Data-Elektronikk “Stødig”

Vi har laget “stødig i flere versjoner. I den første versjonen benyttet vi PLS og styring fra dataskjerm. I versjon benyttet vi en liten µ-kontroller og et display fra en Nokia telefon.
Nå skal vi bygge om på nytt, og benytte ESP32 som har Bluetooth og gir mulighet til styring fra APP.

Mål
Bli kjent med Kicad for å lage skjemategninger for elektronikk.
Bli kjent med grunnleggende komponenter.
Benytte algoritmisk tenking, ved å få inngående kjennskap innenfor et emne man normalt ikke jobber med

Kompetansemål

- planlegge, gjennomføre, vurdere og dokumentere arbeidsoppgaver knyttet til elektroniske kretser og nettverk, individuelt og i samarbeid med andre og begrunne valgene som er gjort
- risikovurdere og utføre arbeidet i overensstemmelse med rutiner for el- og IKT-sikkerhet og helse, miljø og sikkerhet
- utføre arbeidet fagmessig i elektroniske kretser og nettverk i henhold til gjeldende forskrifter, montasje- og installasjonsanvisninger og kunne bruke egnede håndverktøy og maskiner
- velge og bruke egnede instrumenter og programvare for å utføre målinger og feilsøking, og vurdere måleresultatet opp mot forventede verdier
- vurdere kvalitet på eget arbeid og foreslå forbedringer
- anvende og behandle materiell og utstyr i elektroniske kretser og nettverk på en ansvarlig og bærekraftig måte i samsvar med gjeldende system for internkontroll

Komponentliste (Datablad)
ESP32
N-Mosfet

Diverse opplysninger
Seriekommunikasjon
Vi benytter seriekommunikasjon mot PC og over BT

Dekoding av tekst fra serieporten
Når vi mottar en linje over serieporten, så inneholder linja data vi ønsker å sende til BT enhet, vi mottar data fra samme BT enhet (APP) For at disse enhetene skal kunne snakke sammen, så må vi bli enige om en protokoll.

Selve programmet på ESP32

```
#include "BluetoothSerial.h"
#include "driver/gpio.h"

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth er ikke initiert! Kjør 'make menuconfig' for å initsiere BT
#endif


#define LF 10              //Line feed kode
#define CR 13              //Carriage Return


#define bane_A 32           //GPIO32 Input fysisk pinne 7
#define bane_B 33           //GPIO33 Input fysisk pinne 8
#define bane_C 25           //GPIO25 Input fysisk pinne 9
#define bane_D 26           //GPIO26 Input fysisk pinne 10
#define Sirene 14           //GPIO14 Output fysisk pinne 12

BluetoothSerial SerialBT;

// Globale data
// En struktur for å samle globale data for seriekommunikasjon
struct serieportdata
{
  char rxdata[255];       //buffer for motatte tegn over seriporten.
  char txdata[255];       //buffer for tegn som skal sendes over serieporten.
  char *rx_data;
  byte rxpos=0;
  unsigned long rxt=0;    //Timestamp i ms
};
serieportdata serieport;  //Definer en variabel som holder strukturen
serieportdata btport;

struct kommando
{
  char cmdstr[255];
  char tx[255];
  bool ubehandlet=0;
};
kommando cmd;

struct banedata
{
  int poeng=0;          //Antall berøringer
  bool ubehandlet=0;    //Flagg for om siste poengsum er sendt eller ikke
  unsigned int dt=0;    //Tiden for siste sjekk av berøring
  bool touch=0;         //Flagg for om vi har berøring eller ikke
  byte port=0;          //Hvilken GPIO inngangen er koblet til
};

struct loepdata
{
  banedata bane[4];     //4 stk banedata (bane A til D)
  int sum=0;            //Sum antall berøringer
  int forsinkelse=100;  //Delay melleom hver sjekk av berøring.
  bool ferdig=1;        //Flagg for å fortelle om konkurransen er i gang eller ikke
};

loepdata game;          // definerer variablen game til å innholde strukturene laget over


void setup()
{
  Serial.begin(57600);
  serieport.txdata[0]=0;    //Vi setter buffer for transmitt data tomt.

  game.bane[0].port=bane_A;
```

```
  game.bane[1].port=bane_B;
  game.bane[2].port=bane_C;
  game.bane[3].port=bane_D;
  for (int i=0; i<4; i++) pinMode(game.bane[i].port, INPUT_PULLUP);
  pinMode(Sirene,OUTPUT);

  SerialBT.begin("Stoedig_BT");
  Serial.println("Bluetooth er startet");
}



//*****************************************************
// rxData
// Sjekker om det er data på serieporten uten å
// være blokkerende.
//
// Rutinene leser en hel linje avsluttet med #10 /n LF
// (En annen terminering er #13 /r CR
// Eller en kombinasjon av disse to CR LF #10#13 /n/r)
//
// Data motatt legges i buffer rx_data
// Funksjonen returner True (1) dersom en linje er
// motatt og False (0) dersom ingen ferdig linje.
//*****************************************************
bool rxSerial()
{
  char c = (char)Serial.read();
  if (c!=255)
  {
    serieport.rxdata[serieport.rxpos]=c;
    //Serial.println(c,DEC);
    serieport.rxpos++;
    if (serieport.rxpos>254) {serieport.rxpos=0;}
    serieport.rxt=millis();
    if (c == LF)
    {
      serieport.rxdata[serieport.rxpos]=0;
      serieport.rx_data=serieport.rxdata;
      serieport.rxpos=0;
      return(1);
    }
  }
  return(0);
}
//*****************************************************
// txData
// Rutinen sender data som ligger i txdata uten noen
// noen verifikasjon. Men den sørger for et lite delay
// før den sender uten å være blokkerende.
// txdata blir "tømt" etter sending.
//*****************************************************
void txSerial()
{
  if (millis()-serieport.rxt>20) //La bussen være idle i 20ms før vi sender data
  {
    Serial.write(serieport.txdata);
    serieport.txdata[0]=0;
  }
}
//*****************************************************
// rxBT
// Sjekker om det er data på bluetooth uten å
// være blokkerende.
//
// Rutinene leser en hel linje avsluttet med #10 /n LF
// (En annen terminering er #13 /r CR
// Eller en kombinasjon av disse to CR LF #10#13 /n/r)
//
// Data motatt legges i buffer rx_data
```

```
// Funksjonen returner True (1) dersom en linje er
// motatt og False (0) dersom ingen ferdig linje.
//*************************************************
bool rxBT()
{
  if (SerialBT.available())
  {
    char c = (char)SerialBT.read();
    btport.rxdata[btport.rxpos]=c;
    //Serial.println(c,DEC);
    btport.rxpos++;
    if (btport.rxpos>254) {btport.rxpos=0;}
    btport.rxt=millis();
    if (c == LF)
    {
      btport.rxdata[btport.rxpos]=0;
      btport.rx_data=btport.rxdata;
      btport.rxpos=0;
      return(1);
    }
  }
  return(0);
}


void sjekk_kommando()
{
  if (cmd.ubehandlet)  //Betyr at vi har fått en kommando fra serieport eller bt port
  {
    Serial.print("cmd:");       //For debugging
    Serial.println(cmd.cmdstr); //For debugging
    if (cmd.cmdstr[0]=='S' && game.ferdig==1)     //Vi skal starte nytt løp
    {
      strcpy(cmd.tx,"| 0000 | 0000 | 0000 | 0000 || 0000 ||");
      SerialBT.println(cmd.tx);
      Serial.println(cmd.tx);
      for (int i=0; i<4; i++)
      {
        game.bane[i].poeng=0;
        game.bane[i].ubehandlet=0;
      }
      game.sum=0;
      game.ferdig=0;
    }
    else if (cmd.cmdstr[0]=='F')
    {
      strcpy(cmd.tx,"Runde er ferdig");
      SerialBT.println(cmd.tx);
      game.ferdig=1;
      digitalWrite(Sirene,0);
    }
    else Serial.println(cmd.cmdstr);
  }
  cmd.ubehandlet=0;     //Her skal komandoen være behandlet
}

void sjekk_bane(int i)
{
  if (digitalRead(game.bane[i].port)==0)
  {
    game.bane[i].poeng+=1;
    game.bane[i].ubehandlet=1;
    game.bane[i].dt=millis();
    game.bane[i].touch=1;
  }
  else game.bane[i].touch=0;
}

void sjekk_beroring()
{
```

```
  if (game.ferdig==0)
  {
    int j=0;
    int sum=0;
    for (int i=0; i<4; i++)
    {
      if (millis()-game.bane[i].dt>game.forsinkelse) sjekk_bane(i);
      j+=game.bane[i].touch;
      sum+=game.bane[i].poeng;
    }
    if (j>0)
    {
      digitalWrite(Sirene,HIGH);
      for (int i=0; i<4; i++)
      {
        if (game.bane[i].ubehandlet)
        {
          SerialBT.print("P:");
          SerialBT.print(char(65+i));
          SerialBT.print(":");
          SerialBT.print(game.bane[i].poeng,DEC);
          SerialBT.print(":");
          SerialBT.println(sum,DEC);
          game.bane[i].ubehandlet=0;
        }
      }
    }
    else
      digitalWrite(Sirene,LOW);
  }
}

void loop()
{
  if (rxSerial()) //Sjekk om vi har motatt en linje på serieporten
  {
    strcpy(cmd.cmdstr,serieport.rx_data);  //Kopier data fra rx_data til cmdstr
    cmd.ubehandlet=1;                       //Si i fra at hendelsen ikke er behandlet
  }
  if (rxBT())
  {
    strcpy(cmd.cmdstr,btport.rx_data);     //Kopier data fra rx_data til cmdstr
    cmd.ubehandlet=1;                       //Si i fra at hendelsen ikke er behandlet
  }
  sjekk_kommando();
  sjekk_beroring();


}
```

Programmet på APP
https://drive.google.com/drive/folders/14nwcueKfyNZJO5XyrMXD5sEcNsXFO0rR?usp=sharing
Appen er bygget på BT driver og eksempel fra harry1453
https://github.com/harry1453/android-bluetooth-serial

Det er lagt inn noen ekstra knapper og protokoll for start stopp spill og kontinuerlig overføring av poengsum.
Programmet er ikke lagt ut på Android butikken, men må overføres via wiFi eller USB til telefonen.

Ekstra.
Det er vanlig å levere dokumentasjon på pdf format.
Skjemategninger fra KiCad kan skrives ut til .pdf likeså kan evaluering og det meste annet.
Det er mulig å sette sammen .pdf dokumenter til et dokument.
Les om metoden i Dokumentasjon krav