Prototype - notat

Tittel: Iterasjon 4	
Forfattere: Prosjekt - gruppe 2	
Versjon: 1.0	Dato: 24.04.2024

Innhold

1 Introduksjon **2**

2 Metode og testmiljø **2**

3 Test resultat og diskusjon **9**

4 Begrensninger og tiltak **9**

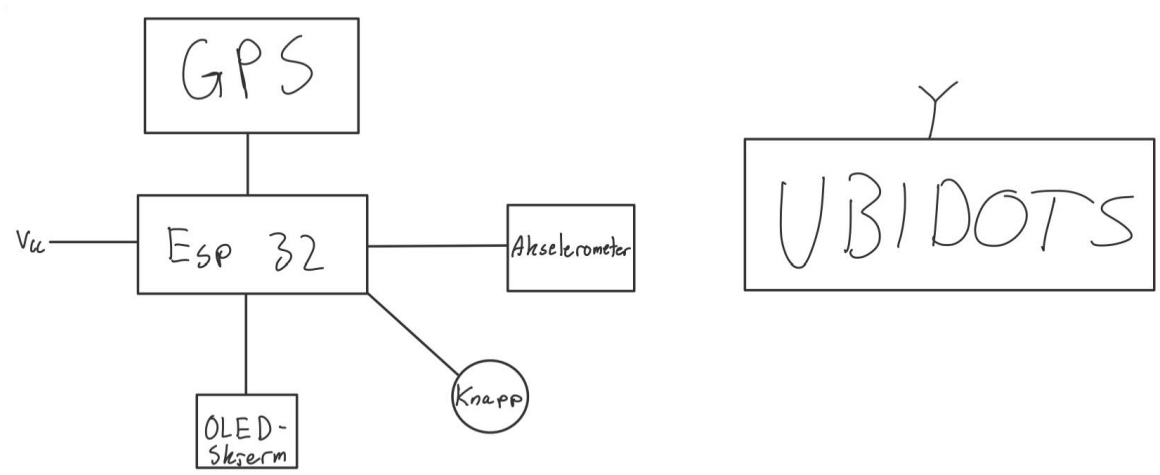
5 Konklusjon **9**

A Ekstra greier **10**

---

1 Introduksjon

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



Figur 1: Prototype iterasjon 4

I denne iterasjonen skal vi gjøre det mulig å bestemme når ting skal publiseres til Ubidots slik at det ikke sendes data hele tiden. Vi skal også implementere en "score" funksjon som overvåker kjøremønsteret og gir oss en mulighet til å belønne "god" kjøring og straffe "dårlig" kjøring.

2 Metode og testmiljø

Under kan du se den nye og oppdaterte koden for denne iterasjonen av prosjektet. Den er delt i fire header filer samt en hovedkode. Header filene inneholde funksjonene til setup, oled-skjermen, knappen og score.

Setup funksjonene så slik ut:

```cpp
/* Header fil som inneholder alle
de ulike setup - funksjonene */

#include <Adafruit_GPS.h>
#include <Arduino.h>
#include <UbidotsEsp32Mqtt.h>
#include "GY521.h"

#define GPSSerial Serial2
Adafruit_GPS GPS(&GPSSerial);

GY521 sensor(0x68);//I2C port for kommunikasjon

// Ubidots credentials
const char *UBIDOTS_TOKEN = "BBUS-4GTYhGckwBnhENTwIxRPtELvX1wWsR";
const char *WIFI_SSID = "NTNU-IOT";
const char *WIFI_PASS = "";

// GPS ubidots labels
const char *DEVICE_LABEL_gps = "gps_position";
const char *VARIABLE_gps = "gps";
const char *VARIABLE_satelitt = "satelitter";
const char *VARIABLE_satelitt_kvalitet = "kvalitet";

//akselerometer ubidots labels
const char *DEVICE_LABEL_akselerometer= "akselerometer";
const char *VARIABLE_akselerometer_verdi = "aks_verdi";
const char *VARIABLE_score = "score";

Ubidots ubidots(UBIDOTS_TOKEN);


void ubidots_setup()
{
  // ubidots connection
  ubidots.setup();
  ubidots.reconnect();
  ubidots.subscribeLastValue(DEVICE_LABEL_akselerometer, VARIABLE_akselerometer_verdi);
  ubidots.subscribeLastValue(DEVICE_LABEL_akselerometer, VARIABLE_score);
  ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_gps);
  ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_satelitt);
  ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_satelitt_kvalitet);
}

void internet_connection()
{
  // koble til internettet
  Serial.println("Initializing Wi-Fi...");
  ubidots.connectToWifi(WIFI_SSID, WIFI_PASS);
  if (ubidots.connected()) {
    Serial.println("Wi-Fi connected!");
  } else {
    Serial.println("Failed to connect to Wi-Fi!");
  }

}

// Funksjon for å kalibrere akselerometeret
void calibrate_accelerometer() {
  Serial.println("Kalibrerer akselerometer");
  int resolution = 100;
  float calx = 0, caly = 0, calz = 0;

  for (int i = 0; i < resolution; i++) {
    sensor.read();
    calx += sensor.getAccelX();
    caly += sensor.getAccelY();
    calz += sensor.getAccelZ();
  }

  sensor.axe = -calx / resolution;
  sensor.aye = -caly / resolution;
  sensor.aze = -calz / resolution;
  Serial.println("Akselerometer er ferdig kalibrert");
}


// Funksjon for å initialisere GPS
void initialize_GPS() {
  GPS.begin(9600);
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);
  Serial.println("GPS er ferdig initialisert");
}

// Funksjon for å initialisere akselerometeret
void initialize_sensor() {
  delay(100);
  while (!sensor.wakeup()) {
    Serial.println("Kan ikke koble til GY521: Sjekk GY521 addresse (0x68/0x69)");
    delay(1000);
  }
  sensor.setAccelSensitivity(0);
  sensor.setThrottle();
  Serial.println("Akselerometer er ferdig initialisert");
}
```

Figur 2: Setup funksjoner

Oled funksjonene så slik ut:

```
#include <U8g2lib.h>

// Setup for SH1106 display ved bruk av U8G2 biblioteket
U8G2_SH1106_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);

float midlertidlig = 0; // Global variabel for total akselerasjon

// OLED Variabler
//int score = 0;  // Eksempel for score verdi


void OLEDSkjerm() {
  unsigned long currentMillis = millis();
  unsigned long previousMillis = 0;  // Lagrer sist gang displayet ble oppdatert
  const long interval = 1000;  // Interval som displayet skal oppdateres på (millisekunder)
  if(toggleState == 1){
    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;

      u8g2.clearBuffer();
      u8g2.setFont(u8g2_font_ncenB08_tr);

      char displayText[30];  // Økt bufferstørrelse for lengre tekst
      sprintf(displayText, "Akselerasjon: %.2f m/s^2", midlertidlig);  // Formaterer tekst med atot

      int textWidth = u8g2.getUTF8Width(displayText);
      int x = (u8g2.getDisplayWidth() - textWidth) / 2;
      int y = u8g2.getDisplayHeight() / 2;

      u8g2.drawStr(x, y, displayText);
      u8g2.sendBuffer();
    }
  }
  else{
    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;

      u8g2.clearBuffer(); //Clear buffer
      u8g2.setFont(u8g2_font_ncenB08_tr);  // Setter font

      char displayText[30];  // Økt bufferstørrelse for lengre tekst
      sprintf(displayText, "Akselerometer er av.");  // Formaterer tekst med atot

      int textWidth = u8g2.getUTF8Width(displayText); //Setter teksten i midten av OLED skjermen
      int x = (u8g2.getDisplayWidth() - textWidth) / 2;
      int y = u8g2.getDisplayHeight() / 2;

      u8g2.drawStr(x, y, displayText);
      u8g2.sendBuffer();
    }
  }
}
```

Figur 3: Oled funksjoner

Knappe funksjonen så slik ut:

```
// Pin definitions
const int buttonPin = 33;  // The pin where the button is connected

// Variables to handle debouncing
unsigned long lastDebounceTime = 0;  // the last time the output pin was toggled
unsigned long debounceDelay = 50;    // the debounce time; increase if the output flickers

// State tracking variables
int lastButtonState = LOW;  // the previous reading from the input pin
int buttonState = LOW;      // the current reading from the input pin

// Toggle variable
int toggleState = 1;  // Variable to keep track of the toggle state

// Function to check the button press, debounce, and toggle the state
int toggleStateFunction() {
  static int internalToggleState = toggleState;  // Store toggle state across function calls
  int reading = digitalRead(buttonPin);  // Read the state of the button

  if (reading != lastButtonState) {
    // Reset the debouncing timer
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != buttonState) {
      buttonState = reading;
      lastDebounceTime = millis();  // Update the time of the change

      if (buttonState == HIGH) {
        internalToggleState = 1 - internalToggleState;  // Toggle the state
      }
    }
  }

  lastButtonState = reading;  // Update the last button state
  return internalToggleState;
}
```

Figur 4: Knappe funksjoner

Score funksjonen så slik ut:

```
////////////////////////////////////////////////////////////////
//////////////////////////Score-Funksjon//////////////////////////
////////////////////////////////////////////////////////////////
const int goodPosotivAccelerationThreshold = 8; //This means that if the acceleration is bellow 8 m/s^2 its a objectively an effective drive
const int goodNegativeAccelerationThreshold = -8; //This means that if the acceleration is higher then -8 m/s^2 its a objectively an effective brake

int score = 0;

float accelerationValue;

unsigned long elapsedGoodTime = 0;  //Tracking how long it has drived "Good"
unsigned long elapsedBadTime = 0;  //Tracking how long it has drived "Bad"
unsigned long elapsedVeryBadTime = 0;  //Tracking how long it has drived "Very Bad"

unsigned long lastGoodTime = 0;  //Remebers when the last time it drove "Good"
unsigned long lastBadTime = 0; //Remebers when the last time it drove "Bad"

unsigned long startTime = 0;
unsigned long currentTime;


void restrainScore(){
  if(score > 100){
    score = 100;
  }

  if(score < 0){
    score = 0;
  }
}


void calculateTime(){
  currentTime = millis();

  if(goodNegativeAccelerationThreshold < accelerationValue < goodPosotivAccelerationThreshold){
    elapsedGoodTime += (currentTime - startTime);
    startTime = currentTime;
    lastGoodTime = currentTime;

    //Serial.println(elapsedGoodTime);

    if(elapsedGoodTime > 60031){
      elapsedGoodTime = 0;
    }
  }

  else if(goodNegativeAccelerationThreshold > accelerationValue > goodPosotivAccelerationThreshold){
    elapsedBadTime += (currentTime - startTime);
    startTime = currentTime;
    lastBadTime = currentTime;

    if(elapsedBadTime > 15031){
      elapsedGoodTime = 0;
    }
  }

  else if((goodNegativeAccelerationThreshold * 2) > accelerationValue > (goodPosotivAccelerationThreshold * 2)){
    elapsedVeryBadTime += (currentTime - startTime);
    startTime = currentTime;

    if(elapsedVeryBadTime > 5031){
      elapsedVeryBadTime = 0;
    }
  }

  else if((goodNegativeAccelerationThreshold < accelerationValue < goodPosotivAccelerationThreshold) && ((currentTime - lastBadTime) > 10000)){
    elapsedBadTime = 0;
    elapsedVeryBadTime = 0;
    lastBadTime = currentTime;
  }

  else if((goodNegativeAccelerationThreshold > accelerationValue > goodPosotivAccelerationThreshold) && ((currentTime - lastGoodTime) > 10000)){
    elapsedGoodTime = 0;
    lastGoodTime = 0;
  }
}


void calculateScore(float accelerationValue, unsigned long elapsedGoodTime, unsigned long elapsedBadTime, unsigned long elapsedVeryBadTime){
  // Implement your scoring algorithm here, considering stat value and time
  // Return the updated score
  if((goodNegativeAccelerationThreshold < accelerationValue < goodPosotivAccelerationThreshold) && (elapsedGoodTime > 60000)){
    score ++;
  }

  else if((goodNegativeAccelerationThreshold > accelerationValue > goodPosotivAccelerationThreshold) && (elapsedBadTime > 15000)){
    score --;
  }

  else if(((goodNegativeAccelerationThreshold * 2) > accelerationValue > (goodPosotivAccelerationThreshold * 2)) && (elapsedVeryBadTime > 5000)){
    score -= 5;
  }
}
```

Figur 5: Score funksjoner

Hovedkoden så slik ut:

```ino
#include "setupFunctions.h"
#include "buttonFunctions.h"
#include "scoreFunctions.h"
#include "oledFunctions.h"


/* Denne lagrer "context" som dynamisk minne på
størrelse på 50 bytes. "malloc" reserverer minnnet
til heapen som er minne som en kan bruke under kjøring
samt er dynamisk som gjør det mulig å endre verdien under
kjøring av programmet */
char* context = (char*)malloc(sizeof(char) * 50);

/* konverterer koordinatene fra grader og minutter om til
desimal grader */
double konverterTilDesimalgrader(double koordinat) {
  int grader = static_cast<int>(koordinat) / 100;
  double desimalMinutter = fmod(koordinat, 100.0);
  return grader + (desimalMinutter / 60.0);
}

void setup()
{
  pinMode(buttonPin, INPUT_PULLUP);
  Wire.begin();
  Serial.begin(115200);
  initialize_GPS();
  initialize_sensor();
  calibrate_accelerometer();
  internet_connection();
  ubidots_setup();
  u8g2.begin();
}


void ubidots_connection()
{
  //om forbindelsen mistes koble til på nytt
  if (!ubidots.connected())
  {
    ubidots.reconnect();
    ubidots.subscribeLastValue(DEVICE_LABEL_akselerometer, VARIABLE_akselerometer_verdi);
    ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_gps);
    ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_satelitt);
    ubidots.subscribeLastValue(DEVICE_LABEL_gps, VARIABLE_satelitt_kvalitet);
    ubidots.subscribeLastValue(DEVICE_LABEL_akselerometer, VARIABLE_score);
  }
}
```

```ino

void gps_connection()
{
  //nye gps data mottat og leses av
  //if (GPS.newNMEAreceived()) {
  GPS.read();
  //}

  //hvis gps ikke finner posisjonen
  if (GPS.fix == false)
  {
    static unsigned long previousMillis = 0;
    const unsigned long interval = 2500; // Intervallet mellom hvert utskrift, i millisekunder
    unsigned long currentMillis = millis();

    if (currentMillis - previousMillis >= interval) {
      Serial.println("Finner ikke posisjon");
      previousMillis = currentMillis;
    }
    //delay(2500);
  }
}

float acceleration_total(){ //Funksjon som henter ut akselerasjonsdata i x,y,z retning for å lage en akselerasjonsvektor som logger total akseleraajon.
  sensor.read();
  float ax = sensor.getAccelX(); //Henter akselerasjon i x retning
  float ay = sensor.getAccelY(); //Henter akselerasjon i y retning
  float az = sensor.getAccelZ(); //Henter akselerasjon i z retning
  float accelerationValue = sqrt(sq(ax)+sq(ay)+sq(az)); // Akselerasjonsvektoren blir laget

  /*Serial.print("  Akselerasjon din er "); //Printer ut akselerasjonsvektoren
  Serial.print(atot);
  Serial.println(" m/s^2");*/
  return accelerationValue;
}

void ubidots_publish() {
  static unsigned long previousMillis = 0;
  const unsigned long interval = 2500; // Intervallet mellom hvert utskrift, i millisekunder
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
```

```
92
93      // Dekrypterer NMEA-setningen
94      if (GPS.parse(GPS.lastNMEA())) {
95        // fortelles oss om kvalitet, fra 1 - 6
96        Serial.print("Kvalitet: ");
97        Serial.println(GPS.fixquality);
98
99        // sier hvor mange satelitter den er koblet til
100       Serial.print("Antall satellitter: ");
101       Serial.println((int)GPS.satellites);
102
103       if (GPS.fix == true) {
104         // Konverterer grader og minutter til desimalgrader
105         double breddegrader = konverterTilDesimalgrader(GPS.latitude);
106         double lengdegrader = konverterTilDesimalgrader(GPS.longitude);
107
108         // print ut lokasjon i decimal grader
109         Serial.print(breddegrader, 8);
110         Serial.print(", "); // Legg til komma her
111         Serial.print(lengdegrader, 8);
112         Serial.println(".");
113
114         // Konverterer desimalgrader til strenger
115         char str_lat[20];
116         char str_lng[20];
117         sprintf(str_lat, "%f", breddegrader);
118         sprintf(str_lng, "%f", lengdegrader);
119
120         // legger til context som "lat" og "Lng" fra breddegrad og lengdegrad
121         //ubidots.addContext("lat", str_lat);
122         //ubidots.addContext("lng", str_lng);
123
124         // Kvalitet og antall satelitter
125         int satelitt = (int)GPS.satellites;
126         int kvalitet = GPS.fixquality;
127
128         // Sender context som er gps data til ubidots
129         //ubidots.getContext(context);
130         //ubidots.add(VARIABLE_gps, 1, context);
131         //ubidots.add(VARIABLE_satelitt, satelitt);
132         //ubidots.add(VARIABLE_satelitt_kvalitet, kvalitet);
133
134         // Henter ut akselerasjon verdiene
135         acceleration_total();
136         float akselerasjon = acceleration_total(); // Example temperature value
137         //ubidots.add(VARIABLE_akselerometer_verdi, akselerasjon);
138
139         calculateScore(accelerationValue, elapsedGoodTime, elapsedBadTime, elapsedVeryBadTime);
140
```

```
141           // Legge inn knappetrykk her
142           if (toggleState == 1) {
143             Serial.println(toggleState);
144             ubidots.addContext("lat", str_lat);
145             ubidots.addContext("lng", str_lng);
146             ubidots.getContext(context);
147             ubidots.add(VARIABLE_akselerometer_verdi, akselerasjon);
148             ubidots.add(VARIABLE_score, score);
149             ubidots.add(VARIABLE_gps, 1, context);
150             ubidots.add(VARIABLE_satelitt, satelitt);
151             ubidots.add(VARIABLE_satelitt_kvalitet, kvalitet);
152             //ubidots.add(VARIABLE_score, score);
153
154             ubidots.publish(DEVICE_LABEL_akselerometer);
155
156             // Forteller oss om data kom trygt frem til ubidots
157             int publishResult = ubidots.publish(DEVICE_LABEL_gps);
158             if (publishResult == 1) {
159               Serial.println("Publisert på ubidots");
160             } else {
161               Serial.print("Fikk ikke til å publisere, feilkode: ");
162               Serial.println(publishResult);
163             }
164           }
165           else
166           {
167             Serial.print("Publiserer ikke til Ubidots");
168           }
169         }
170       }
171     }
172   }
173
174
175
176   void loop()
177   {
178     //knapp funksjon
179     toggleState = toggleStateFunction();  // Update toggleState based on button press
180     // ubidots tilkobling
181     ubidots_connection();
182     //akselerasjon
183     acceleration_total();
184     //score
185     restrainScore();
186     calculateTime();
187     calculateScore(accelerationValue, elapsedGoodTime, elapsedBadTime, elapsedVeryBadTime);
188     //oled
189     OLEDSkjerm();
190     //gps og ubidots
191     gps_connection();
192     ubidots_publish();
193     ubidots.loop();
194
```

I denne nye iterasjonen kan alt av publisering til Ubidots styres av et knappetrykk slik at den ikke konstant publiserer dataene. Dette ble enkelt fikset med en if-setning foran publiseringskoden. Også nytt i denne iterasjonen er et score system som registrerer om brukeren har ”bra” eller ”dårlig” kjøremønster.

Vi testen denne nye koden ved å trykke knappen og se at den gjorde det vi ville, altså printet til Ubidots når knappen var trykket og sluttet og printe når vi trykket på den igjen. Vi sjekket også at score funksjonen virket som vi ville ved å teste den på en kjøretur, hvor vi testet forskjellige kjøremønstre.

3 Test resultat og diskusjon

PÅ figur X ser vi at når knappen var trykket publiserer den som ønsket til ubidots, vi kan også se på figur X at når knappen trykkes igjen slutter den å publisere til Ubidots. Dette var den ønskede funksjonen. BILDE AV AT DET PUBLISERES PÅ UBIDOTS

På figur X ser vi at scoren har gått betydelig ned etter lengre tid med dårlig kjøring. Vi kan også se på figur x at etter en lengre periode med bra kjøring at scoren har gått opp.

BILDE AV SCORE SOM GÅR OPP OG NED VED DÅRLIG OG BRA KJØRING

4 Begrensninger og tiltak

Begrensninger til prototype 4:
- Score går opp hvis bilen står i ro, siden dette er lite akselerasjon og regnes som ”god” kjøring.
- Enda bare et oppkoblet breadboard med alle moduler.
- Mangler en funksjon som sier hvor de værste overtrampene skjer.

Tiltak:
- Bruke ultimate gps funksjon som måler fart til å gi en threshold på hvilke akselerasjonsverdier som burde telles.
- Lage en boks vi kan koble alt opp i slik at det blir ryddig og fint.
- Lage en funksjon som sender hvor de værste overtrampene skjer slik at dette kan overvåkes og kanskje forbedres på sikt.

5 Konklusjon

Vi oppnådde målet om å styre systemet med en knapp og fikset et ”score” funksjon som virket får å overvåke kjøringsmønsteret selv om denne hadde rom for forbedring.

A Ekstra greier

Mindre relevant blabla.