Prototype - notat

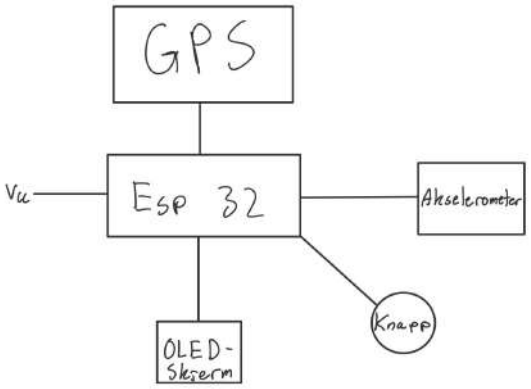
Tittel: Iterasjon 2	
Forfattere: Prosjekt - gruppe 2	
Versjon: 1.0	Dato: 18.04.2024

Innhold

---

1 Introduksjon

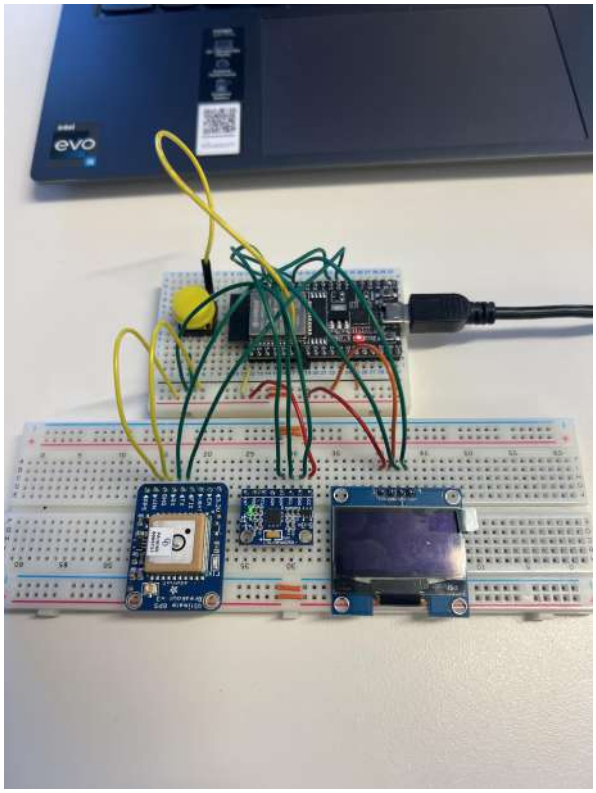
Prototypen er illustrert ved skissen under



Figur 1: Krets Prototype 2

Dette er da en ESP32 koblet til med ulike komponenter, derav en GPS - modul, akselerometer, knapp og en O - LED skjerm. Sammen fungerer dette ved å både tracke posisjonen, samt hente

ut verdien på akselerasjonen i nåtid, dette blir printet ut til OLED skjermen som man kan lese av. Alt dette kan man starte og stoppe med et knappetrykk. Denne kretsen oppkoblet så slik ut:



Figur 2: Oppkoblet Prototype 2

GPS - koordinatene er kodet slik at disse blir printet ut i Serial monitoren til ESP32. Mens akselerasjonen blir displayet på OLED skjermen.

2 Metode og testmiljø

Legger ved kode som ble benyttet for å teste denne prototypen

```ino
#include <Arduino.h>
#include <U8g2lib.h>
#include "GY521.h"
#include <Adafruit_GPS.h>

#define BUTTON_PIN 33 // Definerer Knapp Pin

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

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


//------ GPS -------//
//Kobler GPS kommunikasjon til Serial2 på ESP32
//kan ha den på serial1 med mp ha RX1 og TX1 isÅfall
#define GPSSerial Serial2

//variabel for å få ut GPS til serial kommunikasjon
Adafruit_GPS GPS(&GPSSerial);

// OLED Variabler
int score = 0;  // Eksempel for score verdi
unsigned long previousMillis = 0;  // Lagrer sist gang displayet ble oppdatert
const long interval = 1000;  // Interval som displayet skal oppdateres på (millisekunder)

// Knapp+Akselerometer Variabler
bool lastButtonState = LOW;  // Lagrer the siste status av knappen
bool isOn = false;           // Lagrer nåværende on/off statusen for systemet
unsigned long lastMessageTime = 0; // Variabel for å tracke siste meldings print tid

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


void setup() {
  u8g2.begin();

  Serial.begin(115200);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  Wire.begin();

  delay(100);
  while (sensor.wakeup() == false) //Sjekker oppkobling mot sensor
  {
    Serial.print(millis());
    Serial.println("\Kan ikke koble til GY521: Sjekk GY521 addresse (0x68/0x69)"); //Tilbakemelding blir gitt hvis ESP ikke kan kommunisere med sensoren
    delay(1000);
  }
  sensor.setAccelSensitivity(0);  // Setter sensitivitet på akselerometer til 2g

  sensor.setThrottle();
  Serial.println("Kalibrerer akselerometer");
  int opplosning = 100; //Antall iterasjoner for å kalibrere akselerometer
  float calx = 0; //Variabel definert for x retning
  float caly = 0; //Variabel definert for y retning
  float calz = 0; //Variabel definert for z retning
  for (int i=0; i<opplosning; i++){ //For løkke for kalibrering, tar avlesninger etter hva opplosning settes til
    sensor.read(); //leser sensor
    calx += sensor.getAccelX(); //Lagrer opp avlesninger som en total sum
    caly += sensor.getAccelY();
    calz += sensor.getAccelZ();
  }
  Serial.println("Akselerometer er ferdig kalibrert");

  //  Kalibreringsverdier blir satt inn for å nulle ut akselerometer før bruk.
  sensor.axe = -calx/opplosning; //Deler den totale sommen som er akkumulert over avlsningene på oppløsningen og kalibrerer senseren i x,y,z retning med loggede verdier.
  sensor.aye = -caly/opplosning;
  sensor.aze = -calz/opplosning;

```

```
  // ----- GPS ----- //
  Serial.begin(115200);
  // Starter GPS ved 9600 baud
  GPS.begin(9600);

  /* Denne kommandoen gjør at vi både får sendt ut RMC som står
  for (Recommended Minumum Navigation information) samt GGA som står
  for (Global Positioning System Fix data) som NMEA - setninger */
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);

  // Dette bestemmer oppdateringsfrekvensen til GPS'en
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); // 1 Hz update rate
}

void GPSTracker(){
  /* Denne while setningen sjekker om det er kommet en ny
  NMEA setning med informasjon som skal printes ut */
  while (!GPS.newNMEAreceived()) {
    GPS.read(); // Denne leser av for å se om det kommen noe
  }

  /* henter ut den NMEA informasjonen som ble sist lagret på GPS'en
  og deretter parser denne, dette vil si å analysere NMEA informasjonen
  som er lagret og trekke ut den informasjonen vi ønsker å printe */
  static unsigned long lastPrintTime = 0;  // Holder styr på når siste melding ble sendt
  unsigned long currentTime = millis();
  const unsigned long printInterval = 2500;  // Intervallet mellom meldingene i millisekunder

  if (GPS.parse(GPS.lastNMEA())) {
    if (GPS.fix) {  // Sjekker om GPS har en "fix", altså en posisjon å logge
      if (isOn == true) {
        if (currentTime - lastPrintTime > printInterval) {
          // Printer ut informasjonen som er sendt:
          Serial.print("Lokasjon: ");
          Serial.print(GPS.latitude, 4);
          Serial.print(", ");
          Serial.print(GPS.longitude, 4);
          Serial.println(".");
          lastPrintTime = currentTime;  // Oppdaterer tiden for siste utskrift
        }
      }
      else {
        if (currentTime - lastPrintTime > printInterval) {
          Serial.println("Data printes ikke");
          lastPrintTime = currentTime;  // Oppdaterer tiden for siste utskrift
        }
      }
    }
  }
}

void OLEDSkjerm() {
  unsigned long currentMillis = millis();
  if(isOn == true){
    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", atot);  // Formaterer tekst med atot

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

```
136
137       u8g2.drawStr(x, y, displayText);
138       u8g2.sendBuffer();
139     }
140   }
141   else{
142     if (currentMillis - previousMillis >= interval) {
143       previousMillis = currentMillis;
144
145       u8g2.clearBuffer(); //Clear buffer
146       u8g2.setFont(u8g2_font_ncenB08_tr);  // Setter font
147
148       char displayText[30];  // Økt bufferstørrelse for lengre tekst
149       sprintf(displayText, "Akselerometer er av.");  // Formaterer tekst med atot
150
151       int textWidth = u8g2.getUTF8Width(displayText); //Setter teksten i midten av OLED skjermen
152       int x = (u8g2.getDisplayWidth() - textWidth) / 2;
153       int y = u8g2.getDisplayHeight() / 2;
154
155       u8g2.drawStr(x, y, displayText);
156       u8g2.sendBuffer();
157     }
158   }
159 }
160
161 void programOnOff() {
162   static unsigned long lastDebounceTime = 0;  // Tid siden siste debounce sjekk
163   const unsigned long debounceDelay = 50;     // Debounce delay i millisekunder
164   static bool lastButtonState = HIGH;  // Lagrer siste button state, start som HIGH
165
166   bool currentButtonState = digitalRead(BUTTON_PIN); // Leser nåværende state av knappen
167
168   // Sjekk for endring i button state
169   if (currentButtonState != lastButtonState) {
170     lastDebounceTime = millis();  // reset debounce tid
171   }
172
173
174   if ((millis() - lastDebounceTime) > debounceDelay) {
175     // Om knappen trykes
176     if (currentButtonState == LOW) {
177       isOn = !isOn;  // Endre status til systemet
178     }
179   }
180   lastButtonState = currentButtonState;  // Oppdater siste kjente button state
181 }
182
183 void acceleration_total() {
184   if(isOn == true){
185     sensor.read();
186     float ax = sensor.getAccelX();
187     float ay = sensor.getAccelY();
188     float az = sensor.getAccelZ();
189     atot = sqrt(sq(ax) + sq(ay) + sq(az)); // Oppdaterer den globale variabelen
190
191     /*Serial.print("Akselerasjon din er ");
192     Serial.print(atot);
193     Serial.println(" m/s^2"); */
194   }
195   else {
196     unsigned long currentTime = millis();
197     if (currentTime - lastMessageTime >= 2000) {
198       Serial.println("Akselerometer er av");
199       lastMessageTime = currentTime;
200     }
201   }
202 }
```

```
203
204
205 void loop() {
206   OLEDSkjerm(); // Kaller på funksjon for å printe til OLEDSkjerm
207   programOnOff(); // Kaller på funksjon for å skru av og på program
208   acceleration_total(); // Kaller på funksjon for å logge akselerasjon
209   GPSTracker(); //Kaller på funksjon for å sjekke posisjon
210 }
211
```

Figur 3: Komplett kode

I koden kalibreres akselerometeret, samtidig starter gps-en å lete etter et signal, når den finner denne posisjonen venter den til at knappen trykkes før den printer posisjonen til serial monitoren. Oled skjermen viser "Akselerometer er avnår knappen ikke trykkes, etter knappen trykkes viser den akselerasjons verdien på skjermen. Slik som vist på figur (5).

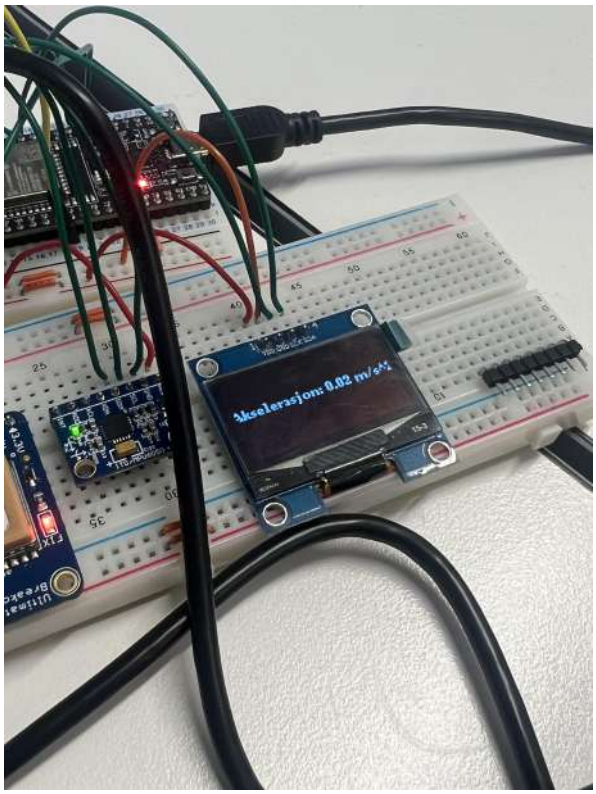
Ved å ha koden kjørende i bakgrunnen kunne vi sjekke både på OLED og Serial monitoren om verdiene stemte. Dette gjorde vi flere ganger der vi også flyttet på oss med varierende fart og lokasjon for å sjekke om den greide å printe rett underveis. Knappen fungerte slik den skulle, om den ble trykket ble det printet ut informasjon, og dersom man trykket igjen stoppet den og informerte om dette i serial monitoren.

3 Test resultat og diskusjon

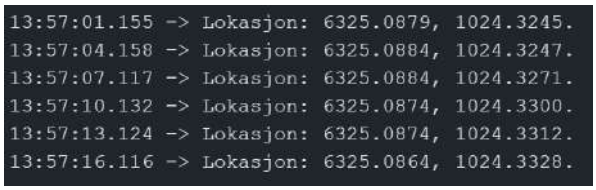
Ved å se på figuren under kan man se verdiene og resultatene vi fikk ut

```
13:48:32.664 -> Akselerometer er av
13:48:34.748 -> Akselerometer er av
13:48:36.748 -> Akselerometer er av
```

Figur 4: Serial monitoren før knappen trykkes



Figur 5: Akselerometerdata printes til OLEDSkjerm når knappen trykkes



Figur 6: Printer lokasjon til Serial monitoren

4 Begrensninger og tiltak

Begrensninger til prototype 2:
- Informasjonen er begrenset til kun serial monitoren.
- Må manuelt taste inn koordinatene for å displaye lokasjonen på et kart.

Eventuelle tiltak:
- Bruke et program som kan displaye lokasjonen i et kart på en oversiktlig måte.

5 Konklusjon

Prototypen funket som den skulle. Fikk printet ut koordinasjonene, sendt informasjon til OLED og alt dette kunne reguleres med et knappetrykk.