Pawtential

Teknisk rapport

Mikka De Leyos | mcdeleyo

Mina Ghairat | minagha

Radwa Godor | radwaag

Jenny Le | jenl

Laurynas Pilibaitis | laurynp

Prosjektoppgave IN1060 Institutt for informatikk

Universitetet i Oslo

28.05.2024

Antall ord: 2282

INNHOLDSFORTEGNELSE

1. INNLEDNING 3

1.2 Presentasjon av artefakt 3

2. VIDEO 4

2.1 Lenke til video 4

2.2 Beskrivelse av video 4

3. TEKNISK LØSNING 4

3.1 Oversikt over de brukte komponenter 5

3.2 Kompontentliste 5

3.3 Kretser 6

3.4 Kodeforklaring 7

3.5 Utfordringer 11

4. AVSLUTNING 12

LITTERATURLISTE 13

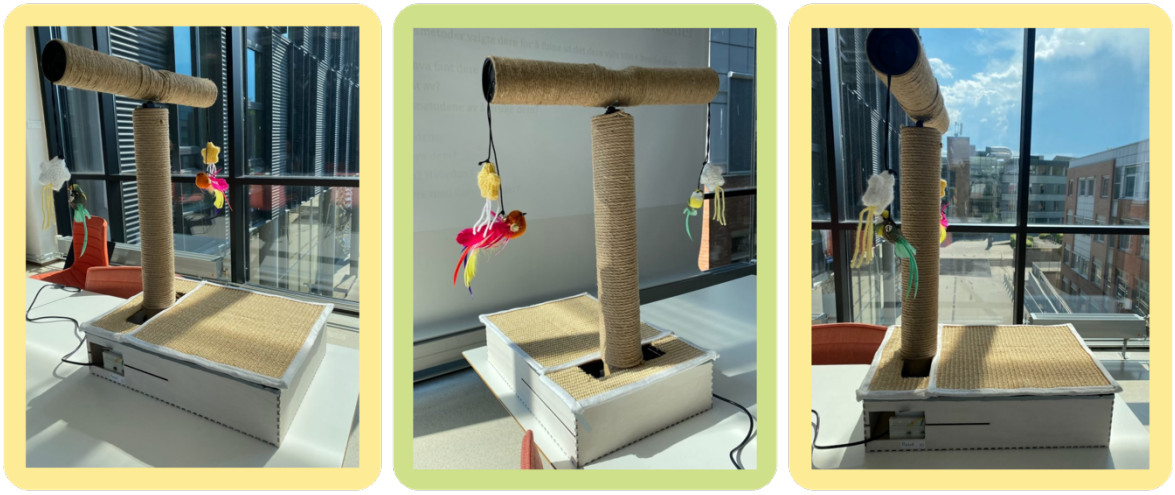
1. INNLEDNING

1.1 Mål for prosjektet

I IN1060 – Bruksorientert design – fikk vi den gylne muligheten til å realisere vår visjon ved å skape en automatisert artefakt ved hjelp av Arduino. Temaet for årets prosjekt var «av og på», og dette har vi forsøkt å implementere ved å skape en artefakt som skal dekke behovet for stimulering hos katter, og ikke minst aktivere deres naturlige jaktinnsikt. Målgruppen vår var katteeiere, og ved å tilfredsstille dette behovet, stimulerer vi også katteeiernes rolle som gode dyreeiere. I denne rapporten presenterer vi vår artefakt, den tekniske løsningen og utfordringer vi har møtt på.

1.2 Presentasjon av artefakt

Catgym er en automatisert katteleke som har blitt bygget ved hjelp av Arduino Student Kit, og er designet for å stimulere katter og aktivere deres jaktinstinkt. Artefakten aktiveres når katten «tråkker» på platen. Under platen har vi plassert en vektsensor som registrerer en vekt på over 200 gram. Når vektsensoren registrerer trykket fra katten, vil hele artefakten aktivere seg – som speiler vår implementasjonen av "av og på". Stangen vil begynne å rotere 180 grader ved hjelp av en mikroservo, og hengelekene vil begynne å «fly» og lage «bjelle»-lyder. Hengelekene står på en klorestativ slik at katten kan både leke og tilfredsstille sitt naturlige instinkt for å klore. I tillegg er det festet et klorematerialе på selve platen. Etter registrering av vekten, er artefakten «på» i et helt minutt før katten må aktivere den igjen med sin kroppsvekt. Det er implementert to knapper: reset-knapp og av/på-knapp. Reset-knappen vil ta en omstart av hele koden, mens av/på-knappen vil stanse koden helt – slik at katteeierne har muligheten til å slå av artefakten uten å dra ut noen ledninger. Catgym har en solid og holdbar form ettersom vi har brukt treverk for å sikre stabilitet. Ikke minst 3D-printet vi en stativholder for å holde klorestativet stabil.



Figur 1: *Endelig artefakt – Catgym!*

2. VIDEO

2.1 Lenke til video

Her er lenken til video av bruksscenario til artefakten:

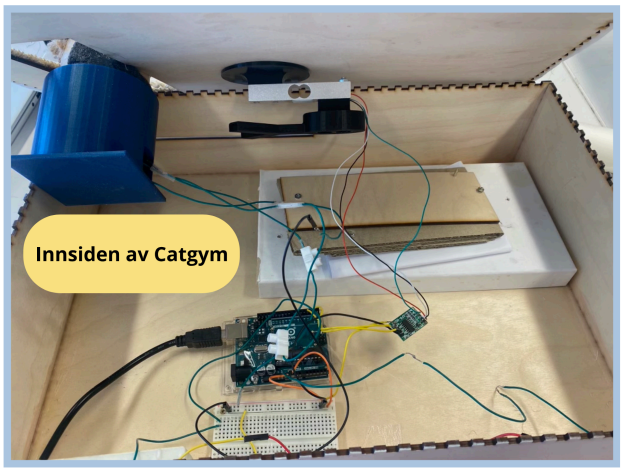
https://www.youtube.com/watch?v=Kb9Ea5tdvoE&t=5s

2.2 Beskrivelse av video

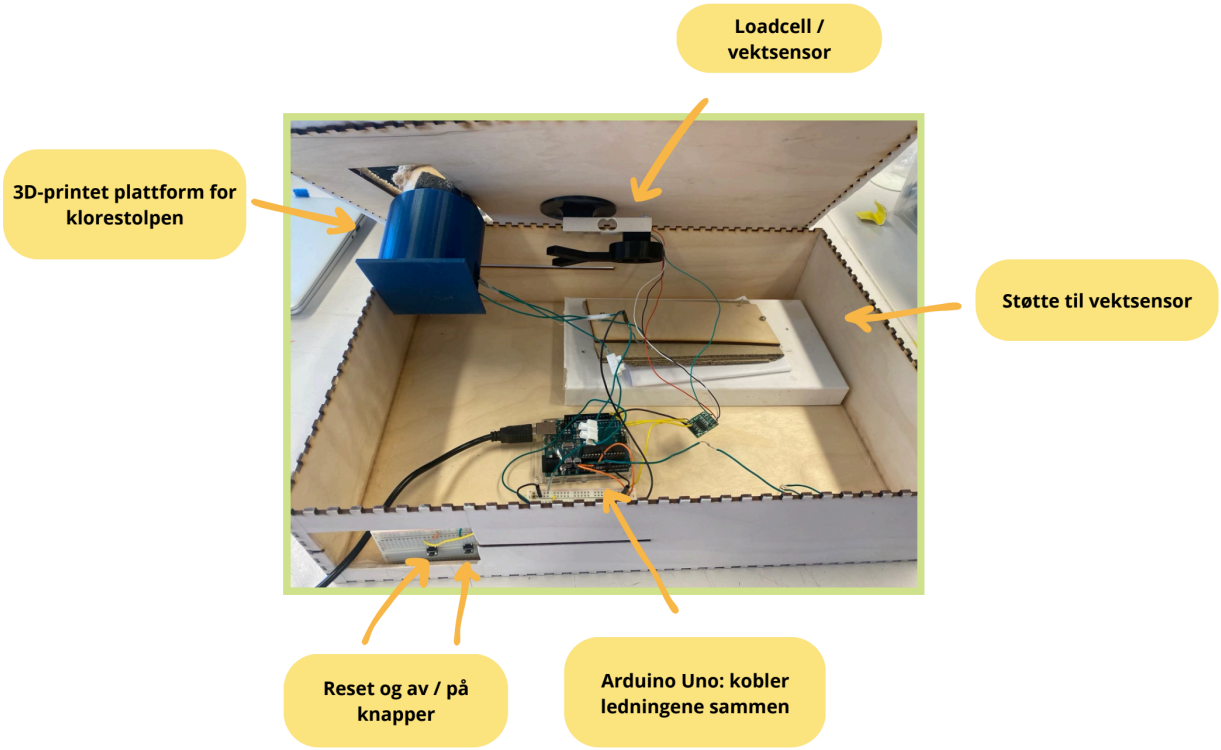
I videoen presenteres et bruksscenario i møte med en frustrert katteeier. Når katteeieren prøver den automatiserte katteleken – Catgym – utføres de ulike funksjonene implementert i artefakten. Videre i videoen får vi med oss to katter som aktivt engasjerer seg i artefakten. Konseptet *aktivitet* gjenspeiles når kattene interagerer med artefaktens funksjonaliteter inkludert klorestativet av sisal. Mot slutten av filmen forklarer katteeier at de endelig har mer tid til å fokusere på seg selv, mens deres katter er i gode, automatiserte hender.

3. TEKNISK LØSNING

Artefakten Catgym er konstruert på følgende måte: rotasjon og vektregistrering. Dets funksjoner er inkludert for å lette på samvittigheten til katteiere, gjennom selvregulerende stimulerring for pusene. Selve boksen er rektangelformet og montert sammen av fire laserkuttede plater. For å unngå uhell rundt det tekniske har vi plassert Arduino inne i den rektangelformede boksen, slik at den ligger der trygt. I tillegg består artefakten av to sylindere; den som ligger på toppen –«*armen*» – roterer 180 grader for hvert 1.5 sekunder, og den på bunnen – «stangen» – blir stående. Mer om artefaktets funksjon blir demonstrert i videoen over.



3.1 Oversikt over de brukte komponenter



Figur 2: *Oversikt over alle komponentene som er blitt brukt i artefakten.*

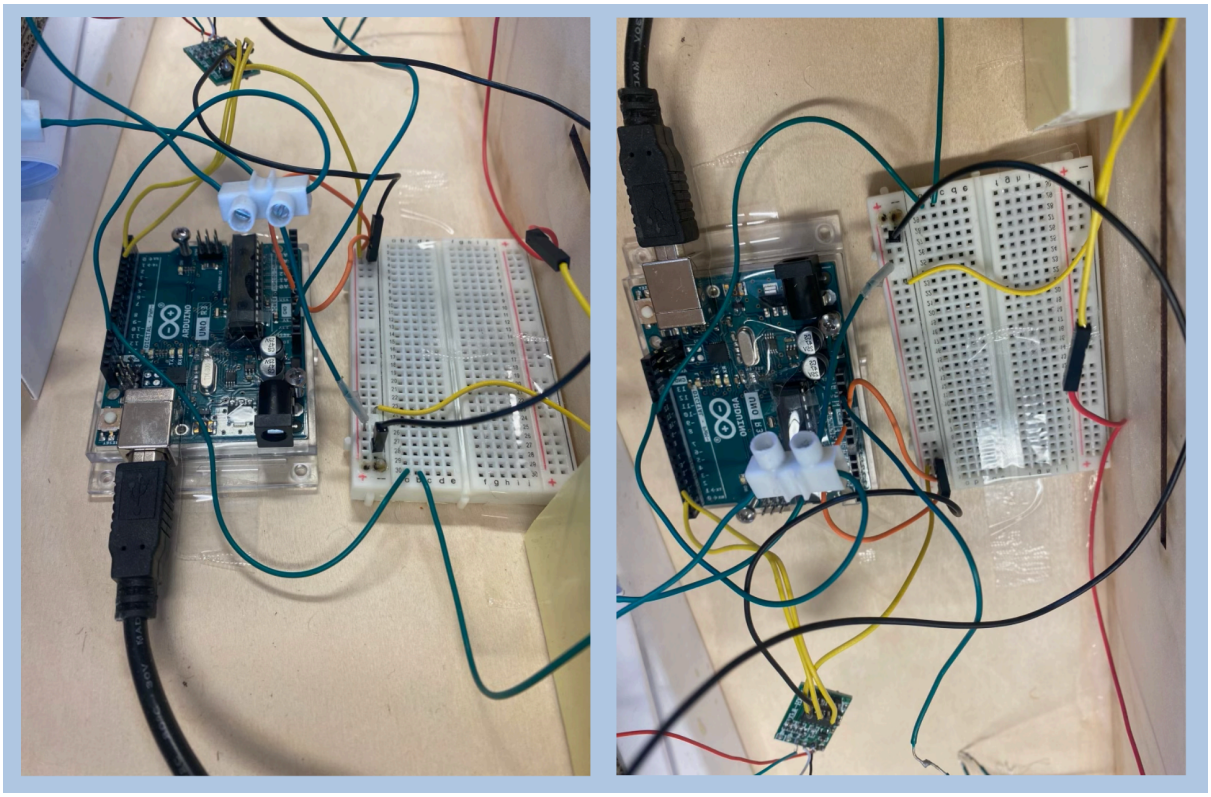
3.2 Kompontentliste

Tabell 1: En liste over komponentene som ble brukt i vår artefakt

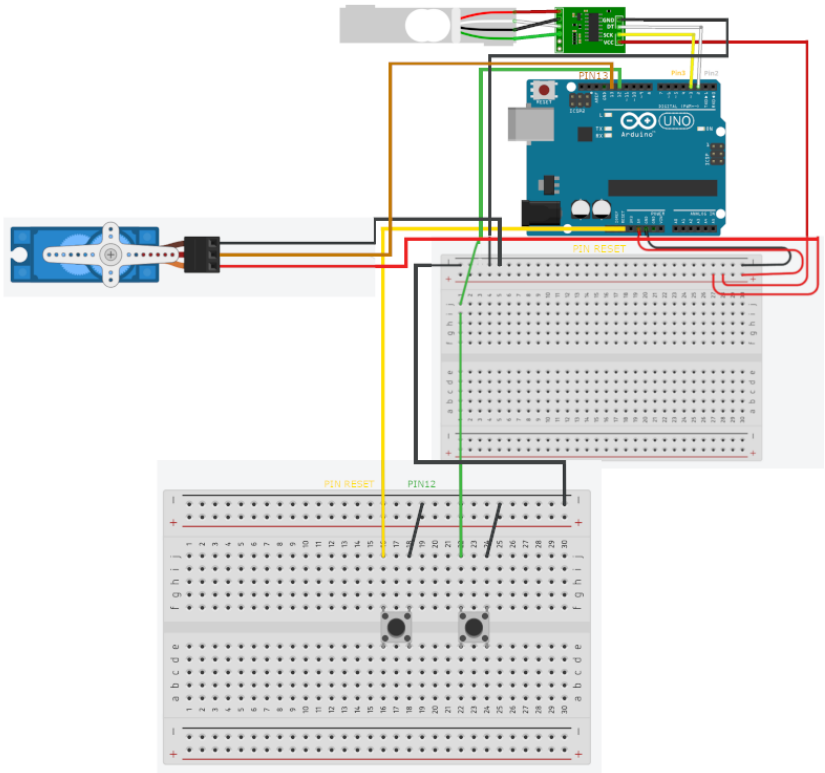
KOMPONENTER	ANTALL	BESKRIVELSE AV FUNKSJONALITET
USB A-B kabel	1	Kobler Arduino mikro-kontroller til PC
Arduino UNO R3 brett	1	Arduino mikro-kontroller
Breadboard	2	Infrastruktur till ledningene
Mikroservo	1	Innebygd kontrollenhet som har evne til å bevege seg i sirkler
Ledninger	x	Leder strømmen videre til komponentene, og kobler GND og 5V i breadboardet
1k ohm resistor	1	Begrenser strømmen
Vektsensor (load cell) 5KG	1	Overfører kompresjon, trykk og vekt om til signal
HX711	1	Leser av vektsensoren, og konverterer analoge signaler til digitale tall (24-bit ADC)
Bryter	2	Knapp

Figur 3: *Komponentliste.*

3.3 Kretser



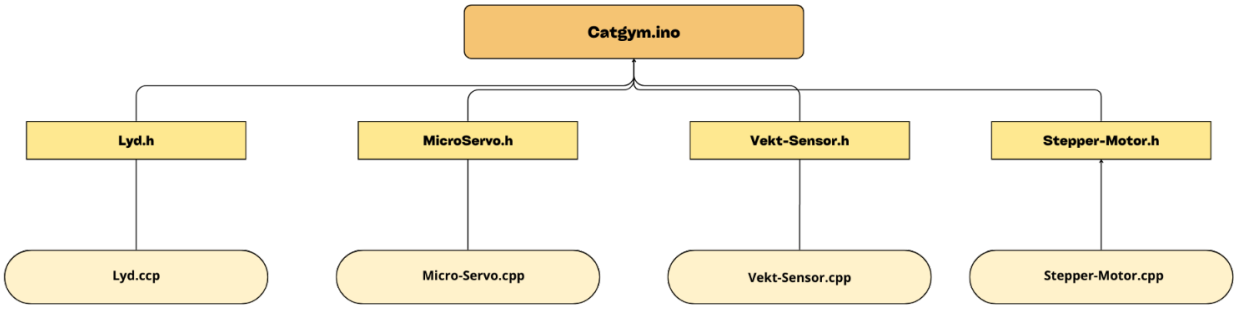
Figur 4 og 5: *Bilder av fysiske kretsen.*



Figur 6: *Digital bildet av kretsen.*

3.4 Kodeforklaring

Vi tenkte at koden skulle bestå av en overordnet klasse kalt Catgym.ino. Inne i hovedklassen Catgym skulle det bli fulgt av eksterne koder / biblioteker; og med #Include syntaksen skulle vi kalle på bibliotekene under. Disse bibliotekene var opprinnelig: Lyd, MicroServo, Vekt-Sensor og Stepper Motor. I *figur 1* er det visualisert en hierarkisk oversikt over den opprinnelige oppbygning av koden og strukturen.



Figur 7*: Overordnet klasse Catgym med underklassene Lyd, MicroServo, Vekt-Sensor og Stepper-Motor*

Figur 2 demonstrer det endelige oppsette av koden. Gjennom flere begrensninger av funksjonaliteter av det endelige artefaktet, vi kom frem til at vi har ikke behov for den nivå av modulisering i koden – siden det er primært bygd opp av to hovedfunksjoner (*Bevegelse()* og *Oppstart()*). Ettersom vi ikke bruker lyd eller steppermotor, er det ikke grunnlag for å beholde disse såkalte *interfacene* fra Arduino biblioteket.

Figur 8*: Endelig oppsett av koden*

Opprinnelig valgte vi å modulere koden – en klasse opprettes for hver funksjon. Catgym.ino skulle i dette tilfellet fungere som hovedfilen der alle de små klassene og kodene skulle kjøres gjennom. For hver funksjon opprettet vi en header fil (.h) som inneholdt deklarasjoner av metoder, variabler og konstanter som i etterkant skulle gjenbrukes. Vi opprettet source filer (.cpp) som brukte deklarasjonene, og implementerte de samme metoder, variabler og konstanter fra header-filene – dette ble til slutt implementert i .ino-filen. I etterkant skjønte vi at funksjonene som MikroServo.h / .cpp tok opp unødvendig lagringsplass, og krevde egentlig bare én funksjon for å kunne implementeres i hovedfilen.

Underveis vurderte vi å bruke flere Arduino-mikrokontrollere – en for hver funksjon, men kom frem til at synkronisering mellom hver Arduino gjennom vektsensor kunne skape problemer, og dermed valgte vi heller å gå videre med kun én istedenfor.

Tabell 2: Liste over biblioteker som har blitt brukt i løsningen:

Bibliotek	Beskrivelse
HX711_ADC.h	*HX711_ADC* bibliotek som tilhører *HX711 modul* som brukes til *load cell* (vektsensor).
Servo.h	*Servo* bibliotek som tillater kontroll av mikro-servo bevegelse gjennom *attach()* og *write()*.
Arduino.h	Gir tilgang til funksjoner som *pinMode(), digitalWrite(), analogWrite(), delay(), Serial.begin()*

Tabell 3, 4, 5: Liste over konstanter, variabler og innebygde funksjoner som har blitt brukt i løsningen.

Konstanter	Beskrivelse
const int dt	Vi bruker pin 3 for dt tilkobling på *HX711 modul* som da sender informasjon fra vektsensoren og oversetter det til Arduino.
const int sck	Vi bruker pin 2 for sck tilkobling på *HX711 modul* som sender informasjon fra vektsensoren og oversetter det til Arduino.
const int knapp	Bruker pin 12 til knappen. Dette brukes for *AvPaa* variabelen.
const long interval	Konstanten er satt til 1500 millisekunder, som tilsvarer 1.5 sekunder. Benyttes til *Bevegelse()* funksjonen for hvor ofte det skal rotere.
const long varighet	Denne konstanten er satt til *60000 millisekunder*, som tilsvarer 60 sekunder eller 1 min. Benyttes til å aktivere *Bevegelse()* i 60 sekunder.
Servo HorisontalServo	Initierer en objekt av *Servo* biblioteket med navnet *HorisontalServo*.
HX711_ADC LoadCell(dt, sck)	Initierer et objekt av *HX711_ADC* biblioteket med navnet *LoadCell*.

Variabler	Beskrivelse
unsigned long previousMillis	Blir satt til 0 ved initiering av variabelen.
unsigned long currentMillis	Blir satt til 0 ved initiering av variabelen. Settes til *millis()* – *klokkeStartTid* ved hver gjennomkjøring av loopen for å ta tiden.
int servoPos	Initialisering av servo posisjon variabelen. Ansvarlig for bevegelsen i servo.

bool AvPaa	Initialisering av status variabel som *«false»*, kontrollerer om prototypen er av eller på.
int knappCurrentState	Denne koden er innstilt uten noe forhånd satt verdi, siden det oppdateres i *loop()* delen av koden gjennom *digitalRead()*.
int knappLastState	Den er innstilt til verdi *HIGH*, slik at når *loop()* starter i koden, så kan koden komme seg gjennom første if-påstand.
unsigned long startMillis	Lagrer siste gangen vektsensor ble aktivert.
bool Aktiv	Boolean variabel som bestemmer om vektsensor skal aktiveres eller ikke ved bruk av *true* eller *false*.
unsigned long stabilizingTid	Stabiliserer og gir tid til oppstart av *HX711 modul*.
bool_tare	Utfører en tare ved oppstarten.

Innebygde funksjoner	Beskrivelse
millis()	*millis()* vil måle og returnere tiden (antall millisekunder) det har gått siden koden begynte å kjøre. Siden flere av funksjonalitetene baserer seg på tidsintervaller, bruker vi *millis()* for å la alle funksjoner kjøre parallelt i koden uten at det stopper for andre under samme kjøring.
pinMode()	*pinMode()* brukes til å konfigurere en spesifikk pinne som enten inngang eller utgang – slik at Arduino forstår hvordan den skal bruke den aktuelle pinnen, altså hvordan pinnen skal oppføre seg ved å skrive enten input eller output. Vi bruker denne funksjonen for å definere og konfigurere av og på-knappen. For å unngå eksterne forstyrrelser som kan påvirke knappens tilstand, har vi tatt i bruk INPUT_PULLUP for å stabilisere tilstanden. INPUT_PULLUP er konfigurert med intern pullup-motstand.
digitalRead()	*digitalRead()* brukes til å lese statusen til en digital pin som er konfigurert som en inngang. Den returnerer enten *HIGH* eller *LOW*, avhengig av om pinnen leser en høy eller lav spenning. Vi benytter denne funksjonen til knappen (push button), som samarbeider med variabelen *knappCurrentState* for å endre *AvPaa* variabelen til *true/false*.
Serial.begin(57600)	*Serial.begin(57600)* brukes til å starte seriell kommunikasjon mellom Arduino og en datamaskin eller en annen enhet via seriell port. *57600* tilsvarer en baudrate (biter per sekund), og er hastigheten på dataoverføring – jo høyere baudraten er, desto raskere blir data overført fra Arduino til enheten. Vi benytter denne funksjonen til feilsøking; forstå hva slags data som

	overføres fra og til datamaskinen og Arduino – og overvåke hva som skjer i programmet vårt.
Serial.println()	*Serial.println()* er avhengig av *Serial.begin()* for å kunne fungere. Funksjonen brukes til å sende data til den serielle porten, og skriver ut nødvendig linjer. Vi benytter denne funksjonen for å overvåke og kommunisere mellom Arduino og datamaskinen, slik at vi har oversikt over hva som skjer i programmet.

void Bevegelse (int vinkel): Denne funksjonen tar inn en parameter – **vinkel** – av typen int. Denne funksjonen kalles på i loop(), og her har vi mulighet til å bestemme ønsket vinkel som mikroservoen skal kunne rotere i. Vi satte den til maksimal rotasjonsvinkel for mest mulig bevegelse.

Vi la til en if-påstand i funksjonen, og som sjekker om variabelen servoPos er 0. Hvis den er 0, vil verdien til servoPos endres til verdien av vinkel-variabelen. Når funksjonen kalles på i loop(), vil den sørge for at mikroservoen roteres basert på vinkelen som sendes som argument. Skulle servoPos vært et annet heltall, vil den endres tilbake til 0.

```
78    void Bevegelse(int vinkel) {
79      if (servoPos == 0) {
80        servoPos = vinkel; // Gaar til en ende av spekteret
81      } else {
82        servoPos = 0; // Gaar tilbake til startposisjonen
83      }
84      HorisontalServo.write(servoPos);
85    }
86
```

Figur 9: *Funksjon Bevegelse()*

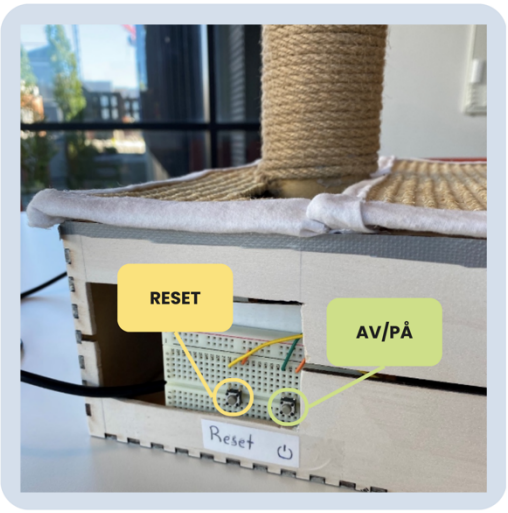
void Oppstart (float vekt): Denne funksjonen tar inn en parameter – **vekt** – av typen float. vekt er en variabel som er blitt initialisert i loop(), og har fått tilordnet en verdi som **LoadCell.getData()-**funksjonen returnerer i det programmet kjører. LoadCell.getData() mottar og håndterer data fra vektsensoren i heltall (int), mens **LoadCell.update()** oppdaterer verdien basert på vekt-verdien som returneres av LoadCell.getData(). Hvis verdien er over 200g, vil variabelen *Aktiv* dermed settes lik true. Når *Aktiv* er true, vil koden i loop() fortsette videre til å kalle på Bevegelse()-funksjonen.

```
87    void Oppstart(float vekt) {
88      if (vekt >= 200) {
89        if (!Aktiv) {
90          // Starter
91          Aktiv = true;
92          startMillis = millis();
93        }
94      }
95    }
```

Figur 10: *Funksjon Oppstart (float vekt)*

Knapper

Under den siste iterasjon innså vi at det ble veldig upraktisk å trekke ut kabelen hver gang artefakten skulle testes / slås på. Av den grunn kom vi på en boolean-løsning som kunne stanse koden ved bruk av en knapp (push button). Knappen bruker en boolean variabel **AvPaa**, og når den settes til false, stanser resten av koden i loop(). Vi har i tillegg implementert en reset-knapp som er direkte koblet til Arduinoens reset pin, og tilbakestiller programmet når knappen trykkes på.



Figur 11*: reset-og av/på-knapp*

3.5 Utfordringer

Utformingen av det tekniske til artefakten resulterte i en del utfordringer – både små og store. En av våre store utfordringer var komponentene for både lyd, horisontal-bevegelse og vektsensor funksjonalitetene. Stepper-motor modulen – som gjorde det mulig å kontrollere stepper-motoren – kom ikke tidsnok, og på bakgrunn av dette fikk vi heller ikke implementert funksjonaliteten (stepper_motor.h). DFPlayer Mini MP3 Player sammen med høyttaleren kom derimot ganske tidlig, men funksjonaliteten (lyd.h) ble ekskludert mot slutten ettersom den ikke fungerte – ville ikke spille av lyd. En mulig løsning var å bestille på nytt, men vi oppdaget fort at det kunne bli risikabelt å måtte vente på de nye komponentene på grunn av stram tidsramme. En siste komponent var vektsensoren. Den måtte bestilles på nytt uavhengig ettersom HX711 modul manglet, og dermed ble det en sen implementasjon av denne funksjonaliteten. Vi måtte revidere planen enda en gang, og det resulterte i at prosjektarbeidet hang et godt stykke bak skjemaet enn forventet.

I forbindelse med vektsensoren, var kalibreringen en krevende oppgave for oss. Hovedgrunnen var at vi trengte en stabil plattform som kunne fordele vekten fra den ene polen av vektsensoren til den andre – og på denne måten kunne vektsensoren også få registrert nøyaktig vekt.

En siste utfordring vi måtte gruble over var hvordan vi kunne få økt hastigheten på rotasjonen. De gangene vi testet artefakten, oppstod det samme problemet flere ganger: armen roterte ikke etter ønsket hastighet. Den roterte altfor sakte ved start, men etter hvert som den fikk kjøre i et par sekunder ble hastigheten mye raskere. Vi ønsket å holde en jevnlig hastighet utover brukstiden, og begynte derfor å feilsøke små områder av gangen for å se hvor problemet befant seg. Det viste seg at årsaken var lite friksjonen mellom mikroservoen og stangen. En mikroservo under belastning krever friksjon for å fungere korrekt. En ny løsning var å skape sikring på et av fire hjørner til mikroservoen med en skrue.

4. AVSLUTNING

I dette prosjektet har vi utviklet en automatisert artefakt som stimulerer og aktiverer jaktinstinktet hos katter. Vi har støtt på flere utfordringer, hvor den mest betydelige var manglende evne til å implementere alle ønskede funksjoner, som bevegelse av hele stangen og fuglelyder. Likevel har vi klart å implementere de mest sentrale funksjonene som tilfredsstilte behovet for stimulering av katter ved hjelp av vektsensor og mikroservo. Vektsensoren spilte en kritisk rolle i inkorporeringen av det overordnede temaet «av og på». Det har vært mange oppturer og nedturer, men grunnet samarbeidet innad i teamet har vi klart å realisere visjonen vår - som var vårt endelige mål. Vi har fått en dypere forståelse av Arduino og bruksorientert design, og disse lærdommene skal vi ta med oss videre i studietiden slik vi kan bli dyktige, kreative designere – og skape bærekraftige, innovative artefakter sammen med brukere.

LITTERATURLISTE

Arduino IDE 2.3.2. https://www.arduino.cc/en/software.com

Bratteteig. T (2021). Design for, med og av brukere: *å inkludere brukere i design av informasjonssystemer*. Universitetsforlaget.

GitHub koden: workworkworkworkworkwork/Catgym.ino (github.com)

Tinkercad. https://www.tinkercad.com/