

Prototype - notat

Tittel: Iterasjon 5

Forfattere: Prosjekt - gruppe 2

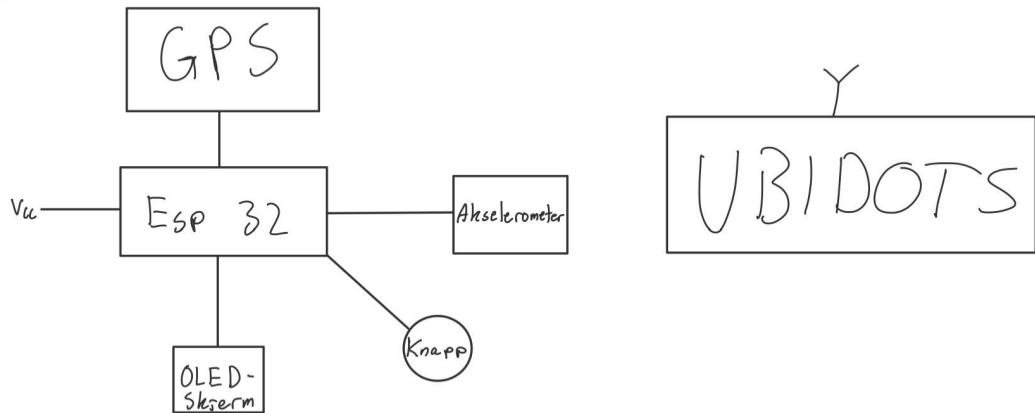
Versjon: 1.0	Dato: 12.03.16
--------------	----------------

Innhold

1	Introduksjon	2
2	Metode og testmiljø	2
3	Test resultat og diskusjon	9
4	Begrensninger og tiltak	10
5	Konklusjon	10

1 Introduksjon

Vi skal ta for oss et system som vist i figur 1.



Figur 1: Prototype iterasjon 5

Denne iterasjonen er vårt ferdige produkt. I denne nye iterasjonen har vi laget en egen boks for alle komponentene, vist i figur 2. Vi har fikset problemet med at den teller å stå i ro som ”god” kjøring, og vi har laget en funksjon som sender oss koordinater på hvor de groveste overtrampene skjer slik at disse kan overvåkes av kommunen.

2 Metode og testmiljø

Under er den nye og oppdaterte kode. Denne koden er delt inn i fem header filer og en hovedkode. Header filene inneholder funksjonene til setup, oled-skjermen, knappen og score.

Oledfunksjoner:

```
1 #include <U8g2lib.h>
2
3 // Setup for SH1106 display ved bruk av U8G2 biblioteket
4 // Initialiserer displayet SH1106 med I2C kommunikasjon og ingen reset pin
5 U8G2_SH1106_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/
6                                         /* U8X8_PIN_NONE);
7
8 // Funksjon som håndterer oppdatering av OLED-skjermen
9 void OLEDskjerm() {
10     unsigned long currentMillis = millis(); // Får den nåværende tiden i millisekunder
11     unsigned long previousMillis = 0; // Lagrer sist gang displayet ble oppdatert
12     const long interval = 1000; // Interval som displayet skal oppdateres på (millisekunder)
13
14     // Sjekker om toggleState er satt til 1
15     if (toggleState == 1) {
16         // Sjekker om nok tid har passert siden forrige oppdatering
17         if (currentMillis - previousMillis >= interval) {
18             previousMillis = currentMillis; // Oppdaterer previousMillis til nåværende tid
19
20             u8g2.clearBuffer(); // Tømmer bufferet for å kunne skrive ny tekst
21             u8g2.setFont(u8g2_font_ncenB08_tr); // Setter fonten som skal brukes
22
23             char displayText[30]; // Buffer for å holde teksten som skal vises
24             sprintf(displayText, "Score: %d", score); // Formaterer teksten med score-verdi
25
26             int textWidth = u8g2.getUTF8Width(displayText); // Får bredden på teksten
27             int x = (u8g2.getDisplayWidth() - textWidth) / 2; // Beregner X-posisjon for å sentrere teksten
28             int y = u8g2.getDisplayHeight() / 2; // Beregner Y-posisjon for å plassere teksten i midten
29
30             u8g2.drawStr(x, y, displayText); // Tegner teksten på skjermen
31             u8g2.sendBuffer(); // Sender bufferet til skjermen for visning
32         }
33     } else {
34         // Hvis toggleState ikke er 1, viser en annen melding
35         if (currentMillis - previousMillis >= interval) {
36             previousMillis = currentMillis; // Oppdaterer previousMillis til nåværende tid
37
38             u8g2.clearBuffer(); // Tømmer bufferet for å kunne skrive ny tekst
39             u8g2.setFont(u8g2_font_ncenB08_tr); // Setter fonten som skal brukes
40
41             char displayText[30]; // Buffer for å holde teksten som skal vises
42             sprintf(displayText, "ForceMap er av."); // Formaterer teksten
43
44             int textWidth = u8g2.getUTF8Width(displayText); // Får bredden på teksten
45             int x = (u8g2.getDisplayWidth() - textWidth) / 2; // Beregner X-posisjon for å sentrere teksten
46             int y = u8g2.getDisplayHeight() / 2; // Beregner Y-posisjon for å plassere teksten i midten
47
48             u8g2.drawStr(x, y, displayText); // Tegner teksten på skjermen
49             u8g2.sendBuffer(); // Sender bufferet til skjermen for visning
50         }
51     }
52 }
53 }
```

Figur 2: Oled-funksjoner

Knappefunksjoner:

```
1 // Pindefinisjoner
2 const int buttonPin = 33; // Pinnen hvor knappen er tilkoblet
3
4 // Variabler for å håndtere debouncing
5 unsigned long lastDebounceTime = 0; // Siste gang utgangspinnen ble endret
6 unsigned long debounceDelay = 50; // Debounce-tid; øk hvis utgangen flimrer
7
8 // Variabler for å spore tilstanden
9 int lastButtonState = LOW; // Forrige avlesning fra inngangspinnen
10 int buttonState = LOW; // Nåværende avlesning fra inngangspinnen
11
12 // Toggle-variabel
13 int toggleState = 1; // Variabel for å holde styr på toggle-tilstanden
14
15 // Funksjon for å sjekke knappetrykk, debounce og toggle tilstanden
16 int toggleStateFunction() {
17     static int internalToggleState = toggleState; // Lagrer toggle-tilstanden mellom funksjonskall
18     int reading = digitalRead(buttonPin); // Les tilstanden til knappen
19
20     if (reading != lastButtonState) {
21         // Nullstil debouce-timeren
22         lastDebounceTime = millis();
23     }
24
25     if ((millis() - lastDebounceTime) > debounceDelay) {
26         if (reading != buttonState) {
27             buttonState = reading;
28             lastDebounceTime = millis(); // Oppdater tiden for endringen
29
30             if (buttonState == HIGH) {
31                 internalToggleState = 1 - internalToggleState; // Endre tilstanden
32             }
33         }
34     }
35
36     lastButtonState = reading; // Oppdater forrige knappetilstand
37     return internalToggleState;
38 }
```

Figur 3: Knappefunksjoner

Setupfunksjoner:

```

1  /* Header fil som inneholder alle
2  de ulike setup - funksjonene */
3
4  #include <Adafruit_GPS.h>
5  #include <Arduino.h>
6  #include <UbidotsESP32MQTT.h>
7  #include "GY521.h"
8
9  #define GPSSerial Serial2
10 Adafruit_GPS GPS(&GPSSerial);
11
12 GY521 sensor(0x68); // I2C port for kommunikasjon
13
14 // Ubidots credentials
15 const char *UBIDOTS_TOKEN = "BBUS-4GTYhGckwBnhENTwIxRPTeLvX1wWsR";
16 const char *WIFI_SSID = "NTNU-IOT";
17 const char *WIFI_PASS = "";
18
19 // GPS ubidots labels
20 const char *DEVICE_LABEL_gps = "gps_position";
21 const char *VARIABLE_gps = "gps";
22 const char *VARIABLE_satellitt = "satellitter";
23 const char *VARIABLE_satellitt_kvalitet = "kvalitet";
24 const char *VARIABLE_gps_speed = "gps_speed";
25 const char *VARIABLE_gps_speed_breach_lat = "gps_speed_breach_lat";
26 const char *VARIABLE_gps_speed_breach_long = "gps_speed_breach_long";
27
28 // akselerometer ubidots labels
29 const char *DEVICE_LABEL_akselerometer = "akselerometer";
30 const char *VARIABLE_akselerometer_verdi = "aks_verdi";
31 const char *VARIABLE_score = "score";
32
33 Ubidots ubidots(UBIDOTS_TOKEN);
34
35 void ubidots_setup()
36 {
37     // ubidots setup, innebygde funksjoner som etablerer kommunikasjon med ubidots
38     ubidots.setup();
39     ubidots.reconnect();
40     // Abonnerer på utvalgte topics inne på ubidots
41     ubidots.subscribeLastValue(DEVICE_LABEL_akselerometer, VARIABLE_akselerometer_verdi);
42     ubidots.subscribeLastValue(DEVICE_LABEL_akselerometer, VARIABLE_score);
43     ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_gps);
44     ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_satellitt);
45     ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_satellitt_kvalitet);
46     ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_gps_speed);
47     ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_gps_speed_breach_long);
48     ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_gps_speed_breach_lat);
49 }
50
51 void internet_connection()
52 {
53     // koble til internettet
54     Serial.println("Initializing Wi-Fi....");
55     ubidots.begin(UBIDOTS_TOKEN, WIFI_SSID, WIFI_PASS);
56     if (ubidots.connected()) {
57         // tilkobling på osm tilkoblingen ble gjennomført
58         serial.println("Wi-Fi connected!");
59     } else {
60         serial.println("Failed to connect to Wi-Fi!");
61     }
62 }
63
64 // Funksjon for å kalibrere akselerometret
65 void calibrate_akselerometer() {
66     Serial.println("Kalibrerer akselerometret"); // Skriv en melding til seriell monitor for å indikere at kalibrering startet
67     int resolution = 100; // Antall målinger som skal brukes for kalibreringen
68     float calx = 0, caly = 0, calz = 0; // Variabler for å samle opp akselerometradata
69
70     // Les akselerometradata 'resolution' ganger
71     for (int i = 0; i < resolution; i++) {
72         sensor.read(); // Les data fra sensoren
73         calx += sensor.getAxelX(); // Legg til X-aksel data
74         caly += sensor.getAxelY(); // Legg til Y-aksel data
75         calz += sensor.getAxelZ(); // Legg til Z-aksel data
76     }
77     // Beregn gjennomsnittet av hver aksel og inverter det for kalibrering
78     sensor.setAxelX((calx / resolution) / -1); // Kalkuleringsverdi for x-aksen
79     sensor.setAxelY((caly / resolution) / -1); // Kalkuleringsverdi for y-aksen
80     sensor.setAxelZ((calz / resolution) / -1); // Kalkuleringsverdi for z-aksen
81     Serial.println("Akselerometret er ferdig kalibrert"); // Skriv en melding til seriell monitor for å indikere at kalibreringen er fullført
82 }
83
84 // Funksjon for å initialisere GPS
85 void initialize_GPS() {
86     GPS.begin(9600);
87     /* Setter opp GPS tilkobling, INC setter opp informasjon om
88     posisjonsoppklaring, mens GGA setter opp informasjon om
89     GPS har en fix, antall satellitter og kvalitet */
90     GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
91     // Setter oppdateringsfrekvensen på Hz
92     GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);
93     Serial.println("GPS er ferdig initialisert");
94 }
95
96 // Funksjon for å initialisere akselerometret
97 void initialize_sensoren()
98 {
99     delay(100); // Vent i 100 millisekunder for å sikre at sensoren er klar
100    // Prøv å vekke sensoren til den svarer
101    while (!sensor.wakeUp()) {
102        Serial.println("Kan ikke koble til GY521! Sjekk GY521 adresse (0x68/0x69)"); // Hvis sensoren ikke svarer, skriv en feilmelding til seriell monitor
103        delay(1000); // Vent i 1 sekund før du prøver igjen
104    }
105    sensor.setAccelSensitivity(); // Sett akselerometersensitiviteten til 0
106    sensor.setThrottle(); // Aktiver akselerometrets throttle-funksjon
107    Serial.println("Akselerometret er ferdig initialisert"); // Skriv en melding til seriell monitor for å indikere at initialiseringen er fullført
108 }

```

Figur 4: Setupfunksjoner

Scorefunksjoner:

```

1 /////////////////////////////////////////////////////////////////////
2 /////////////////////Score-Funksjon///////////////////
3 /////////////////////Score-Funksjon///////////////////
4 const float AccelerationThreshold = (0.3); //Dette er øvre grense på hva god akselerasjon er, det vil si er du over det så blir det sett på "dårlig" kjøring
5
6 int score = 0; //Sjølv forklarende, dette er poengen du har til no
7
8 float accelerationValue; //Dette er verdien til akselerasjonen som vi for fra akselerometret
9
10 unsigned long elapsedGoodTime = 0; //Variablen som lagrer hvor lenge bilen har kjørt "bra"
11 unsigned long elapsedBadTime = 0; //Variablen som lagrer hvor lenge bilen har kjørt "dårlig"
12 unsigned long elapsedVeryBadTime = 0; //Variablen som lagrer hvor lenge bilen har kjørt "ekstremt dårlig"
13 unsigned long elapsedStandStillTime = 0; //Variablen som lagrer hvor lenge bilen har stått i ro
14
15 unsigned long lastGoodTime = 0; //Variabel som husker når den sist kjørte "bra"
16 unsigned long lastBadTime = 0; //Variabel som husker når den sist kjørte "dårlig"
17
18 unsigned long currentTime; //Variabel for millis
19 unsigned long lastTime = 0;
20
21
22 void restrainScore() { //Funksjon som begrenser variablene score slik at den ikke overstiger 100 eller 0
23     if (score >= 100) {
24         score = 100;
25     }
26
27     if (score <= 0) {
28         score = 0;
29     }
30 }
31
32
33 void calculateTime(){ //Funksjon som teller hvor lenge bilen har kjørt hverken "bra", "dårlig" eller "ekstremt dårlig"
34     currentTime = millis();
35
36     if (toggleState == 1) { //Ser først om boksen er på
37         if ((accelerationValue < AccelerationThreshold) && standstill == false) { //Viss bilen er under grensen så teller den hvor lenge den kjører "bra"
38             elapsedGoodTime += (currentTime - lastTime);
39             lastTime = currentTime;
40             lastGoodTime = currentTime;
41
42             if (elapsedGoodTime > 12000) { //Nullstiller tellingen etter den har pasert 2 minutter
43                 elapsedGoodTime = 0;
44             }
45         }
46
47         else if ((accelerationValue > AccelerationThreshold) && standstill == false) { //Viss bilen er over grensen så teller den hvor lenge den har kjørt "dårlig"
48             elapsedBadTime += (currentTime - lastTime);
49             lastTime = currentTime;
50             lastBadTime = currentTime;
51
52             if (elapsedBadTime > 5000) { //Nullstiller tellingen etter den har pasert 5 sekunder
53                 elapsedBadTime = 0;
54             }
55         }
56
57         else if ((accelerationValue > AccelerationThreshold * 1.5) && standstill == false) { //Viss bilen er veldig mye over grensen så teller den hvor lenge den har kjørt "veldig dårlig"
58             elapsedVeryBadTime += (currentTime - lastTime);
59             lastTime = currentTime;
60
61             if (elapsedVeryBadTime > 2000) { //Nullstiller tellingen etter den har pasert 2 sekunder
62                 elapsedVeryBadTime = 0;
63             }
64         }
65
66         else if (standstill == true) { //Viss bilen står i ro så vil den begynne å teller hvor lenge den står i ro
67             elapsedStandStillTime = (currentTime - lastTime);
68             lastTime = currentTime;
69
70             if (elapsedStandStillTime > 24000) { //Nullstiller telleren etter den har pasert 4 minutter
71                 elapsedStandStillTime = 0;
72             }
73         }
74
75         else if ((accelerationValue > AccelerationThreshold) && ((currentTime - lastBadTime) > 90000)) { //Viss bilen kjører "bra" og den har kjørt "bra" i 90 sekunder så vil den nullstille teller til dårlig tid
76             elapsedBadTime = 0;
77             elapsedVeryBadTime = 0;
78         }
79
80         else if ((accelerationValue > AccelerationThreshold) && ((currentTime - lastGoodTime) > 10000)) { //Viss bilen kjører "dårlig" og den har kjørt "dårlig" i 10 sekunder så vil den nullstille teller til bra tid
81             elapsedGoodTime = 0;
82         }
83
84         else {
85             lastTime = currentTime; //Viss boksen er av så vil den bare oppdatere siste tid til no tid
86         }
87     }
88 }
89
90
91 void calculateScore(float accelerationValue, unsigned long elapsedGoodTime, unsigned long elapsedBadTime, unsigned long elapsedVeryBadTime, unsigned long elapsedStandStillTime) { //Funksjon som oppdaterer poengen til brukeren
92     if (toggleState == 1) { //Ser om boksen skal være på
93         if ((accelerationValue < AccelerationThreshold) && (elapsedGoodTime > 12000)) { //Viss bilen er under grensen og har kjørt "bra" i 2 minutter til sammen så vil poengen til brukeren gå opp med 1
94             score++;
95         }
96
97         else if (((accelerationValue > AccelerationThreshold) && (elapsedBadTime > 5000)) { //Viss bilen er over grensen og har kjørt "dårlig" i 5 sekunder til sammen så vil poengen til brukeren gå ned med 1
98             score--;
99         }
100
101         else if (((accelerationValue > AccelerationThreshold * 1.5)) && (elapsedVeryBadTime > 2000)) { //Viss bilen er veldig mye over grensen og har kjørt "veldig dårlig" i 2 sekunder så vil poengen til brukeren gå ned med 5
102             score = (score - 5);
103         }
104         else if (standstill == true && elapsedStandStillTime > 24000) { //Viss bilen står i ro og har stått i ro i 4 minutter så vil poengen til brukeren gå ned med 5
105             score = (score - 5);
106         }
107     }
108 }

```

Figur 5: Scorefunksjoner

Hovedkode:

```

1  #include "setupFunctions.h"
2  #include "buttonFunctions.h"
3  #include "scoreFunctions.h"
4  #include "oledFunctions.h"
5
6  /* Denne lagrer "context" som dynamisk minne på
7   * størrelse på 50 bytes. "malloc" reserverer minnet
8   * til heapen som er minne som man kan bruke under kjøring
9   * samt er dynamisk som gjør det mulig å endre verdien under
10  * kjøring av programmet, dette brukes til å lagre
11  * lokasjonen man får ut */
12  char* context = (char*)malloc(sizeof(char) * 50);
13
14  /* konverterer koordinatene fra grader og minutter om til
15  * desimal grader */
16  double konverterTilDesimalgrader(double koordinat) {
17      // henter ut koordinatene og fjerner desimaler, deretter deles det på 100 for å få ut antall hele grader
18      int grader = static_cast<int>(koordinat) / 100;
19      // fmod deler koordinatet på 100 og gir ut restene fra dette, dette er da desimalminutter
20      double desimalMinutter = fmod(koordinat, 100.0);
21      // adderer sammen helgradene og minuttene, først gjøre om desimalminutter --> desimalgrader ved å dele på 60
22      return grader + (desimalMinutter / 60.0);
23  }
24
25  void setup()
26  {
27      pinMode(buttonPin, INPUT_PULLUP);
28      Wire.begin();
29      Serial.begin(115200);
30      initialize_GPS();
31      initialize_sensor();
32      calibrate_accelerometer();
33      internet_connection();
34      ubidots_setup();
35      u8g2.begin();
36      // ubidots.setCallback(score_callback);
37  }
38
39
40  void ubidots_connection()
41  {
42      //om forbindelsen mistes koble til på nytt
43      if (!ubidots.connected())
44      {
45          ubidots.reconnect();
46          ubidots.subscribeLastValue(DEVICE_LABEL_akselerometer, VARIABLE_akselerometer_verdi);
47          ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_gps);
48          ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_satellitt);
49          ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_satellitt_kvalitet);
50          ubidots.subscribeLastValue(DEVICE_LABEL_akselerometer, VARIABLE_score);
51          ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_gps_speed);
52          ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_gps_speed_breach_lat);
53          ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_gps_speed_breach_long);
54      }
55  }
56
57
58  void gps_connection()
59  {
60      //nye gps data mottatt og leses av
61      GPS.read();
62      //hvis gps ikke finner posisjonen
63      if (GPS.fix == false)
64      {
65          static unsigned long previousMillis = 0;
66          const unsigned long interval = 2500; // Intervallet mellom hvert utskrift, i millisekunder
67          unsigned long currentMillis = millis();
68
69          if (currentMillis - previousMillis >= interval) {
70              // printer ut hvert 2,5 sekund dersom den ikke har en posisjon
71              Serial.println("Finner ikke posisjon");
72              previousMillis = currentMillis;
73          }
74      }
75  }
76
77  float acceleration_total(){ //Funksjon som henter ut akselerasjonsdata i x,y,z retning for å lage en akselerasjonsvektor som logger total akselerasjon.
78  sensor.read();
79  float ax = sensor.getAccelX(); //Henter akselerasjon i x retning
80  float ay = sensor.getAccelY(); //Henter akselerasjon i y retning
81  float az = sensor.getAccelZ(); //Henter akselerasjon i z retning
82  float accelerationValue = sqrtf((ax*ax)+(ay*ay)+(az*az)); // Akselerasjonsvektoren blir laget
83  return accelerationValue;
84  }
85
86  void ubidots_publish()
87  {
88      static unsigned long previousMillis = 0;
89      const unsigned long interval = 2500; // Intervallet mellom hvert utskrift, i millisekunder
90      unsigned long currentMillis = millis();
91
92      if (currentMillis - previousMillis >= interval) {
93          previousMillis = currentMillis;
94
95          // Dekrypterer NMEA-settningen slik at vi får ut informasjon vi ønsker
96          if (GPS.parse(GPS.lastNMEA)) {
97              // fortelles oss om kvalitet, fra 1 - 6
98              Serial.print("Kvalitet: ");
99              Serial.println(GPS.fixQuality);
100
101             // viser hvor mange satellitter den er koblet til
102             Serial.print("Antall satellitter: ");
103             Serial.println((int)GPS.satellites);
104
105             if (GPS.fix == true) {
106                 // Konverterer grader og minutter til desimalgrader
107                 double breddegrader = konverterTilDesimalgrader(GPS.latitude);
108                 double lengdegrader = konverterTilDesimalgrader(GPS.longitude);
109
110             }
111         }
112     }
113  }

```

Figur 6: Hovedkode del 1

```

197     // Konverterer til desimalgrader (fra S. lengdegrad)
198     breddegrader = konverterTilDesimalgrader(GPS.longitude);
199
200     // Print ut lokasjon i decimal grader
201     Serial.print(breddegrader, 8);
202     Serial.print(","); // Legg til komma her
203     Serial.print(lengdegrader, 8);
204     Serial.println(".");
205
206     // Konverterer desimalgrader til strenger
207     // Lager en char med plass til 20 tegn for å lagre lokasjonen i
208     char str_lat[20];
209     char str_lng[20];
210
211     //sprintf(str_lat, "%f", breddegrader);
212     //sprintf(str_lng, "%f", lengdegrader);
213
214     // Kvalitet og antall satellitter
215     int satellitt = (int)GPS.satellites;
216     int kvalitet = GPS.fixquality;
217     int gps_speed = GPS.speed;
218
219     // Henter ut akselerasjon verdiene
220     acceleration_total();
221     float akselerasjon = acceleration_total(); // Example temperature value
222
223     calculateScore(accelerationValue, elapsedGoodTime, elapsedBadTime, elapsedVeryBadTime, elapsedStandStillTime);
224
225     // Legge inn knappetrykk her
226     if (toggleState == 1) {
227         Serial.println(toggleState);
228         // Legger til context som "lat" og "lng" fra breddegrad og lengdegrad
229         ubidots.addContext("lat", str_lat);
230         ubidots.addContext("lng", str_lng);
231         // Sender context som en gps data til ubidots
232         ubidots.getContext(context);
233         ubidots.add(VARIABLE_akselerometre_verdi, akselerasjon);
234         ubidots.add(VARIABLE_score, score);
235         ubidots.add(VARIABLE_gps, 1, context);
236         ubidots.add(VARIABLE_satellitt, satellitt);
237         ubidots.add(VARIABLE_satellitt_kvalitet, kvalitet);
238         ubidots.add(VARIABLE_gps_speed, gps_speed);
239
240         // Lagre lokasjon dersom akselerasjonen (i g krefter) overstiger limitten
241         if (akselerasjon > 0.35)
242         {
243             ubidots.add(VARIABLE_gps_speed_breach_lat, GPS.latitude);
244             ubidots.add(VARIABLE_gps_speed_breach_long, GPS.longitude);
245         }
246
247         // Forteller oss om data kom trygt frem til ubidots
248         int publishResult = ubidots.publish(DEVICE_LABEL_gps);
249         if (publishResult == 1)
250         {
251             Serial.println("Publisert på ubidots");
252             ubidots.publish(DEVICE_LABEL_akselerometre);
253         }
254         else
255         {
256             Serial.print("Fikk ikke til å publisere, feilkode: ");
257             Serial.println(publishResult);
258         }
259     }
260
261     }
262
263 }
264
265 void loop()
266 {
267     // Knapp funksjon
268     toggleState = toggleStateFunction(); // Update toggleState based on button press
269     // ubidots tilkobling
270     ubidots_connection();
271     // Akselerasjon
272     acceleration_total();
273     // Score
274     restrainScore();
275     calculateTime();
276     calculateScore(accelerationValue, elapsedGoodTime, elapsedBadTime, elapsedVeryBadTime, elapsedStandStillTime);
277     // OLED
278     OLEDskjerm();
279     // GPS og ubidots
280     GPS_connection();
281     ubidots_publish();
282     ubidots.loop();
283 }

```

Figur 7: Hovedkode del 2

I denne nye iterasjonen har vi fikset en kode som bruker en funksjon fra gps-modulen som heter gps-speed som tar inn hvor fort gps-en beveger seg. Vi bruker denne med score slik at scoren ikke påvirkes når gps-speed er null. I tillegg har vi en funksjon som sender koordinatene til de groveste overtrampene til Ubidots, denne informasjonen kan kommunen bruke for å se om det er områder i byen som skiller seg ut når det kommer til å kjøre for fort. Denne grensen er satt til 0.35 i g-krefter, etter tester vi har gjennomført. Testene ble gjennomført i egen bil og under kontrollerte forhold.

3 Test resultat og diskusjon

Figur viser at scoren ikke blir påvirket når bilden står i ro slik som den gjorde ved forrige iterasjon. Dette var akkurat det vi ville med denne funksjonen.



Figur 8: Score går ikke opp når bil står i ro

Vi kan også se på figur x at Ubidots har fått sendt inn koordinater for der det skjedde et grovt overtramp. Dette regnes som ekstremt ueffektiv kjøring og bruker mye energi, som er det vi er ute etter å redusere.

Akselerasjon overtramp	
VARIABLE NAME	AKSELERASJONS OVERTRAMP
gps_speed_breach_lat	63,417635
gps_speed_breach_long	10,406337

Figur 9: Overtramp koordinater i Ubidots

Her en boksen vi bygget for å holde på alle modulene, denne er en fin og moderne boks med en fin look til den.



Figur 10: Boks som inneholder moduler

4 Begrensninger og tiltak

Begrensninger til prototype 5:

- GPS er treg å finne nøyaktig posisjon når skrudd på.
- Produktet trenger konstant tilgang til internett.

Tiltak:

- Kunne fikset en bedre GPS-modul, men dette var ikke tilgjengelig for oss.
- Ha en 4G-modul i boksen slik at den (nesten) alltid har tilgang til internett.

5 Konklusjon

I denne iterasjonen har vi fikset de fleste svakheter vi fant ved prototype 4. Vi fikk fikset problemet med at scoren gikk opp når bilen står i ro. Vi fikset også en boks til produktet og

printet på logo slik at den får en fin look. Til slutt fikset vi en funksjon som sender posisjon til de groveste overtrampene slik vi ville ha.