Kommunikasjonsprotokoller på Arduino Uno

Heisann! Arne ternger litt hjelp til noen gjøremål som han ikke rakk å gjennomføre over sommerferien. Han skal blant annet gjøre lysbrytere sine smarte, lage et overvåkingssystem til kontoret hans og bygge et luftkontroll og monitoreringssystem til minidrivhuset hans. Heldigvis har han en del komponenter til overs. Noen av desse prosjektene ønsker han å kontrollere fra PCen, mens andre vil han bruke en OLED skjerm for å vise fram verdier. For å hjelpe Arne med prosjektene sine må du derfor innom en liten introduksjon til ulike kommunikasjonsprotokoller.

Innhold:

UART - Kommunikasjonsprotokollen

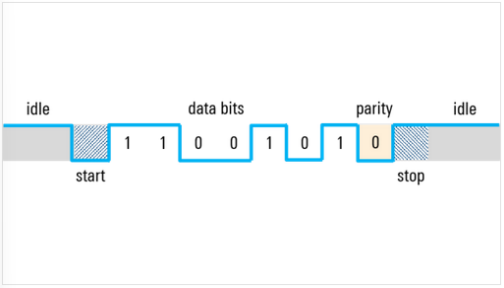
Teori

Generelt om UART

UART(Universal asynchronous receiver/transmitter) er en kommunikasjonsprotokoll som bruker to ledninger for å overføre seriell data mellom to enheter. Seriell kommunikasjon omhandler å sende data èn og èn bit av gangen, i motsetning til parallell overføring der mange databits blir overført på samme tid. I de fleste system blir lav spenning på ledningen sett på som 0, og høg spenning sett på som 1. Dermed kan man sende informasjon ved å justere denne spenningen. Det at dataoverføringen er asynkron vil si at begge enheter kan overføre data når de vil, uten at noe styrer dataflyten. Hver enhet har en RX- og TX-inngang. RX mottar informasjon og TX sender ut informasjon, desse krysskobles mellom de to enhetene (Figur). Videre har begge enheter fastsatt hastigheten som bitsene kan overførest med, dette kalles baudrate. De vanligste baudratene brukt i dag er 4800, 9600, 19.2k, 57.5k, 115.2k.

Dataoverføring med UART

Siden dataoverføring over UART er asynkron er det nødvendig å sende noen kontrollbits for å indikere hvor en melding starter/slutter og i noen tilfeller sjekke om datapakken er korrekt overført. Start og Stopp bits indikerer starten og slutten på dataoverføring. Startbiten er alltid å dra linjen fra høy til lav, dette er etterfulgt av databits. Stoppbiten er da enten a løfte linjen fra lav til høy(om forrige bit var lav), eller forbli høy i en ekstra periode(om forrige bit var høy). Det mulig å bruke to stopp bits, men dette er svært uvanlig i praksis. Data bits er informasjon som skal overførest, det kan være mellom 5 og 9 data bits, men i praksis er 7 eller 8 mest vanlig. Data bits er som regel sendt med minst signifikante bit, LSB, først. Paritetsbiten er et valgfritt tillegg som kommer mellom siste databit og stoppbit. Denne blir brukt til feildeteksjon, gjennom jamn eller ujamn paritet. Dersom systemet har jamn paritet vil denne biten bli satt slik at det er ett partall 1ere i pakken. Dersom systemet har ujamn paritet vil denne biten bli satt slik at det er ett oddetall 1ere i datarammen.



Figur 1: Generell UART-ramme

Figur 2: Overføring av 7-bit ASCII 'S' uten paritet

UART på Arduino Uno

Det er innebygd seriell kommunikasjon på Arduino Uno på pinne 0(RX) og pinne 1(TX), men dersom man ønsker å kommunisere med flere enheter er dette mulig å sette opp med biblioteket SoftwareSerial.h. Videre kan man lese om de innebygde seriell-funksjonene og andre brett sine pinner i Arduino sin dokumentasjon. Noen viktige funksjoner i det innebygde biblioteket til Arduino er:

Funksjon	Beskrivelse
Serial.begin(hastighet)	Initialiserer kommunikasjon, hastighet referer til baudraten
Serial.print(data)	Sender data i form av ASCII tekst
Serial.write(data)	Sender rå binærdata over seriellporten
Serial.available()	Retunerer tallet på bytes som er tilgjenglig i seriell buffer
Serial.read()	Leser en byte med innkommende data fra bufferen

Eksempel

Kommunikasjon med Seriemonitor:

I dette programmet skal en bruker skrive inn en melding til Arduino. Denne meldingen blir deretter printet ut av Arduinoen med tillegget “Jeg mottok: ”

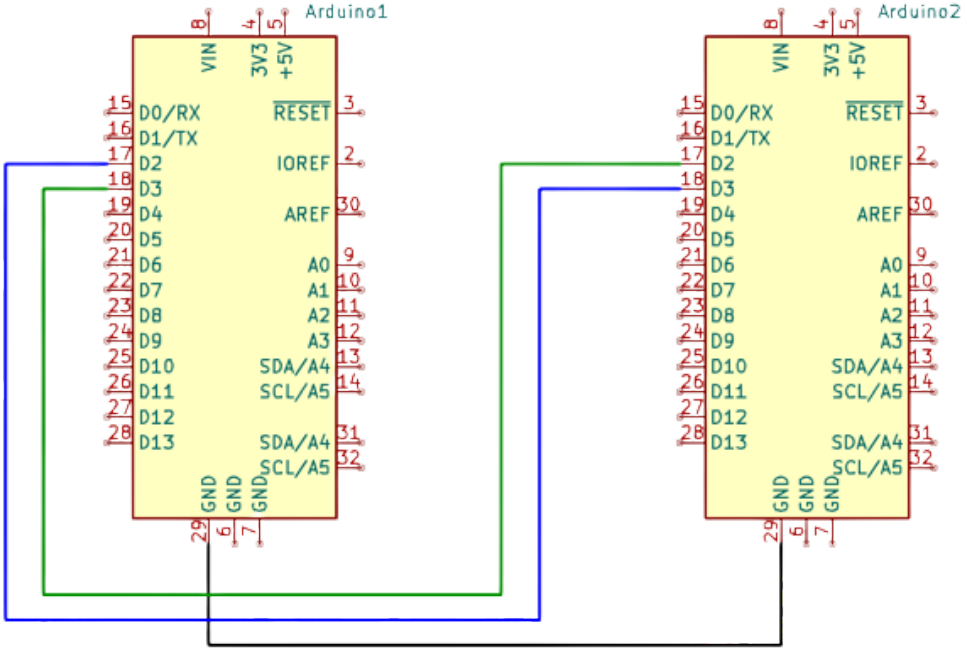
1. Koble 1 Arduino Uno til PCen din via USB
2. Last opp dette programmet fra Arduino IDE

```
void setup() {
  //Starter Seriekommunikasjon med baudrate 9600
  Serial.begin(9600);
}
void loop() {
  //Dersom det er noe i seriekommunikasjonsbufferen
  if(Serial.available()>0){
  //Les av informasjonen som String
    String melding = Serial.readString();
  //Denne funksjonen fjerner eventuelle '\n' og '\r' fra endene til
    meldingen
    melding.trim();
  //Print meldingen
    Serial.println("Jeg mottok: " + melding);
  }
}
```

Kommunikasjon med SoftwareSerial

I dette programmet skal brett 1 overføre strengen "Hei, dette er sender" til brett 2. Men, her blir det brukt pinner som ikke har innebygd UART. Derfor må både sender og mottaker importere biblioteket SoftwareSerial. Dette lar oss bruke andre pinner til UART overføring. Merk her hvordan vi både har Serial, som referer til pinne 0 og 1, og mySerial, som referer til pinne 2 og 3. Her bruker vi konsekvent mySerial til dataoverføring mellom enhetene, og Serial blir brukt til å kommunisere med seriemonitor.

1. Koble sammen brettene som vist



2. På Arduino Uno 1(Sender) last opp følgende kode

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(2,3);
void setup(){
      Serial.begin(9600);
      mySerial.begin(9600);
}

void loop(){
      mySerial.println("Hei, dette er sender!");
      delay(1000);
}
```

3. På Arduino Uno 2(Mottaker) last opp følgende kode

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3);

void setup() {
    Serial.begin(9600);
    mySerial.begin(9600);
}

void loop() {
if (mySerial.available()) {
            Serial.println(mySerial.readString());
      }
}
```

Generering av UART-melding med ADALM2000

1. Last opp følgende program på Arduino Uno

```
void setup(){
      Serial.begin(9600, SERIAL_8N1);
}

void loop(){
      if(Serial.available()>0){
    String message = Serial.readStringUntil('!');
    Serial.print("Jeg mottok: ");
    Serial.println(message);
  }
  delay(500);
}
```

2. Koble sammen DIO0 på ADALM2000 med RX på Arduino Uno
3. I Scopy gå til 'Pattern generator'
    a. Skru på DIO0 og gå til 'Channel Settings'
    b. Sett Pattern UART
    c. Baud Rate: 9600
    d. Stop: 1
    e. Parity: None
4. Endre nå 'Data to Send' feltet med meldingen du vil sende f.eks 'Hallo!'
    a. Merk! I Arduino Uno koden .readStringUntil('!') gjør at programmet vil lese fram til ! er sendt
5. Send data med Single eller Run, dette kan nå lesest av i Serial Monitor

Oppgave - Servostyrte Lysbrytere

Arne vil gjøre lysbryterene i huset sitt smartere. I huset er det mange dimmbare lys og han ønsker å kunne styre desse på ulike måter. Han ønsker å kunne bytte innstillinger fra PCen, og han vil ha moduser der han setter en spesifikk lysstyrke, kontroll av servoen fra ett potentiometer, og at servoen justerer seg automatisk basert på lyset i rommet. I denne oppgaven skal du derfor lage en prototype av et slikt system. Du skal lage en tilstandsmaskin som lar brukeren bytte mellom moduser over UART. Her er noen forslag til innstillinger:

- Sett servoen til 0, 90 eller 180 grader
- Sett servoen til å sakte bevege seg mellom 0 og 180 grader
- Sett servoen til å være justerbar med ett potentiometer
- Sett servoen til å være justerbar med fotoresistor
- Juster servoen med 10 grader

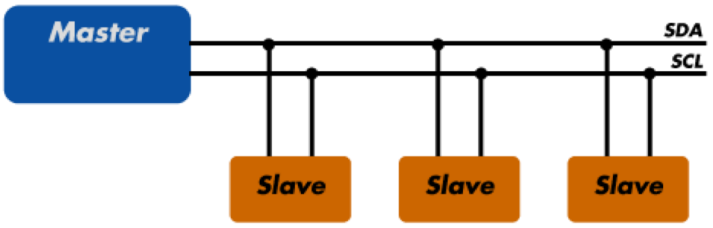
Løsningsforslag: Github

Utfordring: Bruk kommandoord i stedet for enkeltkarakterer

I²C - Kommunikasjonsprotokollen

Teori

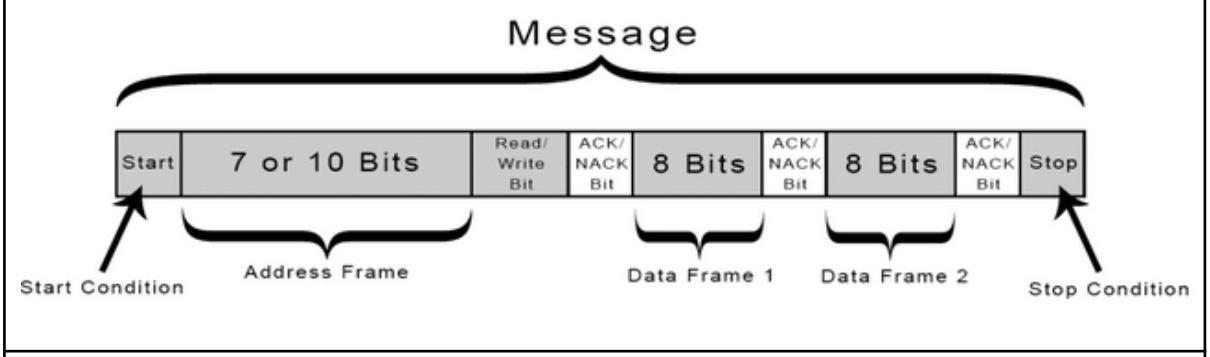
Generelt om I2C



I2C er en populær protokoll som lar flere enheter kommunisere med hverandre med bruk av kunn to linjer, desse linjene kallest en I2C-buss. I2C bruker en struktur som er basert på at en 'kontrollere' initierer kommunikasjon med en 'periferienhet'. Dette vil si at ingen informasjon blir overført uten at kontrolleren har bedt om det. For å gjennomføre dette oppfører de to linjene SCL(Serial Clock) og SDA(Serial Data) annerledes enn for eksempel: UART sine RX og TX linjer. SCL-linjen blir brukt til å sende ut ett klokkesignal fra kontrolleren, dette klokkesignalet bestemmer timingen til dataoverføringen. SDA blir brukt til å overføre seriell data, både kontroller og periferienhet kan sette denne til høy(1) eller lav(0). Ofte er I2C brukt til kommunikasjon mellom mikrokontrollere og sensorer, eller kommunikasjon mellom flere mikrokontrollere. Da kan for eksempel mikrokontrolleren hente ut data fra mange ulike sensorer uten å bruke mange linjer.

Dataoverføring med I2C

Med I2C er data overført i form av meldinger. Desse meldingene kan videre brytes ned i datarammer og kontrollbits. Selve dataoverføringen starter med at kontrolleren sender ut en start-kondisjon og deretter en adresseramme. Adresserammen blir brukt for å identifisere hvilken periferienhet kontrolleren vil kommunisere med. Adressen er enhetsspesifikk, og alle enhetene på bussen sammenligner denne delen av meldingen med sin egen adresse. Etter adressen kommer en bit som forteller periferienheten om den skal *motta* eller *sende* data. ACK/NACK blir sendt mellom hver ramme, dersom en ramme ble tatt i mot korrekt skal mottaker sende en ACK-bit i retur. Etter kontrolleren har mottatt den første ACK-biten, kan overføring av datarammer starte. Desse er alltid 8-bit lange med mest signifikante bit(MSB) først. Mellom hver av ramme skal ACK-bit bli sendt, før neste pakke kan overførest. Det er til slutt kontrolleren som stopper dataoverføringen med en stopp kondisjon.
Steg-for-steg dataoverføring med forklaring



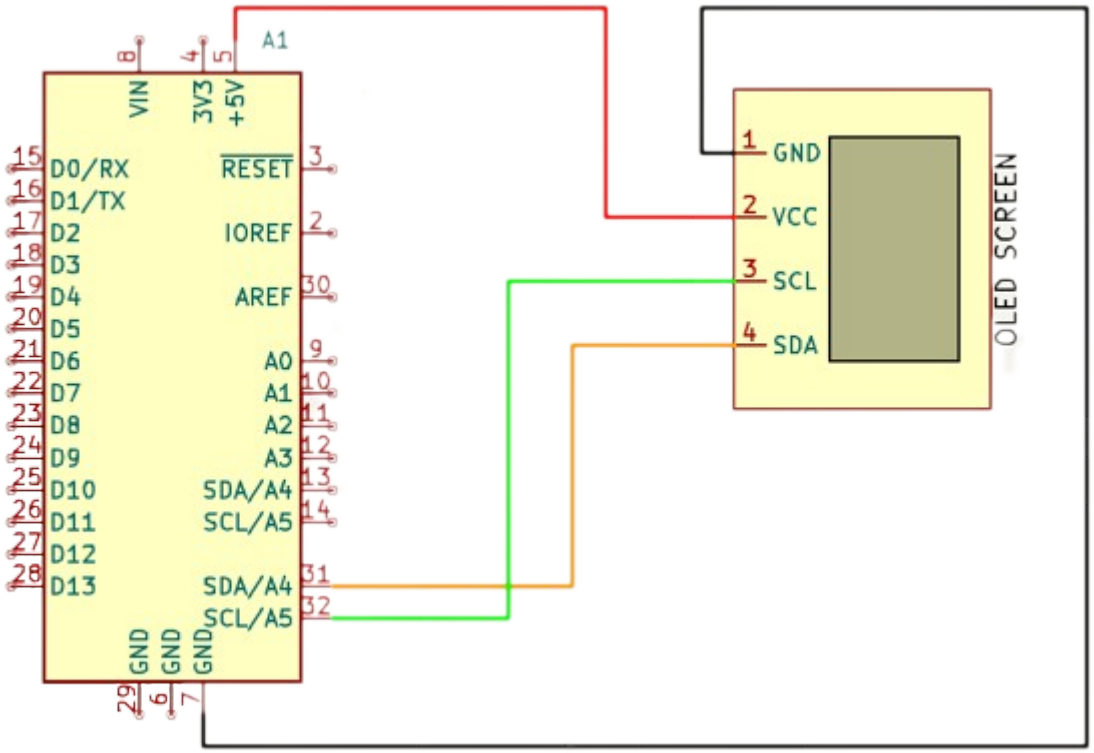
Figur 3: Generell I2C-melding

I2C på Arduino UNO

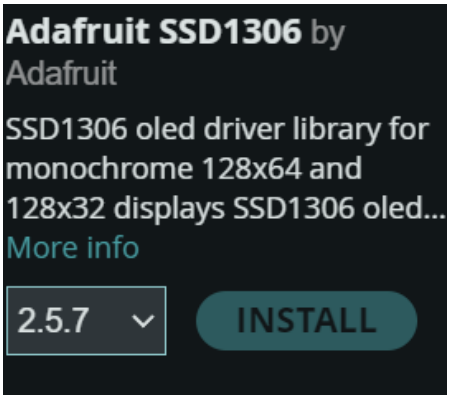
På Arduino Uno kan man bruke biblioteket <Wire.h> for å sette opp I2C kommunikasjon, men merk at det ofte kan finnes biblioteker som er bygd for komponenten du skal bruke. Desse inneholder ofte funksjoner som pakker inn behandling av data, noe som gjør jobben mye enklere.

Eksempel
Enkel OLED skjerm

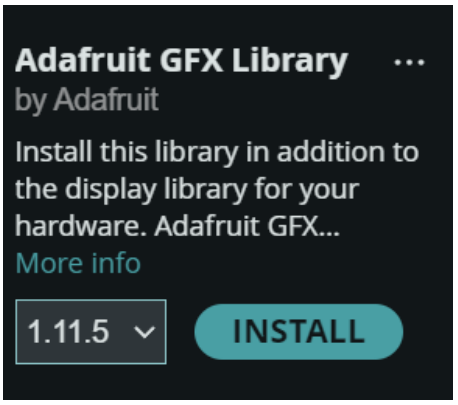
1. Koble Arduino Uno sammen med OLED skjermen slik:



2. Last ned biblioteket Adafruit SSD1306 under Sketch > Include Library > Manage Libraries



3. Last også ned Adafruit GFX biblioteket fra denne menyen



4. Last opp denne koden:

```
//Inkluderer bibliotek OLED skjermen trenger
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
//Oppgir størrelsen til skjermen
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
//Starter opp skjermen (innebygd biblioteksfunksjon)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

//Dette er bildet som visest på skjermen senere
const unsigned char myBitmap [] PROGMEM ={
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xfe, 0x8a, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xf9, 0x55, 0x5f, 0xff, 0xff,
0xff, 0xff, 0xff, 0xe4, 0x55, 0x2b, 0xff, 0xff,
0xff, 0xff, 0xff, 0xd2, 0xa4, 0xab, 0xff, 0xff,
0xff, 0xff, 0xff, 0xa9, 0x12, 0xa4, 0xff, 0xff,
0xff, 0xff, 0xff, 0x44, 0x8a, 0x55, 0x7f, 0xff,
0xff, 0xff, 0xfe, 0x92, 0x55, 0x2a, 0xbf, 0xff,
0xff, 0xff, 0xfd, 0x0d, 0x22, 0xaa, 0x9f, 0xff,
0xff, 0xff, 0xfa, 0x52, 0xad, 0x7f, 0x57, 0xff,
0xff, 0xff, 0xf9, 0x2a, 0xfb, 0xd5, 0xaf, 0xff,
0xff, 0xff, 0xf4, 0x97, 0x5e, 0xff, 0x57, 0xff,
0xff, 0xff, 0xf5, 0x2a, 0xef, 0xad, 0xdb, 0xff,
0xff, 0xff, 0xf4, 0xad, 0xfa, 0xff, 0x67, 0xff,
0xff, 0xff, 0xf5, 0x2b, 0x5f, 0xeb, 0xef, 0xff,
0xff, 0xff, 0xe8, 0xad, 0xed, 0xbe, 0xb3, 0xff,
0xff, 0xff, 0xf5, 0x56, 0xbf, 0xff, 0xef, 0xff,
0xff, 0xff, 0xe5, 0x2b, 0xed, 0xdb, 0x73, 0xff,
```

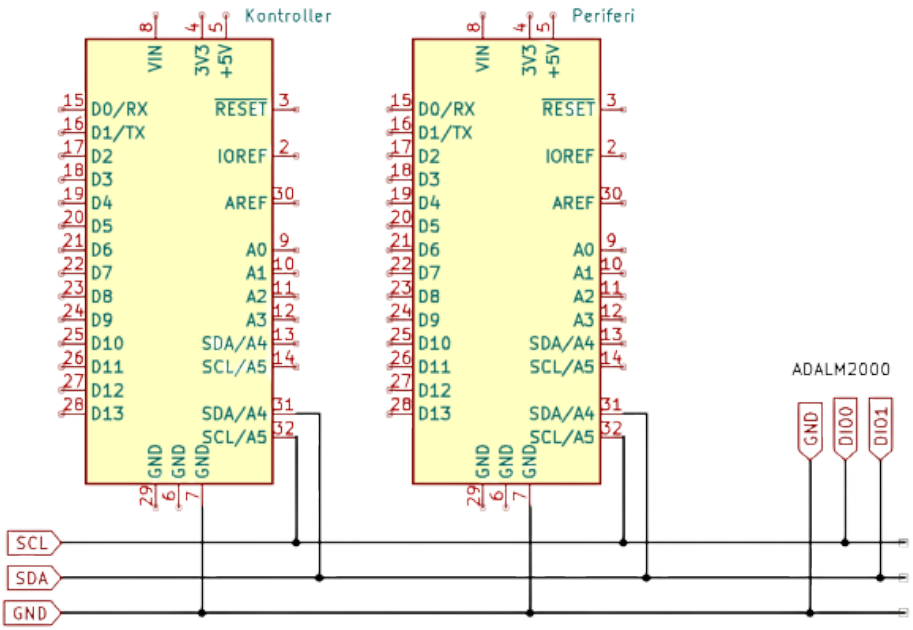
```
0xff, 0xff, 0xea, 0xaa, 0xbf, 0x7f, 0xdb, 0xff,
0xff, 0xff, 0xe4, 0x57, 0x77, 0xf5, 0x6b, 0xff,
0xff, 0xff, 0xea, 0x95, 0xdd, 0xbf, 0xf7, 0xff,
0xff, 0xff, 0xd2, 0x56, 0xbf, 0xed, 0xab, 0xff,
0xff, 0xff, 0xe9, 0x53, 0xed, 0x7f, 0x77, 0xff,
0xff, 0xff, 0x85, 0x2a, 0xbf, 0xed, 0xdb, 0xff,
0xff, 0xff, 0xd2, 0xa9, 0x55, 0xbf, 0x6d, 0xff,
0xff, 0xff, 0xaa, 0x4e, 0xbe, 0xe9, 0xb7, 0xff,
0xff, 0xff, 0x89, 0x25, 0x8b, 0xae, 0xeb, 0xff,
0xff, 0xff, 0xa4, 0x9b, 0xeb, 0x77, 0xbb, 0xff,
0xff, 0xff, 0x92, 0xa4, 0x57, 0xaa, 0xd7, 0xff,
0xff, 0xff, 0xca, 0x91, 0xaa, 0xd1, 0x7b, 0xff,
0xff, 0xff, 0x85, 0x4e, 0xeb, 0xb6, 0xd7, 0xff,
0xff, 0xff, 0xd4, 0xeb, 0x6a, 0xef, 0x7f, 0xff,
0xff, 0xff, 0xc2, 0xb5, 0xd7, 0xbb, 0xd7, 0xff,
0xff, 0xff, 0xd5, 0x5f, 0x69, 0x7d, 0x7b, 0xff,
0xff, 0xff, 0xc4, 0x55, 0xd7, 0xd7, 0xd7, 0xff,
0xff, 0xff, 0xea, 0xae, 0xca, 0xbe, 0xff, 0xff,
0xff, 0xff, 0xe1, 0x57, 0x57, 0xd7, 0xaf, 0xff,
0xff, 0xff, 0xea, 0x5a, 0x82, 0xba, 0xdf, 0xff,
0xff, 0xff, 0xf4, 0xab, 0x55, 0x6f, 0x6f, 0xff,
0xff, 0xff, 0xf2, 0x2a, 0xa2, 0xb2, 0xdf, 0xff,
0xff, 0xff, 0xe9, 0x55, 0x44, 0xab, 0x7f, 0xff,
0xff, 0xff, 0xc4, 0xaa, 0x2a, 0x5a, 0x9f, 0xff,
0xff, 0xff, 0xd5, 0x51, 0x49, 0x55, 0x7f, 0xff,
0xff, 0xff, 0x82, 0xa8, 0xab, 0x53, 0xbf, 0xff,
0xff, 0xff, 0xa9, 0x55, 0x0a, 0x2a, 0xff, 0xff,
0xff, 0xfe, 0x04, 0xaa, 0x2b, 0x57, 0x5f, 0xff,
0xff, 0xfc, 0xa2, 0x55, 0x55, 0xaa, 0xff, 0xff,
0xff, 0xf2, 0x01, 0x55, 0x5a, 0xdb, 0x7f, 0xff,
0xff, 0xe8, 0x48, 0x55, 0x25, 0x6d, 0xff, 0xff,
0xff, 0x82, 0x02, 0xa5, 0x55, 0x35, 0x7f, 0xff,
0xfc, 0x50, 0x10, 0x15, 0x4a, 0xd6, 0xbf, 0xff,
0xf5, 0x0a, 0x40, 0xa2, 0xd5, 0x75, 0x05, 0xff,
0xe9, 0x50, 0x00, 0x15, 0x6a, 0xd4, 0xb3, 0xff,
0x12, 0x22, 0x01, 0x21, 0x56, 0xaa, 0x88, 0x2f,
0xa4, 0x88, 0x84, 0x0a, 0x55, 0x50, 0x65, 0xab,
0x49, 0x42, 0x20, 0x01, 0x2a, 0xd3, 0x14, 0x55,
0x22, 0x28, 0x80, 0x0a, 0x92, 0x40, 0xa2, 0x89,
0x94, 0x92, 0x00, 0x40, 0x4a, 0x82, 0x54, 0x64,
0x49, 0x20, 0x50, 0x00, 0x91, 0x09, 0x12, 0x92,
0x24, 0x8a, 0x04, 0x00, 0x24, 0x80, 0x48, 0x49,
0x82, 0x44, 0xa0, 0x80, 0x08, 0x0a, 0xa2, 0xa4,
0x54, 0x92, 0x08, 0x01, 0x00, 0x48, 0x54, 0x12,
0x22, 0x41, 0x42, 0x20, 0x00, 0x22, 0x82, 0xa9
};
```

```
void setup() {
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Standard I2C
addresse for skjermen: 0x3C
    Serial.println(F("Klarte ikke å koble til skjermen"));
    for(;;);
  }
  //Setter opp generelle innstillinger
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
}

void loop() {
  //Print en setning
  display.setCursor(0,10);
  display.print("The man, the myth, the legend...");
  display.display();
  delay(5000);
  display.clearDisplay();
  //Tegn ett bilde
  display.drawBitmap(0, 0, myBitmap, 64, 64, WHITE);
  display.display();
  delay(5000);
  display.clearDisplay();
}
```

Dekoding av I2C-datastrøm med ADALM2000

1. Koble sammen to Arduino Unoer og ADALM2000 som vist:



2. Last opp denne koden til Arduino 1(Kontroller):

```
#include <Wire.h>
int x = 0;

void setup() {
  //Start I2C-bus
  Wire.begin();
}
void loop() {
  Wire.beginTransmission(9);  // Start dataoverføring med adresse 9
  Wire.write(x);              // Send x til adresse 9
  Wire.endTransmission();    // Stopp overføring
  x++; //Endre x
  if (x > 5) x = 0; // Resetter x om den blir høgere enn 5
  delay(1);
}
```

3. Last opp denne koden til Arduino 2(Periferi):

```
#include <Wire.h>
int x = 0;

void setup() {
  //Kobl på I2C-bussen med adresse 9
  Wire.begin(9);
  //Funksjonen mottarmelding vil kjøre når det mottas en melding
  Wire.onReceive(mottarMelding);
}
void mottarMelding(int bytes) {
  x = Wire.read();    // Les av en karakter fra I2C-bussen
}
void loop() {
  ; //Gjør ingenting(utenom når melding kommer)
}
```

4. Åpne Scopy og gå til Logic Analyser, anbefalte instillinger:
   a. Skru på DIO0 og DIO1
   b. Legg til decoder for i2c
   c. Sett SCL til å være DIO0, og SDA til DIO1
   d. Sample Rate: 2Msps, Nr of samples: 10k, Delay: 0
5. Kjør skopet og zoom inn på de ulike datapakkene
6. Hvilke datapakker gjenkjenner du fra koden?

Oppgave - Overvåking med OLED skjerm

Arne mistenker at noen går inn og ut fra kontoret hans på NTNU. Han vil derfor at du skal sette opp ett system der han kan sjekke bevegelser inn og ut på en OLED skjerm når han kommer tilbake. Du skal derfor sette opp en OLEDskjerm, to valgfrie sensorer som kan detektere at døren har blitt åpnet og en metode for å rullere mellom sensorverdier. Når han er borte vil han at skjermen ikke skal vise noe.

For eksempel:

Løsningsforslag: [Github]

SPI - Kommunikasjonsprotokollen

Teori

Generelt om SPI

Serial Peripheral Interface - SPI er en kommunikasjonsprotokoll som i likhet med I2C kan kommunisere med mange enheter på samme tid. SPI er vanlig brukt i RFID-lesere, SD-lesere og 2.4GHz trådløse mottakere/sendere for å kommunisere med mikrokontrollere. En ting som skiller SPI fra UART og I2C er at data kan overførest uten forstyrrelse i form av start-/stopp-kondisjoner.
I likhet med I2C bruker SPI et kontroller/periferisystem. Der kontrolleren bestemmer hvilken periferienhet som skal sende informasjon. I stedet for å bruke meldinger med adresser bruker SPI fire linjer for å kontrollere dataflyten.

- **MOSI(Master Output/Slave Input):** Linje for å sende data fra kontroller til periferi
- **MISO(Master Input/Slave Output):** Linje for å sende data fra periferi til kontroller
- **SCLK(Klokke):** Klokkesignal for å holde takt
- **SS/CS(Slave/Control Selecty):** Kontrolleren bestemmer periferienhet her

Noen fordeler med dette er at adressesystemet er enklere enn med I2C, både kontroller og periferi kan sende data på samme tid og at dataoverføringen kan være opptil dobbelt så rask som I2C. På den andre siden bruker denne protokollen mest fysiske ledninger, det er ikke et ACK-system som kontrollerer om pakker er kommet fram og det er ingen feilsjekkingsmetode som paritetsbiten i UART.

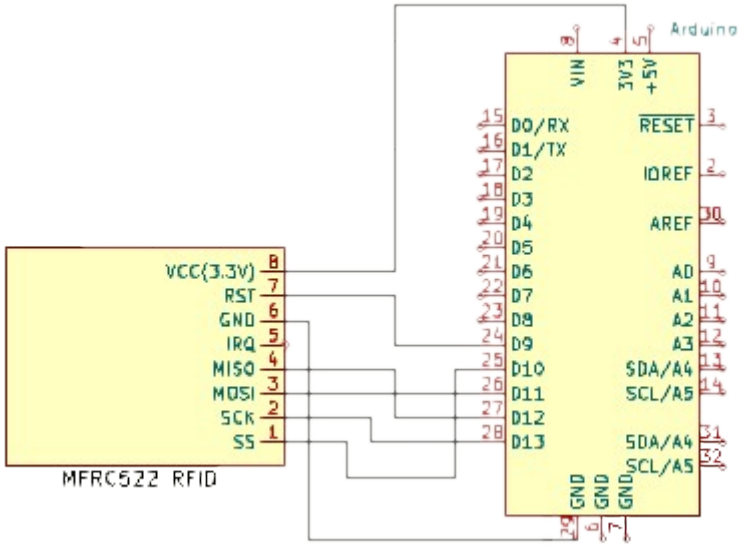
SPI på arduino

I likhet med I2C er det et innebygd SPI bibliotek <SPI.h>, men det er i tillegg skapt mange biblioteker som er skapt for spesifike komponenter. Det er ofte verdt å sjekke om et slikt eksisterer, da de som regel pakker inn kommunikasjon og databehandling slik at komponenten er enklere å bruke.

Eksempel

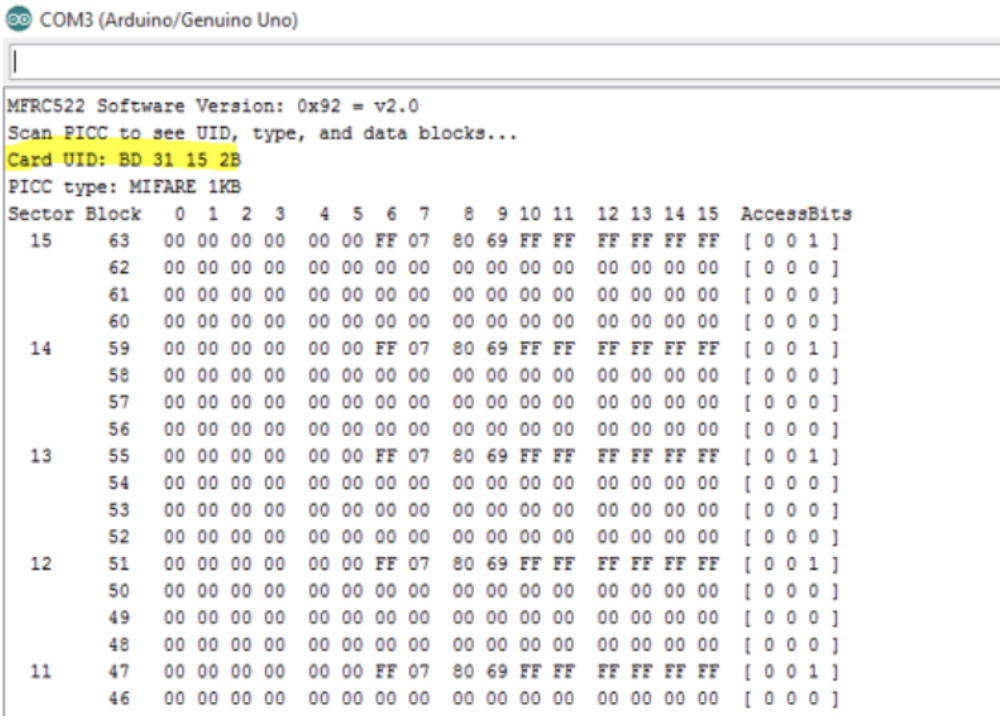
RFID-scanner med SPI

1. Koble sammen kretsen som vist:



MFRC522 RFID	Arduino UNO
SS/SDA	Digital 10
SCK	Digital 13
MOSI	Digital 11
MISO	Digital 12
IRQ	-
GND	GNN
RST	Digital 9
VCC	3.3V

2. Last ned RFID biblioteket skapt av miguelbalboa (merk! Dette biblioteket blir ikke lenger oppdatert)
    a. Åpne Arduino IDE gå til 'Sketch' -> 'Add .ZIP library'
    b. Velg .ZIP filen du lastet ned
3. Les av RFID-kortet ditt
    a. Gå til 'File' -> 'Examples' -> 'MFRC522' -> 'DumpInfo'
    b. Last opp denne eksempelkoden
    c. Åpne Seriemonitor
    d. Plasser RFID-kortet ditt nærme scanneren
    e. Denne informasjonen skal nå visest i seriemonitor:



f. Skriv ned Card UID

4. Last opp koden under til Arduino:

```
//RFID bibliotek og definisjoner
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup()
{
  //Start seriekommunikasjon
  Serial.begin(9600);
  //Start SPI dataoverføring
  SPI.begin();
  //Start mfrc522
  mfrc522.PCD_Init();
  Serial.println("Approximate your card to the reader...");
  Serial.println();
}
void loop()
{
  //Leit etter nye kort
  if ( !mfrc522.PICC_IsNewCardPresent() ){
    return;
```

```
  }
  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial() ){
    return;
  }
  //Leser av UID
  String content= "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++){
     content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : "
"));
     content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.print("Respons : ");
  content.toUpperCase();
  //Dersom kortUID matcher med denne strengen
  if (content.substring(1) == "BD 31 15 2B"){ //Endre til din UID
her!
    Serial.println("Tilgang tillatt");
    Serial.println();
    delay(3000);
  }
 else{
    Serial.println("Kort ikke godkjent");
    delay(3000);
  }
}
```

Demonstrasjonsvideo:

Oppgave - Drivhusmonitorering

Nå siden du har lært hvordan I^2C, UART og SPI fungerer, trenger Arne hjelp til ett lite prosjekt. Han ønsker å bygge en liten miljøstasjon som skal stå i drivhuset hans slik at han kan få perfekte tomater hver gang. Miljøstasjonen må ha temperaturmåling, måling av lysstyrke, ett vindu som kan åpnest til ulike vinkler og en vifte som kan kjøle ned på varme sommerdager. Han ønsker å kunne sette innstillinger til systemet med PCen sin over UART, men han vil i tillegg ha ett display som viser viktig informasjon kontinuerlig etter innstillingene er fikset.
Krav:

- Displayet **må** vise verdier fra minst to sensorer
- Man **må** kunne skru viften AV og PÅ over UART
- Man **må** kunne sette vinduet til å være 0, 45 eller 90 grader åpent over UART
- Man **bør** kunne sette vinduet til å justere vinkel automatisk etter tempertur eller lysstyrke
- Man **bør** kunne sette viften til å skru seg av og på automatisk etter temperatur og lysstyrke
- Displayet (eller annet tillegg) **kan** kunne varsle Arne om noe er galt
- Man **kan** legge til manuell modus fo vinkel- og hastighetsjustering med f.eks potentiometer
- Man **kan** bytte ut UART-styring med styring fra Keypad
- Man **kan** legge til sensor som henter inn luftfuktighet
    - Man **kan** legge til luftkvalitetsindikator basert på temperatur og luftfuktighet

Anbefalt arbeidsfordleing for grupper

- En person lager UART meny
- En person setter opp funksjoner til OLED skjerm
- En person setter opp funksjoner for sensorer og motorer

Løsningsforslag: Github

Videre lesing:

UART

Bruk av seriekommunikasjon arduino:
https://www.makeuseof.com/arduino-mastering-serial-communication/
Arduino sin egen referanse om UART:
https://www.arduino.cc/reference/en/language/functions/communication/serial/
Dypere forklaring av UART-biblioteket:
https://linuxhint.com/software-serial-library-arduino/
Teoretisk forklaring av UART:
https://www.rohde-schwarz.com/us/products/test-and-measurement/essentials-test-equipment/digital-oscilloscopes/understanding-uart_254524.html

I²C

Teoretisk forklaring av I2C(fokus hardware):
https://learn.sparkfun.com/tutorials/i2c/all
Bruk av OLED display:
https://randomnerdtutorials.com/guide-for-oled-display-with-arduino/
Teoretisk forklaring av I2C(fokus dataoverføring):
https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/
I2C med to Arduino Unoer
https://www.instructables.com/I2C-between-Arduinos/

SPI

Arduino sin egen referanse om SPI:
https://docs.arduino.cc/learn/communication/spi
Teoretisk forklaring SPI(Fokus på dataoverføring):
https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html
Teoretisk forklaring SPI(Generell):
https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/
Bruk av RFID med Arduino:
https://randomnerdtutorials.com/security-access-using-mfrc522-rfid-reader-with-arduino/