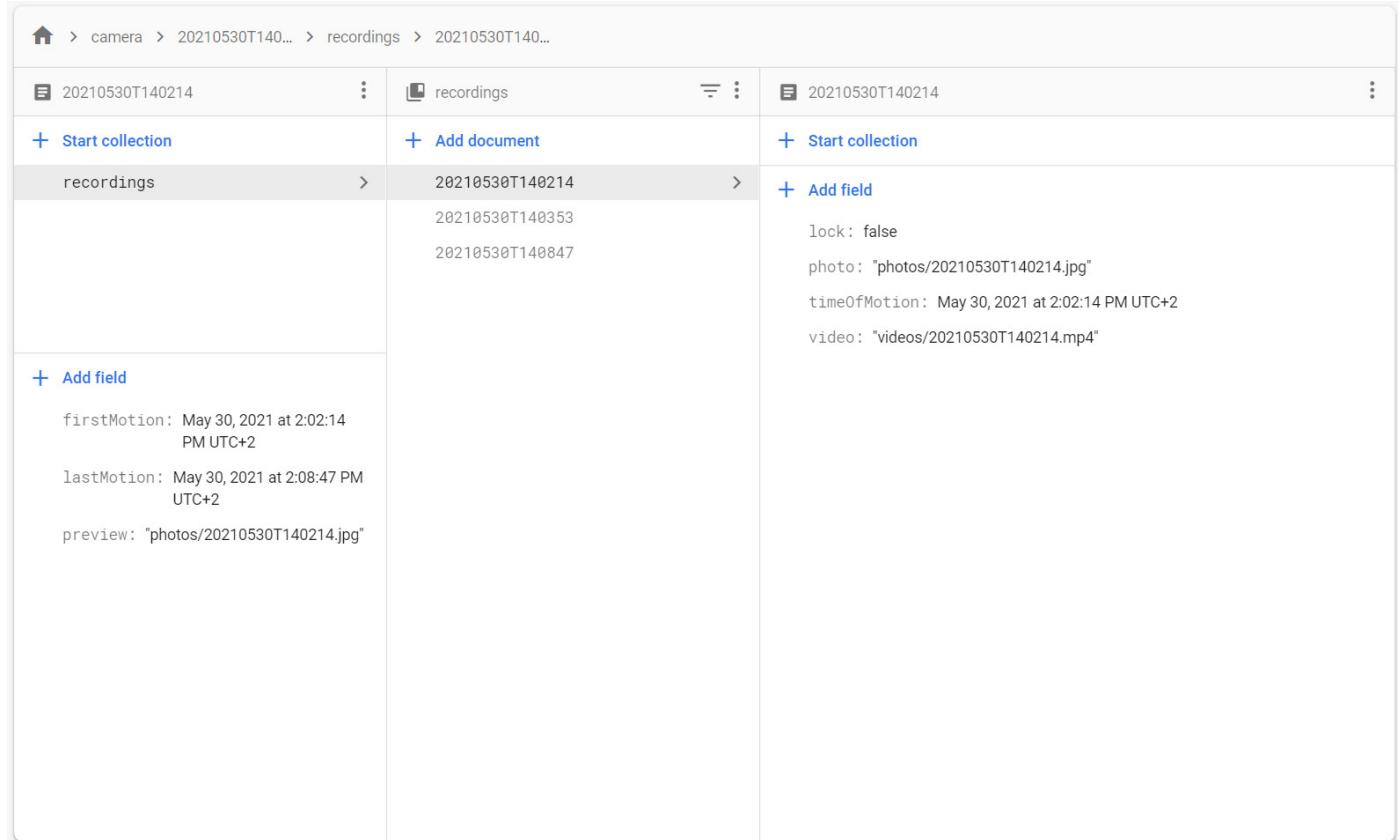
Verder bevat dit document een subcollectie genaamd *recordings* met informatie over een opname. Zolang het tijdstip van detectie dicht genoeg ligt bij de laatste detectie, zal de opname toegevoegd worden aan de *recordings* subcollectie van hetzelfde document.



The screenshot shows the Firebase Realtime Database interface. The path in the top navigation bar is: [Home](#) > [camera](#) > [20210530T140214...](#). The left sidebar shows a collection named **iot-werkstuk** with a subcollection **app** and a document **camera** (which is selected and highlighted in grey). The main content area shows a list of documents under the **camera** document, with the first document **20210530T140214** selected. The right panel shows the details for this selected document, including a subcollection named **recordings**. Below the recordings, there is a section for adding fields, with three fields defined: **firstMotion** (May 30, 2021 at 2:02:14 PM UTC+2), **lastMotion** (May 30, 2021 at 2:08:47 PM UTC+2), and **preview** (a URL to a photo file: "photos/20210530T140214.jpg").

Develop

De recordings subcollectie bevat verwijzingen naar de foto en video in de Storage. Het lock veld zorgt ervoor dat een opname niet automatisch gewist wordt na een bepaalde tijd.



The screenshot shows the MongoDB interface with a tree view of a database structure. The root node is 'recordings'. Under 'recordings', there is a node '20210530T140214' which contains the following fields:

- lock:** false
- photo:** "photos/20210530T140214.jpg"
- timeOfMotion:** May 30, 2021 at 2:02:14 PM UTC+2
- video:** "videos/20210530T140214.mp4"



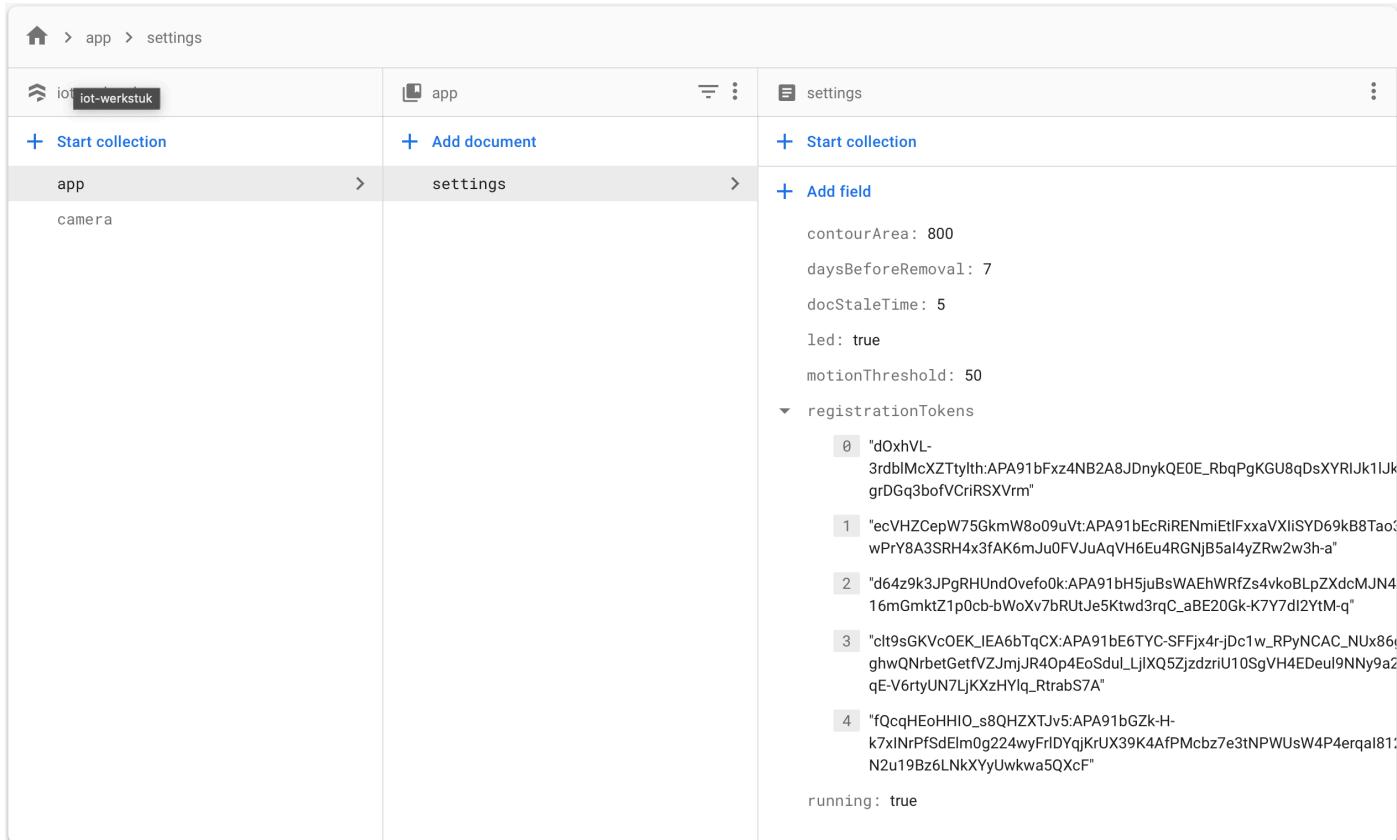
"gelocked", deze video zal niet automatisch verwijderd worden.

"downloaden", Download de video naar je fotorol.

"unlocked", Deze video wordt na een tijdje automatisch verwijderd.

Develop

De app collectie bevat één enkel document met alle configurerbare instellingen en de lijst van tokens die nodig zijn voor het verzenden van push notificaties.



The screenshot shows the MongoDB Compass interface with the following structure:

- Collection:** app
- Document:** settings
- Fields:**
 - contourArea: 800
 - daysBeforeRemoval: 7
 - docStaleTime: 5
 - led: true
 - motionThreshold: 50
 - registrationTokens (array):
 - 0: "dOxhVL-3rdBIMcXZTtyIth:APA91bFxz4NB2A8JDnykQE0E_RbqPgKGU8qDsXYRIJk1IjkgrDGq3bofVCriRSVrm"
 - 1: "ecVHZCepW75GkmW8o09uVt:APA91bEcRiRENmiEtIFxxaVXliSYD69kB8TaowPrY8A3SRH4x3fAK6mJu0FVJuAqVH6Eu4RGNjb5aI4yZRw2w3h-a"
 - 2: "d64z9k3JPgRHUndOvefo0k:APA91bH5juBsWAEhWRFzs4vkoBLpZXdcMJN416mGmktZ1p0cb-bWoXv7bRUtJe5Ktwd3rqC_aBE20Gk-K7Y7di2YtM-q"
 - 3: "clt9sGKV0EK_lEA6bTqCX:APA91bE6TYC-SFFjx4r-jDc1w_RPyNCAC_NuX86ghwQNbctGetfVZJmjJR40p4EoSduL_LjIXQ5JzdriU10SgVH4EDeuI9NNy9a2qE-V6rtyUN7LjKXzHYlq_RtrabS7A"
 - 4: "fQcqHEoHHIO_s8QHZXTJv5:APA91bGZk-H-k7xINrPfSdElm0g224wyFrIDYqjKrUX39K4AfPMcbz7e3tNPWUsW4P4erqal81N2u19Bz6LnkXYyUwkwa5QXcF"
 - running: true

Develop

In de Storage worden alle foto's en video's opgeslagen met de timestamp als naam.
Jaar/maand/dag ...

gs://iot-werkstuk.appspot.com > videos			
	Name	Size	Type
<input type="checkbox"/>	20210530T140214.mp4	208.54 KB	video/mp4
<input type="checkbox"/>	20210530T140353.mp4	1.75 MB	video/mp4
<input type="checkbox"/>	20210530T140847.mp4	212.15 KB	video/mp4
<input type="checkbox"/>	20210530T144358.mp4	461.52 KB	video/mp4

gs://iot-werkstuk.appspot.com > photos			
	Name	Size	Type
<input type="checkbox"/>	20210530T140214.jpg	213.96 KB	image/jpeg
<input type="checkbox"/>	20210530T140353.jpg	215.79 KB	image/jpeg
<input type="checkbox"/>	20210530T140617.jpg	208.64 KB	image/jpeg
<input type="checkbox"/>	20210530T140847.jpg	210.08 KB	image/jpeg

Develop

Backend

De main functie

- initialiseert de *Firebase connectie*
 - maakt een *camera object* aan
- start het luisteren naar het settings document in Firestore*
- voegt enkele *listeners* toe aan het *onSettingsChange* event
 - *start de camera*.

```
from Firebase import Firebase
from Firebase import Firestore
from Camera import Camera
from Actuators import Led
import sys

def main():
    Firebase.init()
    Camera.init()

    Firestore.listenToSettings()

    Firestore.onSettingsChange('running', Led.toggle)
    Firestore.onSettingsChange('led', Led.toggle)

    Camera.start()

try:
    main()
except (KeyboardInterrupt, SystemExit):
    print('Exiting program')
finally:
    Camera.stop()
    sys.exit(0)
```

Develop

De Camera.start functie is het belangrijkste deel in de code. Hier wordt een oneindige iterator gestart die elke frame zal doorgeven aan de Motion module voor analyse. Alvorens de iterator te starten, worden enkele callbacks toegevoegd aan de onMotion en onMotionEnd event. Wanneer de camera gedeactiveerd wordt vanuit het dashboard, zal de loop wachten tot het terug geactiveerd wordt.

```
def start():

    # ...

    # Add callbacks to call when motion is detected
    Motion.onMotion(takePicture)
    Motion.onMotion(startRecording)
    Motion.onMotion(Led.on)

    Motion.onMotionEnd(stopRecording)
    Motion.onMotionEnd(Led.off)

    # Start capturing frames
    rawCapture = PiRGBArray(camera, size = camera.resolution)
    for f in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
        if Firestore.settings and Firestore.settings['running']:
            Motion.checkForMotion(f.array)
        else:
            Motion.close()
            sleep(0.25)

        # Clear the stream in preparation for the next frame
        rawCapture.truncate(0)
```

Develop

Wanneer de camera geactiveerd is, zal elke frame naar de Motion.checkForMotion functie gestuurd worden. Hier gebeurt een analyse die de huidige frame vergelijkt met de vorige, gebruik makend van de cv2 library. Wanneer het verschil tussen de twee frames hoger is dan een vooraf ingestelde waarde, wordt de frame beschouwd als motion frame. Voor zowel het starten als stoppen van de motion event is een threshold voorzien, zodat enkele uitlopers geen onverwacht resultaat geven.

```
def checkForMotion(frame):  
    # ...  
  
    # Convert frame to grayscale and blur  
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  
    gray = cv2.GaussianBlur(gray, (21, 21), 0)  
  
    # ...  
  
    # Find the absolute difference between average and current frame  
    cv2.accumulateWeighted(gray, avg, 0.05)  
    frameDelta = cv2.absdiff(gray, cv2.convertScaleAbs(avg))  
  
    # Threshold and dilate the difference  
    thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]  
    thresh = cv2.dilate(thresh, None, iterations=2)  
  
    # Find contours  
    contours, hierarchy = cv2.findContours(thresh.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)  
  
    # ...  
  
    # Check if a contour exceeds the minimum contour area  
    for c in contours:  
        if cv2.contourArea(c) > Firestore.settings['contourArea']:  
            handleMotionFrame()  
            break
```

Develop

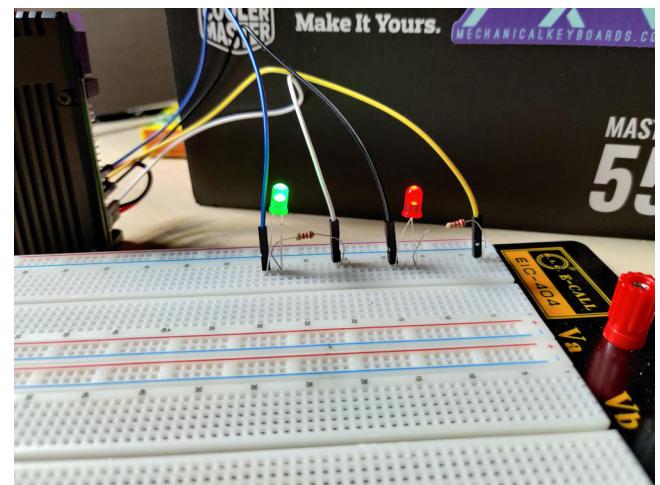
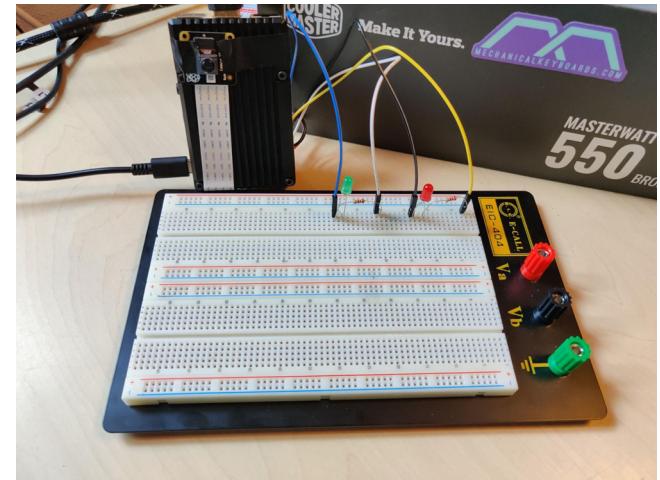
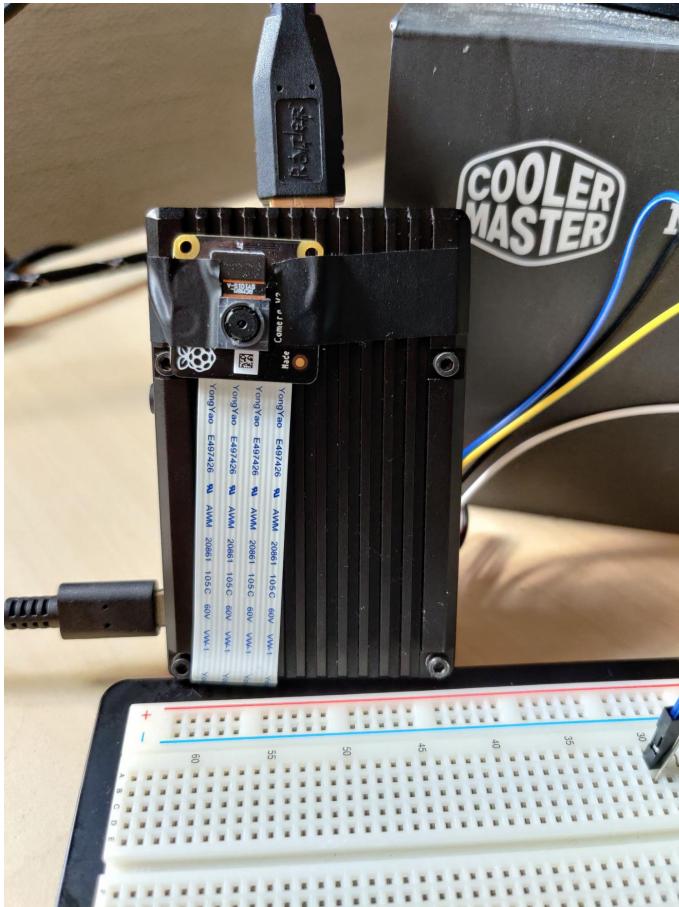
Bij het detecteren van beweging wordt een foto genomen en een opname gestart. Naast het uploaden van deze bestanden naar Firestore, wordt ook een notificatie verstuurd met de foto. Dit gebeurt in de Messaging module en maakt gebruik van Firebase Cloud Messaging. De registration tokens worden opgehaald uit Firestore.

```
def sendPushNotification(registrationTokens, title, imgUrl):
    # Define message payload
    message = messaging.MulticastMessage(
        notification = messaging.Notification(title = title, image = imgUrl),
        tokens = registrationTokens,
    )

    # Send a message to the devices corresponding to the provided registration tokens
    response = messaging.send_multicast(message)

    # Response is a message ID string
    removeFailedTokens(response, registrationTokens)
```

hardware



Deliverables

Installeren alvors runnen app.

- pip install opencv-python or pip install opencv-contrib-python
- sudo apt install -y gpac

Possible errors:

- Error: ImportError: libblas.so.3: cannot open shared object file: No such file or directory
- Solution:

```
pip3 install opencv-python
sudo apt-get install libblas-dev
sudo apt-get install libhdf5-dev
sudo apt-get install libhdf5-serial-dev
sudo apt-get install libatlas-base-dev
sudo apt-get install libjasper-dev
sudo apt-get install libqtgui4
sudo apt-get install libqt4-test
```

Handleiding

Camera:

- Connecteer de camera met de Raspberry Pi
- Geeft toestemming voor gebruik camera.

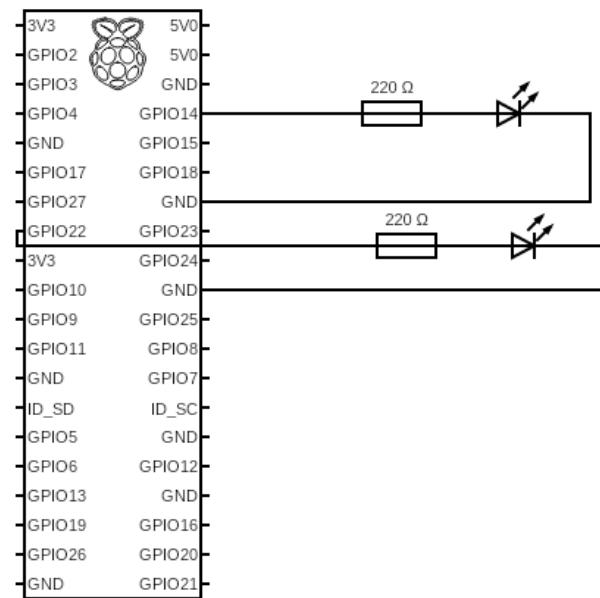
Gui: Menu > Preferences > Raspberry Pi Configuration > Interfaces

CLI: "sudo raspi-config > Interfacing options

- herstart de Raspberry Pi

Led:

- Connect led met GPIO pin 14 & 22
 - Pin 14 -> camera on/off
 - pin 22 -> camera recording ?



Deliverables

Dependencies installeren

Alvorens we het programma gaan kunnen runnen zullen we enkele modules moeten installeren.

PI:

```
sudo apt-get install libcblas-dev
sudo apt-get install libhdf5-dev
sudo apt-get install libhdf5-serial-dev
sudo apt-get install libatlas-base-dev
sudo apt-get install libjasper-dev
sudo apt-get install libqtgui4
sudo apt-get install libqt4-test
sudo apt install -y gpac
```

```
pip3 install opencv-python
pip install firebase -U
pip install firebase-admin -U
pip install google-cloud-storage -U
```

Web

Npm i / yarn

Programma runnen

```
cd surveillance-camera/app/pi/src
" python3 main.py"
```

Om ervoor te zorgen dat oude video's automatisch worden verwijderd na een tijdje moet je een cron job toevoegen.

- *Dit doen we als volgt:*
- *run "crontab - e"*
- *write "0 0 * * * python3 /home/pi/surveillance-camera/app/pi/cron.py" (wijzig pad indien nodig)*
- *opslaan*
- *run "cd surveillance-camera && python3 app/pi/src/main.py"*

Deliverables

Om de app op je smartphone te laten runnen ga je naar de volgende link:

<https://iot-werkstuk.web.app>

Deze installeer je vervolgens als een progressive web app.

Zet op beginscherm



Daar log je in met de volgende gegevens

Email: user@iot.com

Password: password

Firebase

- Wanneer je dit project wilt namaken maak je best je eigen Firebase project aan.
Dit kan je doen door volgende stappen te volgen:
 - Ga naar firebase.google.com en maak een nieuwe project.
 - Connecteer Firebase met je Raspberry pi
 - Project > settings > Service accounts en genereer een nieuwe "private key"
 - Dit zorgt ervoor dat er een JSON file wordt gedownload
 - Plaats deze file binnen volgende folder "/app/pi/src/Firebase OF
 - Kopier inhoud van deze file naar "ServiceAccountKey.example.json"
 - En verander de naam vervolgens naar "ServiceAccountKey.json"
 - Connecteer Firebase met het dashboard
 - Vanuit de Firebase console maak je een nieuwe web app aan
 - Kies een naam en zorg ervoor dat hosting is ingeschakeld
 - In Terminal run je "npm install -g firebase-hosting"
 - Vervolgens ga je via de terminal naar de root van dit project daar run je:
"Firebase login"
"Firebase init"
 - Vervolgens ga je naar Project settings in de Firebase console via:
 - General > Your apps > SDK setup and configuration > Config
 - Deze settings kopiëer je
 - Dan ga je binnen je project naar "/src/core/services.firebaseio.js" en plak je alles.
 - Configureer cloud messaging
 - Binne de project settings ga je naar het Cloud Messaging tab en genereer je een
 - nieuw "Key pair". Kopiëer deze key en plak deze in:
"/src/core/hooks/useRegistrationToken.js"

Deliverables

- *Schakel firestore in*
 - *Build > Firestore database en maak een nieuwe database aan*
- *Schakel Authentication in*
 - *Build > Authentication > Get started*
- *Schakel Email/Password in*
 - *Ga naar het tab "users" en voeg een user toe, deze gegevens zijn de gegevens die je later zal gebruiken om in te loggen in je dashboard.*
- *En tot slot build & deploy*
 - *Ja naar de root van de webapplicatie ("/app/web") en run "yarn install"*
 - *Vervolgens run je "yarn build" gevuld door "firebase deploy"*
 - *En je dashboard is nu online beschikbaar*
 - *Open het dashboard en installeer dit als een PWA op je smartphone*
 - *Vervolgens log je in en je dashboard is klaar voor gebruik*

Timesheet

Fasen	Totale tijd
Voorbereiding	17u40
Werkstuk	96u05
Productiedossier	10u30
Video-opname	8u15
Totaal	132u30