Dette dokument beskriver et bud på hvordan man kan sende et antal bytes via radiomodulet HC12.

Data sendes jo serielt via radiobølger, og der kan meget sandsynligt opstå forstyrrelser undervejs i en transmission.

Fx kan et ”1” blive læst som ”0”, eller omvendt!

Eller der kan mistes data undervejs – måske på grund af interferens, dvs. at der er andre sendere, der samtidigt sender på samme frekvens.

Ligeledes kan der opstå ” ude af sync ” problemer.

Dvs., at når der sendes et antal bytes efter hinanden, forventer modtageren typisk at det samme antal bytes ankommer.

Men glipper sendingen eller modtagelsen af antallet af bytes af en eller anden grund, vil modtagelsen let gå i udu. Komme ud af synkronisation. Ud af synk.

Det kan fx skyldes, at der bliver sendt for få bytes, eller at modtageren måske pga. forstyrrelser tror, der er sendt for få.

Eller at der modtages for mange bytes. Det kan føre til, at sidste byte af en transmission måske opfattes som den første af næste transmission.

Det er der her forsøgt at råde bod på.

8 Bytes sendes:

Her er der valgt at sende en datapakke bestående af 8 bytes.

Datapakken består af en header og en footer til hhv. at angive starten og slutningen af en transmissions-pakke.

En header og footer er ”bare” to valgte hexkoder. Her er valgt 0x02 og 0x03.

Modtageren kan herved indrettes til at vente på, at der ankommer en header, før den skal reagere. Og den kan tjekke, at sidste ankomne byte er en footer.

Herudover er det valgt, at der sendes en byte med tværsummen af de sendte bytes – bortset fra header og footer. Det gør, at modtageren kan udføre et tjek af korrektheden af de modtagne bytes ved at udregne tværsummen af de modtagne bytes og sammenligne værdien med den modtagne tværsum.

Og så skal der jo også sendes de data, der ønskes sendt.

Sendes 8 byte, er der herefter plads til 5 bytes udover header, tværsum og footer.

Disse 5 bytes kan så indrettes til at overføre de data, man ønsker.

I dette eksempel er der her valgt at der sendes 2 bytes, der kan opfattes som ” Sender-ID ” og ” Datatype ”.

Dvs. systemet vil kunne bruges til at modtage fra et antal sendere, der hver kan sende forskellige typer af data.

Ud af de 8 bytes, er der så 3 bytes tilbage til data.

En datapakke ser herefter således ud:

```
{ header, senderID, dataType, 0xB2, 0xA1, 0xA2, tvaersum, footer }
```

Men det vil jo altid kunne tilrettes til ens eget system. Fx kan Sender-ID og Datatype sendes som highnibble og lownibble pakket i 1 byte.

Yderligere kan der jo ændres i programmet, så der sendes flere databytes.

Definering af Array:

Altså, der sendes et Array – (eller en String ?) - via radiosenderen HC12.

Defineringen af valgte Array ser nu således ud:

```
uint8_t sendval[8] = { header, senderID, dataType, 0xB2, 0xA1, 0xA2, tvaersum, footer };
```

Først headeren, der her er valgt til 0x02, en senderID, en Datatype, 3 databytes, (her 0xB2, 0xA1 og 0xA2), en byte med udregnet tværsum, og sidst en footer, 0x03

De tre databytes kan fx være aflæsning af et termometer, en vejecelle-forstærker, eller hvad man nu har behov for at sende.

Senderen:

Data i dette eksempel sendes ikke kontinuerligt. Der er gjort brug af at man kan sætte en uC i dvale i en tidsperiode. Herved spares strøm, men det kan også bruges til at der sendes data med et givent interval.

Senderen er her sat til at sove i 8 sekunder. Når den vågner, incrementer den en tæller, og hvis et bestemt antal gange 8 sekunder er gået, foretages en måling, og der sendes data.

Eksemplet her er baseret på en simpel måling af, om en garageport er åben eller lukket.

Værdien af portstatus placeres i den første databyte. De to øvrige kan fx bruges til at indeholde en temperaturmåling, fx med LM35.

For at klargøre dataene til sending, udregnes en tværsum af alle databytes, dvs. undtagen headeren, footeren. Tværsummen placeres i næst sidste byte.

Modtagerens Timing:

I modtageren er der brugt en timer, der automatisk vil forkaste de modtagne data, hvis der ikke inden for en given tid ankommer det korrekte antal bytes.

Når en header er modtaget, startes der en tidsmåling, - og hvis der ikke er modtaget 8 byte inden en bestemt tid (en timeout-tid) forkastes de modtagne bytes, og der ventes igen på en header.

Hvis den sidst modtagne byte er en footer, udregnes en tværsum af byte1 til byte5, og den sammenlignes med den modtagne tværsum, udregnet og sendt af senderen. Er disse tjeck OK, er der rimelig chance for at transmissionen af hele arrayet er lykkedes.

Til Time-out-funktionen bruges funktionen millis().

I dette eksempel er modtager-programmet indrettet så der i en LCD gives forskellige beskeder. Fx at der ikke kommer korrekt antal data i en pakke, eller at der ikke kommer data overhovedet.

Men også, - hvis transmissionen lykkes, skrives tværsummen, ID, Datatype og om porten er lukket eller åben.

LCD:

Programmet til modtageren er baseret på et LCD-display med 4 x 20 karakterer. Dvs. der er rig mulighed for at ” vise ” resultatet af en transmission.

LCD-Displayet kan – som i eksemplet her – fx vise senderID, Datatype, og tværsum. Eller angive fejl-transmission.

HEX i LCD-display:

Når der på LCD-skærmen skal vises data i HEX, vil værdien der skrives jo kunne være under 10_{Hex}. Fx hvis en værdi i decimal er 13, vil det vises som C.

For at få skrevet et foranstillet ”0x0” bruges følgende kodestump:

```
lcd.print("Tvaersum = ");  //
lcd.print("0x");
if (tvaersum < 16) lcd.print("0");
lcd.print(tvaersum, HEX);  //
```

Senderprogrammet:

Processoren i senderen sættes til at gå i dvale et antal sekunder. Når den vågner, - foretager den en måling, og sender en data-pakke.

Og programmet er indrettet med brug af et antal Subrutiner, for at øge overskueligheden.

```
/*-------------------------------------------------------------------------*
Seriel sender eksempel med ATMega sleep
i Power Down Mode. Der gøres ikke brug af biblioteker.
Processoren Wakes med WatchDog - hver 8 sekunder
Der sendes serielle data efter x gange 8 sekunder

Temperaturmåling adderet 9/8-24

Der sendes et array, med Header, SenderID, dataType, 3 databytes, Tværsum og Footer,

Startet 30/6-2024
revideret 26/7-2024, 9/8-24, 29/05-25

/ Valle
 *-------------------------------------------------------------------------*/

const int LED = 13;              //LED on pin 13
const unsigned int wait = 1000;  //milliseconds

uint8_t counter = 0;     // antal sleep a´ 8 sekunder
uint8_t antalx8sek = 2;  // antal x 8 sekunder sleep

uint8_t garagePortPin = 5;  // Pin

// Def af string, med default værdier:
uint8_t header = 0x02;
uint8_t senderID = 0x0F;  // sender-ID
uint8_t dataType = 0xC1;
uint8_t tvaersum = 0x00;
uint8_t footer = 0x03;

uint16_t reading = 0;
uint8_t sendval[8] = { header, senderID, dataType, 0xB2, 0xA1, 0xA2, tvaersum, footer };
// Arrays har numre fra 0 og frem, her 0 til 7

#include <SoftwareSerial.h>

SoftwareSerial HC12(8, 9);  // RX, TX. ( RX til Tx på HC12 )

//*********************************

void setup(void) {

  HC12.begin(9600);

  pinMode(LED, OUTPUT);    //LED-pin er output
  digitalWrite(LED, LOW);
```

```
  pinMode(garagePortPin, INPUT);
  analogReference(INTERNAL); // Sæt intern ref til 1,1 Volt

  setupWatchDogTimer();   //
  delay(1000);
}  // EndSetup

//********************************

void loop(void) {

  if (counter >= antalx8sek) {  // der er gået en tid. hvad skal ske ?
    counter = 0;                // reset Sleep-# counter

    //flash LED, - for test
    digitalWrite(LED, HIGH);
    delay(100);
    digitalWrite(LED, LOW);
    delay(100);

    maalPort();    // Mål garageport-status
    maalTemp();
    udregnTvaersum();
    sendString();  // Send Data

    delay(100);  // All bytes sent ?
  }
    ATMegaGoToSleep();  // ATMega Sleeps, return next line after wakeUp
}  // EndLoop

//********************************

void maalPort() {

  sendval[3] = digitalRead(garagePortPin);
}
//********************************
void maalTemp(){
 // mål temperatur. Rå data stoppes i sendval 4 og 5
 reading = analogRead(A1);
  sendval[4] = highByte(reading);
  sendval[5] = lowByte(reading);
}
//********************************
void udregnTvaersum() {    // udregn Tværsum
  sendval[6] = (sendval[1] + sendval[2] + sendval[3] + sendval[4] + sendval[5]);
  //sendval[6] = tvaersum;
}
//********************************

void sendString() { // Send Array
  for (int i = 0; i < 8; i++) {
    HC12.write(sendval[i]);
    delay(10);
  }
}
//********************************

void ATMegaGoToSleep() {
  bitSet(SMCR, 0);        //sleep_enable();
  sei();                  //ensure interrupts enabled so we can wake up again
  asm volatile("sleep");  // Volatile for at "hjælpe" compileren
```

```
  //wake up here
  bitClear(SMCR, 0);  // Disable Sleep
}  // End-goToSleep
//*******************************

void setupWatchDogTimer() {  // Setup Watchdog Timer
  WDTCSR = 0b00011000;       //0x18;      change enable and WDE - also resets
  WDTCSR = 0b00100001;       //0x21; // Prescalers only - get rid of WDE og WDCE bit
  bitSet(WDTCSR, 6);         // Enable interrupt mode
  bitSet(SMCR, 2);           //set_sleep_mode(SLEEP_MODE_PWR_DOWN);
}  // End setupWatchdDogTimer. Husk ISR
//*******************************

// Når Atmega vågner efter WDT timeout, udføres et interrupt.
ISR(WDT_vect) {
  counter++;  // for hver 8 sek øges counteren !
}  // End-WatchDog-ISR
```

Modtagerprogrammet:

```
/* HC12-String-modtager

  Startet 30/6-2024
  Senest modificeret 8/1-25

  Programmet modtager en string fra HC12.
  Der tjekkes om der er kommet rigtig antal bytes, SenderID, Datatype, 3 databytes,
  Tværsum kontrolleres, - og der er indbygget timeout, der løser problemet,
  hvis der ankommer for få bytes i en datapakke, eller der er kommet ud af sync.
  9/8-24, tilføjet overførsel af temperaturmåling. Ankommer som rå data med highByte i
  rxBuffer 4 og lowByte i rxBuffer 5
  / Valle
 */

byte modtaget = 0;

// Array-def:
uint8_t rxBuffer[8];    // = { header, senderID, dataType, 0xA1, 0xA2, 0xA3, tvaersum, footer
};
uint8_t header = 0x02;  // default values
uint8_t senderID = 0x00;
uint8_t dataType = 0x00;
uint8_t tvaersum = 0x00;
uint8_t footer = 0x03;
bool tvaersumtjeck = 0;  //

uint8_t bufferDefault = 0x01;
uint16_t tempReading = 0; // @ Temperatur

float tempC = 0.0;  // udregning af temp

// værdier for Timeout
unsigned long dataModtagetStamp = 0;       // hvis der ikke kommer data
unsigned long dataUdeblevetDelay = 15000;  // 15 sekunder uden data
uint16_t dataUdeblevetTimer = 0;           //
unsigned long timeoutStarttime = 0;  // def af var til timeout
unsigned long timeouttime = 1000;

#include <SoftwareSerial.h>
```

```
SoftwareSerial HC12(8, 9);  // RX, TX. ( RX til Tx på HC12, og .. )

#include <LiquidCrystal.h>

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// ******************************

void setup() {
  lcd.begin(20, 4);
  lcd.print("HC12-Array-modtager");
  HC12.begin(9600);
  // Serial.begin(9600);
  dataModtagetStamp = millis();  // sæt stamp for data modtaget
}  //***  EndSetup  ******
// ******************************

void loop() {
  if (HC12.available()) {  //
    modtaget = HC12.read();
    //
    if (modtaget == header) {
      rxBuffer[0] = modtaget;
      timeoutStarttime = millis();                                  // for en pakke
      for (int i = 1; i < 8; i++) {                                 // læs alle øvrige bytes
        while (!HC12.available()) {                                 // Vent til de ankommer
          if ((millis() - timeoutStarttime) > timeouttime) break;   // Hop ud af loop
        }                                                           // vent til data eller time
        rxBuffer[i] = HC12.read();                                  // Gem data
      }
      // alle bytes er nu modtaget

      if (rxBuffer[7] == footer) {  // Data er OK
        skrivDataModtaget();

        udregnOgTjekTvaersum();

        if (tvaersumtjeck == 0) {
          skrivTvaersumFejl();
        } else {              // == true
          tvaersumtjeck = 0;  // læg flag ned
                              // Test fra hvilken sender og datatype
          if ((rxBuffer[1] == 0x0F) && (rxBuffer[2] == 0xC1)) {
            doStuf();
            skrivSenderID();
            skrivPortStatus();
            udregnOgSkrivTemp();
          }
          sletModtageBuffer();
        }
      }
    }
  }

  // hvis der går lang tid uden data ankommer

  if ((millis() - dataModtagetStamp) > dataUdeblevetDelay) {
    lcd.setCursor(0, 1);
    lcd.print("Data ikke modtaget");  // i ?? sek
    lcd.setCursor(0, 2);
    lcd.print("                    ");
```

```
    lcd.setCursor(0, 2);
    lcd.print(dataUdeblevetTimer++);  // antal delay-sekunder
    lcd.setCursor(0, 3);
    lcd.print("Tjek dataforbindelse");
    dataModtagetStamp = millis();  // ny periode
  }
}// EndLoop
// ******************************

void udregnOgTjekTvaersum() {
  tvaersum = (rxBuffer[1] + rxBuffer[2] + rxBuffer[3] + rxBuffer[4] + rxBuffer[5]);
  //
  if (tvaersum == rxBuffer[6]) {
    tvaersumtjeck = 0x01;  // tværsum er OK
  }
} // ******************************

void sletModtageBuffer() {  // Ryd op !!
  // Slet ModtageBuffer
  for (int i = 0; i < 8; i++) {
    rxBuffer[i] = bufferDefault;
  }
  dataModtagetStamp = millis();  // gem stamp for at data er modtaget korrekt
  dataUdeblevetTimer = 0;        // når data kommer igen, skal tiden resettes
} // ******************************

void doStuf() {
  // doStuf kan indrettes, så der udføres forskelligt afh. at senderID og datatype.
} // ******************************

void skrivPortStatus() {
  lcd.setCursor(0, 3);
  if (rxBuffer[3] == 0x00) {
    lcd.print("Port Lukket ");
  } else if (rxBuffer[3] == 0x01) {
    lcd.print("Port Aaben  ");
  }
} // ******************************

void udregnOgSkrivTemp() {  // temperatur:  Kommer i rxBuffer 4 og 5, ( High og Low ).
  tempReading = rxBuffer[4];
  tempReading = tempReading << 8;
  tempReading = tempReading + rxBuffer[5];  // 2 bytes til 16 bit.
  tempC = (float)tempReading; // konverter til float-type
  tempC = ((tempC * 110.0) / 1023.0);  // omregn til C med 1,1 V ref !!
  // print temperatur
  lcd.setCursor(12, 3);
  lcd.print(tempC, 1);  // 1 decimal
  lcd.print("C   ");
} // ******************************

void skrivDataModtaget() {
  lcd.setCursor(0, 1);
  lcd.print("data modtaget     ");
  delay(2000);
} // ******************************

void skrivSenderID() {  // og ID og Type
  lcd.setCursor(0, 1);
  lcd.print("Tvaersum = ");  // test
  lcd.print("0x");
  if (tvaersum < 16) lcd.print("0");
  lcd.print(tvaersum, HEX);  //
```

```
  //
  lcd.setCursor(0, 2);
  lcd.print("ID: ");
  lcd.print("0x");
  if (rxBuffer[1] < 16) lcd.print("0");
  lcd.print(rxBuffer[1], HEX);
  //
  lcd.print(" Type: ");
  lcd.print("0x");
  if (rxBuffer[2] < 16) lcd.print("0");
  lcd.print(rxBuffer[2], HEX);
} // ****************************

void skrivTvaersumFejl() {
  lcd.setCursor(0, 1);
  lcd.print("data fejlede");
  delay(1000);
  lcd.setCursor(0, 1);
  lcd.print("  --        ");
}
```

/ Valle